ANNÉE 2013

UNIVERSITÉ DE RENNES 1

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

École doctorale VAS

présentée par

Bacem WALI

préparée à l'unité de recherche VisAGeS–INRIA–INSERM U746 et
l'équipe MediCIS, LTSI–INSERM 1099
Sciences de la vie et de l'environnement

---

# Interopérabilité sémantique entre les outils de traitement d'images en neuroimagerie

Thèse soutenue à Rennes
le 21 Juin 2013

devant le jury composé de :

Mme ANITA BURGUN
Professeur à l'université Paris Descartes, Paris / Président de jury

Mr Djamal BENSLIMANE
Professeur à l'université claude Bernard Lyon 1, Lyon / Rapporteur

Mr Farouk TOUMANI
Professeur à l'université Blaise Pascal, Clermont-Ferrant / Rapporteur

Mr FRANCK MICHEL
Ingénieur de recherche I3S, Sophia Antipolis, Nice / Examinateur

Mr Marc CUGGIA
Maitre de conférence à l'université de Rennes 1, Rennes / Examinateur

Mr Bernard GIBAUD
Chargé de recherche INSERM, Rennes / Directeur de thèse

## Remerciements

Je tiens tout d'abord à remercier mon directeur de thèse, Mr Bernard Gibaud, Chargé de Recherches INSERM, qui m' a encadré et m'a accompagné tout le long de mes années de thèse. Je tiens à saluer son caractère conciliant, ses qualités humaines et ses valeurs morales, qui sont toutes aussi importantes que ses qualités scientifiques, et qui ont contribué, de manière significative, à mener à bien ce travail.

Mes respects et ma gratitude vont également aux membres du jury qui m'ont fait l'honneur de juger ce travail. Je remercie tout d'abord Mme Anita Burgun, Professeur des universités et praticien hospitalier.

Je remercie conjointement Mr Djamal Benslimane, professeur des universités à l'université Claude Bernard à Lyon et membre du laboratoire LIRIS et Mr Farouk Toumani, professeur des universités à l'université Blaise Pascal à Clermont Ferrand et membre du laboratoire LIMOS, d'avoir accepté de juger ce travail, ainsi que pour leurs remarques et l'intérêt qu'ils ont manifesté.

Je remercie également Mr Marc Cuggia, Maître de conférences des universités et praticien hospitalier au CHU de Rennes d'avoir accepté d'examiner ce travail.

Je remercie également Mr Franck Michel, ingénieur de recherche au laboratoire I3S à Sophia Antipolis et ex-contributeur au projet NeuroLOG, d'avoir accepté d'examiner ce travail.

J'adresse mes sincères remerciements aux collaborateurs de l'équipe I3S et de l'équipe MIS, Alban Gaignard, Johan Montagnat, Javier Rojas Balderrama, Gilles Kassel et Frédéric Fürst, qui m'ont fait bénéficier de leur expérience, et avec lesquels la collaboration a toujours été agréable.

Je remercie les autres membres de l'équipe de recherche qui m'a accueilli, en particulier, Tristan Moreau, Florent Lalys, David Bouget, Brivaël Trelhu, Adrien Férial, Omar El-Ganaoui, Pierre Jannin, Farooq Ahmad, Alexandre Abadie, Daniel Garcia Lorenzo, dans une ambiance à la fois stimulante et agréable.

Je n'oublie pas de remercier mes copines, pour leur sympathie et leur amitié en particulier Joana et Karine. Je remercie mes copains Ghassen et Kais. Ils ont toujours été là pour me soutenir, me divertir et me changer les idées quand il le fallait.

Je termine par exprimer ma gratitude, ma reconnaissance éternelle et mes remerciements les plus chaleureux aux êtres les plus chers à mon cœur qui m'ont soutenu tout le long de cette thèse : ma mère Amal, mon père Abdelouaheb ainsi que mes sœurs, Salma et Ibtissem et mon frère Slim.

# Table des matières


**1 Introduction** **1**

**2 Contexte de la thèse** **5**
2.1 Introduction . . . 5
2.1.1 Intégration de données en biomédecine . . . 6
2.1.2 Partage de données et de traitement pour la recherche . . . 7
2.1.3 Synthèse . . . 15
2.2 Contexte du travail : Le projet NeuroLOG . . . 15
2.2.1 Intégration de ressources en neuroimagerie . . . 16
2.2.2 Gestion de données et de métadonnées . . . 16
2.2.3 Partage des outils de traitement d'images . . . 21
2.2.4 Gestion des workflows et des pipelines . . . 23
2.2.5 Ontologie OntoNeuroLOG . . . 24
2.3 Analyse des besoins . . . 27
2.3.1 Limitation des descripteurs XML . . . 27
2.3.2 Description sémantique des fonctionnalités des services . . . 28
2.3.3 Composition automatique et sélection de service . . . 28
2.3.4 Manque de connaissances associées aux fichiers produits . . . 29
2.3.5 Assistance de l'utilisateur lors de la composition et l'exécution des services 29
2.3.6 Surveillance (monitoring) d'exécution des services . . . 30
2.4 Synthèse des objectifs généraux de la thèse et méthodologie . . . 30

**3 État de l'art** **33**
3.1 Vers une Architecture Orientée Services . . . 34
3.1.1 Architecture SOA (Service Oriented Architecture) . . . 34
3.1.2 Services web . . . 35
3.1.3 Utilisation de l'architecture SOA et des services web en biomédecine . 40
3.1.4 Synthèse . . . 41
3.2 Interopérabilité et web sémantique . . . 43
3.2.1 Types d'interopérabilité . . . 44
3.2.2 Web sémantique . . . 44
3.2.3 Ontologies . . . 50
3.2.4 Raisonneurs . . . 52
3.2.5 Ontologies et raisonnement en biomédecine . . . 53
3.2.6 Entrepôts de données sémantiques . . . 55
3.2.7 Moteurs de recherche et interrogation de données sémantiques . . . 57
3.2.8 Synthèse . . . 61
3.3 Composition des services . . . 62
3.3.1 Utilisation des registres de services web . . . 62
3.3.2 Registres de services en bioinformatique et biomédecine . . . 63

3.3.3 Synthèse . . . 65
3.4 Outils sémantiques pour la composition des services . . . 66
3.4.1 Service web sémantiques . . . 67
3.4.2 OWL-S : Ontology web Language for Services . . . 67
3.4.3 WSMO : Web Service Modeling Ontology . . . 74
3.4.4 METEOR-S . . . 80
3.4.5 Synthèse . . . 82
3.5 Workflow . . . 84
3.5.1 Composition des workflows . . . 86
3.5.2 Synthèse . . . 88
3.6 Bilan . . . 88

**4 Extension de OWL-S pour composer des services Web générés par des applications d'encapsulation de services 93**
4.1 Introduction . . . 93
4.1.1 Situation par rapport au travail . . . 94
4.1.2 Problèmes des services jGASW . . . 95
4.1.3 Motivation pour la sélection et la réutilisation de OWL-S . . . 96
4.2 Méthode . . . 96
4.2.1 Réutilisation de OntoNeuroLOG . . . 98
4.2.2 Extension de OWL-S . . . 98
4.2.3 Ajout des mécanismes de raisonnement . . . 101
4.3 Implémentation . . . 103
4.4 Discussion . . . 107
4.5 Conclusion . . . 109

**5 Nouveau modèle pour le partage de services de traitement d'images en Neuro-imagerie 111**
5.1 Introduction . . . 111
5.2 Outils et contexte du travail . . . 112
5.3 Proposition d'un modèle de description de service . . . 114
5.3.1 Ontologie des outils de traitement de services . . . 114
5.3.2 Articulation avec OntoNeuroLOG . . . 117
5.4 Implémentation des services sémantiques . . . 120
5.4.1 Annotation sémantique des services simples et composites . . . 121
5.4.2 Invocation sémantique des services simples et composites . . . 121
5.4.3 Génération et application de règles pour produire des métadonnées . . . 121
5.4.4 Vérification de la compatibilité entre le traitement et l'opération . . . 122
5.5 Aspect techniques de l'implémentation . . . 124
5.5.1 Architecture de la plateforme et du module sémantique . . . 124
5.5.2 Annotation des services simples ou composites . . . 124
5.5.3 Annotation des pré/postconditions . . . 127
5.6 Discussion . . . 131
5.7 Conclusion . . . 133

**6 Discussion** **135**
6.1 Évaluation du travail par rapport aux objectifs . . . 135
6.1.1 Intégration de la sémantique dans le client NeuroLOG . . . 135
6.1.2 Intégration de la sémantique dans le middleware NeuroLOG . . . 136
6.1.3 Gestion des données sémantiques . . . 137
6.2 Positionnement par rapport à l'état de l'art . . . 137

**7 Conclusion** **143**
7.1 Problème général . . . 143
7.2 Travail effectué . . . 143
7.3 Apport de la démarche . . . 144
7.4 Perspectives . . . 145

**A Annexe A** **147**
A.1 Exemple de composition de services JGASW avec OWL-S . . . 147
A.2 Extension de la description sémantique OWL-S . . . 155
A.3 Code modifié dans l'API OWL-S pour la composition . . . 159

**B Annexe B** **165**
B.1 Validation d'une opération et d'une classe de traitement . . . 165
B.2 Validation d'une orchestration . . . 170
B.3 Vérification du type de l'input sélectionnée par l'utilisateur et le type de la variable à la quelle l'input est assignée . . . 173
B.4 Utilisation du raisonneur HermiT pour la vérification de types pour les inputs et les outputs . . . 174
B.5 Génération d'une règle CORESE à partir de son annotation . . . 178
B.6 Diagramme de séquence . . . 183

# Table des figures


2.1 Architecture logicielle de caBIG (approche opérationnelle) [8] . . . 13
2.2 Architecture logicielle du système NeuroLOG [183] . . . 17
2.3 Interface d'interrogation de l'application relationnelle . . . 18
2.4 Les deux niveaux de METAmorphoses [188] . . . 20
2.5 Utilisation de CORESE dans l'architecture NeuroLOG [194] . . . 21
2.6 Interface cliente de jGASW au niveau de la plateforme NeuroLOG . . . 22
2.7 Partage et sélection des outils au niveau du client NeuroLOG . . . 22
2.8 Résultats retournés après l'exécution de l'outil de traitement d'images . . . 23
2.9 Exemple de workflow construit à l'aide de l'interface MOTEUR . . . 24
2.10 esquisse Script GWENDIA en XML . . . 25
2.11 Représentation en couches de l'ontologie OntoNeuroLOG . . . 27

3.1 L'architecture utilisée pour les services web [195] . . . 36
3.2 Les couches conceptuelles de la modélisation des services web [95] . . . 37
3.3 Utilisation de la messagerie XML dans SOAP [95] . . . 38
3.4 Description basique d'un service web [95] . . . 39
3.5 Les couches du web sémantique [87] . . . 47
3.6 Structure d'un triplet RDF . . . 48
3.7 Exemple d'un graphe RDF . . . 49
3.8 Transformation de connaissances représentés en OWL en un programme datalog disjonctif, effectué par KAON2 . . . 53
3.9 Exemple de règle CORESE . . . 60
3.10 Architecture "trois tiers" de Corese [61] . . . 60
3.11 Structure de base de UDDI [139] . . . 62
3.12 Architecture du projet $^{my}$GRID : utilisation du registre UDDI-M [135] . . . 64
3.13 Les différentes sous couches de l'ontologie de haut niveau : Ontologie de services de OWL-S [118] . . . 69
3.14 L'ontologie service profile dans la spécification OWL-S [118] . . . 69
3.15 L'ontologie Process dans la spécification OWL-S [118] . . . 70
3.16 Mapping entre les éléments des WSDLs et les éléments des process OWL-S via le Grounding [118] . . . 72
3.17 Les quatre éléments de base de WSMO [27] . . . 74
3.18 Architecture de IRS-III [109] . . . 76
3.19 Architecture de METEOR-S [162] . . . 80
3.20 Termes standards des systèmes de gestion de workflow [184] . . . 85

4.1 Grounding automatique d'un service jGASW en utilisant l'API WSDL2OWLS 97
4.2 Exemple d'utilisation de l'extension de la couche Process . . . 100
4.3 Composition des services jGASW à l'aide de l'extension qui a été proposée et du lien ajouté entre les paramètres . . . 101

4.4 Transformation d'un Profile en data-processing . . . . . 102
4.5 Annotation sémantique des services jGASW utilisant l'extension OWL-S . . . 106

5.1 Modèle pour la description sémantique des services (outils de traitement d'images dans NeuroLOG) . . . . . 116
5.2 Modèle pour la description sémantique des services (outils de traitement d'images dans NeuroLOG) . . . . . 117
5.3 Modèle de description d'exécution sémantique . . . . . 118
5.4 Modèle d'enrichissement sémantique à l'aide des règles sémantiques . . . . . 119
5.5 Modèle de liaison entre les descripteurs de base des services et les descriptions sémantiques . . . . . 120
5.6 Articulation de l'ontologie des services avec les axes des ontologies de DOLCE 120
5.7 Découpage de l'ontologie des dataset en datatype et rôle . . . . . 121
5.8 Exemple illustratif de la transformation d'une opération en une classe de traitement en utilisant les rôles . . . . . 123
5.9 Architecture de la plateforme NeuroLOG . . . . . 124
5.10 Architecture du module sémantique . . . . . 125
5.11 Annotation des services élémentaires : annotation de l'outil de recalage : Registration . . . . . 128
5.12 Annotation d'un workflow de traitement : débruitage-recalage . . . . . 129

B.1 Diagramme de séquence expliquant les phases d'implémentations : partie 1 . . 184
B.2 Diagramme de séquence expliquant les phases d'implémentations : partie 2 . . 185
B.3 Diagramme de séquence expliquant les phases d'implémentations : partie 3 . . 186

# Liste des tableaux

6.1 Présentation d'exemples de systèmes de composition de services . . . . . 139

Chapitre 1

# Introduction

La science ne cesse d'avancer et progresser dans les différents domaines, en particulier les neurosciences. Les neurosciences regroupent l'ensemble des disciplines scientifiques qui contribuent à une meilleure compréhension du fonctionnement du cerveau humain et de ses pathologies. A terme, elles visent à améliorer le diagnostic et le traitement des affections neurologiques, grâce à l'appuie de plus en plus important de la neuroimagerie.

La neuroimagerie recouvre différentes techniques permettant de visualiser le cerveau et ainsi de mieux connaître son anatomie et son fonctionnement normal ou pathologique. Elle possède de nombreuses applications en neurosciences. Les avancées technologiques en neuroimagerie ont permis de produire une quantité de données très importante largement disponibles grâce aux technologies informatiques [64]. En effet, des importantes ressources ont été mises en place pour mieux acquérir et analyser les données. Une tendance accrue vers un partage de ces données a émergé ces dernières années. L'augmentation du nombre, de la complexité et de l'hétérogénéité de ces ressources rendent difficile leur analyse et nécessaire leur intégration et partage posant naturellement la question d'interopérabilité des outils de traitements et des données[37] [36].

Il ne suffit pas de partager les ressources telles qu'elles sont mais il faut aussi les rendre interopérables. En effet, pour pouvoir analyser les données on doit le plus souvent leurs appliquer de nombreux prétraitements. Il faut donc pouvoir enchaîner des outils de traitement sous la forme de pipeline de traitement permettant d'avoir des analyses plus détaillées, plus rigoureuses et plus efficaces.

A ce jour, de nombreux outils sont disponibles pour l'analyse des images mais malheureusement ils sont dans la plupart des cas non partagés et non interopérables. Pour les mettre à la disposition des utilisateurs nous pouvons aujourd'hui les partager à travers le web. Pour les rendre interopérables, nous leur associons des descriptions permettant de leur ajouter de la sémantique. La combinaison des deux dans le cadre du web sémantique est devenue le moyen le plus sûr pour assurer l'interopérabilité et pour rendre les traitements "compréhensibles" par les agents logiciels et les machines. Défini par Tim Berners-Lee [18], le web sémantique permet de structurer les ressources à travers le Web en leur ajoutant des connaissances sous forme de métadonnées et en les rendant interopérables. Plusieurs auteurs définissent l'interopérabilité sémantique comme la capacité des services à échanger des données et communiquer entre eux [203]. Les services et les données sont fournies par des partenaires différents, par conséquent, l'interopérabilité se base sur des consensus et des conventions entre les partenaires. Des connaissances sont ajoutées pour décrire les spécifications techniques des services et c'est à l'aide de ces connaissances que l'interopérabilité est assurée.

L'interopérabilité sémantique repose sur l'inter-connectivité des composants en suivant

les normes standards informatiques du web sémantique. Ces standards sont présent sur la description des données, des services ou des paramètres relatifs à ces services en vue de définir une vue commune compréhensible par tout type d'agent, humain ou machine [44]. A l'aide des connaissances ainsi ajoutées, le contenu informel des composants du web est désormais plus explicite et plus compréhensible. L'automatisation, l'enchaînement des services et la composition de modules de traitement complexes est désormais envisageable.

Sur le Net on dispose d'un nombre important de services [152] qui proviennent de différents domaines d'application et sources hétérogènes. Le fait qu'ils aient des origines et des formats différents rend plus difficile leurs échanges de données et par conséquent leurs enchainements et leur interopérabilité.
Dans la plupart des cas, l'enchaînement des services consiste à lier les paramètres de sortie avec les paramètres d'entrée du service suivant. Cependant, en neuroimagerie la simple description de paramètres n'est pas suffisante pour avoir un enchaînement cohérent de services. Nous souhaitons bénéficier des atouts du web sémantique pour ajouter une description plus précise aux services qui prend en considération non seulement les types de données mais aussi les rôles de chaque paramètre et opération effectué par le service. Nous souhaitons aussi ajouter des mécanismes de raisonnement qui contrôleront la validité des données, des outils et des modules de traitement utilisés.

L'objectif de cette thèse est de proposer des solutions sémantiques de composition adaptées aux outils de traitement d'images déployés sous forme de services partagés sur internet. Nous allons pour cela exploiter les fonctionnalités des ontologies pour automatiser en totalité ou partiellement la composition de services et le contrôle de validité. Nous utilisons aussi les ontologies pour fournir une description des rôles des outils et de leurs paramètres. Nous mettons en place un mécanisme qui permet l'annotation des outils et des workflows de données crées à partir de ces outils, ainsi qu'un mécanisme qui permet d'ajouter des métadonnées aux données produites pour renforcer la cohérence des entrepôts de données sémantiques utilisés. Les ontologies proposées doivent s'inspirer des ontologies déjà existantes dans d'autres domaines d'applications et elles doivent respecter les principes ontologiques formels pour améliorer la description des données et ainsi bénéficier des atouts de l'intelligence artificielle pour créer des mécanismes de raisonnement aidant à réaliser des enchaînements cohérents de services. Ces ontologies doivent aussi suivre un consensus de modélisation extensible et ouvert pour qu'elles soient réutilisables dans autres contextes et avec d'autres plateformes similaires.

Les différentes contributions de ce travail sont présentées en quatre parties différentes contenant chacune un ensemble de chapitres.

La première expose le contexte de ce travail. Elle commence par un premier chapitre introductif dans lequel nous définissons le domaine dans lequel nous travaillons et la problématique générale que nous abordons. Dans le deuxième chapitre nous expliquons le contexte de notre travail, en l'occurrence le projet dans lequel nous travaillons. Pour cela nous présentons globalement les différents outils utilisés pour le partage et l'enchaînement des outils de traitement, en signalant les différentes limites de ces outils. Nous concluons cette partie avec la définition des différents objectifs de cette thèse.

La deuxième partie traite de l'état de l'art, elle détaille donc les outils et les techniques utilisés dans des domaines différents et qui ont servi à la composition des services, tout en

discutant leurs avantages et inconvénients par rapport au domaine de la neuroimagerie et au contexte du projet dans lequel nous travaillons.
Elle décrit aussi les différents outils utilisés pour l'enrichissement sémantique, la gestion des données sémantiques ainsi que les différents raisonneurs utilisés pour raisonner sur les métadonnées, en donnant à chaque fois les avantages et inconvénients de chacun par rapport au domaine de la neuroimagerie et au projet que dans le cadre duquel nous travaillons, de façon à orienter notre choix vers celui ou ceux qui seront les mieux adaptés à notre domaine.
Puis dans un second chapitre nous confrontons plus en détail les objectifs proposés au niveau de la deuxième partie avec les outils discutés au niveau de l'état de l'art pour pourvoir des solutions.

La troisième partie est la partie associé à nos résultats composée de deux chapitres présentant respectivement deux contributions différentes dont la première est une extension d'un des outils au profit de la plateforme avec laquelle nous travaillons et la deuxième présente une solution plus sophistiquée qui remplit la quasi-totalité des objectifs qu'on souhaitait atteindre dans la plateforme.

La quatrième et dernière partie, comprend deux chapitres, le premier fait l'objet d'une discussion détaillée de l'apport des travaux proposés par rapport aux travaux étudiés au niveau de l'état de l'art. Elle revient plus précisément sur les choix de modélisation, les méthodes utilisées ainsi que les stratégies de choix d'implémentation. Elle positionne nos travaux par rapport à d'autres travaux et projets existants. Finalement un dernier chapitre résume et met en avant les perspectives qu'ouvre ce travail.

Chapitre 2

# Contexte de la thèse

## Sommaire


**2.1 Introduction . . . . . 5**
2.1.1 Intégration de données en biomédecine . . . . . 6
2.1.2 Partage de données et de traitement pour la recherche . . . . . 7
2.1.3 Synthèse . . . . . 15
**2.2 Contexte du travail : Le projet NeuroLOG . . . . . 15**
2.2.1 Intégration de ressources en neuroimagerie . . . . . 16
2.2.2 Gestion de données et de métadonnées . . . . . 16
2.2.3 Partage des outils de traitement d'images . . . . . 21
2.2.4 Gestion des workflows et des pipelines . . . . . 23
2.2.5 Ontologie OntoNeuroLOG . . . . . 24
**2.3 Analyse des besoins . . . . . 27**
2.3.1 Limitation des descripteurs XML . . . . . 27
2.3.2 Description sémantique des fonctionnalités des services . . . . . 28
2.3.3 Composition automatique et sélection de service . . . . . 28
2.3.4 Manque de connaissances associées aux fichiers produits . . . . . 29
2.3.5 Assistance de l'utilisateur lors de la composition et l'exécution des services 29
2.3.6 Surveillance (monitoring) d'exécution des services . . . . . 30
**2.4 Synthèse des objectifs généraux de la thèse et méthodologie . . . . 30**


Dans ce chapitre nous donnons un aperçu sur la notion d'intégration de ressources et plus précisément lors de son application en biomédecine. Nous commençons par définir la notion d'intégration, lister les différentes techniques mises en jeux et finalement montrer l'importance de l'aspect sémantique pour fournir une solution efficace. Ensuite, nous présentons l'apport des ontologies dans le contexte d'une intégration sémantique. Et finalement nous présentons le projet NeuroLOG dans le cadre duquel nous travaillons et les différents aspects d'intégration de données qu'il supporte. La dernière section conclut avec une mise au point sur le travail fait dans le cadre du projet NeuroLOG et ses limites puis expose les nouvelles méthodes et stratégies suivies pour les repousser.

## 2.1 Introduction

Actuellement, les sources d'information sont de plus en plus distribuées et hétérogènes. Cette diversité rend plus difficile l'intégration des données d'origines diverses et la recherche

et le partage d'information spécialement dans un environnement en pleine expansion tel que le web.

Le partage d'information ou l'intégration de données peut être considéré comme la combinaison de plusieurs sources de données différentes, de sorte que l'accès à n'importe quelle donnée, soit organisé via une seule vue sur l'ensemble des données [104]. Pour assurer l'intégration de données plusieurs défis doivent être relevés. Le premier concerne l'hétérogénéité des ressources. Le deuxième concerne la multiplicité des approches d'intégration et la nécessité de les faire interopérer.

Dans le contexte de l'hétérogénéité des ressources on note l'hétérogénéité des systèmes qui résulte d'une utilisation variée des OS (Operating Systems) ou logiciels et codes. Ensuite on trouve l'hétérogénéité syntaxique qui concerne les formes de représentation des données (ou hétérogénéité structurelle) qui concerne les modèles utilisés. Enfin l'hétérogénéité sémantique est à notre égard la plus lourde à défier et le résultat d'une variété d'interprétation du sens de la donnée. Cela représente une contrainte très forte pour la recherche dans le milieu biomédical. Ce milieu est extrêmement large ce qui peut amener des ambigüités dans l'usage des différents termes utilisés par différents laboratoire de recherches ou dans différentes langues. Ceci peut pénaliser l'avancement de la recherche dans ce domaine, et de nouvelles méthodes doivent êtres mises en œuvre pour limiter cette hétérogénéité.

La recherche sur l'intégration de données s'est orientée vers la sémantique des données et l'intégration automatique des sources de données. En effet, en l'absence d'intégration automatique des données, les scientifiques devaient trouver manuellement les sources de données nécessaires, analyser leurs formats de représentation puis intégrer manuellement les différents résultats [75]. Compte tenu de l'augmentation du nombre et de la taille des ressources sur Internet, aujourd'hui une approche automatisée devient essentielle.

Comme Schonbach et al. l'ont expliqué, l'intégration des données représente une étape préliminaire et nécessaire pour l'analyse des données [173]. Cela est particulièrement vrai pour le domaine biomédical dans lequel l'application de méthodes à un ensemble de données relevant d'un domaine d'étude constitue la voie habituelle pour l'acquisition de nouvelles connaissances. Aujourd'hui avec l'augmentation de la taille des informations sur Internet, l'augmentation des besoins des utilisateurs a aussi augmenté. L'extraction, l'analyse et la comparaison des données à partir de ressources distribuées hétérogènes sont devenues à la fois un challenge et une nécessité.

### 2.1.1 Intégration de données en biomédecine

La biomédecine est un domaine fondé sur la richesse et la diversité des ressources, leurs complexité et leur aspect distribué. Les ressources peuvent représenter des images (irm, radiologies), des études de cas scientifique et des outils et des modules d'analyse. Plusieurs laboratoires et organisations travaillant dans le domaine de biomédecine ont construit leurs propres sources de données utilisant des techniques et des langages différents. Leur diversité rends très difficile leurs réutilisation car elle nécessite beaucoup d'efforts.
Lors de l'intégration de données provenant de sources biomédicales plusieurs conditions doivent être remplies afin de permettre une réutilisation plus facile de ces données. Certaines de ces exigences proviennent de l'évolution historique du partage de données au niveau

de l'entreprise, d'où l'idée des données intégrées, ou de fédération de données. Ceci nécessite au moins l'une des techniques décrites dans le paragraphe précédent en y ajoutant des aspects qui sont relatifs au domaine biomédical comme :

**Transparence de l'accès aux données biomédicales par l'utilisateur :** Du côté technique, il y a masquage des sources de données à l'utilisateur. L'utilisateur n'a pas besoin de savoir comment l'implémentation est réalisée (techniques d'accès aux ressources, systèmes d'interrogation).

**Exhaustivité :** Les données représentées par le système intégré doivent être complètes et cohérentes.

**Correction sémantique et de redondance :** Cette clause concerne les outils de fédération de données distribuées qui fédèrent les schémas de représentation de données de différentes plateformes. Le schéma fédéré doit être le résultat d'un raffinement (extraction des mots clés et des nomenclatures des termes techniques, élaboration des schémas globaux etc) des différents schémas des données intégrés.

**Actualité :**Pour certaines études scientifiques biomédicales, il est nécessaire de disposer des données les plus récentes. Un bon système d'intégration de données doit permettre non seulement l'agrégation des données et leur affichage mais aussi leur mise à jour.

**Gestion des doublons :**L'intégration de données dans un contexte biomédical nécessite de reconnaître et fusionner les informations provenant de diverses sources mais qui représentent des doublons. La sémantique doit nous permettre de déterminer s'il y a différence ou pas entre deux objets provenant de deux différentes sources de données.

### 2.1.2 Partage de données et de traitement pour la recherche

Ces dernières années, les technologies utilisées en imagerie médicale ont évolué et les études scientifiques en dépendent de plus en plus. De ce fait, les recherches dans le milieu de l'imagerie ne cessent d'évoluer à leur tour. Les chercheurs sont amenés à mettre en place des outils de traitement d'images ayant des critères spécifiques aux domaines étudiés, aux types d'études et parfois même aux pathologies ou maladies étudiées. Par exemple, des outils peuvent être utilisés pour corriger les défauts des acquisitions d'images ou détecter des régions d'intérêt de façon adaptée aux besoins de chaque pathologie. La diversité des méthodes d'acquisition d'images ainsi que la diversité des domaines et des spécialités en médecine a donné lieu à une diversification des méthodes d'analyse d'images et à une diversification des études scientifiques rendues possibles grâce aux images et aux traitements.

Face à cette évolution, il s'avère fortement souhaitable de pouvoir partager ces travaux, leurs résultats et les méthodes utilisées (problématique que l'ont peut dénoter sous le terme de "partage de ressources") afin de les mettre à la disposition des autres chercheurs ou laboratoires.

Aujourd'hui la majorité des laboratoires mènent leurs travaux dans le cadre d'études scientifiques multicentriques (ce sont des études faites sur des sites différents, voire dans certains cas avec des données issues d'autres études scientifiques. Le fait d'avoir des sources de données hétérogènes (sous différents formats, avec des codages intrinsèques aux besoins des unités de recherches qui les traitent...) représente en soi un obstacle à la réutilisation de ces données. Mais ce n'est pas le seul problème, en fait, les données résultantes issues de l'exé-

cution des outils et des modules de traitement d'images sont en pratique très difficilement ré-exploitables en terme de résultat du fait de l'absence de métadonnées associées, décrivant leurs provenances (elles appartiennent à quelle étude scientifique, comment ont-elles étaient obtenues etc).
De nombreuse initiatives ont mené des travaux dans ce domaine, comme le Biomedical Informatics Research Network (BIRN) ou le Cancer Biomedical Informatics Grid (caBIG) aux Etats Unis ou bien le projet européen @NeurIST.

#### 2.1.2.1 Biomedical Informatics Research Network (BIRN)

BIRN [1] est une initiative parrainée par le NIH américain (National Institutes of Health) qui propose une infrastructure de partage de données composée d'outils logiciels ainsi que de services. Le but principal du projet est de créer un milieu favorable pour la recherche et la collaboration multi-intitutionnelle dans le domaine biomédical. Il vise à encourager de nouvelles recherches et découvertes en facilitant les communications par le biais de partages de données, d'analyses et de comparaisons de données à travers plusieurs sites de recherche, documents scientifiques et laboratoires.

Les stratégies suivies dans BIRN sont particulièrement innovantes pour nous puisque la neuroimagerie représente le principal centre d'intérêt. Le choix de spécialisation dans l'étude de la neuroimagerie dans BIRN a été motivé par le nombre de travaux menés dans le cadre de cette discipline ainsi que le volume et la richesse du contenu des images médicales. Au fil du temps le projet a réalisé des avancées importantes dans le domaine de la neuroimagerie et a rassemblé un nombre important de partenaire qui ont contribué à la construction d'un entrepôt de ressources logicielles et de données largement exploité et compris par les communautés scientifiques exerçant dans les mêmes domaines de recherche [120].

##### Organisation

Actuellement, BIRN est une initiative sponsorisée par le National Institute of Health (NIH). Il est composé d'un ensemble de trois collaborateurs scientifiques menant leurs travaux de recherche autour l'imagerie médicale cérébrale, la génétique des troubles neurologiques chez l'homme et les modèles animaux associés. Pour permettre à ces groupes de collaborer, le centre de coordination de BIRN (BIRN-CC) a été créé. Il assure la mise en place des infrastructures nécessaires pour réaliser le partage de ressources à grande échelle. L'objectif primordial de BIRN est de collecter des ressources provenant de plusieurs centres de recherches afin que les scientifiques puissent disposer d'un nombre important d'études et d'échantillons.

BIRN est organisée en trois sous-projets :

- **Function BIRN Test Bed :** emploie la neuroimagerie fonctionnelle pour explorer les causes fondamentales de la schizophrénie et pour évaluer l'impact de nouveaux traitements
- **Brain Morphometry Test Bed :** se focalise sur la mutualisation des données acquises à travers des sites de neuroimagerie, pour étudier si les différences morphologiques

1. http ://www.nbirn.net

spécifiques permettent un diagnostic des dysfonctionnements spécifiques de la mémoire, tels que la dépression, la maladie d'Alzheimer, et les déficits cognitifs légers
- **Mouse BIRN Test Bed :** utilise l'imagerie multimodale et multi-échelle de modèles de désordres neurologiques chez la souris, pour mieux comprendre la schizophrénie, la maladie de Parkinson, la sclérose en plaques, et le cancer du cerveau

### Architecture

L'intégration de données dans le projet BIRN se fait à l'aide d'un médiateur Metropolis-II. Ce médiateur suit une approche GAV (Global-As-View) pour intégrer des données hétérogènes structurées ou non dans des bases de données. Ces données proviennent généralement de différents sites distribués distants géographiquement. Au sein de BIRN un ensemble d'outils a été mis en place permettant l'enregistrement des schémas de partage et des données, la définition et la construction des requêtes pour l'interrogation des données et la définition des ontologies permettant de structurer les données enregistrées. Le plus connu de ces outils est le brocker SRB (Storage Resource Broker) qui est un système de fichiers distribué, il gère les localisations des fichiers par une base de données appelée Meta-Catalog (MCAT) qui contient les informations de base des fichiers. Le MCAT permet de rassembler et sauvegarder des informations utiles concernant les données enregistrées comme la taille et la version et les dates de modification des fichiers.

Dans le top de cette architecture un grand nombre d'outils d'acquisition, d'analyse et de visualisation de données ont été développé et donnés gratuitement à la communauté de recherche BIRN.

- Human Imaging DataBase (HID) : est une base de données extensible implémentée en Oracle qui utilise une description hiérarchique des expériences et des protocoles liés à cette hiérarchie d'expérience. Elle contient aussi une plateforme extensible pour la définition et l'enregistrement des systèmes d'évaluation clinique. L'information enregistrée permet l'annotation de la qualité de l'évaluation des données. Le HID contient un IHM web et un moteur d'intégration de données pour fédérer les bases de données.
- Extensible Neuroimaging Archive Toolkit (XNAT) : est un logiciel de plateforme construit pour faciliter la tâche de gestion, de productivité et des données associées dans le domaine de la neroimagerie. Il utilisent un modèle de données XML duquel une base de données relationnelle est générée. Il implémente les workflows pour aider à obtenir une meilleure qualité, sécurité et intégrité de données depuis l'acquisition et l'enregistrement jusqu'à leur partage public et analyse.
- XML-Based Clinical Experiment Data Exchange Schema (XCEDE) : Il représente un ensemble de modules de portée générale permettant de représenter des métadonnées différentes et de faciliter les modes de transfert de données entre les différentes bases de données et logiciels. Ces modules ont été mis en place et gérés par XCEDE comme par exemple *Experiment hierarchy* qui représente plusieurs niveaux représentant les types de données expérimentales à des niveaux de granularité différents (projet, sous réserve, visite, étude, et l'acquisition épisode). *Event data* sert à représenter le temps d'annotation et le tag de données comme des stimuli durant l'irm fonctionnelle fMRI etc. *Binary data interface* fournit une interface abstraite de lecture de données destinée

à pouvoir lire n'importe quel format de données et de fournir une interface commune pour les données binaires. Finalement *Data provenance* permet la documentation des étapes d'exécution des workflows de traitement à temps réel.

L'analyse de données repose sur plusieurs outils : FBIRN Image Processing Stream (FIPS) c'est un package dédié à l'analyse, le traitement et la gestion de données fMRI provenant de plusieurs sites partenaires. Il assure des analyses identiques dans tous les sites partenaires. maintien la flexibilité des paramètres et des composant des workflow lors de l'analyse. Limiter l'accès et privatiser données sensibles et finalement assurer une veille automatique sur la qualité de données utilisées.

Le BIRN portal qui représente l'interface web d'architecture flexible via lequel l'utilisateur peut s'authentifier à l'aide d'un login et un mot de passe et manipuler une interface évolutive de tous les traitements et les expertises possible du système. Donne un espace de travail configuré spécialement pour les tests et les utilisateurs.

Le système de gestion de workflow (Workflow Management System) qui est lui même divisé en "portal interface" et un serveur d'exécution. Le système permet le téléchargement des fichiers images et d'autres données logiques ou des données de la grille de calcul. L'exécution du serveur a accès à plusieurs environnement d'exécution (Condor, jBPM, LONI pipeline).

Le système médiateur conçu permet l'intégration des données non seulement au niveau des schémas physiques comme le propose la majorité des travaux mais aussi au niveau des modèles conceptuels. Le mécanisme de fonctionnement de ce médiateur repose sur les adaptateurs (wrappers) qui intègrent les différents schémas des différentes sources de données représentés à l'aide d'une ontologie qui peut être reconnue par le schéma ontologique global du médiateur. Un adaptateur a pour fonction de maintenir l'ontologie locale de la source à savoir les termes et les relations utilisés et de sauvegarder les interprétations (liens utilisés pour faire des correspondances entre les définitions du schéma local et les définitions du schéma global du système médiateur).

### Plateforme d'intégration

La majorité des sources de données sont relationnelles, elles nécessitent un lien direct avec les relations utilisées dans d'autres base de données. Tous les schémas de relations possèdent des descripteurs qui sauvegardent des mots clés réservés pour la recherche par mots clé. Le mot clé associé à un attribut est un type de données sémantique qui pourra servir à le désigner dans une ontologie de domaine générique. Les utilisateurs de BIRN doivent impérativement fournir les informations nécessaires pour faciliter la compréhension et la classification de leurs données. Ces données sont principalement les métadonnées associées aux ressources que l'utilisateur souhaite partager. Par exemple, chaque outil de traitement à partager doit être stocké avec une description complète qui inclut les informations sur son utilisation, ses entrées ses sorties ainsi que leurs types. Pour ce faire, BIRN développe des outils qui ont pour but de faciliter l'intégration de ces outils à l'aide des descriptions fournies par les utilisateurs qui souhaitent partager leurs outils.

#### 2.1.2.2 Cancer Biomedical Informatics Grid (caBIG)

caBIG [2] (cancer Biomedical Informatics Grid) est un projet lancé en 2004, soutenu par le NCI américain (National Cancer Institute). L'institut national du cancer (partie du National Institute of Health) et maintenu par le Center of Biomedical Informatics and Information Technology (CBIIT). La principale motivation de caBIG est de stimuler le développement d'approches novatrices pour la prévention et le traitement du cancer en facilitant le partage, la découverte et l'intégration des informations, les analyses des patients dans toute la communauté de cancer. caBIG vise à fournir une infrastructure qui permet la création, la communication et le partage d'outils bioinformatiques, de données et de résultats de recherche, en utilisant des standards de partage de données et des modèles de partage de données. caBIG favorise ainsi l'échange rapide d'informations entre tous les secteurs de la recherche et des soin afin que les chercheurs et cliniciens puissent collaborer de telle façon à partager et intégrer dans leurs travaux leurs dernières découvertes. Le résultat attendu est d'accélérer le processus de la recherche biomédicale, conduisant à des résultats cliniques améliorés et plus efficaces.

**Organisation**

La nécessité d'une synthèse efficace de l'information provenant de sources multiples, principalement des communautés de recherche contre les cancers, imprègne toutes les étapes de la recherche biomédicale. Pour cela la recherche fondamentale et clinique est de plus en plus dépendante des technologies d'information de pointe pour la gestion, l'échange et l'analyse des diverses données biomédicales.
caBIG propose un certain nombre de services à ses utilisateurs comme par exemple :

- développer ou modifier des outils interopérables (ex : software, infrastructure) ;
- adapter les applications pour leur utilisation dans des configurations différentes de leurs configurations originelles ;
- former d'autres participants sur le modèle de données, le développement d'outils, le développement de logiciels et la documentation ;
- contribuer à la rédaction de rapports sur les stratégies à adopter concernant, par exemple, le respect de la confidentialité des patients et l'architecture de sécurité.

Ces activités sont organisées dans des espaces de travail supervisés par l'Institut National du Cancer. Les espaces de travail sont organisés en trois catégories :

1. Domain Workspaces (focalisés sur un domaine spécifique)
   - Clinical Trial Management Systems (CTMS) : développe un ensemble complet d'outils modulaires, interopérables et basés sur une norme, conçus pour satisfaire les divers besoins en matière de gestion des essais cliniques.
   - Integrative Cancer Research (ICR) : produit les outils et les interfaces modulaires et interopérables qui assurent l'intégration entre les applications d'informatique biomédicale et les données.
   - In Vivo Imaging (IMAG) : crée, optimise et valide des outils et des méthodes pour extraire la signification des données images.

2. https ://cabig.nci.nih.gov/

- Tissue Banks and Pathology Tools (TBPT) : développe un ensemble d'outils pour inventorier, suivre, extraire, et visualiser des biospecimens et des annotations relatives à des banques de prélèvements géographiquement dispersées.

2. Cross-Cutting Workspaces (focalisés sur la définition et la réalisation de l'interopérabilité)
   - Architecture (ARCH) : développe la plateforme qui intègre les différents types de données et qui permet l'interopérabilité des outils d'analyse. Ce groupe définit l'interopérabilité syntaxique, et développe caGrid qui est l'architecture sous-jacente du réseau et la plateforme qui fournit la base pour la connectivité, le déploiement d'outils, et les données à partager entre les participants de caBIG.
   - Vocabularies and Common Data Elements (VCDE) : évalue et intègre des systèmes et des normes pour des vocabulaires et des éléments d'information communs et le développement d'ontologies, aussi bien que les systèmes logiciels pour le traitement du contenu. Ils définissent également l'interopérabilité sémantique, fournissent des directives pour l'adoption des normes et l'harmonisation des éléments de données communs (Common Data Elements).
3. Strategic-Level Workspaces (focalisés sur les questions d'intégration entre les différents espaces de travail)
   - Data Sharing and Intellectual Capital (DSIC) : étudie les questions et développe des recommandations concernant le partage de données, la protection de la confidentialité des données, la propriété intellectuelle, et la sécurité ainsi que d'autres politiques, liées à caGrid, touchant à d'autres questions de normalisation et de propriété industrielle.
   - Documentation and Training (DT) : définit des directives, des processus, des modèles et des outils pour développer une documentation pertinente des logiciels et créer des supports de diffusion vers toute la communauté de caBIG.
   - Strategic Planning (SP) : assiste les seniors de caBIG pour développer une planification stratégique et une vision sur les activités de développement.

### Architecture

caBIG a été défini comme un ensemble de sources de données reliées entre elles pour former des plateformes de recherche sur le cancer. Les données proviennent de diverses disciplines de recherche distribuées sur différents sites. Un objectif important était de concevoir toutes les nouvelles applications et les infrastructures de façon modulaire, en soutenant la création d'outils complexes à l'aide d'une approche appelée "bloc lego".

Le déploiement de caBIG est fait selon une infrastructure fédérée sur une grille appelée caGrid. caGrid est conçue selon une architecture orientée services dans laquelle une ressource est exposée à son environnement en tant que service de la grille doté d'interfaces bien définies. Les services et les clients interagissent ensemble à travers les protocoles de communication et les services d'invocation de la grille. La grille de calcul caGrid propose deux types de services : "grid services" qui permet être enregistrés comme un nœud de la grille "Data services" et "Analytical Services". caGrid fournit une infrastructure standard pour les développeurs. A travers cette interface les développeurs peuvent publier leurs services en utilisant des métadonnées définies en UML (Unified Modeling Language) au sein du système de médiation et

de partage. Les utilisateurs peuvent, par la suite, accéder à cette grille de services et aux données, via des programmes, en utilisant la politique de gestion de contrôle d'accès local ainsi que des données fortement typées avec le format XML (eXtensible Markup Language).

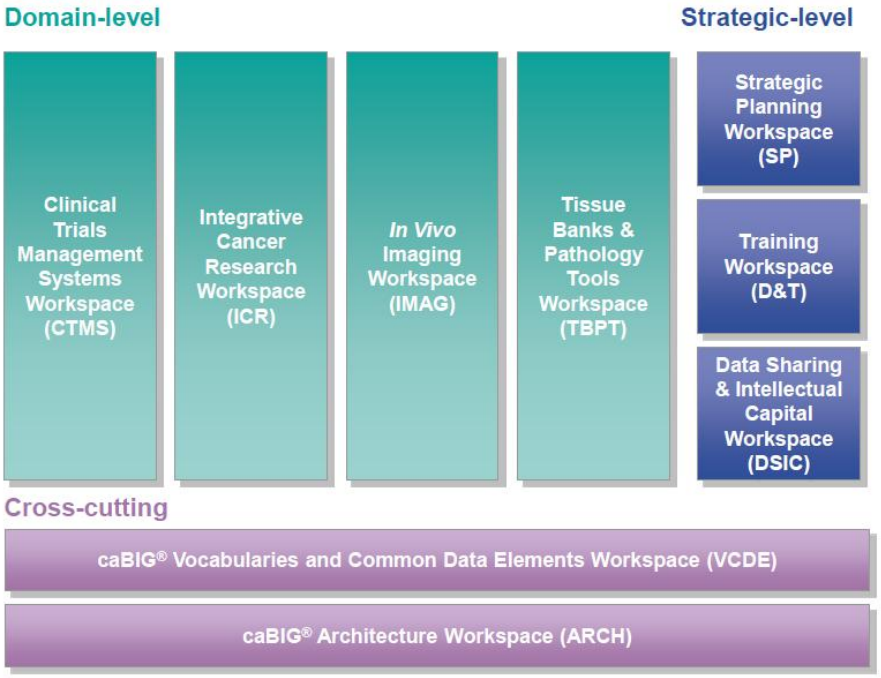


FIGURE 2.1 – Architecture logicielle de caBIG (approche opérationnelle) [8]

### Solution d'intégration

Les efforts fournis dans caBIG visent à créer et adopter des normes communes pour l'échange de données et l'interopérabilité des outils. Cet objectif a conduit à définir ses propres standards et normes ainsi qu'un ensemble d'outils et de plateformes. caBIG a sensibilisé les communautés de la recherche biomédicale de la nécessité et des avantages potentiels des architectures orientées services, des plateformes d'interopérabilité sémantique et des facilitées qu'ils apportent pour la réutilisation des outils de gestion des données de recherche.

Trois conditions ont été définies pour assurer l'interopérabilité et l'intégration de données, les modèles d'information communs, des normes de représentation de données et de terminologie et des API documentées. Pour assurer l'interopérabilité le NCI a développé caCORE (cancer Common Ontologic presentation environment). caCORE est un système de représentation ontologique fait spécialement pour représenter l'ensemble des données à traiter au niveau de caBIG. L'interrogation des sources de données passe par ce système de représentation de données. L'architecture de caCORE est composée d'un modèle EVS (Entreprise Vocabulary Services), d'un thésaurus basé sur la logique de description et d'un système de gestion d'ontologies. Un entrepôt de données est dédié spécifiquement pour les métadonnées associées au sources de données. Pour que les sevices caGRID puissent être gérés par caCORE ils doivent être encapsulés à l'aide des ontologies de représentation de données.

#### 2.1.2.3 Integrated Biomedical Informatics for the Management of Cerebral Aneurysms (@neurIST)

@neurIST [3] cherche à développer une infrastructure informatique pour la gestion de données hétérogènes associés au diagnostic et au traitement des anévrismes cérébraux. Les données utilisées dans le cadre de ce projet couvrent tout le niveau moléculaire et cellulaire grâce à des représentations de tissus, d'organes et de patient. Les données sont également hétérogènes et sont représentées sous différents formats. De nouvelles méthodes sont nécessaires pour gérer, intégrer et d'interroger ces données et les rendre accessible à l'utilisateur final.

**Organisation**

@neurIST est un projet européen de quatre ans, démarré le 1er janvier 2006, doté d'un budget de 17 millions d'euros. Réunissant un consortium majoritairement européen de trente partenaires académiques, dont l'UNIGE, les HUG et l'EPFL ainsi que des industriels. Il permet la fédération de sources de données hétérogènes. Pour cela la système étend les ressources médicales que les chercheurs créent en leur ajoutant des connaissance via des outils de gestion de données pour aider à la prise de décision clinique.

**Architecture**

Le projet a adopté une structure en couche qui est la suivante :

1. La couche application fournit des services qui répondent aux besoins de besoins des utilisateurs lors de l'utilisation des application mises à leurs dispositions.
   - @neuFuse permet l'analyse complexe de la biomécanique de l'anévrisme cérébral pour des géométries spécifiques à chaque patient obtenues à partir d'images médicales, @neuEndo étend cette analyse à l'aide d'autres traitement ;
   - @neuLink permet l'extraction de connaissances à travers l'intégration d'informations provenant de sources multiples et de différents types ;
   - @neuRisk Offre une aide à la prise de décision clinique ;
2. La couche middleware a fournit la couche application les outils et un services nécessaires pour accéder aux données et aux ressources informatiques distribuées géographiquement. Elle dissimule la complexité de l'accès à distance de l'utilisateur et est indépendante des aspects relatifs à des maladies spécifiques.
   - @neuInfo offre une gestion de données générique et du cadre d'intégration qui prend en charge le déploiement de services de données.
   - @neuCompute permet aux fournisseurs de services de virtualiser les applications informatiques de haute performance disponibles sur les clusters des grilles de calcul.
3. Dans la couche des ressources, la "biomédical infostructure" (Biois) offre un accès sécurisé aux bases de données cliniques réparties géographiquement. Les données sont sécurisées car elles sont anonymisées avant qu'elles soient transférées ou insérées dans des bases de données lors de leurs utilisations dans le cadre de l'exécution d'un outil de traitement.

3. http ://www.aneurist.org/aneurist1/index.php

4. La réalisation des Biois dans @neurIST utilise le modèle de référence clinique d'information (CRIM), qui est une description cliniquement pertinente de tous les renseignements sur les patients . Le modèle couvre les données liées au traitement des anévrismes ;
5. La couche sécurité du système vise à assurer un unique accès aux données des patients via le système. Le RelationshipManager fournit l'infrastructure de sécurité pour le contrôle d'accès distribué basé sur les spécifications webServices, y compris WS-Trust et SAML. Des mécanismes de protection des renseignements personnels tels que pseudonymisation et la dépersonnalisation sont utilisés pour séparer entre les informations personnellement identifiables des données médicales relatives à un patient.

### 2.1.3 Synthèse

Tous ces projets ont le même objectif ; c'est le partage de données en biomédecine. La problématique a été abordée différemment dans chacun de ces projets et les moyens mis en œuvre pour apporter des solutions logicielles adéquates sont forcément assez hétérogènes, et dépendent de chaque contexte d'étude et des données expérimentales à gérer. Les différentes techniques utilisées dans le cadre de ces projets sont soit basées sur les grilles de calcul ou sur les fichiers de description de données (ceci va être expliqué dans les sections qui suivent). Dans notre travail, le projet NeuroLOG [4] a les mêmes objectifs généraux mais utilise des techniques combinées basées sur des méthodes et des architectures existantes améliorées de telle façon à bénéficier des travaux récents issus notamment des domaines du web sémantique et des ontologies.

La section qui suit explique le projet NeuroLOG en détaillant les différents objectifs, les méthodes et solutions mises en œuvre pour le partage de ressources en neuroimagerie.

## 2.2 Contexte du travail : Le projet NeuroLOG

NeuroLOG est un projet financé par l'ANR (Agence Nationale pour la Recherche) qui vise à mettre en place une plateforme logicielle pour l'intégration des processus, des données et des connaissances en imagerie médicale. Le projet NeuroLOG essaye de combiner les différentes techniques de partage de données utilisées dans différents travaux antérieurs, pour leur ajouter une couche sémantique pour la médiation et le partage de données via les langages de description de données comme RDF [5] et OWL [6] (Ontology web Langage). Cette couche sémantique se fonde sur l'adoption d'une ontologie commune matérialisant le langage commun des spécialistes de la neuroimagerie.

Le projet NeuroLOG a une dimension pluridisciplinaire et rassemble un certain nombre de partenaire ayant chacun travaillé sur une thématique bien définie et proposant des solutions aux problématiques soulevées dans le cadre de ce projet. Les partenaires du projet NeuroLOG sont : le laboratoire I3S qui possède une solide expérience du domaine des grilles de calcul et qui s'intéresse particulièrement à la mise en œuvre d'applications médicales s'exécutant en local ou sur les grilles de calcul dans le cas du calcul intensif ; L'Institut des Neurosciences

4. http ://neurolog.i3s.unice.fr/neurolog
5. http ://www.w3.org/RDF/
6. http ://www.w3.org/TR/owl-features/

(GIN) de Grenoble qui est une équipe de neuroimagerie fonctionnelle et métabolique ; Visioscopie, une société spécialisé dans la conception de matériel et de logiciel pour la radiologie à Nice ; le LARIA équipe ingénierie des connaissances à Amiens ; Business Object, une SS2I du groupe SAP travaillant sur les systèmes de gestion de données et l'aspect fédération de données ; L'IRISA projet VisAGeS à Rennes, s'occupant de l'aspect médiation et partage de donnée dans le domaine biomédical en neuroImagerie ; L'INRIA Sophia Antipolispour, le projet asceclopios en imagerie médicale ; LRI équipe Inférence et Apprentissage à Orsay, et finalement l'IFR49 Laboratoire d'Imagerie Fonctionnelle à Paris.

Dans ce qui suit nous décrivons le projet NeuroLOG ainsi que son objectif puis nous présentons l'architecture logicielle mise en place pour le partage de ressources en neuroimagerie.

### 2.2.1 Intégration de ressources en neuroimagerie

Dans le contexte du projet NeuroLOG, l'intégration de ressources recouvre le partage de ressources expérimentales et logicielles acquises à partir des travaux récents développés par les collaborateurs du projet.
Les ressources expérimentales regroupent les sujets ou patients, en décrivant les différentes caractéristiques du sujet, sain ou pathologique, les études menées, en spécifiant les investigateurs et les expériences impliquées ainsi que les examens, en décrivant le contexte d'acquisition, comme le protocole d'acquisition et l'état du sujet durant l'acquisition et d'autres données susceptibles d'intéresser les chercheurs dans le domaine de la neuroimagerie.
Les ressources logicielles à partager dans le cadre du projet NeuroLOG sont les outils de traitement d'images qui sont développés par les partenaire de NeuroLOG comme par exemple les outils de recalage ou de segmenentation ainsi que les workflows et les modules d'analyse qui sont le résultat de l'enchainement (la composition) des outils de traitement d'images. Il servent à définir des nouveaux traitements en faisant interopérer plusieurs outils de traitement d'images simples ou composés comme par exemple les outils de détection de lésion dans le cerveau qui sont le résultat de plusieurs outils simples (débruitage, segmentation et recalage...).

La figure 2.2 montre les différents modules de la plateforme NeuroLOG. L'architecture NeuroLOG se compose de quatre modules principaux numérotés de I à IV au niveau de la figure 2.2. Le premier module représente l'interface utilisateur pour l'interrogation des données partagées et la réutilisation des outils de traitement d'images, le deuxième représente le module sémantique qui permet la représentation des données relationnelles avec des métadonnées sémantiques, le troisième module sert à représenter les données collectées à partir des sites partenaires de NeuroLOG sous la forme d'une vue unique et finalement le quatrième module montre le déploiement outils de traitement et d'analyse d'images.

### 2.2.2 Gestion de données et de métadonnées

Ce module permet la gestion des données des différents laboratoires partenaires du projet NeuroLOG. Il gère à la fois les données relationnelles et les données sémantiques. Il permet par exemple la recherche de données dans des bases de données distribuées hétérogènes par l'intermédiaire de composants appelés adaptateurs de données qui les adaptent d'une façon à

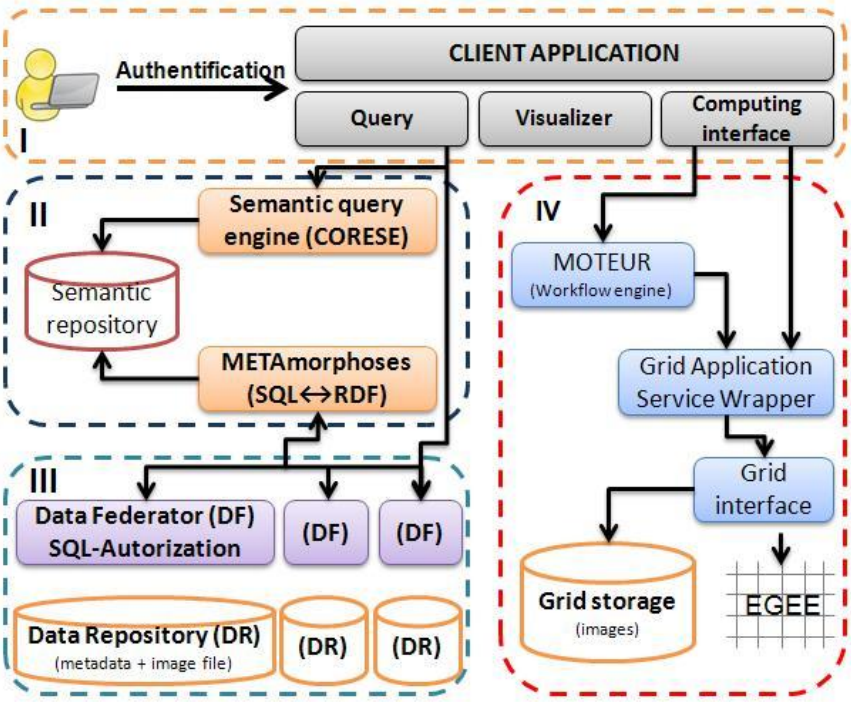


FIGURE 2.2 – Architecture logicielle du système NeuroLOG [183]

ce que l'échange (l'interrogation des données et la réponse du système de gestion de données) entre les utilisateurs finaux et les sources de données soit transparent et efficace.

La gestion des données se fait en deux phases différentes. Au niveau de la première phase, toutes les données sont collectées sous la forme de vues relationnelles. Dans la deuxième phase, ces données sont transformées en données sémantiques selon un schéma ontologique. Les données sémantiques sont ensuite enrichies par des règles sémantiques ce qui permet à l'utilisateur d'interroger ces données à l'aide de requêtes plus riches sémantiquement.

La figure 2.3 représente l'interface globale d'interrogation des données relationnelles au sein de la plateforme NeuroLOG. Ces données relationnelles proviennent des différentes bases de données des partenaires de NeuroLOG. A l'aide de l'outil Data Federator (fédérateur de données) on construit une vue relationnelle unifiée de toutes ces données relationnelles en vue de pouvoir les interroger d'une façon unique en utilisant des requêtes standards. Les données ainsi collectées par Data Federator sont rendues disponibles à l'ensemble des participants, de manière uniforme, grâce à des vues unifiées cachant l'hétérogénéité sous-jacente. Pour cela, au niveau de chaque site, une instance Data Federator client s'exécute pour publier les données destinées au partage. Le serveur Data Federator construit une vue globale sur les différentes sources de données. Le schéma relationnel de la vue globale est construit selon le schéma sémantique de description de données en l'occurrence le schéma de l'ontologie OntoNeuroLOG.

L'interface d'interrogation des données sémantiques (module sémantique) permet à l'utilisateur d'interroger le module de recherche sémantique CORESE en utilisant des requêtes

SPARQL plus riches sémantiquement que les requêtes relationnelles.

A l'aide de l'outil METAmorphoses les données relationnelles sont transformées en données sémantiques sous formes d'instances RDF. Finalement, le schéma de l'ontologie, les différentes instances RDF ainsi que les règles d'enrichissement sémantique sont chargés dans le moteur de recherche de données sémantiques CORESE. C'est ainsi que l'utilisateur pourra construire ses requêtes sémantiques et interroger les différentes bases de données. Il est vrai que le système médiateur ne s'intéresse qu'aux métadonnées associées aux images, métadonnées qui sont stockées dans les différentes bases de données relationnelles. Au delà l'utilisateur souhaite récupérer les données images qui sont référencées dans les bases de données ; la DML (Data Management Layer) [126] a été développée spécialement pour répondre à ce besoin. La DML s'occupe de la récupération et du transfert de données d'une façon sécurisée.

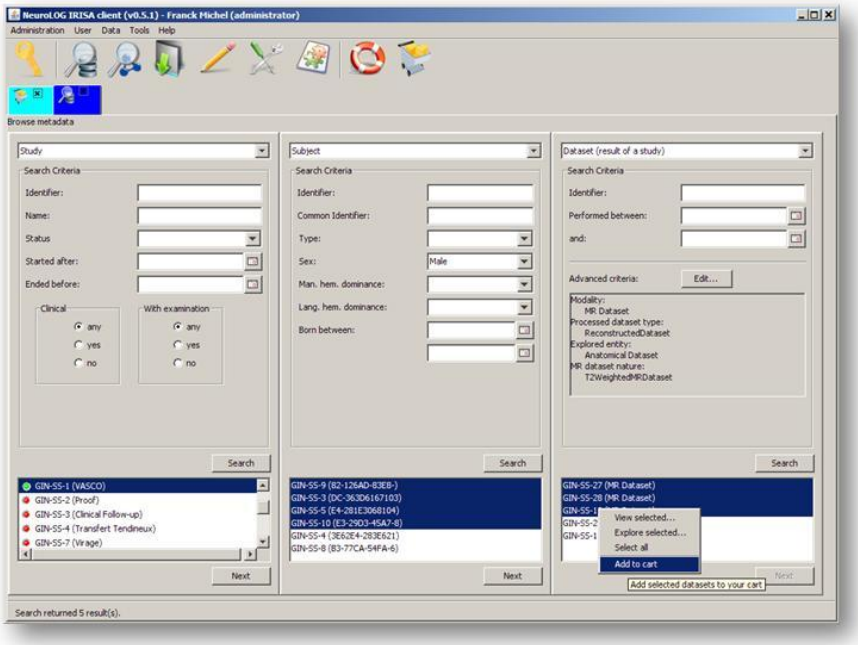


FIGURE 2.3 – Interface d'interrogation de l'application relationnelle

Les sections suivantes expliquent en détail les différents outils utilisés dans la plateforme NeuroLOG.

#### 2.2.2.1 Data Federator

Data Federator est un Fédérateur de données qui permet l'intégration de données dans les applications d'entreprise (EII). Il fournit une vue uniforme, cohérente et intégrée des sources de données distribuées et hétérogènes. Les sources de données peuvent être réparties sur un réseau, gérées par différents systèmes de gestion de données et administrées sous les différents domaines des organisations.

Il est le successeur de l'outil LeSelect développé à l'INRIA, dans le cadre du projet CA-

RAVEL [7], puis industrialisé par la société Medience. La société Business Objects [8] a pris la suite de son développement, elle offre des solutions de partage de données aux entreprises. Comme LeSelect, Data Federator permet de rassembler et gérer l'accès à des données hétérogènes et distribuées, situées dans des sites géographiquement distants. Son fonctionnement est basé sur les vues communes. Il construit donc une vue uniforme (il ne charge pas les données) sur l'ensemble des données publiées. La vue est construite en se basant sur le schéma global dans le cas de l'utilisation de NeuroLOG, elle se fonde sur l'utilisation de l'ontologie OntoNeuroLOG.

Dans l'architecture de NeuroLOG, Data Federator joue un rôle très important. Il fournit la vue unique de toutes les données des sites partenaires NeuroLOG. Il est doté d'une interface homme-machine conviviale, riche et robuste offrant un système efficace et facile à maintenir. Il rassemble les drivers des principaux SGBD du marché (Systèmes de Gestion de Bases de Données) ce qui lui permet de reconnaitre toute les formes de requêtes utilisées par ces SGBD. Il maîtrise et exploite toutes les techniques de gestion de données offertes par les SGBDs comme les mécanismes d'indexation et de gestion de clés primaires et étrangères, des mécanismes d'optimisation de requêtes, et la gestion des rôles pour les différents type d'utilisateurs etc.

#### 2.2.2.2 METAmorphoses

METAmorphoses [188] est un outil flexible et facile à utiliser pour la génération d'annotations RDF à partir d'une base de données relationnelle. Les annotations sont générées selon un template de (transformation) incluant un mapping entre le schéma de la base de données relationnelle et un schéma particulier d'ontologie ; ce template est représenté en XML. Le schéma de transformation XML de METAmorphoses peut être appliqué à n'importe quelle ontologie ce qui le rend flexible dans le sens où il est capable de transformer les données relationnelles en instance de n'importe quelle ontologie ou langage de description de données.

Comme le montre la figure 2.4, METAmorphoses utilise deux couches de transformation appelées "Mapping Layer" et "Template Layer" qui représentent en réalité deux fichiers XML contenant chacun un schéma XML de mapping de données. Le rôle de la "Mapping Layer" se résume dans les trois points suivants :

1. elle gère la correspondance entre le schéma de la base de données et l'ontologie,
2. elle préserve la cohérence des données RDF générées et
3. elle définit les balises ("tags") pour le "Template Layer".

La fonction de la "Template Layer" se résume en deux points :

1. elle utilise les balises ("tags") définies dans le "Mapping Layer", et finalement
2. elle produit les annotations RDF selon le schéma proposé par la couche "Mapping Layer".

7. http ://raweb.inria.fr/rapportsactivite/RA99/caravel/caravel_tf.html
8. http ://www.france.businessobjects.com

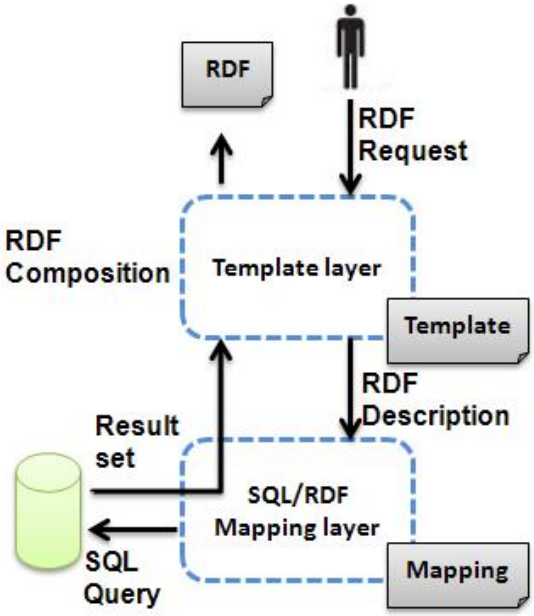


FIGURE 2.4 – Les deux niveaux de METAmorphoses [188]

#### 2.2.2.3 CORESE : Conceptual Resource Search Engine

C'est un moteur RDF qui permet le requêtage des données sémantiques basées sur les graphes conceptuels (GC). Il permet le traitement de RDF Schema et déclarations RDF dans le formalisme de GC décrit en détail dans le paragraphe 3.2.7 du chapitre 3.

Son architecture se compose de trois parties différentes, selon une architecture trois tiers. La première couche « Presentaion layer » comprend une API JAVA principalement une servlet JAVA et trois serveurs différents ; un serveur de gestion de données sémantiques RDF et ontologies (principalement OWL), un serveur de gestion de données XML et un serveur d'interrogation de données sémantiques basé sur SPARQL. La deuxième couche « Business Logic Layer » comporte tous les outils qui gèrent les données sémantiques. Finalement la dernière couche « Data Layer » contient tous types de données (classes et instances).

CORESE permet d'interroger les données RDF générées par METAmorphoses. Pour cela il charge le schéma de l'ontologie NeuroLOG, et les instances RDF. CORESE y rajoute des règles d'inférence spécifiques qui permettent de rajouter des connaissances sur les données. Il permet alors d'exécuter des requêtes sémantique plus riche que de simples requêtes SQL étant donné que les informations sémantiques sont plus riches, car elles exploitent une partie de la connaissance représentée dans l'ontologie, notamment concernant :

- les images médicales, les praticiens pourront alors faire des recherches d'images médicales qui portent non seulement sur les caractéristiques des images (comme les requêtes SQL) mais la connaissance sur les images représentée dans l'ontologie ;
- les outils logiciels de traitement d'images médicales, les praticiens auront la possibilité d'obtenir des renseignements sur le pipeline ou l'outil de traitement d'images le mieux

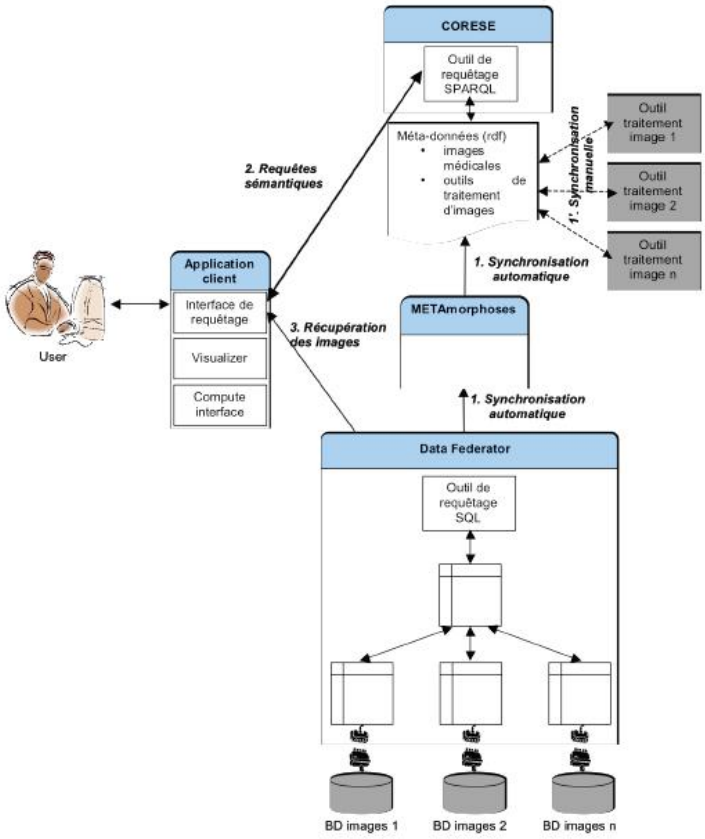


FIGURE 2.5 – Utilisation de CORESE dans l'architecture NeuroLOG [194]

adapté à leur besoin.
Dans le cadre de cette thèse c'est l'ontologie des outils de traitement d'images qui va être enrichie pour mieux modéliser les outils de la plateforme.

### 2.2.3 Partage des outils de traitement d'images

Cette couche est très spécifique à NeuroLOG. C'est à travers cette couche que les outils de traitement d'images qui sont à l'état exécutable deviennent des services déployables (sous forme de services web) et invocables à travers la plateforme NeuroLOG.

Elle repose sur l'outil jGASW (java Generic Application Service Wrapper) [13] développé à Sophia Antipolis par l'équipe I3S. En pratique ce service est accessible au moyen d'une IHM qui permet à l'utilisateur de décrire l'exécutable en termes d'entrées/sorties. Grâce à cette description jGASW génère une classe java et un fichier WSDL qui les compile et déploie l'outil sous forme de service web au sein de la plateforme NeuroLOG. Pour pouvoir exécuter ces outils, jGASW utilise la WSDL comme un point d'invocation de l'outil et réutilise la description XML donnée par l'utilisateur pour reproduire la ligne de commande qui va avec le traitement voulu. La figure suivante montre l'IHM de l'outil jGASW au niveau du client NeuroLOG.

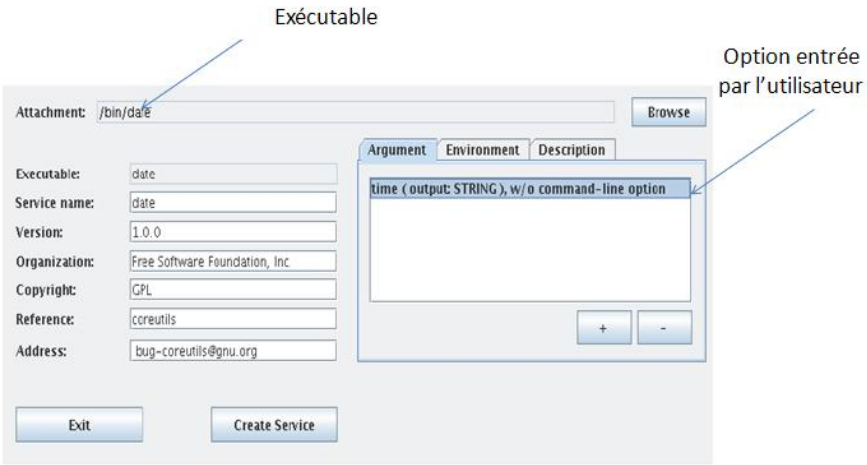


FIGURE 2.6 – Interface cliente de jGASW au niveau de la plateforme NeuroLOG

Après avoir encapsulé les outils de traitement d'images, ces derniers sont déployés sous un serveur Tomcat[9]. La liste de ces outils de traitement d'images transformés en services web est consultable à travers une interface au niveau de la plateforme NeuroLOG. L'utilisateur peut choisir un outil pour l'exécuter et récupérer les résultats de son exécution. Les trois figures suivantes montrent le processus de sélection d'un outil et son exécution.

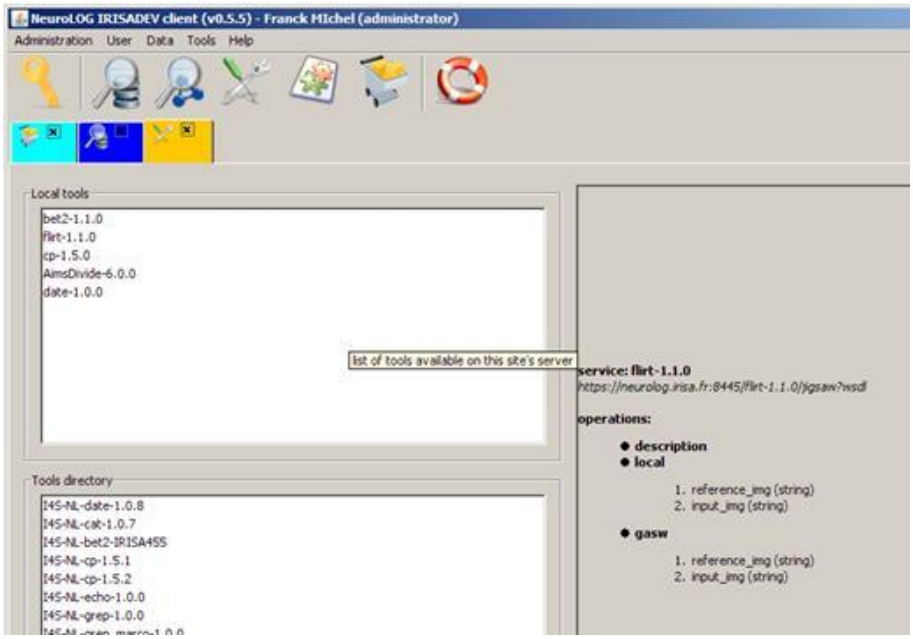


FIGURE 2.7 – Partage et sélection des outils au niveau du client NeuroLOG

A partir du catalogue de services web représenté dans la figure 2.7 l'utilisateur peut sélectionner un outil et l'exécuter.

9. http ://tomcat.apache.org/

La figure 2.8 montre l'interface de lancement de l'exécution d'un des outils déployés dans le catalogue des services/outils de traitement d'images. Elle montre aussi comment l'outil jGASW récupère le résultat d'exécution d'un service web. Il retourne trois URL qui représentent, respectivement trois fichiers. Deux fichiers standards sont respectivement la sortie standard au niveau du terminal d'exécution de l'outil de traitement d'images (le nom attribué automatiquement à ce fichier est std.out) et la sortie standard au cas où il y a eu erreur d'exécution (le nom attribué automatiquement à ce fichier est std.err). Tous les autres fichiers (images ou autres) représentent les résultats de l'exécution de l'outil de traitement d'images.

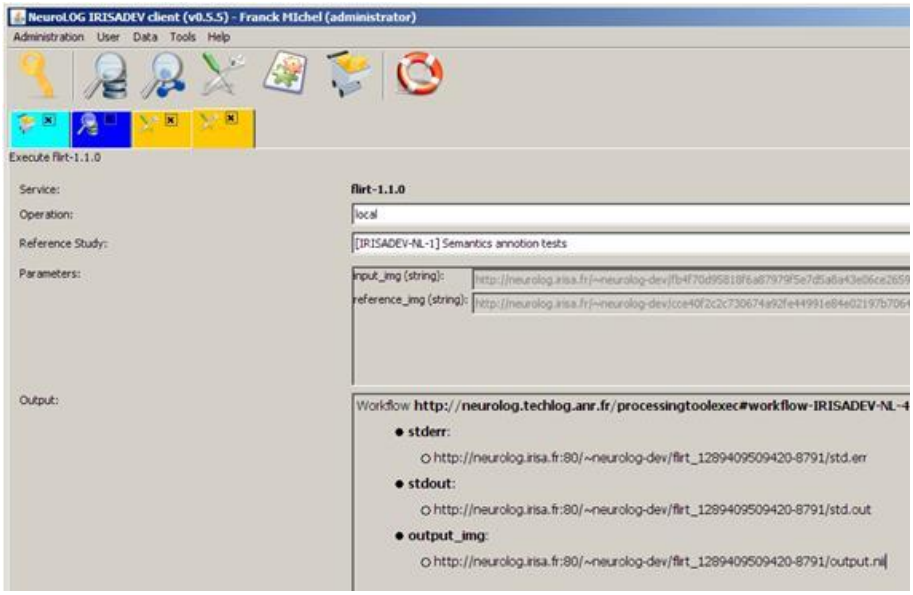


FIGURE 2.8 – Résultats retournés après l'exécution de l'outil de traitement d'images

### 2.2.4 Gestion des workflows et des pipelines

Ce paragraphe décrit l'aspect de gestion de workflow et des pipelines [10] au sein de la plateforme NeuroLOG. L'outil responsable de l'enchaînement des services (outils de traitement d'images sous forme d'exécutable et de shell, beanshell, services non standard encapsulés sous forme de service web) est MOTEUR [65]. MOTEUR est une application logicielle développée en langage JAVA dédiée spécifiquement aux procédures demandant des compétences computationnelles très évoluées. Elle permet de construire des applications complexes sous forme de flux de travail (voir figure 2.9) en faisant enchaîner des applications de différents types et origines permettant la réutilisation du code des applications scientifiques en local ou sur des grilles de calcul GRID [11] tout en facilitant la gestion et la manipulation des données. MOTEUR utilise l'approche basée sur les services. Il encapsule tout type de services pour offrir une flexibilité maximale en termes de types de services enchaînés. Il utilise plusieurs

10. Pipeline, c'est une technique utilisée pour faire le suivi de l'acheminement des données lors de la composition plusieurs services
11. http ://en.wikipedia.org/wiki/Grid_computing

techniques d'optimisation lors de l'encapsulation, l'acheminement des données d'un service à l'autre et lors de l'accès aux grilles de calcul. Il gère aussi les temps différés d'attente des applications lors de leur exécution. Les pipelines sont décrits de façon informelle et formelle (en utilisant le langage de gestion de flux de données Gwendia [133]). Les documents officiels Gwendia (voir exemple figure 2.10) peuvent être trouvés sur le wiki du projet NeuroLOG, dans la section des applications.

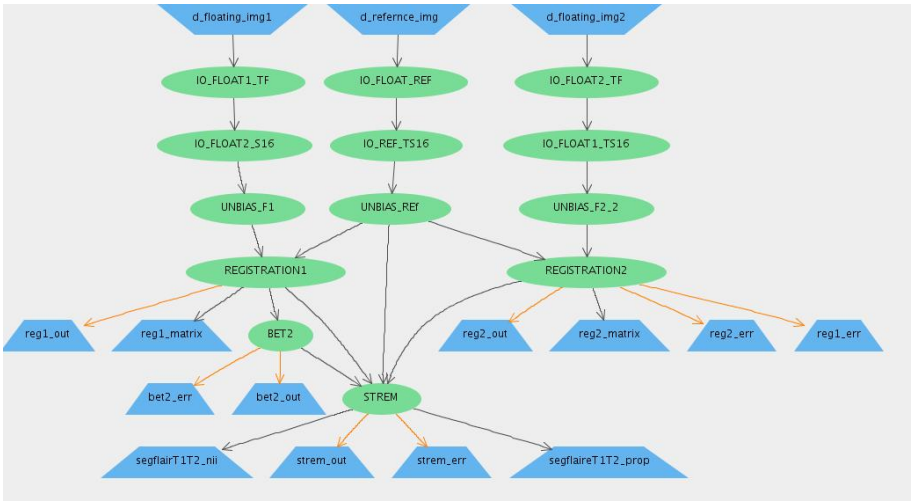


FIGURE 2.9 – Exemple de workflow construit à l'aide de l'interface MOTEUR

### 2.2.5 Ontologie OntoNeuroLOG

Dans cette section nous décrivons dans ses grandes lignes l'ontologie sur laquelle la plateforme NeuroLOG se base pour partager les ressources en neuroimagerie. Nous décrivons aussi la méthode générale utilisée pour la construction et la maintenance de cette ontologie.

#### 2.2.5.1 Objectif de la création de l'ontologie OntoNeuroLOG

L'objectif principal de l'ontologie OntoNeuroLOG est de définir un langage commun riche avec lequel on pourra décrire toutes les ressources du projet NeuroLOG, et donc remédier à l'hétérogénéité des ressources initiales. En l'absence d'un système médiateur, l'interrogation des ressources distribuées hétérogènes reste une tâche difficile. L'utilisateur doit nécessairement connaitre la nature des ressources des différents sites ainsi que le vocabulaire utilisé pour décrire ces données. Dans le projet NeuroLOG chacun des partenaires possède une base de données, des métadonnées et des schémas logiques à suivre pour interroger ses données. L'ontologie joue donc un rôle clé au sein du système médiateur [160] de NeuroLOG pour représenter d'une façon unifiée les ressources partagées dans le système NeuroLOG. Elle permet à l'utilisateur :

- de gérer et partager les données produites et exploitées par les utilisateurs du projet
- de réaliser de nouvelles études en s'appuyant sur les données distribuées hétérogènes ainsi partagées

```
<?xml version="1.0" encoding="UTF-8"?>
<workflow name="new workflow">

  <interface>
    <source name="d_floating_img1" type="string" />
    <source name="d_refernce_img" type="string" />
    <source name="d_floating_img2" type="string" />
    <sink name="nlmfloat1_err" type="string" />
    <sink name="nlmfloat1_out" type="string" />
    <sink name="nlmref_out" type="string" />
    <sink name="nlmref_err" type="string" />
    <sink name="reg1_out" type="string" />
    <sink name="reg1_err" type="string" />
    <sink name="reg2_out" type="string" />
    <sink name="reg2_err" type="string" />
    <sink name="segflairT1T2_nii" type="string" />
    <sink name="segflaireT1T2_prop" type="string" />
    <sink name="bet2_err" type="string" />
    <sink name="bet2_out" type="string" />
    <sink name="strem_err" type="string" />
    <sink name="strem_out" type="string" />
    <sink name="nlmfloat2_out" type="string" />
    <sink name="nlmfloat2_err" type="string" />
    <sink name="reg1_matrix" type="string" />
    <sink name="reg2_matrix" type="string" />
    <sink name="reg1_err_2" type="string" />
  </interface>

  <processors>
    <processor name="NLMEANSFloat1" >
      <in name="input" type="string" depth="0" />
```

FIGURE 2.10 – esquisse Script GWENDIA en XML

- de combiner des outils existants de traitement d'images pour définir de nouveaux pipelines de traitement des données
- d'évaluer ces pipelines hétérogènes sur les grands ensembles de données produites par les centres d'imagerie
- de formuler des requêtes plus riches plus précises grâce aux images annotées
- de faire des raisonnements dans le cadre de l'évaluation des requêtes
- d'interroger les différentes bases de données des différents sites des partenaires NeuroLOG en formulant une seule requête.

La conception de l'ontologie OntoNeuroLOG s'appuie sur les travaux du projet NeuroBase et notamment l'ontologie OntoNeuroBase [14][193]. A travers cette ontologie les partenaires NeuroLOG souhaitent concevoir un modèle sémantique commun offrant une vue unifiée de toutes les données et de tous les outils pour qu'ils soient partagés.

Ainsi, OntoNeuroLOG est conçue selon plusieurs principes généraux suivant :

- Les concepteurs ont adopté une approche multi-couches et réutilisé des ontologies existantes situées à différents niveaux d'abstraction comme les ontologies fondationelles de haut niveau ou des ontologies de domaine à niveau plus bas[62].
- Pour faciliter le mapping et l'alignement des ontologies et pour que l'ontologie OntoNeuroLOG soit réutilisable elle a été conçue en utilisant la méthodologie semi-informelle OntoSpec [84]. Cette méthodologie est sémantiquement riche ; elle permet la définition des relations temporellement indexées et des méta-propriétés spécifiques aux domaines

d'application définies par la méthodologie OntoClean [138].
- L'opérationnalisation de OntoNeuroLOG a été conçue en OWL-Lite, l'un des dialectes de OWL défini par le W3C qui se situe à un niveau d'expressivité compatible avec les possibilités du moteur d'interrogation sémantique CORESE utilisé dans la plateforme.

Comme le montre la figure 2.11 l'ontologie est construite en multi-couches et multi-composants pour faciliter son alignement avec d'autres ontologies et pour faciliter sa réutilisabilité. La structure globale de l'ontologie est représentée sous forme d'axes bien définis qui donnent un aspect modulaire [165].

De façon plus détaillée, OntoNeuroLOG s'appuie sur :

√ DOLCE (ontologie descriptive de l'ingénierie linguistique et cognitive) [121]. Comme le montre la figure 2.11 cette ontologie se situe au plus haut niveau, elle modélise les concepts et les relations les plus abstraits d'une façon indépendante des domaines sous-jacents. Ce type d'ontologie s'appelle ontologie fondamentale ; les concepts et les relations qui y sont définis partagent une base philosophique commune.

√ Des ontologies de base (core ontologies), qui fournissent le minimum de concepts et de relations génériques et non spécifique d'un domaine d'application. Minimum veut dire que les ontologies de base utilisées ne devraient comprendre que les catégories de relations et de classes les plus réutilisables et largement applicables. Ces types d'ontologies sont indispensables pour faciliter le partage d'entités entre les différents domaines. Comme le montre la figure 2.11 l'ontologie est composée de I & DA (Information and Discourse Acts) [59], qui est une ontologie de base initialement construite pour classer les documents en fonction de leur contenu. Elle a été utilisée notamment pour la modélisation des images médicales, considérées comme des types de documents. Participant Role est une ontologie de base utilisée pour décrire la manière ou la fonction selon laquelle l'image participe à une action, par exemple lors d'un traitement utilisant un ensemble d'outils de traitement d'images. I & DA, Participant Role, l'ontologie OntoKADS basée sur CommonCad [26] et l'ontologie des fonctions et artefacts [85] s'appuie sur la couche supérieure DOLCE.

√ Sur la base de ces deux couches, différentes ontologies de domaine dédiées à la conceptualisation du domaine de neuroimagerie ont été construites formant une ontologie d'application unique.

L'ontologie d'application est subdivisée en plusieurs parties indépendantes et dépendantes. Chaque partie modélise des connaissances liées à un aspect spécifique. Par exemple l'ontologie des datasets modélise les types des images par exemple, image de résonance magnétique, ou des classes exprimant le type d'information exploré par exemple anatomical dataset et functional dataset, et certain rôles des images dans le cadre de leur traitement. L'ontologie des inscriptions modélise les images en tant que par exemple des fichiers physiquement présent sur un disque.

L'ontologie COPS : Core Ontology for Programs and Software [98] modélise les logiciels et les programmes, qui s'exécutent indépendamment sur les machines.

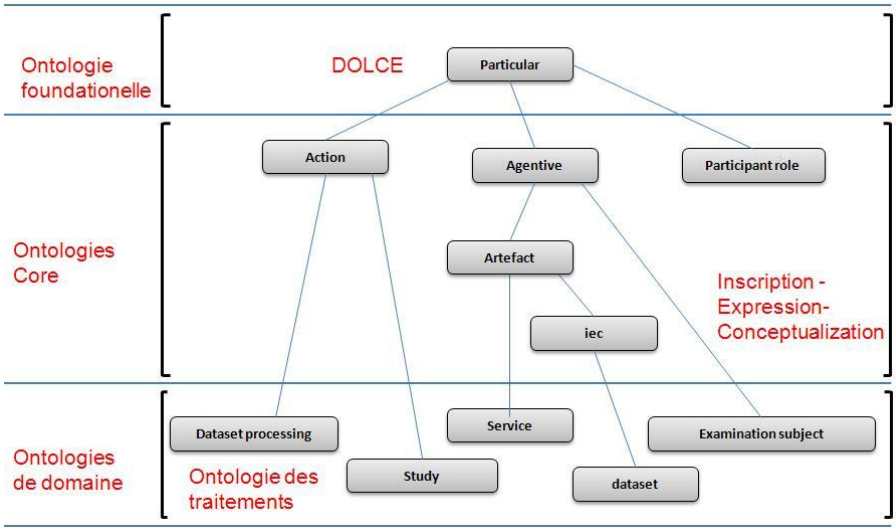


FIGURE 2.11 – Représentation en couches de l'ontologie OntoNeuroLOG

## 2.3 Analyse des besoins

Dans cette section, nous allons donner une analyse des problèmes existants avec les solutions actuelles des services et auxquels nous souhaitions remédier à travers des services web sémantiques. Nous donnons ensuite un aperçu rapide de la solution que nous proposons pour l'intégration des modèles sémantique de composition et de sélection de services. À la fin de ce chapitre, nous résumons les contributions les plus proches de celles qu'on souhaite intégrer dans le cadre de cette thèse.

### 2.3.1 Limitation des descripteurs XML

Comme on a pu le voir, à l'heure actuelle la description des services au niveau de la plateforme NeuroLOG est assez hétérogène. Lorsqu'il s'agit de services simples, ces descriptions sont des fichiers WSDL (basé sur XML) et étendus à l'aide d'un descripteur jGASW appelé jGASW descriptor, ou bien des beanshell [12] ; s'il s'agit d'un service composite, le workflow géré par l'outil MOTEUR est donc décrit par un script GWENDIA, lui aussi en langage XML.

Ces scripts basés principalement sur XML, se situent à un niveau strictement syntaxique, ils ne facilitent pas la tâche d'enchaînement de ces services dans le but d'obtenir des modules d'analyse plus complexes. Ces documents décrivent les types de données d'entrée et de sortie d'un outil, la structure des messages qu'il doit recevoir et envoyer, les protocoles de communication qu'il supporte, et l'URI qui le localise mais ne définissent aucune sémantique susceptible de vérifier la pertinence du traitement, ou aider l'utilisateur à concevoir cet outil ou de faire des raisonnements sur les données utilisées ou produites. Dans cette perspective, on constate que l'enchaînement et l'invocation des services au sein de la plateforme Neuro-

12. Un ensemble d'instructions en langage java, facilement intégrable et interprétable au niveau de l'interpréteur Java avec des objets basés sur les caractéristiques du langage de script

LOG ne sont pas réellement autonomes, car la façon dont on les utilise repose très largement sur l'interprétation humaine.

### 2.3.2 Description sémantique des fonctionnalités des services

Dans le milieu de la neuroimagerie la fonctionnalité du service désigne bien le type et la nature du traitement qu'il peut faire comme par exemple le recalage, la segmentation etc. Ce traitement peut avoir plusieurs caractéristiques qui peuvent être utiles lors de la création, l'exécution ou la composition avec d'autres services.

Les caractéristiques du traitement portent généralement sur le type et le rôle des entrées et des sorties du service comme le fait d'avoir nécessairement une image de référence lors d'un recalage ou un certain type de donnée lors d'une segmentation. Elles peuvent porter aussi sur la sémantique qu'on souhaite représenter lors de la positionnement de l'entrée en vue d'exécuter le service comme par exemple l'avoir en image flottante ou en image de référence dans la cas du recalage. Ou bien aussi peut renseigner sur le contexte de l'utilisation de l'outil ou de son exécution comme le fait que ça soit dans le cadre d'une étude ou autres ...). Il peut aussi aider parfois à la sélection du service lors de la composition d'un workflow.

Le but de décrire les fonctionnalités des services peut faciliter leurs sélection si par exemple on souhaite les mettre à la disposition de l'utilisateur sou forme de catalogue de service. Il peut aussi aider à vérifier l'annotation de l'outil par l'utilisateur en confrontant sa description avec la description de la fonction qui lui était attribuée.

### 2.3.3 Composition automatique et sélection de service

**Complexité des données biomédicales entraîne la complexité de la composition et la découverte de services :** Le but de la composition de services est d'arranger différents services sous forme de workflow pour répondre à certains besoins de l'utilisateur. Dans le milieu biomédical, la composition de service est presque devenu une nécessité car la spécificité des données biomédicales l'impose.
Les données biomédicales sont de nature très complexe par rapport aux données traditionnelles(exemple : un fichier irm peut avoir une entête ; une matrice avec ou des données additionnelles empaquetées dans un seul fichier). Nous avons donc besoin de les représenter et de les définir correctement sans aucune perte d'information et de savoir comment les utiliser lors de leurs traitements. Par exemple, certains attributs peuvent avoir une grande plage de valeurs possibles, donc nous avons besoin d'un ensemble de types de données approprié afin de couvrir tous les types de données qui pourraient exister. Un autre problème de taille est que le schéma des données dépend de la nature des recherches dans le domaine biomédical. Pour cela l'enchaînement des services et leur interopérabilité n'est pas toujours une tâche triviale, leur sélection non plus.

**Manque de sémantique au niveau des descripteurs de base des services :** La composition permet la création de nouveau workflows de traitement de données. Le résultat de la composition est appelée un service composite et il peut être réutilisé dans autre composition, conduisant ainsi à un appel récursif de services. Idéalement donc, un service doit être complètement autonome afin que les services soient interopérables et qu'ils puissent

être découvert automatiquement par des agents logiciels ou d'autres services. Les descripteurs XML généralement utilisés ont été principalement axés sur les détails opérationnels et syntaxiques concernant la mise en œuvre et l'exécution des services. Ils ne disposeraient pas de la sémantique qui faciliterait la tache de composition et de sélection. Ce manque de sémantique dans les spécifications limite la capacité des utilisateurs (machines ou humains) à garantir un enchaînement cohérent et à associer la consistance dans les résultats ou même une adéquation entre les données qu'on a envie de traiter et l'enchainement de services qu'on se propose d'utiliser. Pour remédier à ce problème il est donc nécessaire de décrire les services d'une façon riche et rigoureuse facilitant ainsi l'interopérabilité et facilitant la tâche de composition aux utilisateurs de ces services.

Dans le cadre du projet NeuroLOG, la composition des services est manuelle. L'utilisateur se connecte à l'interface graphique de l'outil MOTEUR et compose les services sans qu'il se soucie de la compatibilité des types de données ou de la cohérence de l'enchaînement des services construit. Ceci est dû au manque d'assistance sémantique. L'assistance sémantique est l'un des objectifs majeurs lors de la composition de services dans le projet NeuroLOG. Cet aspect devrait être inspiré des propositions antérieures faites dans le cadre de la composition des services dans le domaine biomédical. Cette option permet de vérifier le type de données des sorties des services connectées à des entrées d'autres services en vue de réaliser des workflow de données. Le deuxième objectif que l'on souhaite résoudre via l'annotation des types de données des entrées et des sorties est le contrôle sur les valeurs sélectionnées par l'utilisateur donc vérifier la compatibilité entre le type de données de l'image données par l'utilisateur et la plage de types de données que l'entrée du service peut accepter

### 2.3.4 Manque de connaissances associées aux fichiers produits

Pour faciliter l'exploitation des images issues des traitements, il faut avoir des connaissances supplémentaires précises sur des images produites par traitement. Par exemple, dans le cadre des études de cas scientifiques il faut attacher les images produites à leurs études scientifiques pour que les mesures et les résultats scientifiques soient exploitables. Il faut aussi rattacher les images à leurs sujets et aux traitements qu'ils ont générés pour assurer la traçabilité des données (sur quelle machine les traitements été réalisés, quel traitement les a produits, etc)

### 2.3.5 Assistance de l'utilisateur lors de la composition et l'exécution des services

Le but de vouloir assister l'utilisateur revient à la cause principale qui est la diversité des traitements dans la neuroimagerie et la richesse de la plateforme qui renferme différentes données images et outils provenant de plusieurs sites partenaires. Nous souhaitons avoir des traitements qui ont du sens, des workflows qui sont corrects en termes d'étapes et de procédures comme le fait d'avoir au moins un outil de segmentation lors de la construction d'un workflow de détection de lésions cérébrales. Nous souhaitons aussi proposer une interface conviviale et facile à utiliser par les utilisateurs non-spécialistes du domaine afin de leur permettre de mieux sélectionner, exécuter et enregistrer soigneusement les outils, les données

et les études qu'ils sont entrain de créer.

### 2.3.6 Surveillance (monitoring) d'exécution des services

L'importance des techniques de surveillance augmente au fur et à mesure où les environnements de fonctionnement des services deviennent de plus en plus dynamiques et les systèmes d'information basés sur des services web sont appelés à fonctionner de façon autonome ou semi-autonome. L'application des techniques de surveillance "monitoring" lors de la composition et de l'invocation des services est une étape essentielle que l'on souhaite avoir lors de l'utilisation des workflows d'outils de traitement d'images.

Des mécanismes de suivi d'exécution sont nécessaires pour fournir aux agents humains ou logiciels une information appropriée sur le déroulement de l'exécution et les résultats. Les informations fournies par les mécanismes de suivi peuvent être utilisées soit au cours de l'exécution pour apporter une réponse dynamique au cours d'une exécution donnée, ou après la fin de l'exécution pour des fins d'analyse et de contrôle. Par exemple, lors de l'exécution, le suivi peut être utilisé pour effectuer la mesure et l'évaluation de la qualité de services (QoS). En outre, étant donné que les services web sont souvent utilisés dans le cadre de modèles de processus complexes, la nécessité d'une analyse de diagnostic de simulation et d'une optimisation des processus de tels modèles s'imposent. Ce type de sénario serai beaucoup utilisé dans des domaines sémantique appliquée au Business Process Management et le data mining des processus. Le monitoring sert aussi à surveiller les temps de réponse et la disponibilité des ressources. Il assure le bon déroulement des appels de services en supervisant les protocoles, les types de données et les fonctionnalités. Il diminue le temps et les coûts lors de la conception, l'invocation, la composition, l'intégration et la maintenance des services complexes.

## 2.4 Synthèse des objectifs généraux de la thèse et méthodologie

L'un des objectifs consiste à faciliter la composition des services de traitement d'images dans le domaine de recherche biomédicale. Comme nous avons pu le constater la composition des services est une tâche complexe. Il existe de nombreux paramètres et défis à relever.

Avant d'aborder la composition proprement dite nous allons tout d'abord résoudre le problème du manque de description des services dans la plateforme NeuroLOG, ceci en les enrichissant à l'aide d'une description sémantique. Il faut définir et formaliser ces connaissances afin de faciliter l'utilisation de ces ressources pour les utilisateurs de la plateforme. Il faut pour cela fournir des informations explicites sur les paramètres des services, les contextes d'utilisation des services ainsi que les services eux mêmes afin de faciliter leur composition, leurs sélection et la ré-exploitation des données gérées grâce à des métadonnées créées par des modules sémantiques associés à ses services.

#### Améliorer la description des services

L'amélioration de la description de service consiste à mieux décrire les fonctions (la nature du traitement du service), les contraintes de type et de rôle sur les entrés/sorties pour assurer une meilleure composition de services.

Pour améliorer la description des services, il faut proposer une solution qui autorise l'obtention d'information plus précise. Une des solutions couramment utilisée est une solution se basant sur la sémantique. Dans le cadre de ce travail nous allons utiliser les ontologies pour faciliter l'opération d'interopérabilité et faire face à la fois à l'hétérogénéité des données et des services et pour enrichir les descripteurs de bases des services par des descripteurs plus adapté qui pourront aider à la composition et au raisonnement sur les données.

#### Améliorer la description des fonctionnalités des services

Pour décrire les fonctionnalités des services en noeuroimagerie l'ontologie des data processing (voir figure 2.11) a été conçu lors de la première phase de travaux du projet NeuroLOG. Cette ontologie décrit les différents traitements que les outils de traitement d'images en neuroimagerie peuvent faire. Dans ce travail de thèse, le but est d'enrichir cette ontologie de telle façon à mieux décrire les traitements et à pouvoir les réutiliser pour faire des vérifications entre les traitements et la description sémantique des services et des workflows. Pour valider la sureté du traitement de l'outil par rapport à sa description sémantique nous allons développer quelques algorithmes spécifiques.

#### Apporter un support à la création des workflows

Les outils de création de workflow dans le milieu de bioinformatique comme dans le milieu biomédical ne disposent pas généralement de sémantique, ils sont fait de telle façon il traitent seulement les problèmes de volumes et quantité de données, c'est le cas de l'outil MOTEUR dans NeuroLOG. A travers la sémantique on souhaite faciliter les tâches de conception aux utilisateurs non spécialistes. Nous souhaitons proposer un éditeur intelligent qui permettra d'assister l'utilisateur lors de la procédure de composition de service simple ou de workflows. Lors de la procédure de composition, des mécanismes de vérification de la composition qui vont être appelé afin de valider les compatibilité entre les types de données utilisés, les rôles de données, des traitements des services utilisés au sein d'un workflow et par rapport au traitement global que le workflow doit accomplir.

#### Contrôler les exécutions des services et des workflows

Lors de la création des workflows il s'agit de vérifier la validité de l'enchaînement des outils. Ceci en vérifiant les types et les rôles des entrées sorties des outils ainsi que de la fonctionnalité globale du workflow par rapport aux outils qui y figurent dedans.

Lors de l'exécution de service nous souhaitons ajouter les contrôles nécessaires sur les images sélectionnées par l'utilisateur en termes de rôle et de type de données et leurs validité par rapport à la description de services le rôle et le type que la variables auxquelles elles sont assignées par l'utilisateur.

#### Gérer des métadonnées

Il s'agit de réinsérer les données et les métadonnées respectivement dans la base de données relationnelle et la base de données sémantique en faisant attention à ce que les données soient

correctes et homogènes.

Profiter des systèmes d'inférence sémantique pour rajouter le maximum de sémantique qui pourra aider à l'analyse l'interprétation des résultats obtenus lors d'une étude scientifique, d'un traitement ou d'une expérience clinique ultérieures et rajouter des mécanismes de vérification qui aide au maintien de la cohérence et la solidité des résultat produits et insérés au sein de la plateforme.

Pour celà nous allons étudier les différents mécanismes développé à l'aides des langages d'enrichissement de métadonnées comme le langages SWRL(Semantic Web Rule Language) et qui nous seront facile à intégrer dans la plateforme NeuroLOG et bénéfique dans le sens ou ils nous seront le plus approprié pour cette fonctionnalité. Nous optons à priori au mécanisme des règles CORESE.

Nous allons aussi étudier les différents outils de gestions de données sémantiques qui sont susceptibles d'être intégré et utilisés au sein de la palteforme NeuroLOG.

Chapitre 3

# État de l'art

## Sommaire


**3.1 Vers une Architecture Orientée Services . . . . . 34**
3.1.1 Architecture SOA (Service Oriented Architecture) . . . . . 34
3.1.2 Services web . . . . . 35
3.1.3 Utilisation de l'architecture SOA et des services web en biomédecine . . 40
3.1.4 Synthèse . . . . . 41
**3.2 Interopérabilité et web sémantique . . . . . 43**
3.2.1 Types d'interopérabilité . . . . . 44
3.2.2 Web sémantique . . . . . 44
3.2.3 Ontologies . . . . . 50
3.2.4 Raisonneurs . . . . . 52
3.2.5 Ontologies et raisonnement en biomédecine . . . . . 53
3.2.6 Entrepôts de données sémantiques . . . . . 55
3.2.7 Moteurs de recherche et interrogation de données sémantiques . . . . . 57
3.2.8 Synthèse . . . . . 61
**3.3 Composition des services . . . . . 62**
3.3.1 Utilisation des registres de services web . . . . . 62
3.3.2 Registres de services en bioinformatique et biomédecine . . . . . 63
3.3.3 Synthèse . . . . . 65
**3.4 Outils sémantiques pour la composition des services . . . . . 66**
3.4.1 Service web sémantiques . . . . . 67
3.4.2 OWL-S : Ontology web Language for Services . . . . . 67
3.4.3 WSMO : Web Service Modeling Ontology . . . . . 74
3.4.4 METEOR-S . . . . . 80
3.4.5 Synthèse . . . . . 82
**3.5 Workflow . . . . . 84**
3.5.1 Composition des workflows . . . . . 86
3.5.2 Synthèse . . . . . 88
**3.6 Bilan . . . . . 88**


## Introduction

Ce chapitre présente l'état de l'art des travaux de cette thèse. D'abord il présente les spécificités et les caractéristiques de différentes techniques utilisées dans le partage des services

et des traitements puis il introduit l'évolution sémantique qui a accompagné ces outils lors de leur développement ainsi que les progrès réalisés dans le domaine de la composition de services web. Et enfin il explique les efforts déployés dans certain projets pour faciliter le partage des traitements dans le domaine biomédical en utilisant les techniques sémantiques.

## 3.1 Vers une Architecture Orientée Services

Les entreprises modernes ont besoin de répondre d'une façon rapide et efficace aux opportunités. Ceci résulte du contexte actuel marqué par une concurrence généralisée (au niveau des achats, des ventes et des prestations et des services) et de la globalité et de la standardisation des marchés (présentation selon un format unique comparable et facilement exploitable par les utilisateurs des services sur le web).

Pour pouvoir mettre en place des solutions tout en respectant les principes métier [1] ainsi que leur agilité, les entreprises sont censées simplifier les processus de développement métier tout en gardant la possibilité d'exposer diverses applications et tout en gardant l'aspect réparti et progiciel d'une manière standardisée. Une approche contemporaine pour répondre à ces questions cruciales est incarnée par le biais de l'architecture orientée services SOA (Service Oriented Architecture), qui met en avant des services qui peuvent être facilement assemblés pour former un ensemble de processus métier autonomes et faiblement couplés [2]. Ces dernières années, le concept SOA n'a cessé de gagner du terrain. SOA détient donc une part majoritaire du marché du développement de solutions logicielles informatiques et il s'avère qu'elle a un impact très fort dans l'organisation et le développement de nombreuses solutions logicielles dans plusieurs domaines notamment la biomédecine et la neuroimagrie [198], [107], [216].

L'architecture orientée service fournit le cadre de développement des applications distribuées [151], où les composants logiciels sont fournis sous forme de services modulaires et réutilisables. L'architecture SOA a des apports énormes du point de vue architectural par exemple la flexibilité du processus métier grâce aux services faiblement couplés dont elle dispose, la baisse des coûts d'intégration, la diminution de la complexité d'intégration et l'augmentation du potentiel de réutilisation et la flexibilité du système.

### 3.1.1 Architecture SOA (Service Oriented Architecture)

Le terme «SOA» a été utilisé la première fois en 1996 par Gartner pour dénoter le besoin d'une architecture multi-niveaux/multi-couches qui aidera les organisations à partager leurs applications et leurs données entre plusieurs stations [174]. Le concept est toutefois plus large que ça. Il peut faire interagir des services ou des composants logiciels ensemble en transformant des ressources existantes en services réutilisables ou en composant des nouveaux services pour répondre à des besoins.

1. Le concept métier fournit les services nécessaires au traitement des demandes des clients. Il contient les données métier et la logique associée. Tous les traitements métier de l'application sont centralisés dans une couche appelé la couche métier

2. C'est un concept qui consiste à décomposer les grandes plateformes logicielles en plusieurs composant logiciels indépendants qui communiquent entr'eux en échangeant des messages en suivant un protocole de communication bien défini et un format standard

*« L'architecture orientée services constitue un style d'architecture basée sur le principe de séparation de l'activité de la couche métier en une série de services. Ces services peuvent être assemblés et liés entre eux selon le principe de couplage faible pour exécuter l'application désirée. Ces services sont définis à un niveau supérieur de la traditionnelle approche composants »*

Gartner - Septembre 2005

L'idée est basée sur les concepts de la programmation orientée objet. Le principe est d'encapsuler du code dans du code orienté objet pour le réutiliser ensuite sous forme de service. Depuis 1996, plusieurs publications ont redéfini ce terme. Certaines définitions mettent l'accent sur les aspects techniques de l'architecture SOA, par exemple, [94], [140], [83], tandis que d'autres combinent des aspects techniques et commerciaux (par exemple [56] [148]).

L'architecture SOA a été appliquée dans des domaines différents et elle a réussi à gagner la confiance des gens en terme de facilité d'intégration et d'utilisation. Selon chaque domaine les services prennent des formes d'architectures et de vues différentes. Leurs implémentations, déploiements et exécutions sont à chaque fois adaptés à leurs contextes d'utilisation. Cela peut être relié au fait que la recherche SOA a été influencée par trois communautés de recherche différentes : science des affaires, sciences de l'information et informatique [12]. Si l'on prend d'abord les aspects commerciaux en considération, un service peut être considéré comme une fonctionnalité d'entreprise particulière ou une étape d'un processus métier. Les services sont publiés, trouvés, composés et exécutés selon un modèle composé de trois rôles différents.

### 3.1.2 Services web

Plusieurs définitions du terme "services web" ont été fournies par IBM et Microsoft. En effet, les services web ont reçu l'attention de plusieurs communautés de recherche, et ont été le terrain de nombreuse recherches dans l'industrie. Les services web sont des modules logiciels qui fournissent un service et qui peuvent être consultés et appelés via un réseau. Une utilisation à court terme des services web dans l'industrie consiste à fournir des interfaces web, pour ses systèmes de commande et de facturation, permettant à d'autres entreprises d'invoquer et utiliser ses services via Internet [34]. Pour ce faire, l'architecture des services web est conçue pour être la plus facile à intégrer et la plus facile à adapter aux systèmes hétérogènes disponibles sur internet.

#### 3.1.2.1 Architecture des services web

L'architecture des services web est basée sur les interactions entre trois rôles : fournisseur de services, registre de services et demandeur de services. Les interactions impliquent la publication, la recherche et la liaison entre les opérations utilisant le concept SOA [55]. Ensemble, ces rôles et ces opérations agissent sur les services web et leurs artefacts : le module logiciel d'un service web et sa description. Le concept de service reste le même que celui décrit dans le paragraphe précédent : un fournisseur de service définit une description de service pour le service web et il le publie au sein du réseau pour qu'il soit exploité par des

demandeurs de services ou des registres de services. Ces demandeurs de services utilisent donc des outils de recherche et des opérations de recherche pour récupérer la description du service localement (sur leurs stations) ou à partir du registre de services. Puis ils l'utilisent afin de pouvoir l'enchaîner avec d'autres opérations de services (il peut s'agir du même service) l'invoquer ou le faire interagir avec les autres services web. La description des rôles des fournisseurs de services et des demandeurs de service sont généralement des constructions logiques indépendantes des plateformes dans lesquelles les services web sont exécutés. Une même entité peut se présenter à la fois en tant que fournisseur ou en tant que demandeur. La figure 3.1 illustre l'architecture des services web ainsi qu'un ensemble de protocoles avec lesquels les différentes couches de cette architecture communiquent entr'elles.

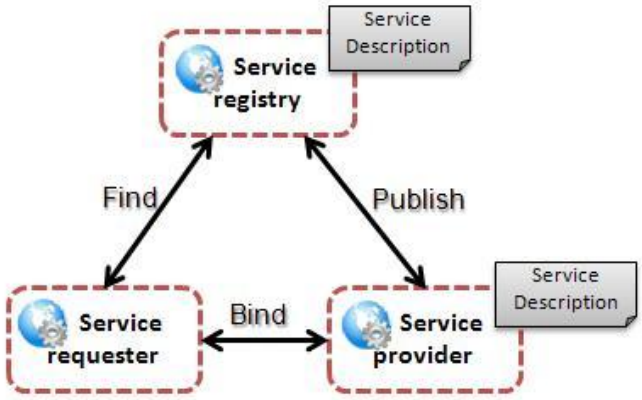


FIGURE 3.1 – L'architecture utilisée pour les services web [195]

Dans ce qui suit nous détaillons les différentes couches dont les services web sont constitués. Nous pouvons répartir l'architecture des services web sous forme de plusieurs couches. Tout d'abord, nous présentons une pile conceptuelle des différentes couches. Ensuite, nous allons discuter des critères pour choisir le protocole de réseau. Nous allons également examiner les couches basées sur XML [3] qui représentent les protocoles de messagerie distribuée.

**La structure en pile de la conception des services web :** pour effectuer les trois opérations de publication, recherche et composition d'une manière interopérable, il doit y avoir une pile de services web qui utilisent différentes normes à chaque niveau. La figure 3.2 montre une pile conceptuelle de services web. Les couches supérieures s'appuient sur les capacités fournies par les couches inférieures. Les éléments verticaux représentent des exigences qui doivent être abordées à chaque niveau de la pile. Le texte sur la gauche représente les technologies standards qui s'appliquent à chaque couche de la pile.

Le fondement de la pile conceptuelle des services web est basé sur le réseau. Les services web doivent être accessibles via le réseau et doivent être invocables par un demandeur de service. Les services web qui sont disponibles publiquement sur Internet utilisent des protocoles réseau couramment déployés. En raison de sa notoriété, HTTP [4] est le protocole réseau stan-

3. http ://www.w3schools.com/xml/
4. C'est le protocole de transfert sur internet le plus courant. C'est par l'intermédiaire de celui-ci que sont

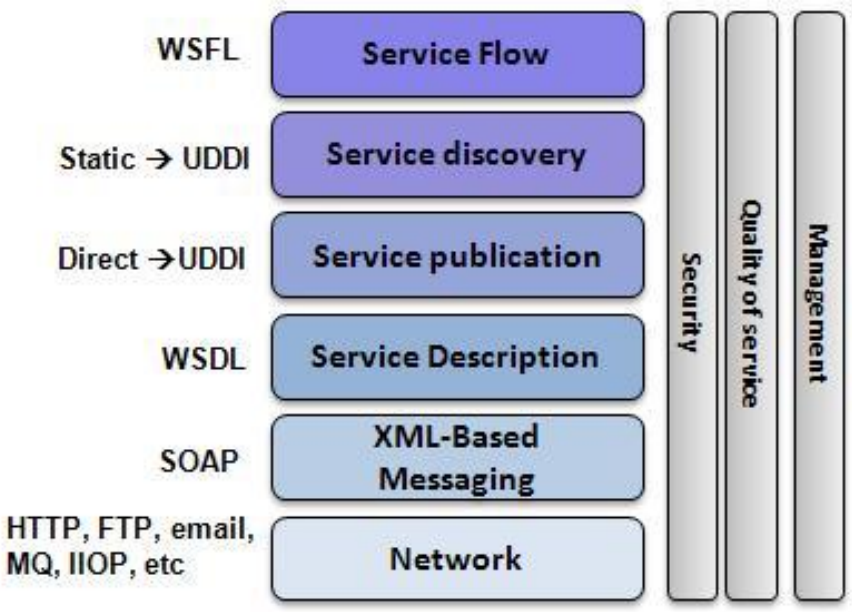


FIGURE 3.2 – Les couches conceptuelles de la modélisation des services web [95]

dard pour les services web et il est le plus disponible et le plus utilisé sur Internet. D'autres protocoles Internet peuvent être utilisés comme SMTP [5] et FTP [6] qui ne seront pas détaillés dans ce travail.

La couche qui suit la couche réseau est celle du messaging (échange de messages entre les partenaires soit le demandeur et le receveur) ; SOAP [7] est la principale couche du messaging, basée sur les flux XML. En premier lieu, cette couche utilise la sérialisation [8] comme moyen de présentation des données. Les objets de base (chaines de caractères ou entiers) ou les objets complexes (objets constitués de tableaux ou classes d'objets de base) sont décrits selon un schéma XML bien défini. SOAP est le protocole de messagerie XML choisi pour plusieurs raisons : c'est un mécanisme standardisé pour la communication de données centralisées qui permet d'envelopper des documents, des messages et des appels à des procédures et des méthodes ou services distants. Il s'avère que c'est un protocole simple à mettre en place (à intégrer), il est basé sur HTTP comme protocole de la couche inférieure de base et utilise une enveloppe XML comme schéma de présentation de données. Il est préférable que le protocole de base utilisé soit HTTP POST, car il définit un mécanisme standard pour intégrer des extensions aux entêtes de messages SOAP et un codage standard des opérations ou des fonctions. Le mode de messagerie ainsi que la technologie utilisée dans la messagerie SOAP aide à la mise en place et à l'implémentation des fonctionnalités de base des services web

transmises les pages web (au langage HTML) et que votre navigateur web vous présente de façon structurée.

5. http ://fr.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol
6. http ://fr.wikipedia.org/wiki/File_Transfer_Protocol
7. http ://fr.wikipedia.org/wiki/SOAP
8. La séralisation est le fait de présenter de façon séquentielle une information organisée initialement de façon structurée, comme la sérialisation par rapport à un schema XML

comme publier, chercher et composer des opérations. La figure suivante illustre le mode d'utilisation de la messagerie de SOAP :

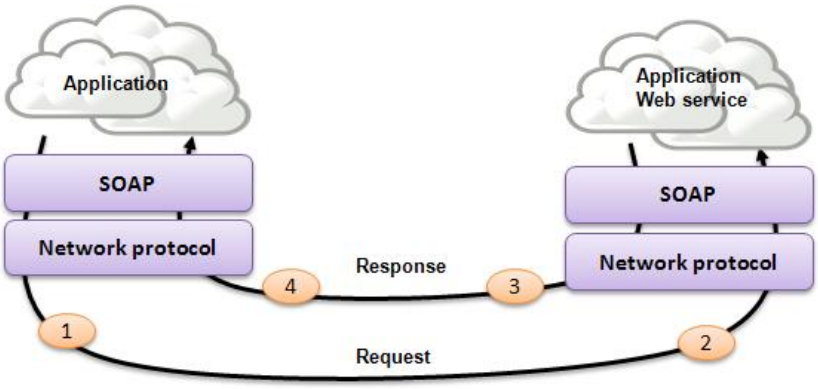


FIGURE 3.3 – Utilisation de la messagerie XML dans SOAP [95]

La couche WSDL [9](Web Service Description Langage) présentée dans la figure 3.2 permet la description des fonctionnalités d'un service web. Tout d'abord, WSDL est un standard basé sur XML pour la description des services web. Elle représente le strict minimum pour permettre l'interopérabilité entre les services. La WSDL définit l'interface et les mécanismes d'invocation et de sélection des services. Bien évidemment, la WSDL ne permet pas à elle seule tous les mécanismes de composition de services. Des descriptions additionnelles sont nécessaires pour spécifier les intentions métier de l'utilisateur ou bien la qualité de l'exécution ou de la composition des services. Le document WSDL peut être alors complété par d'autres documents de description de service pour décrire les aspects métier. Par exemple, dans un contexte qui déroule une procédure métier un outil a été développé, à savoir UDDI (voir paragraphe 3.3.1.1). La composition des services est par contre mené par le biais de la plateforme WSFL.

Si on rentre dans les détails de la WSDL, on se rend compte que c'est grâce à elle que le fournisseur de services communique, toutes les spécifications pour invoquer le service. La description du service est aussi la clé pour réaliser une architecture à couplage faible. L'intérêt principal du couplage faible est la réduction de quantité de code spécifique aux communautés de développement, d'exploitation et de proposition des services. Par exemple, ni le demandeur ni le fournisseur ne doivent se soucier des détails de l'implémentation du service ni de la plateforme sur laquelle il doit être exécuté, ni des éventuels autres services susceptibles d'être intégrés ou invoqués, ni surtout des bibliothèques nécessaires. Le principe est clair et simple : le fournisseur fournit une interface de service dans laquelle il définit et décrit ce qui est nécessaire pour l'invocation. Le demandeur n'a qu'à visualiser et utiliser cette interface telle qu'elle est. L'architecture de services web est fondée sur les WSDL. Elles permettent la description basique du service web. WSDL a été soumis au W3C pour standardisation et il représente un document XML qui décrit des services web comme un ensemble de terminaux fonctionnant à base de messagerie SOAP. Ces messages seront donc ou bien orientés document et utilisés en local, ou bien orientés appel de procédure à distance RPC (Remote Procedure Call). Les

9. http ://www.w3.org/TR/wsdl

opérations et les messages sont décrits de façon abstraite, puis liés à un protocole réseau concret et aux formats de messages voulus, pour donner naissance à un point d'invocation (end-point) qui caractérise le service et à partir duquel on peut l'invoquer. L'utilisation de WSDL dans l'architecture des services web divise classiquement la description de service de base en deux parties : une partie descriptive du service (ou interface du service) et une partie opérationnelle du service (ou invocation du service). Cela permet à chaque partie d'être définie séparément et indépendamment, et réutilisées par d'autres parties.

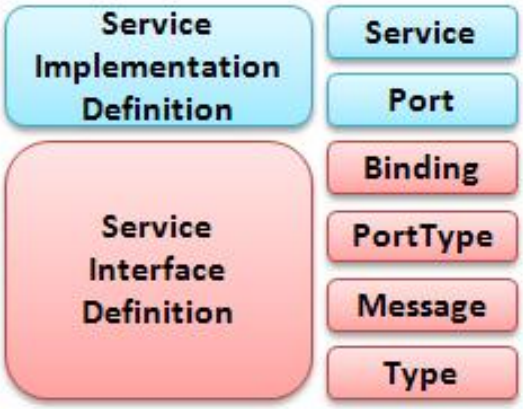


FIGURE 3.4 – Description basique d'un service web [95]

Une définition de l'interface de service est une définition abstraite du service et elle est aussi une définition réutilisable au travers de laquelle le service pourra être instancié et mis en œuvre. L'interface de service est comme une interface défini dans un langage de définition d'interface (IDL), un langage de programmation comme les interfaces Java ou plus génériquement elle définit le type d'un service web. Cela permet à différents types de services communs standards de l'industrie d'être définis et mis en œuvre par plusieurs exécutants de service.
Les interfaces de service peuvent être normalisées via les organisations industrielles telles que RosettaNet [10], ou HL7 [11] pour l'industrie de la santé. L'interface de service contient des éléments qui s'appellent les éléments de la WSDL. Une partie de ces éléments sont réutilisables par exemple WSDL:binding, WSDL:portType, WSDL:message et WSDL:type tel que décrit dans la figure 3.4. Dans l'élément WSDL:portType, il y a la définition des opérations du service web. Les opérations permettent de définir les messages XML qui peuvent apparaître comme des flux d'entrée et des flux de sortie. Il faut dire qu'une opération est similaire à une signature de méthode dans un langage de programmation orienté objet. L'élément WSDL:message spécifie quels types de données XML constituent un message. L'utilisation de types de données complexes dans le message est décrite dans l'élément WSDL:type. La WSDL:binding sert de liaison avec le protocole, le format des données, la sécurité et d'autres attributs d'autres interfaces de service particuliers. L'implémentation du service n'est pas un document de description comme les WSDL. Mais elle représente la réalisation concrète

10. http ://www.rosettanet.org/
11. http ://www.hl7.org/

du service et décrit comment ce dernier est mis en œuvre par un fournisseur de services donné. Un service web est modélisé comme un WSDL:service. Un élément de service contient une collection (généralement un seul) de WSDL:port qui représente un point d'invocation correspondant à une URL ou un emplacement d'adresse réseau. La définition de l'interface de service ainsi que la définition et la spécification des paramètres d'accès à l'implémentation de service constituent une WSDL complète. Cette WSDL contient l'information nécessaire et suffisante pour décrire aux demandeurs de ce service comment l'invoquer, et l'intégrer, le réutiliser et le faire interagir avec d'autres services web. Le demandeur de service peut exiger d'autres renseignements à propos d'autres critères de la part du fournisseur de services.

Pour résumer, la description complète de services web repose sur la description basique des services web qu'est la WSDL. La description complète de services web aborde des questions telles que: Quel est l'hôte de ce service et à quel type d'entreprise appartient-il ? Quels sont les produits ou les biens associés à ce service ? Dans quelles catégories est-il classé (par exemple, finance, banques, assurances etc.) ? Y a t-il d'autres aspects de services (tels que la Qualité de Service) qui auraient pu influencer le choix d'un demandeur ? Quels mots clés peuvent être fournis afin qu'il soit plus facile de trouver ce service ?
Le plus importante de ces questions est la dernière. On dispose actuellement d'un ensemble de logiciel qui s'exécutent de façon répartie sur des systèmes différents, fournissant des prestations suite à leur auto-exécution et grâce à leur auto-exposition. Malheureusement, la description du service web, bien qu'elle soit suffisante pour son invocation, reste très technique (liée à l'aspect technique de l'implémentation du service) et manque de l'aspect informatif. Pour remédier à ce problème, différentes contributions de complexité variable ont été proposées par différentes communautés de recherche en fonction des besoins et des contextes d'utilisation. Dans ce qui suit nous présentons quelques travaux récents et nous discutons ensuite l'impact de ces travaux dans le domaine de la biomédecine et plus précisément la neuroimagerie.

### 3.1.3 Utilisation de l'architecture SOA et des services web en biomédecine

Comme ceci a été expliqué, l'architecture SOA (Service-Oriented Architecture) et la technologie des services web constituent un moyen standard pour intégrer différentes applications développées avec différent langages et fonctionnant sur différentes plates-formes [150]. Ils représentent aussi un moyen puissant pour traiter les données et les ressources logicielles à travers les systèmes répartis et fédérés.
L'architecture SOA permet une approche plus simple de la conception de logiciels et leurs mise en œuvre car, dans cette architecture, les problèmes complexes peuvent être décomposés en problèmes plus petits et donc plus faciles à gérer. En outre, la nature des services (leur présentation sous forme de boites noires qui s'exécuteront sur des plateformes indépendantes et différentes) permet aux utilisateurs de services d'être déchargés des aspects d'implémentation potentiellement complexes du domaine d'application. Ceci concerne le domaine de la santé notamment lors de l'utilisation d'outils de traitement d'image, qui peuvent être localisés sur différents sites, applications et serveurs. L'association de l'architecture SOA et des systèmes basés sur des unités logicielles indépendantes et faiblement couplées en occurrence les services web, a de nombreux avantages, le plus important étant l'allègement des tâches d'implémentation lourdes lors de la réutilisation de ces logiciels qui souvent diffèrent en termes de méthode

de programmation, de langage utilisé ou même de mode d'utilisation.
Le deuxième avantage important est de renforcer la réutilisation des ressources informatiques existantes. Dans une architecture SOA, les capacités des logiciels disponibles peuvent être mises à profit par l'exposition de la fonctionnalité par l'intermédiaire des interfaces de service de la plate-forme. En outre, un service donné peut être réutilisé par plusieurs applications et d'autres services afin de répondre à des exigences opérationnelles différentes. En outre, la recherche en imagerie médicale conduit fréquemment l'utilisateur à essayer un ensemble d'outils, de façon à déterminer celui qui est le plus à même de répondre au problème posé. Ceci renforce le besoin de partager des outils ayant plusieurs fonctionnalités utilisables dans des domaines d'applications très variés.
Le troisième avantage de l'architecture SOA est la capacité à s'adapter aux exigences des autres plateformes qui sont éventuellement changeantes et flexibles. Dans cette architecture, les systèmes sur lesquels repose les interfaces de service peuvent être modifiés selon les besoins, et de nouvelles exigences peuvent être remplies rapidement en s'appuyant sur les services existants et en créant de nouveaux services. Enfin, un avantage important d'une architecture SOA réside dans le potentiel de réduction des coûts. Ceci touche aussi bien l'aspect médical qui aujourd'hui nécessite un patrimoine informatique et des efforts énormes en termes de compétences et de maitrise des nouvelles technologies à un coût qui doit rester minimal. Cette architecture apparait donc comme bénéfique surtout au niveau de la souplesse qu'elle apporte pour faire face à de nouvelles exigences, et de la simplification résultant de la modularisation des données et services en biomédecine, qui ne peuvent que réduire le temps et les coûts impliqués dans la conception, la mise en œuvre, et la gestion quotidienne des systèmes de santé.

### 3.1.4 Synthèse

Nous avons constaté que l'utilisation de l'architecture SOA et des services web simplifie la mise en œuvre et la maintenance des systèmes de santé, facilite la réutilisation des ressources existantes, et permet une adaptation rapide des composants existants pour répondre aux nouveaux besoins. En particulier, l'utilisation de cette approche dans le cadre de plusieurs projets comme BIRN [12], caBIG [13], caGRID [14], @NeurIST [15] nous a permis de mesurer le degré de simplification des tâches d'intégration dans différentes applications.

L'utilisation de l'architecture SOA et des services web permet aussi la réutilisation d'infrastructures géographiquement séparées et donc de bénéficier de ressources disponibles au niveau d'autres organisations de recherche et de résultats d'études scientifiques également issues d'autres centres de recherche. L'utilisation de ces architectures peut s'avérer pertinente vis à vis de différents aspects du partage de connaissances médicales, de données expérimentales ou cliniques et de ressources de traitement. Tout d'abord, le premier objectif est de représenter les ressources et les connaissances dont on dispose à propos de ces ressources dans des formats standardisés faciles à comprendre et à réutiliser. Pour cela, il est judicieux

12. http://www.birncommunity.org/
13. https://cabig.nci.nih.gov/
14. http://www.cagrid.org/display/cagridhome/Home
15. http://www.aneurist.org/aneurist1/index.php

de réduire les obstacles par l'adoption généralisée des solutions SOA disponibles. En outre il a fallu rendre disponibles des services standards pour faciliter la mise en œuvre et la maintenance des plateformes.
Généralement, les instituts de recherche et les organisations ou entreprises spécialisées dans le domaine de la santé, cherchent à faciliter l'accès aux données et aux ressources de traitement sur le web par le biais des protocoles Internet standard. L'implémentation des services au sein de l'architecture SOA facilite l'accès aux différentes plates-formes logicielles et à n'importe quelle ressource encapsulée sous forme d'un service web et permet par conséquent une intégration facile d'une telle ressource dans des applications distribuées.
Ces instituts, organisations et entreprises, ont aussi besoin d'exposer la fonctionnalité des services pour que l'utilisateur ait un support sur lequel il s'appuie pour sélectionner l'outil pertinent selon les fonctionnalités demandées, les traitements voulus et les résultats souhaités. Avec son architecture structurée en composantes faiblement couplées, la conception de logiciel et leurs mise en œuvre est plus simple, car repose sur la décomposition des problèmes complexes en problèmes plus circonscrits facilement gérables.
Les objectifs les plus cruciaux dans cette architecture sont d'améliorer la réutilisabilité des logiciels grâce à la réutilisation améliorée de ressources informatiques existantes, et l'amélioration de l'adaptabilité par rapport aux besoins des organisations qui ont tendance à évoluer constamment. L'intégration de ressources et leur réutilisation seront plus facile si on rend plus facile la prise en main de l'intégration des systèmes de santé en les rendant adaptables les uns aux autres. L'orchestration de services indépendants (dont les interfaces sont normalisées) proposée par l'architecture SOA permettra l'adoption généralisée de logiciels à faible couplage, une solution efficace pour l'évolution du système de soins de santé. En outre, le déploiement des services web peut être plus facile en raison du fait que les architectures des plateformes peuvent être décrites d'une manière directe en termes d'orchestration de services pour répondre à un besoin métier ou fonctionnel.
Dans la suite de cette section nous allons présenter l'impact de l'utilisation des services web dans le domaine biomédical. Nous allons expliquer aussi la relation entre les applications qui intègrent des services web dans le domaine biomédical, les problèmes rencontrés avec des outils de biomédecine, et le fonctionnement de plateformes basées sur les services web.
Un grand nombre d'outils sont disponibles pour la biomédecine, et certains de ces outils sont capable d'analyser des données biologiques comme les séquences génomiques, des images comme les images du cerveau ou même des analyses plus spécifiques comme la détection des anomalies génétiques ou la recherche de certain type d'anomalies dans des images du cerveau. Toutefois, en raison de l'absence de normes pour les ensembles de données et la définition des interfaces des outils utilisés, l'application de ces outils sur des données différentes n'est pas une tâche triviale. Au cours des dernières années, les services web sont devenus un moyen usuel de partage des données et des outils distribués sur le web et ils sont aujourd'hui utilisés par les chercheurs du monde entier. Dans ce contexte on étudiera et on discutera deux grand projets (voir les deux sections 3.3.2.2 et 3.3.2.3) $^{my}$Grid [16] et BioMOBY [17]. Nous discuterons aussi comment la recherche et le développement technologique ayant conduit à la diffusion des

16. http://www.mygrid.org.uk/
17. http://www.biomoby.org/

services web et rendu possible leurs composition, peut aider à résoudre quelques problèmes et aspects d'implémentation résiduels et faciliter leur intégration.
Les instituts de recherche ainsi que les nombreux autres organismes de recherche conçoivent sans cesse de nouveau outils afin de toujours mieux exploiter les données issues de la recherche en biomédecine. La plupart de ces outils sont spécialisés dans la fouille et l'analyse de données mais le progrès de la recherche exige de pouvoir compléter ces analyses (pour analyser, par exemple, les mêmes phénomènes à différentes échelles, ou étudier les similarités de phénomènes différents). Ce besoin nécessite non seulement de partager des outils mais aussi de pouvoir comparer et analyser les données provenant de sources multiples suite à l'utilisation de ces outils.

Dans le cadre du projet NeuroLOG, ces besoins ont apparus de façon progressive. Au début, il y a eu expression d'un besoin relatif au traitement de données et au partage des outils de traitement. Au delà, est apparu le besoin d'enchaîner les traitements sous forme de workflows qui puissent être facilement réutilisés ou même adaptés par les autres chercheurs. La technique de workflow n'est pas la plus facile à mettre en place. Elle n'est pas difficile en soi, mais nécessite d'assurer l'intégration. En fait, nous disposons rarement de toutes les informations souhaitables à propos des outils et des données exploitées par ces outils pour pouvoir assurer l'interfonctionnement entre les outils ce qui constitue le problème de d'interopérabilité.
Plus généralement, on pourra conclure que les organisations de soins de santé pourraient tirer quelques-uns des avantages de l'architecture SOA, même si les institutions fournisseurs des services conçoivent leurs offres indépendamment les uns des autres. Une condition nécessaire pour la réutilisation et l'interopérabilité des services est que la fonctionnalité et les interfaces de services puissent être représentées de façon normalisée.

## 3.2 Interopérabilité et web sémantique

La diversification des systèmes informatiques a entraîné une grande disparité de types de matériel et de logiciel. Ces systèmes ont été conçus différemment et utilisant souvent des protocoles de communication différents. Ces systèmes ne peuvent dialoguer que s'ils partagent une définition commune des informations. Cette définition commune est la condition nécessaire à l'interopérabilité entre les applications. Par définition, l'interopérabilité est la capacité des applications ou logiciels informatiques issus de plusieurs origines à coopérer entr'eux. Le principal défi de l'interopérabilité est de mobiliser les moyens les plus simples et le plus faciles à adopter par les utilisateurs de façon à éviter l'hétérogénéité des plateformes, qui est la source principale de l'absence d'interopérabilité. Elle est généralement assurée au niveau des couches physiques et des protocoles de base, mais moins assurée au niveau des couches supérieures des applications logicielles.
La complexité de l'interopérabilité diffère d'un niveau à un autre ; si nous prenons par exemple l'intégration au niveau de l'interface utilisateur qui servira pour une application donnée comme un point d'entrée sortie, nous remarquons que l'interopérabilité peut faire partie d'un processus de conception métier qui consiste à faire en sorte que les interfaces clientes soit tout simplement adaptables, intégrables et donc interopérables les unes avec les autres.

Au delà, l'interopérabilité entre l'interface l'utilisateur et la couche logique ou métier relève vraiment de la fonctionnalité du logiciel, qui sera capable de faire inter-opérer différents composants au sein de la même plateforme. Finalement, l'interopérabilité entre les couches logiques métier et les couches de données suppose l'utilisation de protocoles de communication. Les solutions techniques concourant à l'interopérabilité sont les services web (expliqués dans le paragraphe précédent), les systèmes de partage et d'intégration de données, et le web sémantique (expliqué dans le paragraphe suivant) qui permet de dépasser une simple interopérabilité syntaxique, pour offrir une véritable interopérabilité sémantique.

### 3.2.1 Types d'interopérabilité

L'interopérabilité évoque deux problèmes majeurs, des conflits d'ordre syntaxique et d'autres d'ordre sémantique.
L'interopérabilité syntaxique est l'aptitude de deux systèmes à coopérer sans qu'ils soient conçus et implémentés de la même façon et avec la même syntaxe [153]. Le besoin de l'interopérabilité syntaxique se manifeste lorsqu'on souhaite échanger des données non structurées entre deux systèmes qui peuvent appartenir ou non à un même domaine d'application.
Le problème de l'interopérabilité syntaxique peut être résolu avec l'utilisation de standards tels que XML, RDF et les services web. Avec l'apparition de ces standards, le problème d'interopérabilité syntaxique s'est allégé et a été largement résolu.

Contrairement au problème précédent, le problème de l'interopérabilité sémantique est un problème persistant, qui dépasse l'aspect syntaxique et se concentre plus sur l'aspect sémantique. Il apparaît lors de la communication entre des plateformes logicielles de domaines différents et vise à s'assurer que la signification exacte des informations échangées est bien compréhensible par les différentes applications ou plateformes communicantes. L'interopérabilité sémantique est définie dans [204] de la façon suivante: « La difficulté de l'intégration ressources qui ont été développées en utilisant des vocabulaires différents et des perspectives différentes . Dans [153], elle est définie comme « la capacité des systèmes à comprendre le sens et l'utilisation de la terminologie à partir de domaines différents ». L'interopérabilité sémantique est considérée comme la composante la plus difficile de l'interopérabilité. Selon [153] l'interopérabilité sémantique suppose la représentation en plus de la connaissance susceptible de fournir un moyen de résoudre les conflits sémantiques, créant ainsi un environnement d'informations sémantiquement compatibles.

### 3.2.2 Web sémantique

Dans le domaine de la santé, la précision sémantique est d'une grande importance. En effet, les résultats de recherches de données qui sont pauvres, inexactes ou incomplets peuvent avoir des conséquences graves, elles peuvent nous tromper lors de la prise de décision ou leur réutilisation. Des recherchent récentes ont montré qu'Internet est digne de confiance au troisième lieu après les médecins et les pharmaciens dans les questions de santé d'après la revue professionnelle en ligne des pratiques collaboratives [18], et elle est souvent utilisée comme une

18. http://www.collaboratif-info.fr/actualite/lordre-des-medecins-veut-mieux-integrer-le-web-dans-la-relation-medecins-patients

ressource d'information médicale fiable grâce au nombreux sites de santé qui se trouvent sur le Net. Mais cela reste particulièrement troublant étant donné que les dernières statistiques montrent que 72% des personnes qui utilisent internet pour rechercher par rapport à un sujet de santé ne vérifient pas leurs sources. La vérification de la fiabilité des sources d'informations et de la qualité des informations obtenues suite à une recherche est essentielle sachant que la plupart des gens commencent leur recherche de santé en ligne avec une recherche Google avant d'envisager aller chercher l'information sur des sites spécifique santé comme Pubmed, MayoClinic [19] ou l'Institut national de la santé NIH (National Institute of Health). Il est clair qu'il y a un besoin de pouvoir consulter l'information de santé en ligne et d'avoir des systèmes fiables pour la récupération précise et approfondie de l'information.

Le modèle traditionnel des moteurs de recherche Web, se montrent assez satisfaisant pour les utilisateurs ordinaires. Les moyens de recherche sur internet et dans le domaine de la santé nécessitent généralement des mots clé sensibles sinon ils rendent des résultants redondants et fragmentés. En outre les thèmes de santé sont multiples et utilisent un vocabulaire médical très spécialisé. Vous désirez trouver les symptômes de la maladie, les options de son traitement, les expériences des anciens patients, ou des interactions médicamenteuses ? alors assurez vous que vous avez bien saisi le terme médical correct pour votre requête ? mais après un retour peu spécifique de la part du moteur de recherche, quel résultat de recherche devriez-vous choisir ? Et sur quel critère vous devriez choisir ? Comme on le voit la recherche d'informations médicale en ligne est très difficile à gérer. Il y a beaucoup de choix, peu de possibilités de raffinement des résultats, et une forte probabilité d'être déçu. La réponse réside dans une vision de l'avenir de l'Internet appelé le Web sémantique.
Si nous prenons par exemple un fichier image, il contient plusieurs coupes, une échelle, un type etc. En fait, il porte plusieurs informations qui sont très utiles et importantes pour l'analyse, les interprétations ainsi que le choix d'utilisation dans certain cas. Ces informations ne sont pas toutes accessibles au sein de ce fichier. Le plus souvent document lui-même ne fournit pas l'information suffisante pour se décrire et se faire reconnaître sur le web. Nous avons donc besoin de décrire plus explicitement les données, de les rattacher à d'autres données selon leurs interactions. Il faut en outre que les machines soient plus intelligentes et comprennent la sémantique des données qu'elles manipulent. Par conséquent, la vision de l'avenir du web vise à ajouter de la « connaissance » aux données et aux services sous le forme d'une description « sémantique » qui les rend compréhensibles c'est à dire qui permet aux utilisateurs de mieux formuler des requêtes (demandes) et aux machines de mieux répondre à ces requêtes; on appelle ceci le « web sémantique ».

Le web sémantique tel qu'il a été décrit par Tim Berners Lee [18] [176] est une vision qui décrit comment doit être et sur quoi repose la génération suivante du web. Il le décrit sous forme d'une vue d'un ensemble de ressources (outils, données et informations) qui sont en réalité déconnectées entre elles et fonctionnent indépendamment les unes des autres mais interconnectées entre elles grâce à cette vue. Cette vision favorise la collaboration entre les humains (chercheurs, vendeurs, clients) et machines (hardware et software) pour les faire inter-opérer et fonctionner plus efficacement. Ceci se caractérise par une aptitude à la compréhension et aux raisonnements pour mieux gérer une quantité importante de ressources et

19. http://www.mayoclinic.com/

d'informations. L'objectif principal du web sémantique est alors de rendre explicite le contenu des ressources sur le web. Les machines et les logiciels pourront de ce fait mieux interagir ensemble. En effet, la description des ressources représente un moyen efficace pour ajouter des informations pertinentes à ces ressources comme par exemple le domaine de travail, l'utilité, le mode de fonctionnement ou même parfois les interactions avec les autres objets du web.

Le contexte d'utilisation du web sémantique vise à plus automatiser l'accès aux données et l'exécution des services présentés à travers le web. Il s'agit d'une sorte d'autoconsommation puisque les premiers consommateurs du contenu du web sont des agents logiciels, plutôt que les utilisateurs humains. Pour que les agents logiciels arrivent à se comprendre et à appréhender le contenu des ressources sur le web, ils doivent disposer de données représentées formellement.

Tel qu'il a été proposé par a ces débuts, le web sémantique est représenté sous forme de plusieurs couches hiérarchiques figure 3.5. Les deux premières couches sont les plus basiques, elles assurent l'interopérabilité syntaxique. Une URI [20] est l'acronyme de « Uniform Resource Identifier », et dénote une chaîne de caractères standardisée universalisée utilisée pour identifier ou nommer une ressource. L'URL [21] est l'acronyme de «Uniform Resource Locator». Elle est aussi une URI mais plus reconnue et plus populaire sur le web et désigne le plus souvent une page web. Elle a un lien direct avec les instances (ressources physiques) des documents sur le web. L'IRI [22] est l'acronyme de « Internationalized Resource Identifier » qui est une forme d'URI qui n'utilise que des caractères ASCII, elle est plus utile dans un contexte international. Unicode est un système universel d'encodage standard qui sert à représenter des données textuelles. La table contient à peu près un million de caractères qui spécifient n'importe quel caractère dans n'importe quelle langue sans une aucune séquence d'échappement ou code de contrôle. Avant Unicode, il y avait plusieurs systèmes de codage différents, qui rendent la communication et l'intégration à travers les frontières particulièrement complexes.

**XML :** est l'acronyme de « Extensible Markup Language ». Il fournit une syntaxe pour décrire la structure des documents, et parfois pour générer ou manipuler des instances de documents. XML représente un moyen standard pour composer l'information afin qu'elle puisse être partagée plus facilement à travers le web. Il peut être aussi un moyen pour structurer les données et les documents, il offre toujours la possibilité d'extension.

**XML Query :** (XQuery alias) est un langage normalisé pour les documents combinant des bases de données et des pages web. Il est le plus utilisé sur le web grâce à sa facilité d'utilisation et sa puissance et rapidité d'exécution. XQuery permet de remplacer des instructions de traitement par des scripts très facile à manipuler et à comprendre en donnant la possibilité de ré-exécuter un programme déjà implémenté dans langage de programmation comme Java ou C++.

Il suffit donc de se référer à ces éléments de base (Unicode, URI et XML) pour standardiser les informations échangées et mettre en place un système universel de codage de l'information et disposer d'un moyen standard pour identifier et localiser les ressources. Ainsi XML fournit un socle unique qui nous permet le partage et l'intégration cohérente de nombreuses ressources hétérogènes. Toutes ces techniques aideront à intégrer les contenus et services à travers le web.

20. http://fr.wikipedia.org/wiki/Uniform_Resource_Identifier
21. http://fr.wikipedia.org/wiki/Uniform_Resource_Locator
22. http://fr.wikipedia.org/wiki/Internationalized_Resource_Identifier

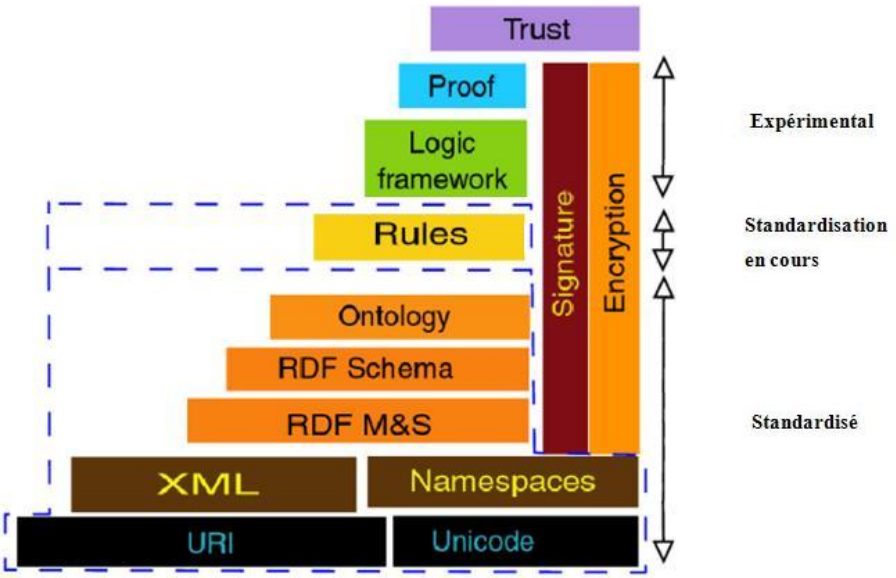


FIGURE 3.5 – Les couches du web sémantique [87]

Dans ce qui suit, nous allons commencer à voir dans la pratique comment nous utilisons Unicode, URI et XML via un langage plus évolué, le Resource Description Framework (RDF) [115].

**RDF :** RDF est un standard du W3C (World Wide Web Consortium). Il permet de gérer les métadonnées et il est principalement conçu pour être utilisé sur le web. Il fournit un langage pour décrire les ressources sur le web. Son infrastructure permet l'encodage, l'échange et la réutilisation de métadonnées structurées. RDF est une application de XML qui impose des contraintes structurelles nécessaires pour fournir des méthodes claires pour exprimer la sémantique. RDF fournit en outre un vocabulaire lisible par l'humain et exploitable par la machine destiné à encourager la réutilisation et l'extension de la sémantique des métadonnées au sein des communautés.

La conception de RDF est destinée à répondre aux objectifs suivants [31] :

- Avoir un modèle de données simple
- Avoir une sémantique formelle
- Utiliser une URI extensible basée sur un vocabulaire bien défini
- Utiliser une syntaxe basée sur XML
- S'appuyer sur des schémas XML pour décrire les données
- Permettre à n'importe quel utilisateur du web de décrire une ressource ; avec RDF une ressource est présentée par un ou plusieurs triplets RDF. Un triplet RDF est composé des trois objets suivant: un sujet, un prédicat et un objet.

Il est construit sur les concepts fondamentaux suivants [23] :

- Un sujet est tout objet qui a la capacité d'avoir une URI, ce qui inclut toutes les pages web, ainsi que les éléments individuels d'un document XML. Par exemple http://www.monsite.com/unexemple.html
- Un prédicat est une ressource qui a un nom et peut être utilisée comme une propriété

d'objet, par exemple pour Auteur ou Titre. Dans de nombreux cas, la seule caractéristique dont nous nous soucions vraiment est le nom ; toutefois une propriété doit être une ressource afin qu'elle puisse avoir ses propres propriétés.

- Un objet ou une valeur peut être une description, un nom ou un numéro qui décrit le sujet via le prédicat.
- Un triplet est la combinaison d'une ressource, une propriété, et une valeur comme le montre la figure 3.6.

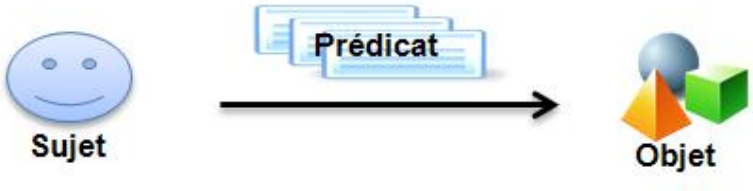


FIGURE 3.6 – Structure d'un triplet RDF

La structure de toute expression RDF est un ensemble de triplets RDF. Soit les triplets RDF suivants :
Le document wikipedia:
« http://en.wikipedia.org/wiki/Tony_Benn » « a pour titre » « Tony Benn », publié par « Wikipedia » et son contenue s'intéresse à « Tony Benn ». La ressource décrite est représentée par une URI. Chaque valeur peut qui la décrit est tout simplement une chaîne de caractères ou elle peut être aussi une ressource sur le web, par exemple la page qui décrit « Tony Benn » est « http://www.tonybenn.com/ »
Une façon est possible d'exprimer ces propriétés en RDF est:

```
1  <rdf:RDF
2    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3    xmlns:foaf="http://xmlns.com/foaf/0.1/"
4    xmlns:dc="http://purl.org/dc/elements/1.1/">
5  <rdf:Description rdf:about="http://en.wikipedia.org/wiki/Tony_Benn">
6      <dc:title>Tony Benn</dc:title>
7          <dc:publisher>Wikipedia</dc:publisher>
8           <foaf:primaryTopic>
9                          <foaf:Person>
10 <foaf:page>rdf:resource="http://blog.developpez.com/jplu/"/>
11                         </foaf:Person>
12           </foaf:primaryTopic>
13      </rdf:Description>
14 </rdf:RDF>
```

Un ensemble de ces triplets est appelé un graphe RDF.

**Graphe RDF :** il est possible de construire un graphe de nœuds et d'arcs, qui décrivent les ressources, leurs propriétés et leurs valeurs (figure 3.7). Comme vu dans la figure, nous pouvons par exemple affirmer des faits sur un service web avec l'ensemble de triplets suivants : Le service (identifié par son URL WSDL) a une Evaluation. L'évaluation a une valeur égale

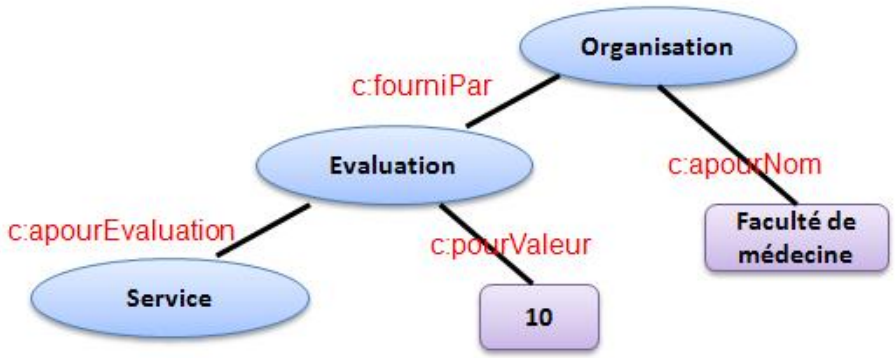


FIGURE 3.7 – Exemple d'un graphe RDF

à 10 qui a été délivrée par un établissement. L'établissement est identifié par sa page web URL « Faculté de médecine ».
La capacité à décrire les ressources va largement au delà: en fait si on assigne plusieurs fonctions à ce service on pourra les représenter à l'aide de deux prédicats désignant deux fonctionnalités différentes et les rattacher à deux ressources sur le web représentant les fonctionnalités elles-mêmes. Les données RDF peuvent être consultées à l'aide de langages de requêtes qui permettent d'interroger un modèle de graphe [161]. On peut noter l'existence de RDQL [23] (RDF Data Query Language) parmi les langages d'interrogation de données RDF. Il permet d'interroger un graphe de données RDF selon un schéma RDF. RDQL est un langage SQL qui permet de sélectionner un sous graphe d'un graphe RDF selon le schéma souhaité. Dans notre exemple, il est possible d'interroger le graphe de la figure 3.7 comme suit : « Retrouvez les noms de toutes les institutions qui ont noté un service particulier, et qui ont donné la valeur 10 ».

**Schéma RDF :** On ne peut pas prévoir de limites pour la description des données à l'aide des métadonnées sauf si on envisage un vocabulaire contrôlé. Les communautés d'utilisateurs de RDF ont éprouvé le besoin de définir des vocabulaires (termes) pour spécifier les concepts utilisés dans leurs descriptions, par exemple pour catégoriser les types des documents et les types des relations entre les documents ou les objets figurant dans un document.
Suite à notre exemple de la figure 3.7, le service peut être décrit en utilisant des classes telles que a:ServiceVente et des propriétés telles que a:LienEvaluation, a:apourFournisseur. De même, il peut s'avérer intéressant de décrire les organisations qui veulent utiliser des classes telles que o:Organisation et des propriétés d'usage telles que o:apourNom, o:apourAdresse. RDF seul ne fournit aucun moyen permettant de définir de telles applications spécifiques des classes et des propriétés. RDF a été étendu en RDF Schema (RDFS) pour ajouter des fonctionnalités comme classes et propriétés [24].
RDFS ne fournit pas un vocabulaire spécifique qu'on est censé appliquer lors de la définition des métadonnées, mais il fournit un corpus générique de classes et de relations entre les classes.

23. http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/

En résumé, RDFS est un schéma XML avec lequel l'utilisateur peut typer les instances de données RDF. L'organisation à l'aide d'un schéma RDF est un moyen pour avoir une présentation hiérarchique des métadonnées, facilement gérable et compréhensible, par exemple, une catégorie o:OrganisationStandard pourrait être définie comme une sous-classe de o:Organisation, ou bien o:apourAdresse peut être définie comme une sous-propriété de o:apourLocalite. Les relations sous-classe et sous-propriété sont transitives, et la fermeture transitive de ces classes peut être exploitée lors de l'interrogation des triplets d'un graphe RDF. Par exemple supposons que la description de notre service faite en RDF, dise qu'il existe des « Cours » qui intègrent des « Séances » de « Travaux pratiques », qu'une « Séance » est-un type de « Cours ». Soit aussi un autre schéma RDF qui décrive que toute « Séance » de « Travaux pratique » est aussi un « Cours ». Dans un tel cas la requête qui cherche tout les « Cours » retournera aussi toute les séances de Travaux pratiques.
Malgré cette facilité d'expressivité, de raisonnement et d'interprétation avec l'utilisation de RDF et RDFS il y a encore des connaissances que nous ne pouvons pas exprimer à l'aide de telles descriptions. RDFS est limité, il permet juste de décrire des relations sous-classe/super-classe, alors qu'il peut y avoir d'autres relations entre les termes que nous tenons à exprimer, comme par exemple les « Séances » de « Travaux pratiques » ayant attribué au moins une note. Ce type de connaissance exige une expressivité assez évoluée ainsi qu'un raisonnement plus spécifique et donc un langage plus évolué.
C'est à ce besoin que répondent les ontologies, qui apparaissent au niveau de la couche qui suit RDFS dans la pile du web sémantique figure 3.5. Mais tout d'abord nous allons discuter l'utilisation de XML et RDF(S) dans le domaine de la biomédecine.
L'utilisation de XML et RDF en biomédecine [106] [130] est un sujet important depuis plusieurs années dans le domaine de l'informatique biomédicale. Parmi les langages utilisés pour représenter des données médicales. XML s'est imposé comme le premier langage de description de données.
Des données médicales structurées sont générées par une multitude de sources, notamment pour exprimer le résultat de mesures de paramètres biologiques effectuées chez des patients. Les médecins et les chercheurs peuvent stocker leurs données médicales dans le cadre de différents sous-systèmes, éventuellement distribués géographiquement. D'autres domaines de spécialité tels que la génétique ou l'imagerie médicale peuvent avoir besoin de ces données, ce qui conduit à partager ces données ainsi que les connaissances qu'on a à propos de ces données.

### 3.2.3 Ontologies

Une définition claire du terme ontologie n'est pas facile, car il est utilisé dans de nombreux contextes différents et dans différents domaines d'étude. Guarino et Giaretta donnent un bon aperçu des interprétations possibles [71]. Ontologie est un terme issu de la philosophie et caractérise une actualité dans la quelle on essaye de décrire la réalité et l'existant sous formes d'entités et de relations entre entités pour suivre leurs interactions [209]. Dans le contexte de la science une ontologie n'est pas destinée à modéliser le monde réel, mais plutôt donne une interprétation correcte de l'existant, du système d'information ou plus précisément des données partagées [186].

Les ontologies constituent un vocabulaire généralement conçu sous forme de taxonomie. Le vocabulaire permet de définir une signification non ambiguë des termes. Les taxonomies permettent d'organiser les termes dans une structure hiérarchique. Par exemple, dans une ontologie on peut bénéficier de relation de généralisation/spécialisation et connecter les concepts entr'eux via des relations plus spécifique ou moins spécifique [210]. Une ontologie peut être aussi plus complexe qu'un simple vocabulaire et taxonomie. Dans le domaine scientifique, les ontologies sont souvent fondées sur la logique de description. Le langage le plus usuel de représentation des connaissances pour les ontologies est OWL [24] (Ontology Web Language).

**Ontologie fondationnelle :** Une ontologie fondationnelle est une ontologie qui définit des concepts généraux/globaux qui sont valide pour plusieurs domaines. Lorsqu'on utilise des ontologies fondationnelles on peut plus facilement aligner les ontologies entr'elles, c'est à dire mettre en correspondance des concepts introduits dans différentes ontologies ou calculer des similarités entre les concepts. Une ontologie de domaine étend éventuellement une ontologie fondationnelle et elle couvre les connaissances relevant d'un seul domaine. Cette typologie sera revue d'une façon plus détaillée dans la définition des objectifs du travail réalisé dans le cadre de cette thèse.

**Logiques de description :** Les logiques de description (DL) font partie de la famille des langages de représentation des connaissances qui sont le plus souvent utilisés pour représenter les connaissances d'un domaine d'application d'une manière structurée et suivant une logique bien définie [11]. Les éléments de base des logiques de description sont les concepts et les rôles. Le concept désigne la classe d'objets et le rôle désigne une relation binaire entre les classes [78]. Comme les logiques de description sont un ensemble de langages pour la représentation des connaissances, ils ont des ensembles de symboles et une syntaxe pour décrire le monde sous forme d'expressions logiques. Les logiques de description sont issues des réseaux d'héritage [102]. Le réseau d'héritage simple se base sur la relation hiérarchique « est-un » entre les concepts et les propriétés. Les logiques de description sont très similaires aux modèles des réseaux d'héritage, mais possèdent une expressivité plus riche. L'ensemble des assertions et des faits constituent une base de connaissances. Une base de connaissance typique, comprend deux composante : la T-box et la A-box, qui sont utilisées pour représenter les connaissances extensionnelles et les connaissances intensionnelles [137].
La T-Box contient les connaissances terminologiques qui sont des connaissances intentionnelles. Les déclarations de base qui se font dans la T-Box sont des définitions exprimées en logique de description. La définition d'un nouveau concept est basée sur les concepts précédemment définis. Par exemple, une définition de la femme pourrait être représentée comme suit: $Femme \equiv Personne \sqcap Femelle$ Cette expression signifie que l'ensemble des femmes peut être déterminé en faisant intersecter l'ensemble des personnes avec l'ensemble des femelles. Nous pouvons également écrire l'expression comme: $Femme \sqsubseteq Personne$. Ce qui signifie que la femme est une personne. Cette expression est appelé axiome ou plus précisément axiome d'inclusion.
La définition d'un concept de la T-Box suit les deux principes suivants :

1. une seule définition d'un nom de concept est autorisé
2. Les définitions cycliques ne sont pas autorisées : ce qui signifie que la définition d'un

24. http://www.w3.org/TR/owl-features/

concept qui fait référence à lui-même ou à d'autres concepts qui font référence à lui-même indirectement (lien d'héritage par exemple) [137].

Une assertion de concept définit à quel concept appartient une instance. Une assertion de rôle définit une relation entre deux instances. Par exemple, $Male \sqcap Personne$ (Mark) est une affirmation du concept Mark et aPourEnfant (Mark, Taylor) est une affirmation de rôle pour décrire la relation entre Mark et Taylor. La tâche de raisonnement de base de la A-Box est une forme de contrôle qui assure la cohérence des connaissances. Elle peut être aussi un processus de recherche et récupération de connaissances [137].

### 3.2.4 Raisonneurs

Comme nous l'avons mentionné précédemment, les logiques de description sont conçues pour offrir des mécanismes de raisonnement sur les données classées. Ces capacités peuvent être utilisées pour répondre à différents besoins.
Dans le cadre du développement des ontologies pour (1) vérifier la cohérence entre les concepts définis et les axiomes qui leur sont associés, (2) en déduire des relations de subsomption entre les concepts de l'ontologie. La vérification de la subsomption permet la génération automatique d'une hiérarchie déduite à partir de la classification des concepts définie dans l'ontologie. Le développement d'une ontologie se fait généralement par l'utilisation d'éditeurs d'ontologies. La figure suivante est une capture d'écran de l'éditeur d'ontologie Protégé [92] qui affiche une ontologie de domaine en neuroimagerie. On voit la hiérarchie de subsomption des concepts et la liste des axiomes associés à ces concepts.

Les raisonneurs DL utilisent les algorithmes des tableaux optimisés pour la subsomption, et le contrôle de cohérence entre les concepts. Parmi les raisonneurs DL largement utilisés on peut citer :

**RacerPro**[25] **:** C'est un logiciel commercialisé, mais qui propose une licence libre pour l'utilisation dans le monde de la recherche. Il fournit des services d'inférence de connaissances terminologiques ainsi que pour les représentations de la connaissance sur les individus. Basé sur des techniques d'optimisation et de nouvelles techniques qui ont été développées dans le domaine de la recherche sur les logiques de description au fil des ans, il fournit aussi une architecture pour des tâches de raisonnement typiques. Il est pratique et compatible avec le standard OWL du W3C. Il permet de faire des raisonnements sur des ontologies fondées sur la logique de description. Il supporte des descriptions logiques avec des restrictions de cardinalités qualifiées sur les rôle).

**Pellet**[26] **:** est aussi un raisonneur qui possède deux licences dont une est commerciale et l'autre pour la recherche en environnement de développement ouvert. Il est basé sur des algorithmes développés pour les logiques de description expressives. Il supporte toutes les constructions OWL DL y compris owl:oneof et owl:hasValue. Pellet propose une solution complète consistant en un algorithme décidable et efficace pour OWL-DL

**FaCT++**[27] **:** est un raisonneur d'utilisation libre codé en c++. Il supporte la logique SROI2. Il est intégrable dans l'API Protégé. Il implémente aussi l'algorithme des tableaux.

25. http://www.racer-systems.com/
26. http://clarkparsia.com/pellet/
27. http://owl.man.ac.uk/factplusplus/

**KAON2** [28] : Contrairement à la plupart des raisonneurs DL disponibles, KAON2 n'utilise pas l'algorithme des tableaux mais de nouveaux algorithmes qui réduisent une base de connaissances SHIQ (D) à un programme datalog (figure 3.8). Ces nouveaux algorithmes permettent d'appliquer des techniques bien connues issues des bases de données déductives, tels que les jeux de magie. Il est aussi étendu avec des règles SWRL DL-safe [29].

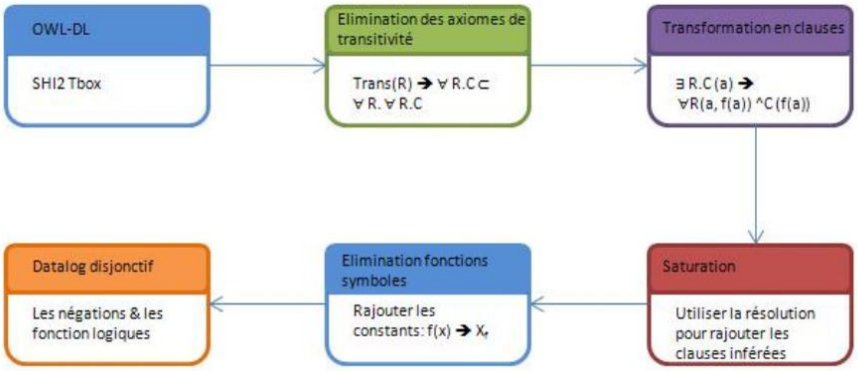


FIGURE 3.8 – Transformation de connaissances représentés en OWL en un programme datalog disjonctif, effectué par KAON2

**HermiT** [30] : est un nouveau raisonneur OWL basé sur un nouvel algorithme de calcul appelé « hyper-tableau ». HermiT intègre un certain nombre de techniques d'optimisation et diverses techniques pour la classification d'ontologies. Il propose des nouvelles compétences en logique de description par exemple une meilleure stratégie qui essaie de minimiser le nombre d'individus réutilisés au lieu d'en générer de nouveaux pour chaque classification d'ontologie. Nos tests montrent que HermiT est généralement beaucoup plus rapide que d'autres raisonneurs, et il est capable de classer un certain nombre d'ontologies que d'autres raisonneurs ne sont pas en mesure de traiter.

### 3.2.5 Ontologies et raisonnement en biomédecine

Le domaine de la biomédecine est d'ores et déjà un utilisateur important des technologies du web sémantique[123] [171]. Celles-ci sont d'une aide précieuse pour organiser les ressources disponibles de manière plus structurée (article, images, manuels de procédures chirurgicales, catalogues de médicaments) [93], mais aussi leur ajouter des descriptions sémantiques explicites compréhensibles par l'utilisateur et la machine, qui peuvent aider à l'analyse et l'interprétation de ces données. Ceci concerne notamment la recherche d'article scientifique sur le net [52] grâce à des outils du web sémantique qui facilitent l'indexation des sites web, comme par exemple PubMed [31] qui propose un service web de la librairie nationale américaine de la médecine. De même GATEWAY [32] propose une interface commune d'interrogation de bases

28. http://kaon2.semanticweb.org/
29. **http://weblog.clarkparsia.com/2007/08/27/understanding-swrl-part-2-dl-safety/**.
30. http://hermit-reasoner.com/
31. http://www.ncbi.nlm.nih.gov/pubmed/
32. http://gateway.nlm.nih.gov/

de données médicales.
Toutefois une partie de ces ressources biomédicales englobe les outils de traitement de données biomédicales (outils de traitement d'images, outils de traitement de séquences biologiques d'ADN ou modules d'analyse sous forme de workflow de données) qui grâce au web sémantique peuvent être partagés, invoqués et mieux déployés au profit des utilisateurs. Par exemple l'utilisation des IRIs permet de mieux gérer et exploiter les identifiants des entités biomédicales, qui prolifèrent d'une manière très rapide dans les entrepôts de données partagées. Aussi, les langages de description de données comme XML, RDF et OWL aident à mieux décrire les ressources et à les rendre mieux compréhensibles par à la fois l'homme et la machine. Ils permettent aussi de définir des relations hiérarchiques entre les différentes entités de données permettant de catégoriser ces ressources selon des axiomes décrivant leurs capacités physiques ou techniques et donc les comparer, les analyser ou décider dans quel cadre on pourra les réutiliser. En l'occurrence, les outils de traitement d'images tels qu'ils sont développés aujourd'hui, manquent de sémantique. Cette sémantique permettra de mieux les décrire, expliciter leurs fonctionnalités, leurs domaines de validité et mieux les enchainer dans le cadre de pipeline de traitement.
Chacune des fonctionnalités proposées par le web sémantique a un apport significatif pour le domaine de la santé et des sciences du vivant. En fait on ne vise pas seulement la classification des ressources biomédicale mais aussi la détection d'incohérences lors de l'enregistrement et la sauvegarde des données. Lors de la réutilisation de ces données et notamment dans le contexte de notre travail, les incohérences peuvent affecter le bon fonctionnement de l'outil qu'on voulait exploiter, ou nous aider à se décider par rapport un traitement souhaité.
De façon plus précise, la description des outils sera faite probablement à deux niveaux d'abstraction. Le premier décrira les types de données en entrée et en sortie, de façon à pouvoir les vérifier lors de leur enchaînement et le deuxième visera à décrire et vérifier les fonctionnalités de l'outil et les valider par rapport aux traitements souhaités et aux données utilisées.
Les capacités du web sémantique permettent de mettre en place ce type de vérification, soit en utilisant des axiomes représentés à l'aide du langage OWL, exploitables dans des raisonnements effectuées à l'aide des raisonneurs DL. Ces raisonneurs exploitent les axiomes, et permettent de réaliser des traitements de classification et de vérification sémantique qu'on applique à ces ressources, ce qui permettra de minimiser les erreurs que l'humain ne peut matériellement détecter.
A long terme l'utilisation des technologies du web sémantique va certainement améliorer la recherche dans le domaine de la biomédecine. Elle permettra l'intégration de données multi-domaines en fournissant des vocabulaires formels permettant d'articuler entr'elles des données de différents domaines grâce à des ontologies fondationnelles. Elle permettra aussi l'ajout de mécanismes de vérification sémantique axés sur les axiomes qui assureront la consistance des ressources utilisées et enregistrées. Elles permettront de mettre à la disposition des utilisateurs des outils puissants qui faciliteront les tâches de recherche. Elles fourniront par exemple un moyen efficace pour formuler des nouvelles hypothèses pour la recherche basées sur les expérimentations cliniques ou des nouveaux outils basés sur l'enchaînement de plusieurs outils partagés par différents centres de recherches.

### 3.2.6 Entrepôts de données sémantiques

Comme nous l'avons vu RDF est le standard de représentation de données sémantiques. Trois solutions principales peuvent être envisagées pour stocker de façon permanente des triplets RDF. Les entrepôts de données RDF sont généralement (i) des systèmes de gestion de fichiers RDF, ou (ii) des systèmes de gestion de bases de données relationnelles (SGBDR), ou (iii) des systèmes de stockage de données gérant des graphes en mémoire.

Les entrepôts de données à base de fichiers constituent une solution adaptée aux cas où les données à gérer restent minimes. Il est plus facile de stocker des données/métadonnées dans un fichier RDF que dans une base de données, par contre un problème majeur se pose avec ce type d'entrepôt de données qui est la nécessité de parcourir séquentiellement les fichiers RDF, ce qui est couteux en temps d'accès aux informations.
Une autre approche disponible pour implémenter un entrepôt de données sémantiques est de recourir à un SGBDR (Systèmes de Gestion de Bases de Données Relationnelles) c'est une approche pratique et évolutive en ce qui concerne la taille croissante des triplets RDF. En effet, l'entrepôt RDF est indépendant de la base de données et il bénéficie largement de cette architecture dans le sens où il gère les données en mémoire puis il fait des mises à jour dans la base de données à la demande. Dans ce type de base de données la performance de l'entrepôt sémantique dépend fortement de la performance des bases de données utilisées.
Contrairement aux deux premières méthodes le stockage natif des données sémantiques à l'aide de graphes présente beaucoup d'avantages. Les entrepôts de données sémantiques sont conçus de telle façon qu'ils mettent en œuvre des stratégies d'optimisation de la gestion des triplets RDF. Généralement ce type d'entrepôt de données adopte la structure d'un graphe. Parmi les entrepôts de données sémantiques largement utilisés on peut citer :

**Jena [122] :** Jena est une plateforme développée en langage Java qui vise à fournir aux utilisateurs un ensemble de composant logiciels couvrant la plupart des préoccupations liées au développement d'applications du web sémantique. Elle fournit un environnement de programmation unifié facilitant la gestion des données RDF (S) et des ontologies OWL. Jena fournit deux sous-systèmes de persistance de données RDF et OWL, à savoir TDB et SDB :
- TDB est basé sur un système de graphe, et fournit une API qui permet la gestion de fichiers locaux stockés sur un disque dur. Il utilise des algorithmes de manipulation de graphes de haute performance qui facilitent l'indexation de ces fichiers. Il propose une application flexible pour la gestion de graphes RDF. Ceci permet de manipuler les données pertinentes dans ces graphes d'une façon rapide.
- SDB, lui est un autre système de la plateforme Jena basé sur les bases de données relationnelles classiques. Ces bases de données communiquent via une interface Java en cas de demande de l'utilisateur pour rendre persistantes les données stockées dans les graphes RDF et les enregistrer dans la base de données.

Jena est diffusé sous la forme de deux versions stables. La première appelés Jena1 propose la gestion de fichiers de types RDF, XML, N3 et N-triple ainsi qu'un système basique d'interrogation de données RDQL.
La deuxième appelée Jena2 vient compléter la première et la rendre plus découplée[33]. Elle utilise Jena1 en tant qu'un module de base pour standardiser tout type de données séman-

33. Intégrable dans une architecture distribuée sous forme de client/serveur

tiques principalement OWL et DAML+OIL [34] en RDF et RDFS. La couche qui manipule les graphes permet la déduction de nouvelles connaissances grâce à (i) des traitements sémantiques, (ii) des règles d'inférences ou (iii) un accès à des ressources déjà existantes sur le Web. Jena utilise le langage RDQL, qui facilite l'extraction des données stockées sous forme de graphe. Ces requêtes peuvent être transformées en requêtes relationnelles pour récupérer des données persistantes dans la base de données relationnelles (appelée back-end store).

**Joseki** [35] **:** Joseki est un serveur web qui permet d'accéder à des données sémantiques à travers le langage de requêtes sémantiques SPARQL [36]. Il implémente à la fois les systèmes d'interrogation SPARQL et les systèmes de mise à jour SPARQL/UPDATE. Il fournit une interface de service web pour interroger et mettre à jour les graphes RDF. Des données RDF sont gérées en interne par le biais de Jena.

**Virtuoso [54] :** est un middleware permettant de faire communiquer différents types de base de données. Il combine les fonctionnalités d'un système de gestion de bases de données relationnelles SGBDR, de base de données virtuelles, RDF et XML. Il a été implémenté d'une façon qui lui permet d'utiliser plusieurs protocoles d'accès à des sources de données à la fois. La base de données universelle de virtuoso peut inclure les données stocké dans des fichiers physiques, dans les mémoire physiques et utilise les transactions des systèmes d'exploitation qui gèrent ces données. Il consacre un seul processus qui joue le rôle d'un listner sur un port spécifique pour HTTP ou SOAP. Sur le plan opérationnel, la conception de virtuoso permet de bénéficier des différents processus des systèmes d'exploitations et des différentes unités de traitements de données auxquelles il est connecté.

**Allegrograph** [37]**:** Comme Virtuoso, Allegrograph est un produit commercial. Il offre une haute performance basée sur des disques de stockage utilisant un client/serveur HTTP et un système d'interrogation basée sur SPARQL ainsi que des raisonnement basés sur et RDFS (RDFS++). Plusieurs entrepôts de triplets RDF, peuvent être assemblés dans un seul entrepôt virtuel, permettant ainsi la fédération des différents jeux de données sémantiques distribuées.

**Sésame [25] :** Sésame est une solution open-source pour le stockage et l'interrogation des données RDF(S). Grâce à son API, Sésame peut être connecté à plusieurs autres entrepôts de stockage tels que les fichiers standards ou une base de données relationnelles. L'outil offre des capacités de raisonnement grâce à l'appui d'un raisonneur RDFS .Toutes les manipulations de données passent par des requêtes SPARQL standard. Alibaba [38] est une extension de Sésame qui permet de lier des objets RDF et OWL avec des classes Java.

**OWLIM [88] :** est un entrepôt de données sémantiques très puissant basé sur Sésame. Deux versions sont disponibles, SwiftOWLIM vise l'édition libre, et BigOWLIM le marché d'entreprise. Les deux versions fournissent une stratégie de persistance qui permet de garantir la préservation et la cohérence des données, et le support de systèmes de raisonnement sur des données OWL et RDF(S).

**Tupelo [60] :** Tupelo est un système de gestion de données et de métadonnées destiné

34. http://www.daml.org/2001/03/daml+oil-index.html
35. http://joseki.sourceforge.net/documentation.html
36. http://www.w3.org/TR/sparql11-query/
37. http://www.franz.com/agraph/allegrograph/
38. http://notes.3kbo.com/alibaba

au domaine de e-science, basé sur les technologies du web sémantique. Il offre une variété de services génériques pour la gestion des données et des métadonnées en utilisant les deux implémentations (i) best-of-breed de bases de données sémantiques comme Jena et SESAME, ainsi que les technologies de stockage ordinaires tels que les fichiers plats. Il vise à améliorer l'interopérabilité des métadonnées en conservant le sens des métadonnées lors de leurs déplacement ou leurs manipulation et s'intéresse à la façon dont ces données ont été traitées pour pouvoir les réutiliser dans des contextes similaires. Tupelo rend les données et les métadonnées portable sur une variété de contextes et scénarios de déploiement, y compris les applications de bureau, des applications Web, et des architectures distribuées plus complexes. Grâce à l'utilisation du système d'identification globale et à l'utilisation d'une sémantique explicite, les métadonnées créées et gérées par Tupelo peuvent être facilement exportées et utilisées par une grande variété de plateformes logicielles utilisant le web sémantique.

### 3.2.7 Moteurs de recherche et interrogation de données sémantiques

L'apparition du web sémantique a permis de partager un grand nombre de données via le Net et via des grandes unités de stockage ou entrepôts de données spécifiques aux organisations et sociétés. Dans un contexte très varié et vaste comme la bioinformatique ou la biomédecine, nous avons besoin de fouiller l'information avec des outils spécialisés. Ces moteurs de recherche sémantique. Répondent à ce besoin. Il y a plusieurs moteurs de recherche sémantique disponibles aujourd'hui, qui diffèrent dans leurs objectifs, leur portée et les approches et les mécanismes de recherche qu'ils utilisent ainsi que les différentes plateformes pour lesquelles ils ont été développés [207]. Ce paragraphe discute un certain nombre d'outils en mettant l'accent sur leurs points communs et leurs différences.

Tout d'abord, la recherche/récupération des données sémantiques est difficile car elle dépend de plusieurs critères comme le domaine de recherche, le ou les outils mis en jeu et le type de résultat que l'utilisateur souhaite obtenir. Par exemple, une simple recherche consistera à soumettre la requête à un seul entrepôt de données et à récupérer les résultats. Une recherche plus complexe, consistera en la combinaison de plusieurs résultats intermédiaires issue de l'interrogation de plusieurs entrepôts de données sémantiques. La recherche est notamment plus complexe lorsque les données ne sont pas complétées par des métadonnées qui expriment leurs sens. Pour surmonter ce problème et extraire des informations de façon intelligente, les technologies du web sémantique ont un apport intéressant et ont le pouvoir d'améliorer les performances de la recherche classique.

Un des cas pratiques classique concerne la recherche d'articles scientifiques [80]. L'extension que le web sémantique apporte aux documents et données sur le web permet de réaliser des recherches dites plus profondes. Par exemple, les articles scientifiques archivés et classés selon leur contenu (auteur, titre, résumé et maison d'édition) peuvent être recherchés avec une grande précision. En effet, les relations que le web sémantique tisse entre les entités permettent de décider si telle ressource trouvée est pertinente ou et permet de rassembler des informations sur les données recherchées quelles que soit leurs localisation physique.

Un autre type de recherche utilisant les capacités du web sémantique est la modularisation et la catégorisation [15]. Elles met l'accent sur (i) l'utilisation des taxonomies pour la modularisation et les taxonomies, les concepts et ainsi que les relations utilisées et les entités

qui constituent l'ontologie lors de la catégorisation puis (ii) le contexte et la structure des ontologies qui signifie les différents langage avec lesquels une ontologie peut être formulée [114] ainsi que les sens sémantique des relations et des entités. D'autres utilisent des algorithmes de classification qui facilitent la navigation dans des graphes et utilisant des outils performants comme Jena, OLWIM etc [49] [166].
Plusieurs moteurs de recherche sémantique ont été développés ces dernières années :

**SemSearch [103]** est l'un des plus populaires. Il vise à exploiter au mieux les capacités du web sémantique en bénéficiant de son expressivité riche pour élargir les champs de recherche et ramener des résultats pertinents à l'utilisateur. Il cache la complexité de la recherche sémantique aux utilisateurs finaux et facilite donc la recherche. Contrairement à d'autres outils basés sur les mot clés, SemSearch permet d'exprimer et traiter les requêtes complexes des utilisateurs sans qu'il n'y ait de problème de surcharge de connaissances par l'utilisateur : le plus souvent les utilisateurs doivent avoir une connaissance approfondie de la base de connaissance et en occurrence de l'ontologie ou de langage spécifique d'interrogation de données sémantiques). Par ailleurs, SemSearch fournit des interfaces sophistiquées pour rendre des résultats finaux à la fois riches et compréhensibles par les utilisateurs finaux.

**Ontobroker [58]**, [43] utilise les techniques de l'intelligence artificielle pour améliorer l'accès aux sources d'information hétérogènes, dispersées et semi-structurées telles que celles présentes dans le World Wide Web. Il repose sur l'utilisation des ontologies pour annoter les pages web, formuler des requêtes, et en tirer des réponses. Ontobroker propose un langage d'ontologies spécifique dédié à l'annotation des documents HTML.
Ontobroker s'appuie sur deux outil : un webcrawler et un moteur d'inférence. Le webcrawler recueille les pages web à partir du web, extrait leurs annotations, et les parcourt selon leurs formats internes. Le moteur d'inférence, lui, combine les faits en sa possession avec la terminologie et les axiomes de l'ontologie, et en déduit les réponses aux requêtes des utilisateurs. Pour ce faire, il réalise un travail assez complexe qui se base sur les techniques standard des bases de données déductives.

**On2broker [57]** est le système successeur de Ontobroker. Les principales décisions de conception de On2broker sont la séparation claire de la gestion de la requête et des moteurs d'inférence et l'intégration des nouveaux standards du web tels que XML et RDF. Le moteur d'inférence fonctionne comme un démon en arrière-plan. Il prend les faits à partir d'une base de données, déduit des faits nouveaux, et retourne les résultats dans la base de données. Le moteur de recherche n'interagit pas directement avec le moteur d'inférence.

**CORESE [39] :(COnceptual REsource Search Engine)** Corese est un moteur de recherche sémantique basé sur un moteur d'inférence RDF(S) utilisant le formalisme des graphes [40]. Il permet le traitement de RDFS, une partie d'OWL Lite et des énoncés RDF (RDF statements) en s'appuyant sur le formalisme des graphes conceptuel GC. Il peut effectuer des requêtes SPARQL et exécuter des règles sur des graphes RDF. Corese peut charger toute variante du langage OWL (Lite, Full, DL) mais n'interprète que les sous-ensembles basés sur RDF(S) et OWL Lite.
L'objectif principal de CORESE est de (i) proposer une interface facile à manipuler pour

39. http://www-sop.inria.fr/edelweiss/software/corese/
40. http ://www.jfsowa.com/cg/index.htm

l'interrogation des données sémantiques et (ii) renvoyer à l'utilisateur une représentation adéquate des résultats. Pour ceci, la méthode de calcul de CORESE repose sur les opérations de projection de graphes conceptuels, qui permettent d'extraire un graphe résultat à partir d'une requête de données. Ces données sont ensuite transformées selon la représentation que l'utilisateur souhaite au moyen de feuilles de style avec un contenu RDF(S) ou XML [41].
CORESE combine les avantages de l'utilisation du langage standard RDF pour exprimer et échanger des métadonnées, et le requêtage et les mécanismes d'inférence disponibles dans le formalisme GC [182]. Parmi les formalismes utilisés en Intelligence Artificielle pour la représentation des connaissances, GC est largement apprécié car il est basé sur un modèle formel solide et fournit une expressivité riche qui rend les données plus claires et lisibles à l'utilisateur. En outre, des mécanismes d'inférence et de requêtage ont été développés et sont disponibles pour manipuler les GC. Il existe une adéquation réelle entre les deux modèles à savoir les classes RDFS et les concepts représentés en GC. Plus précisément, les triplets RDF sont mappés en une base de faits représentés en GC, la hiérarchie de classes définie dans un schéma RDF est mappée à une hiérarchie de types de concepts dans le formalisme GC, et la hiérarchie des propriétés décrites dans le schéma RDF est mappée sur une hiérarchie de types de relations en GC. La hiérarchie des types de concepts et la hiérarchie des types de relations constituent ce qu'on appelle « support » dans le formalisme GC: ils définissent le vocabulaire conceptuel qui va être utilisé dans le GC.
Corese permet à l'utilisateur de faire des inférences selon un mécanisme simple basé sur le système de règles en XML/SPARQL exprimées dans un format qui lui est unique. Ce sont des règles de production qui permettent de compléter et d'enrichir la base de connaissance. Le moteur RDF(S) de Corese prend en charge ces règles, produit les connaissances qu'elles permettent de décrire et les intègre ou les propose à l'utilisateur. Ces règles suivent la forme IF condition THEN conclusion, où condition représente une requête à vérifier et conclusion représente une ou plusieurs nouvelles assertions RDF à ajouter dans le graphe global (exemple figure 3.9).

Enfin, CORESE possède une fonction de recherche approchée qui permet de fournir une réponse approchée à une requête, dans le cas où celle-ci ne donne lieu à aucune réponse [40]. Cette fonction utilise une fonction de calcul de distance entre les concepts d'une ontologie, en se basant sur le calcul de la longueur des chemins entre les concepts et en faisant varier cette longueur avec la profondeur ; ainsi, deux concepts frères de profondeur n+1 sont plus proches que deux concepts frères de profondeur n.

CORESE est conçu selon une architecture "trois-tiers" (figure 3.10) , il est disponible au travers d'une API et utilisable directement par des applications du web sémantique. Il peut également être intégré dans un serveur (SEWESE) permettant la création dynamique d'applications web, à partir d'une base de connaissances représentée en RDF ou en OWL, tout en se basant sur des formalismes de transformation (XSLT) et sur des technologies standards du web (XML, JSP).

Le moteur de recherche **SHOE [76]** est l'un des premiers à prendre en charge les formulaires dans la recherche sémantique. Il fournit des formulaires web sophistiqués qui permettent aux utilisateurs de spécifier leurs requêtes. Ces formulaires sont dédiés spécifiquement aux

41. http://www.w3.org/XML/

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE rdf:RDF [
<!ENTITY cos "http://www.inria.fr/acacia/corese#">
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<!ENTITY s "http://www.inria.fr/acacia/schemas#>
<rdf:RDF xmlns:rdf="&rdf;" xmlns:cos="&cos;" xmlns:s="&s;" >

        <cos:rule cos:name='r1'>
                <cos:if>
                PREFIX s: &lt;http://example/of/ontology#&gt;
                {
                        ?m rdf:type s:Person .
                        ?m s:head ?t .
                        ?t rdf:type s:Team .
                        ?t s:hasMember ?p .
                        ?p rdf:type s:Person
                }
                </cos:if>

                <cos:then>
                {
                        ?m s:manage ?p 1
                }
                </cos:then>
        </cos:rule>

</rdf:RDF>
```

FIGURE 3.9 – Exemple de règle CORESE

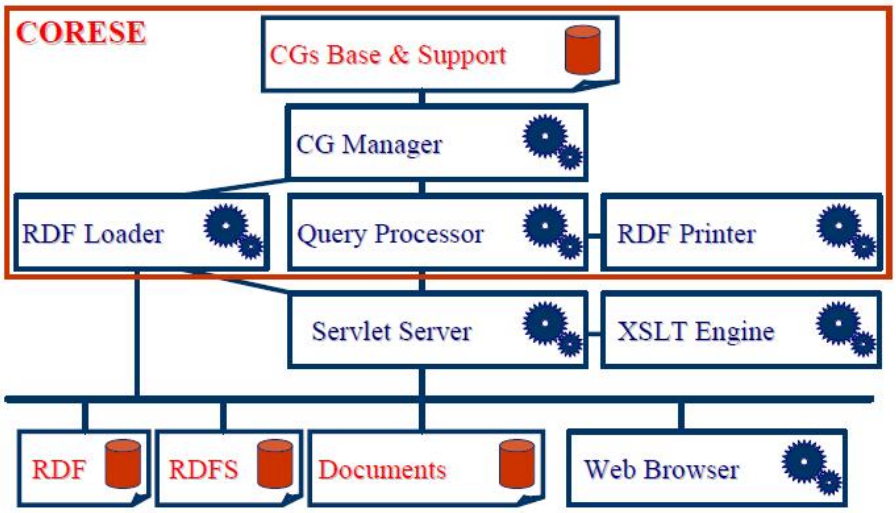


FIGURE 3.10 – Architecture "trois tiers" de Corese [61]

utilisateurs qui sont assez familiers avec les entrepôts de données sémantiques (ontologies et RDF) et les bases de connaissances. Les utilisateurs non familiers avec la technique auront des difficultés avec le mode d'utilisation de ces formulaires. En particulier ils auront des difficultés lors la formulation des requêtes.

Le moteur de recherche **TAP [72]** et le moteur de recherche présenté dans [169] sont

des moteurs de recherche sémantique basés sur la recherche par mots-clés. Le processus de recherche des moteurs de recherche comprend souvent deux étapes principales : (i) trouver une concordance entre les données et le mot-clé spécifié par l'utilisateur et (ii) récupérer les instances qui sont directement liées aux données manifestant une telle concordance. Ces moteurs de recherche fournissent des moyens complets et intelligents pour la classification et le regroupement des résultats des recherches effectuées par les utilisateurs. Leur problème majeur est le manque de sémantique et d'exploitation des données sémantiques sur le web, d'où l'insuffisance de leurs résultats.

**AQUALOG [108] et ORAKEL [38]** sont des exemples de moteurs de recherche basés sur le système question/réponse et sur les ontologies. Ils utilisent les technologies de traitement du langage naturel. Il s'agit de reformuler les requêtes exprimées en langage naturel sous forme de triplets ontologiques. Bien que ces outils semblent être idéaux pour les utilisateurs naïfs, leurs performances restent très limitées et fortement influencées par la performance des techniques utilisées en traitement du langage naturel.

### 3.2.8 Synthèse

Comme tous les systèmes, les systèmes de santé doivent associer l'interopérabilité syntaxique et sémantique pour garantir l'interfonctionnement harmonieux des composants logiciels permettant de rechercher et traiter les informations. Ceci représente dans le domaine scientifique un enjeu tout particulier en permettant l'intégration cohérente de données et de connaissances issues de domaines différents. Elle permet aussi d'élaborer un champs de travail plus large permettant d'avoir des perspectives intéressantes et communes dans la science. Ceci permet au centres de recherches d'échanger des informations et d'être au courant de toutes les nouveautés scientifiques.
L'interopérabilité syntaxique est notamment basé sur les terminologies médicales classiques. Elle soulève le problème de l'intelligence des informations traitées par les différents systèmes. Les ontologies sont aujourd'hui un moyen très puissant qui permet de décrire les ressources en référence à un référentiel sémantique explicite, leurs donnant ainsi le caractère intelligible souhaité. Le développement des ontologies a été accompagné par le développement des d'outils de raisonnement et de gestion de données sémantiques utilisés aujourd'hui par plusieurs plateformes biomédicales comme BIRN, caBIG etc.
Dans le contexte de notre projet NeuroLOG, et spécifiquement dans le cadre de cette thèse on s'intéresse au partage et la réutilisation des outils de traitement d'images. En fait on souhaite construire une plateforme qui permette le partage des outils, leur exécution et le partage des résultats des traitements effectués par ces outils. Ceci suppose une démarche normalisatrice pour s'assurer que les utilisateurs de cette plateforme puisse effectivement consulter, utiliser et réutiliser les résultats des traitements qu'ils ont effectués dans le cadre de leurs travaux de recherche et les partager avec d'autres centres de recherche dans le cadre de collaborations. Lors de la conception du projet NeuroLOG, le choix s'est porté vers une approche fondée sur l'adoption d'ontologies explicitant le consensus sur une sémantique partagée des informations. La section suivante a pour but de présenter les principales caractéristiques des services partagés, notamment leur rôle dans le partage des traitements dans le domaine biomédical. Elle démontre aussi le besoin d'interopérabilité pour la constitution de workflows ainsi que la

validation automatique des résultats des traitements.

## 3.3 Composition des services

On dispose sur le Net d'un grand nombre de services web qui peuvent traiter des demandes particulières. Lorsqu'une requête complexe ne peut pas être traité par un seul service, une combinaison de plus d'un service est obligatoire pour la traiter. Le processus consistant à combiner des services web pour réaliser des tâches complexes est appelé composition de services web. Le service résultant est appelé un service composite qui est défini comme un ensemble de services atomiques ou composites ainsi qu'un flux de contrôle et de données entre les services.

La composition des services web est un sujet de recherche d'actualité qui attire l'attention de plusieurs communautés de recherche du fait de sa complexité et de sa dépendance au domaine d'application.

### 3.3.1 Utilisation des registres de services web

#### 3.3.1.1 UDDI

UDDI fournit un mécanisme pour la gestion des descriptions des services web. Bien qu'il soit souvent considéré comme un mécanisme annuaire ou un annuaire tout court, il définit aussi une norme et une méthode de structuration de données pour représenter des informations qui décrivent les services en XML. Il y a quatre structures fondamentales de données dans un annuaire de service UDDI, comme illustré sur le schéma suivant :

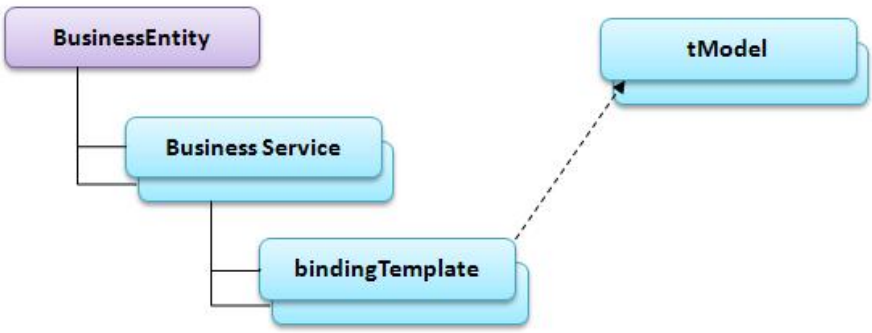


FIGURE 3.11 – Structure de base de UDDI [139]

Quand on souhaite créer un registre ou annuaire UDDI on démarre avec une businessEntity. Chaque élément de businessEntity s'occupe de la modélisation d'un certain type d'informations sur une entreprise, y compris des renseignements de base (par exemple, nom de l'entreprise, coordonnées), des informations de catégorisation (par exemple, de quel genre d'entreprise il s'agit?), et des informations d'identification. Une businessEntity contient une collection de businessServices, une pour chaque service web que l'entreprise souhaite publier. Chaque élément businessService contient des informations techniques et descriptives concernant la businessEntity du service web correspondant. Un businessService contient une col-

lection de bindingTemplates. Un bindingTemplate décrit des informations liées à la façon d'accéder au service (par exemple, le point d'accès au service) et décrit comment le businessService utilise les diverses spécifications techniques. Une spécification technique dérive d'un tModel, dont elle hérite toutes les spécifications. Un tModel peut être utilisé pour créer des concepts nombreux et différents tels que: un type de service défini dans une taxonomie, une technologie utilisée tels que HTTPS. La collection d'éléments d'un bindingTemplate associée au businessService représente l'empreinte digitale des différentes technologies utilisées par les entités métiers businessService.

### 3.3.2 Registres de services en bioinformatique et biomédecine

#### 3.3.2.1 UDDI en biomédecine et en bioinformatique

Le registre UDDI est utilisé pour faciliter la découverte des services web et les mettre à la disposition des utilisateurs. Dans les domaines biomédical et bioinformatique, le contexte d'utilisation d'UDDI change généralement à cause des particularités de ces domaines. Les applications sont développées dans un domaine spécifique et dépendent de plusieurs types de données et structures de contrôle d'accès et de protocoles de sécurité, ce qui rend l'utilisation de ce type de registre moins fréquente et plus difficile à mettre en place.
Les difficultés que nous pouvons rencontrer lors de la mise en place d'un registre UDDI dans un système bioinformatique ou médical sont liées à la liaison avec la nature des services proposés par les plateformes de ces domaines, par exemple le HIS (Hospital Information System), le RIS (Radiology Information System) et le PACS (Picture Achiving and Communication System) [7].
Un client cherche des services qui collent bien avec un certain service déjà utilisé, il va solliciter un registre UDDI qui détient un certain nombre de WSDL qu'il va analyser pour lui donner une réponse effective concernant sa requête. Généralement les plateformes biomédicales ou bioinformatique qui détiennent des services représentent une structure de descripteur différente par rapport aux standards ne pourront pas bénéficier de UDDI [131]

Pour la découverte de services le registre peut être considéré comme un registre de pages blanches qui fournit des informations concernant les fournisseurs de services et les services offert. Ceci n'aide pas à la catégorisation des services, ou à leurs invocations selon leur protocole de sécurité. Il faudra donc ajouter de l'information de contexte au niveau du client UDDI. Dans ce second contexte UDDI est considéré comme un registre de pages jaunes, il fournit des informations qui sont pertinentes et qui décrivent les caractéristiques techniques des services web déployés au sein du registre.
Afin de pouvoir faire communiquer les plateformes qui détiennent des services web de nature médicale ou bioinformatique on note une tendance à étendre les fonctionnalités de UDDI pour les adapter aux besoins spécifique de ces domaines.

#### 3.3.2.2 Le projet $^{my}$GRID [212]

$^{my}$Grid est un projet eScience mené au Royaume-Uni. Il cherche à développer un environnement open source de partage de services en bioinformatique principalement, à un niveau abstrait pour aider à la construction, à la gestion et au partage de données dans des

expériences in silico en biologie (une expérience in silico est une procédure qui utilise des compétences d'analyse de données fournies par des outils informatiques pour tester des hypothèses ou démontrer des faits connus).
Dans le cadre du projet $^{my}$Grid, le fait d'utiliser la technologies des services a motivé le développement d'une version fédérée et étendue du registre UDDI appelé UDDI-M [129]. Cette nouvelle version du registre permet la publication de services en ajoutant des métadonnées sur leur emplacement, les propriétés, les versions, les coûts, la qualité de service, la sécurité, etc.

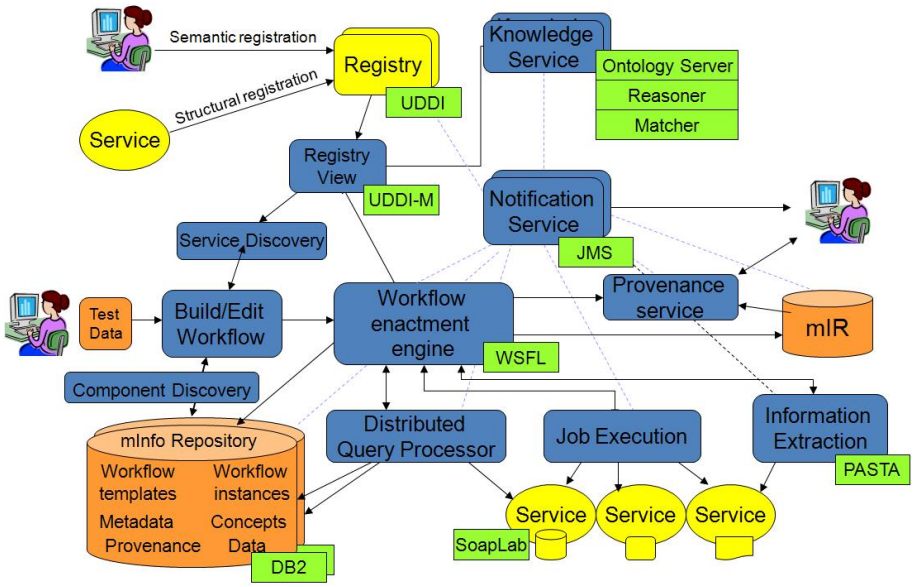


FIGURE 3.12 – Architecture du projet $^{my}$GRID : utilisation du registre UDDI-M [135]

Comme les données et les services sont annotés en utilisant des concepts tirés d'ontologies, ils peuvent ensuite être exploités par d'autres applications. Ainsi, quand un utilisateur veut trouver des services bioinformatiques, les composer et les appliquer aux données, les annotations sémantiques associées aux services disponibles peuvent être recherchées dans le même registre et utilisées pour une vérification de compatibilité. Le registre peut également incorporer des annotations venant d'autres applications externes. Ces applications ont généralement le rôle de personnalisation des services à utiliser et la facilitation dans le choix des services que l'utilisateur souhaite exploiter pour construire des workflows de données.

#### 3.3.2.3 La collaboration BioMoby [211]- $^{my}$GRID

BioMoby est un projet open source qui vise à mettre en place une architecture pour la découverte et la distribution des données biologiques à travers des services web. Afin de faciliter les interactions entres ces différentes ressources centralisées et distribuées, elles sont toutes enregistrées dans le MOBY central. Le MOBY central est une espèce de registre comportant des métadonnées décrivant les différentes ressources utilisées dans le projet BioMoby. Il détient un ensemble d'ontologies décrivant les différents types de données biologiques, les

différents formats de ressources utilisés ainsi que les différentes méthodes d'analyse de données utilisées. La sémantique utilisée dans ces ontologies permet de s'assurer que seuls les fournisseurs de services pertinents sont identifiés et garantit que les données utilisées peuvent être transmis à un prestataire de service. L'interaction entre un consommateur et un fournisseur de services peut être partiellement ou entièrement automatisée. BioMoby ajoute le paradigme des services web, en utilisant UDDI et en occurrence UDDI-M de $^{my}$GRID en exploitant les fonctionnalités proposées par ce registre en terme d'interrogation de données sémantiques relatives aux services web.

#### 3.3.2.4 Le registre GRIMOIRES (Grid RegIstry with Metadata Oriented Interface) [144]

Le registre GRIMOIRES est un prototype de registre de services UDDI développé dans le cadre du projet $^{my}$GRID et qui permet la sauvegarde des métadonnées sémantiques basée sur le système de découverte Feta [110]. Il représente un hôte de services et de workflows qu'un scientifique pourra utiliser pour créer des expériences scientifiques complexes. Dans le registre GRIMOIRES, les métadonnées peuvent être attachées aux entités définies dans UDDI et dans les WSDL sous forme de Triplet RDF. Il peut aussi fonctionner comme un registre UDDI simple au cas ou il n'y a pas des métadonnées à ajouter ou réutiliser.

#### 3.3.2.5 Le registre de bioinformatique BioRegistry [116]

BioRegistry est un projet qui vise à fournir des moyens pour représenter et exploiter les connaissances associées à des bases de données biologiques.
Il est implémenté comme un catalogue pour les bases de données biologiques qui facilite la recherche et la découverte de ressources utilisant un nouveau système d'interrogation et de sélection. Son système permet une recherche de données selon les besoins des utilisateurs en proposant non seulement des outils plus sophistiqués pour la recherche de données mais aussi en indiquant durant une recherche le critère de recherche le plus important par rapport à la demande de l'utilisateur. Il permet aussi la constitution automatique d'un catalogue des bases de données générées automatiquement à partir des modèles et d'extraction.

### 3.3.3 Synthèse

Comparés au registre standard UDDI, les registres développés spécifiquement en biomédecine et bioinformatique apportent quelques extensions. Nous avons vu que l'extension de UDDI pour la E-science donne naissance à de nouveaux outils et défis. Un premier défi concerne la capacité des outils déjà existants à répondre à des problèmes scientifiques d'ordre fonctionnel et opérationnel comme leur réutilisation dans un environnement distribué. Le second défi est de pouvoir réutiliser des données issues de l'exécution de ces services, de pouvoir les visualiser, les comparer et les analyser dans le cadre d'autres études.
Lorsqu'ils sont relevés, ces défis nous permettront de prendre en considération d'autres outils de traitement de données mais par contre ne permettront pas la découverte de services et le raisonnement sur les données utilisées ou produites. L'ajout de fonctionnalités pour relever ces défis aidera à son tour l'utilisateur à analyser, à distinguer et à mieux utiliser ses données

et ses outils.
En fait cet aspect ne peut être assuré que si on associe aux services une description sémantique riche, qui permettra aux utilisateurs de mieux comprendre d'une part le fonctionnement des outils (ce qu'ils font, quelles données ils peuvent traiter) et d'autre part de rendre intelligible les données issues de ces traitements ce qui est indispensable pour permettre leur utilisation et réutilisation éventuelle.

Le prochain chapitre présente les outils/plateformes de partage de services qui permettent à l'utilisateur d'ajouter la sémantique nécessaire pour la composition des services. Nous analyserons leurs capacités, limites et nous ferons le point par rapport aux objectifs de la thèse.

## 3.4 Outils sémantiques pour la composition des services

La technologie des services web a constitué un potentiel énorme pour le développement des applications dynamiques. Elle a amélioré d'une manière significative la communication des applications et leur intégration dans le cadre des plateformes logicielles interopérables. Trop souvent ce type d'intégration se limite à un processus de composition de services web qui permet aux clients de définir le processus métier qu'ils veulent appliquer. Le processus de composition offre généralement un moyen de coordination de ces entités en répondant à des exigences relatives au processus métier ; on parle alors de composition au niveau du processus métier. D'autres exigences du domaine d'application relèvent du niveau fonctionnel et on parle alors de processus de composition fonctionnelle. Ce dernier suscite généralement l'intérêt des communautés de recherche.

Comme point culminant, la composition fonctionnelle (Functional Level Composition) permet la génération d'une séquence de services web en fonction de leurs profiles et objectifs et en fonction des interactions entre les différents services atomiques selon leurs inputs/outputs ainsi que de leurs conditions et effets. Dans un tel contexte, l'objectif de composition est défini comme l'ensemble de fonctionnalités que le service composé devrait mettre en œuvre, en fonction des entrées, des sorties, des conditions préalables, et des effets (à savoir, le niveau fonctionnel). Parmi les techniques qui utilisent les FLC on trouve le Matchmaking combiné avec les dépendances sémantiques [39] et le AI planning (Artificial Intelligence planning) basées sur la composition de services [178].

La composition axée sur les PLC (Process Level Composition) met en place généralement un moteur permettant la gestion du processus de composition et s'occupe des types de liaisons d'échange de messages et de protocoles à utiliser tout le long de ce processus. Dans [159] et [158] la composition de services étudiée est basée sur les objectifs globaux en expliquant le comportement complexe d'un outil composé de plusieurs services web visant un objectif prédéterminé. La plupart des techniques de composition de services web prennent en entrée i) un ensemble de règles, ii) des services partiellement spécifiés ou iii) des contraintes sur le comportement du processus. Dans sa définition la plus simple, la composition de services permet de lier des services web entr'eux d'une façon organisée. La façon dont on organise les enchaînements reflète un besoin fonctionnel qui permet de créer de nouvelles fonctionnalités en combinant les fonctionnalités offertes par des services existants.

**Types de composition de services :**

- **Composition manuelle :** La composition manuelle des services web comme définie par les concepteurs de services composites ou de workflows, consiste à analyser manuellement la tâche à réaliser et à décider de l'enchaînement des différents services pour que le traitement offert par l'ensemble des services atteigne les objectifs souhaités.
- **Composition automatique :** Afin de réaliser une composition automatique de service, les services doivent d'abord être décrits sémantiquement de telle sorte que la description puisse être manipulée par la machine. Ensuite, des technique de planification automatique doivent être utilisées pour spécifier l'enchaînement des services compte tenu des objectifs qu'on souhaite attendre. Ces objectifs sont à leur tour décrits sémantiquement.

Dans les sections suivantes de ce chapitre on détaille les outils sémantiques de composition de services et on souligne les apports majeurs de ces techniques dans le domaine de la biomédecine.

### 3.4.1 Service web sémantiques

Un enjeu important du web sémantique est le déploiement des services web sémantiques. Comme indiqué précédemment, une lacune évidente des services web conventionnels réside dans leur description qui se limite à niveau syntaxique et s'avère limitée en particulier lors de la composition ou pour la découverte des services. L'ajout d'annotations sémantiques exprimées dans des langages comme OWL, RDF etc permet de mettre en place des services web sémantiquement transparents, c'est à dire qui peuvent être découverts ou utilisés avec succès par des clients, sans négociations préalables entre les développeurs de clients et de services [28]. La plupart des mécanismes proposés s'appuient sur l'extension des descriptions actuelles des services web avec des descriptions exprimées selon un formalisme logique et expliquant le sens des services web fournis en se référant à des connaissances exprimées dans des ontologies formelles.
Dans ce qui suit, nous introduisons quelques outils issus du web sémantique tels que OWL-S [42], WSMO [170], et SAWSDL [43]. Enfin, nous nous concentrons sur OWL-S qui est un formalisme que nous avons utilisé dans les travaux de cette thèse.

### 3.4.2 OWL-S : Ontology web Language for Services

#### 3.4.2.1 Description

OWL-S est l'abréviation de Ontology Web Langage for Services. Comme l'indique cet acronyme, OWL-S est une ontologie OWL construite spécialement pour les services web. Elle a été développée principalement par le projet DARPA Agent Markup Language [77]. Les versions antérieures étaient désignées sous le nom de DAML-S et étaient basées sur DAML+OIL au lieu de OWL. Le but de OWL-S est de fournir une description sémantique interprétable par les machines afin qu'ils puissent être déployés, sélectionnés, réutilisés, composé et contrôlés

42. http://www.w3.org/Submission/OWL-S/
43. http://www.w3.org/TR/sawsdl/

automatiquement à travers le web. OWL-S est une recommandation du W3C. Les participants du groupe de travail [117] donnent trois exemples de ce qui est attendu de OWL-S :

Il s'agit tout d'abord de la **découverte automatique de services web**, permettant à un processus automatisé la localisation des services web qui fournissent une classe particulière de service. Avec la description sémantique fournie par OWL-S, les informations nécessaires pour la découverte de services web peuvent être spécifiées sous la forme de balises sémantiques qui viennent enrichir un registre de service spécifique agissant comme un entrepôt de service ou une ontologie améliorée qui facilite et automatise la localisation des services web en l'interrogeant à l'aide d'un moteur de recherche web.

**La description automatique de services web** facilite ensuite l'invocation automatique d'un service web par un programme informatique ou un agent intelligent ; en effet, le balisage proposé par OWL-S fournit une description riche des différents composants du service. Le groupe de travail qui est à l'origine de OWL-S propose une API qui comprend la sémantique des arguments lors de l'envoi de paramètres pour l'invocation d'un service donné ou lors de la récupération des valeurs des paramètres après l'invocation du service.

Il s'agit enfin de **la composition automatique de services web** et de l'interopérabilité qu'impliquent la sélection automatique, la composition, et l'inter-fonctionnement des services web pour accomplir une tâche complexe qui vise un objectif bien déterminé. La plupart du temps cet objectif est exprimé en OWL ou OWL-S, pour décrire sémantiquement un objectif à un haut niveau de description. Actuellement, l'utilisateur doit sélectionner les services web et spécifier manuellement leur composition, et s'assurer que les logiciels nécessaires pour l'inter-fonctionnement des services sont bien mis en place. Avec la spécification OWL-S, les informations nécessaires pour sélectionner et composer les services mis en jeu pour l'exécution d'un service web seront sauvegardées sur les sites client qui ont fait appel au service web. Pour réaliser la composition automatique, des logiciels dédiés permettent la gestion des spécifications OWL-S, et qui permettent aussi la gestion des objectifs et des tâches en se basant sur d'autre logiciels (comme des outils de raisonnement et de gestion de données sémantiques). Pour cela, OWL-S fournit des spécifications qui permettent de spécifier des contraintes et des mécanismes (conditions préalables et séquences d'exécutions de service). Cette spécification se base sur un langage qui décrit la composition de services et les interactions mises en jeu via des flux de données échangés.

**Composants**

Lorsqu'elle a été conçue, l'ontologie OWL-S a essayé de répondre aux besoins de l'utilisateur et aux différents contextes d'utilisation. Ceci a conduit à une structuration spécifique basée sur trois questions fondamentales :

1. Qu'est-ce que le service doit fournir aux clients potentiels ? La réponse à cette question est donnée par l'ontologie spécifique "Profile" (ServiceProfile), qui est utilisée pour décrire les fonctionnalités du service.
2. Comment est-il utilisé ? La réponse à cette question est donnée dans l'ontlologie "Process" (ServiceModel).
3. Comment peut-on l'invoquer et l'utiliser à partir de son serveur ? La réponse à cette question est donnée dans l'ontologie "Grounding".

La figure ci-dessous donne une illustration de la structure de l'ontologie OWL-S:

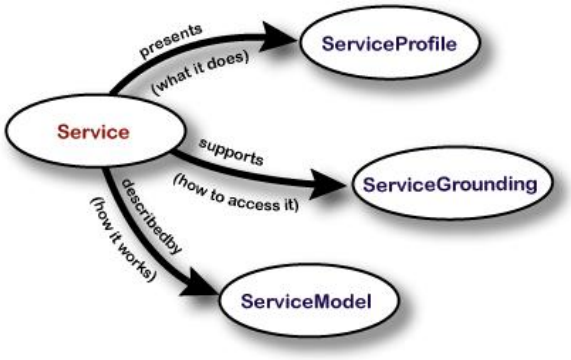


FIGURE 3.13 – Les différentes sous couches de l'ontologie de haut niveau : Ontologie de services de OWL-S [118]

**Service :** L'entité Service est considérée comme le point de départ de la publication d'un service. Elle présente une instance unique de ce service décrite par trois autres ontologies comme indiqué dans la figure précédente 3.13. Les détails des trois ontologies qui caractérisent un service OWL-S sont décrits dans le paragraphe suivant.

**Service Profile :** Le Service Profile décrit ce que le service fait en termes de capacités. Il permet aussi de savoir quels sont les meilleurs fournisseurs de ce service. La description fonctionnelle du service est exprimée en termes de transformation produite par le service. Plus précisément, elle décrit les entrées et les sorties du service. Le profil décrit également les conditions préalables exigées par le service et les effets attendus qui résultent de l'exécution du service. C'est ce que nous appelons I.O.P.E. (entrées, sorties, conditions et effets). Enfin, le profil de service peut décrire les exigences non-fonctionnelles.

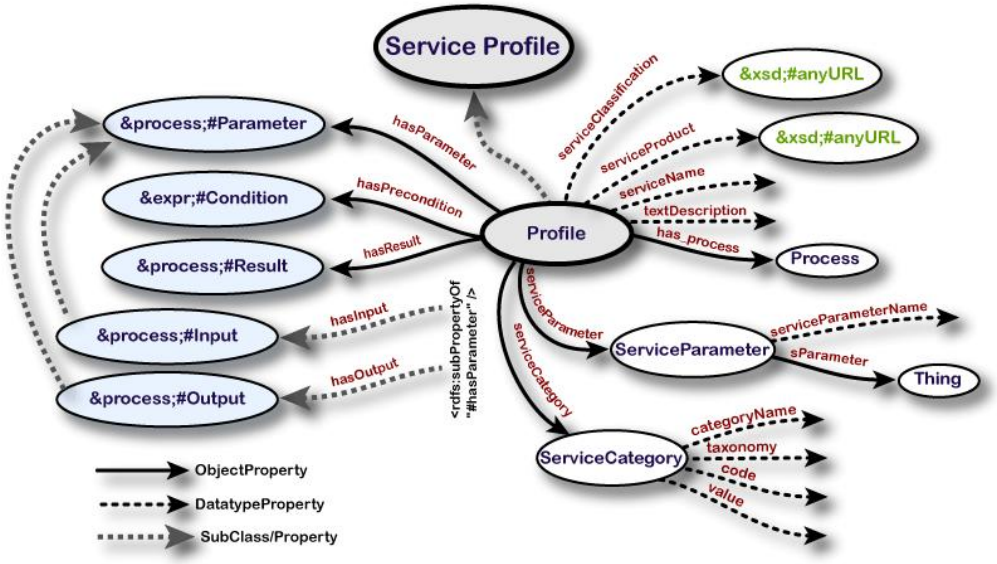


FIGURE 3.14 – L'ontologie service profile dans la spécification OWL-S [118]

Le Service Profile fournit une description concise du service à un utilisateur ou registre donné, mais une fois que le service a été sélectionné le profil n'intervient plus dans les étapes ultérieures comme l'invocation. Le client aura alors à utiliser le Service Model pour gérer le fonctionnement et le contrôle du service avec les autres logiciels ou services.

Le ProcessModel spécifie (a) le fonctionnement du service et (b) comment les clients peuvent interagir avec lui en définissant un protocole d'interaction demandeur-fournisseur. Le Process Model OWL-S se rapporte principalement à la description des fournisseurs d'un tel service. Cependant, la description peut être aussi utilisée pour décrire les demandeurs de ce service.

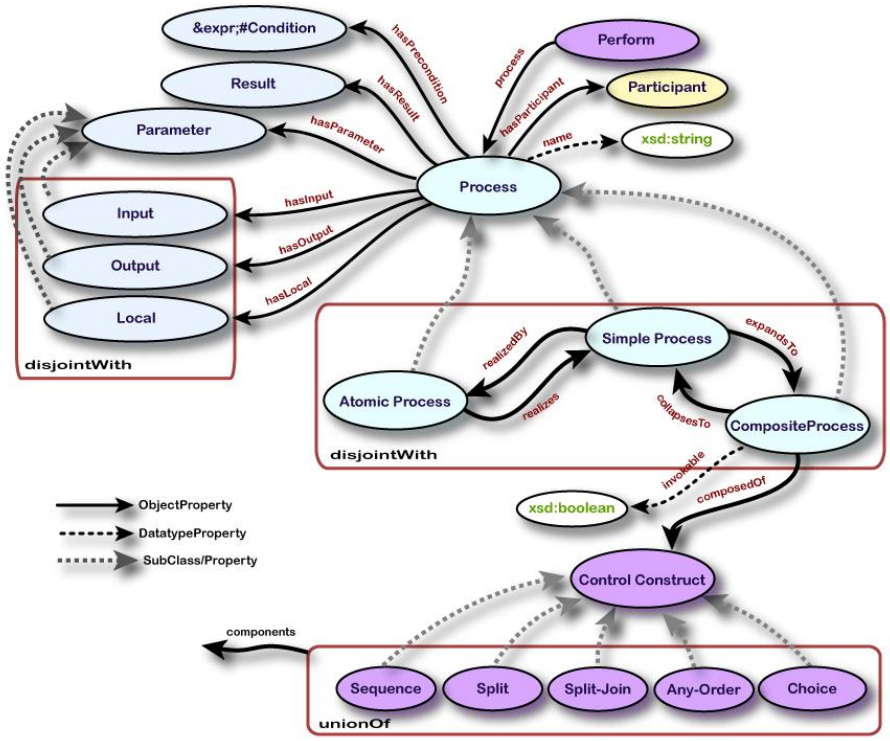


FIGURE 3.15 – L'ontologie Process dans la spécification OWL-S [118]

**Service Model :**Le Service Model, appelé également modèle de processus, décrit comment le service fonctionne et comment un client peut interagir avec le service. Un service peut avoir n'importe quel nombre d'entrées (y compris aucune), représentant les paramètres nécessaires à l'invocation du service. Il peut avoir aussi un certain nombre de sorties. Les entrées et sorties du service peuvent être sous forme de fichiers ou d'informations ou de paramètres nécessaires pour la manipulation des données de sorties. Il peut avoir un nombre quelconque de conditions préalables, qui doivent toutes être satisfaites et appliquées dans l'ordre pour que le/les services soient invoqués avec succès. Enfin, le service peut avoir n'importe quel nombre d'effets qui ajoutent de l'information aux sorties de ce service. Les sorties et les effets peuvent dépendre les uns des autres, par exemple des conditions qui valident si un service a bien été invoqué ou si un service a été invoqué avec succès ou pas. Le ServiceModel établit une distinction entre les processus atomiques, simples et composites.

- Les processus atomiques sont similaires à un service web classique utilisé dans Java ou .NET. Ils sont directement invocables, n'ont pas de sous-processus et sont composés d'une seule étape exécutée une seule fois. Ils prennent un certain nombre d'entrées sous forme de message/ Flux XML. Lors de leur exécution ils décortiquent et utilisent le message XML puis retournent en sortie un paramètre/sortie encapsulé sous forme d'un flux XML. Un processus atomique est généralement relatif à un service qui est prêt à être exécuté et qui comporte une description non sémantique de base, par exemple en WSDL. Il faut donc pour chaque processus atomique définit, une liaison entre sa spécification sémantique et sa spécification fonctionnelle.
- Les processus simples fournissent un mécanisme abstrait pour fournir plusieurs vues pour un processus donné. Ils ne sont pas invocables et n'ont pas donc de liaison avec une spécification de base. Ils sont utilisés comme des éléments abstraits. Ils sont souvent utilisés pour simplifier les processus composites pour des finalités de planification et de raisonnement.
- OWL-S permet la combinaison des processus simples en processus composites d'une manière récursive ou chaînée à l'aide de structures de contrôle. La version 1.2 prend en charge les structures de contrôle suivante: Séquence, Any-Order, Choice, If-then-else, Split, Split-Join, Repeat-then, Repeat-Until.

**Service Grounding :** Le Service Grounding fournit les détails qui spécifient comment un utilisateur peut accéder à un service en spécifiant un protocole de communication, les paramètres à utiliser dans le protocole et les techniques de sérialisation. Le Service Profile et le Service Model sont considérés comme des représentations abstraites alors que Service Grounding est une représentation concrète. Le Service Grounding fait le mapping direct des entrées/sorties du service atomique avec leurs valeurs concrètes (message sous forme de flux XML, fichier ou autres). Ces données sont trop souvent représentées dans un format spécial sérialisable et transmissible. L'implémentation la plus connue du modèle OWL-S a été faite dans un domaine spécifique et concret standardisé par le W3C à savoir les WSDL. La spécification du service Grounding est très similaire à la WSDL car la réflexion initiale a été faite par rapport aux WSDLs. Ceci est illustré dans la figure 3.16.

Comme la figure 3.16 l'indique, les deux languages se chevauchent. Les WSDLs décrivent l'opération abstraite ainsi que les types et les messages à l'aide d'un schéma XML. La spécification OWL-S utilise des classes OWL pour définir des types abstraits (pour les processus atomiques et les entrées-sorties). WSDL/XML Schema ne peuvent pas exprimer la sémantique d'une classe OWL, et OWL-S ne peut pas exprimer les informations de base pour un service web. Il est donc assez naturel d'utiliser OWL-S pour définir des types sémantiquement plus riches qui aideront à la compréhension des fonctions et des modes de fonctionnement des services, et des WSDLs pour définir les formats de messages.

#### 3.4.2.2 Utilisation de OWL-S

OWL-S est considéré comme la plateforme et la spécification la plus performante au niveau de la composition de services et de la description sémantique des services [117]. Plusieurs travaux ont tenté d'étendre OWL-S pour pallier aux manques relatif à différent domaines d'application. Dans le domaine de l'intelligence artificielle, par exemple, OWLS-XPlan [91]

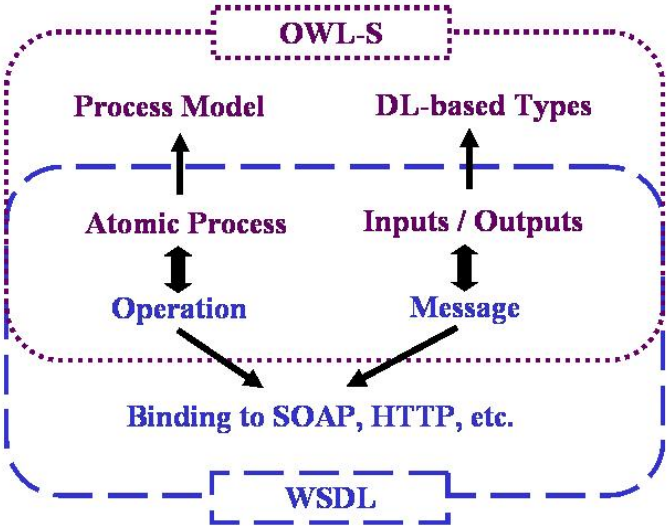


FIGURE 3.16 – Mapping entre les éléments des WSDLs et les éléments des process OWL-S via le Grounding [118]

représente un service de composition de services OWL-S selon une méthode de planification très connue. Les services web sont représenté comme des tâches de planification en décrivant les objectifs et les actions possibles. La composition de services OWL-S dans cette plateforme est basée sur les problèmes de planification [119] en leur ajoutant des définitions des types de données (entrée/sorties), des prédicats et des actions. OWLS-XPlan ne permet que l'enchaînement de services web en OWL-S de manière séquentielle. Les appariements des services web doivent être obligatoirement réalisés par des correspondances identiques ; le type de donnée doit être le même pour la sortie et pour l'entrée dans le cas d'un workflow. De plus, la plateforme ne fournit pas un environnement d'exécution de workflow [90].

**COCOA :** Une autre approche nommée **COCOA** (**CO**nversation-based service **CO**mposition middlew**A**re) a été introduite dans [132]. C'est une plateforme de composition de services web prenant en charge des comportements complexes à la fois des services et des tâches (les tâches sont les requêtes utilisateurs auxquelles le système doit répondre et sont généralement exprimées par des processus OWL-S). Afin de réaliser la composition, COCOA convertit les processus OWL-S en FSA (Finite State Automata). Ceci permet de transformer la composition de services en un problème d'analyse d'automates.

**IRS-III [74] :** IRS-III (Internet Reasoning Service, version III) est une plateforme de composition de services web sémantiques. Afin d'implémenter ce type d'application, IRS-III propose quatre étapes : l'annotation sémantique des services web, la recherche et la sélection des services, et leur composition. L'architecture de IRS-III est composée de cinq composants mettant en œuvre la composition de services web sémantiques. La bibliothèque de services web sémantiques (SWS Library), l'interpréteur de chorégraphie (Choreography Interpreter) en WSMO et l'interpréteur d'orchestration ou process des services web décrit initialement en OWL-S puis traduit en orchestration.

**SHOP2 [179] :** La plate-forme SHOP2 appartient au projet SHOP [44] de l'Université du Maryland. Le premier objectif de ce projet est de fournir une plate-forme de planification hiérarchique. Le concept de base de SHOP2 [179] repose sur le fait que cette plate-forme est basée sur les techniques du domaine de l'intelligence artificielle, et plus spécifiquement sur le concept de réseau de tâche hiérarchique (Hierarchical Task Network ou HTN) qui est une méthode de planification qui crée des plans par la décomposition des tâches. SHOP2 distingue trois types de tâches : les tâches de but, qui déterminent les propriétés pour atteindre l'état final ; les tâches primitives, qui peuvent être directement invoquées et les tâches composées, qui dénotent les changements désirés en faisant référence à d'autres tâches (de but ou primitives).

La définition des processus peut se faire de deux manières différentes, soit l'utilisateur de la plate-forme définit le processus par l'intermédiaire d'un HTN, soit la description du processus existe déjà en OWL-S. Dans le second cas, le module OWL-S to SHOP2 Translator se charge de la traduction en HTN. Le module SHOP2 to OWL-S Plan Converter convertit un processus défini dans la syntaxe de la plate-forme (HTN) en une ontologie OWL conforme à la spécification de OWL-S.
Pour pouvoir exécuter les services web de cette plateforme, ces derniers doivent obligatoirement posséder une description OWL-S. À partir de la description du processus, l'Execution Monitoring System fait appel aux services web. Ce module garde en cache les résultats des services appelés pour les réutiliser en cas de besoin. Ceci permet une plus grande rapidité de l'exécution de la composition.

**Autres ré-utilisations de OWL-S :** Malgré la vision sémantique très large d'Evren Sirin [45] pour créer une plateforme de composition de tout type de service web, l'implémentation de la spécification OWL-S se limite à des WSDLs ce qui fait qu'aujourd'hui la spécification OWL-S est destinée à être universellement applicable à la représentation sémantique de services web traditionnels [68]. Cependant OWL-S a été utilisé dans plusieurs domaines y compris la biomédecine [42] [22] [33]. OWL-S a été utilisé pour la modélisation des tâches médicales administratives au sein des hôpitaux [42], la modélisation des processus de gestion des données au sein des assurances médicales en l'occurrence la gestion des dossiers des patients des des médicaments et des règlements ou remboursements. A priori l'ontologie OWL-S a été étendue avec une ontologie de domaine pour répondre à des besoins fonctionnels d'un système de gestion de workflow destiné à permettre aux utilisateurs (médecins et personnel administratifs) de gérer, de créer des flux de données compatibles avec de nouveaux flux de données médicaux et de les exécuter à la volée. Cette ontologie facilite la gestion personnalisée des soins de santé en rassemblant toutes les connaissances nécessaires pour un scénario de soins de santé personnalisés complexe impliquant les soins aux patients, les polices d'assurance, et la prescription de médicaments.

Dans [33] ils utilisent le langage OWL-S pour décrire leurs services web ainsi que OWL pour leur ontologie du domaine et le langage de règles SWRL pour ajouter des contraintes sur les entrées et les sorties des services web de façon à améliorer leur composition. Ceci est appliqué au domaine des bonnes pratiques cliniques. L'article met l'accent sur la valeur

44. Simple Hierarchical Ordered Planner (http://www.cs.umd.edu/projects/shop/)
45. Premier contributeur dans la réalisation de l'API OWL-S. Page personnelle: http://www.mindswap.org/ evren/

ajoutée apportée par l'enrichissement sémantique des services web, notamment pour la prise de décision lors d'un acte clinique ou opératoire. Un exemple illustratif concerne la prise en charge clinique de la rétinopathie diabétique, où les utilisateurs finaux sont des professionnels de santé qui ne sont pas familiarisés avec les technologies du web sémantique.

La spécification OWL-S a été améliorée dans plusieurs travaux pour donner naissance à d'autres outils mieux adaptés à d'autres domaines d'application, comme la spécification d'un éditeur standard au langage de description de services OWL-S permettant de générer des descriptions sémantiques de services web générées et faits selon les spécifications de bases des services web et basé sur l'API OWL-S et protégé [53]. Elle a été aussi alignée sur une ontologie fondationnelle [128] pour satisfaire des besoins dans le domaine de la biomédecine. Elle a été utilisée également dans d'autres domaines, par exemple les technologies des grilles de calcul [81] [20].

La maniabilité de la spécification OWL-S tient au fait qu'elle soit basée sur OWL qui est un langage puissant largement utilisé dans le web sémantique [53] et tout particulièrement en biomédecine et bioinformatique. D'autres domaines ont naturellement exploité les particularités de OWL-S, notamment pour la vérification de la composition automatique et la découverte de services. Plusieurs réflexions ont finalement mené à adapter OWL-S à certains outils en faisant une correspondance entre les concepts (Profile, Composite Process, simple Process) comme par exemple dans [177] [2], ou les concepts de BPEL4WS ou Taverna dans le domaine de bioinformatique [100].

### 3.4.3 WSMO : Web Service Modeling Ontology

#### 3.4.3.1 Description

Web Service Modeling Ontology (WSMO) [46] est une spécification similaire à OWL-S. C'est l'une des contributions majeures dans le web sémantique pour permettre la composition, la découverte automatique et l'exécution d'une chaîne de services web. WSMO est basé sur le Web Service Modeling Framework (WSMF). WSMO distingue les ontologies, les services web, les objectifs et les médiateurs, comme le montre la figure 3.17.

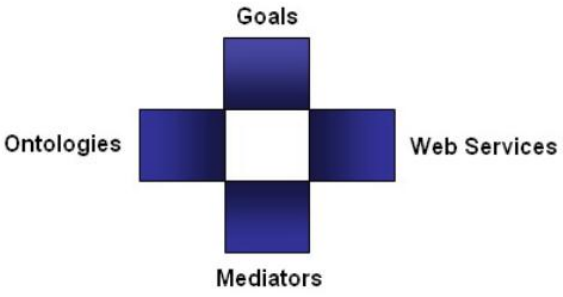


FIGURE 3.17 – Les quatre éléments de base de WSMO [27]

**Ontologies:** ils fournissent toute la terminologie et la sémantique formelle nécessaire compréhensible par une machine et permettant de décrire les informations utilisées par les

46. http://www.w3.org/Submission/WSMO/

utilisateurs concernant les services web.

**Goals:** ces éléments fournissent les moyens de préciser les objectifs des demandeurs de services lors de la consultation d'un service web, décrivant à un haut niveau d'une tâche concrète à accomplir.

**Web Services:** ils fournissent une description sémantique des services web, y compris leurs propriétés fonctionnelles et non fonctionnelles, ainsi que d'autres aspects pertinents pour leurs interactions.

**Mediators:** ce sont des éléments de modélisation qui assurent la fonction de connecteur et qui permettent de résoudre les problèmes d'hétérogénéité et donc d'assurer l'interopérabilité entre des composants hétérogènes.

Parmi les concepts les plus intéressants dans WSMO nous citons : (i) l'orchestration et (ii) la chorégraphie. Actuellement, il existe des initiatives concurrentes pour la modélisation des processus métier en tant que services composites, qui visent à gérer les activités des processus métier et des protocoles d'interaction métiers comprenant des services collaborant à la réalisation des processus métiers. Les termes orchestration et chorégraphie ont été largement utilisés pour décrire les protocoles d'interaction de la couche métier.

Dans ce qui suit nous allons brièvement examiner ces spécifications de niveau supérieur pour la composition des services. Il faut toutefois noter que la distinction entre orchestration et chorégraphie est difficile [149], ces deux concepts étant proches d'un point de vue fonctionnel.

- **L'orchestration :** décrit comment les services interagissent entr'eux au niveau de l'échange de messages. Ceci recouvre la logique métier et l'ordre d'exécution des différentes entités en interaction d'un point de vue contrôle. L'orchestration se réfère à un processus métier exécutable qui peut avoir une longue durée de vie de nature transactionnelle et mettent en jeux plusieurs transactions et différents accords entre les entités. Avec l'orchestration, le processus métier et les interactions sont toujours contrôlés du point de vue échange de message et par rapport aux composantes métier impliqués dans le déroulement du processus. Il est évident que la modélisation des processus métier et du processus de composition au niveau fonctionnel peuvent faire partie de l'orchestration. Les structures de contrôle souvent présentées sous forme de balises sont utilisées lors de la composition au niveau fonctionnel (FLC) (comme par exemple, les constructions de base liés à la séquence, l'exécution simultanée). Elles sont en grande partie intégrées dans l'orchestration. D'autre part l'orchestration au niveau des messages échangés est vraiment basée sur la composition au niveau process (PLC). Actuellement, les langages les plus sophistiqués pour modéliser l'orchestration sont des langages basés sur le standard XML ; le plus connue d'eux est le Business Process Execution Language for services web (BPEL4WS) [5].
- **La chorégraphie :** est généralement associée à des échanges de messages, aux règles de l'interaction et aux accords qui se produisent entre plusieurs points de terminaison des processus métiers, plutôt qu'un processus métier spécifique qui est exécuté par un seul tiers. La chorégraphie recouvre une collaboration plus vaste que l'orchestration. Elle est décrite d'un point de vue plus large (commun à tous les contributeurs), et définit le comportement des participants au sein de la collaboration lors de la réalisation d'un processus métier. La chorégraphie suit les séquences de messages qui peuvent

impliquer plusieurs parties et sources multiples, y compris les clients, les fournisseurs et partenaires.
La modélisation des processus métier mise en jeu dans une composition peut être réalisée par une chorégraphie. En effet, la chorégraphie de services est surtout intéressante dans les messages échangés et lors du contrôle du comportement des différents acteurs intervenant dans une composition, ce qui facilite la modélisation au niveau processus (PLC). Contrairement à PLC, FLC n'est pas approprié à l'approche basée sur le modèle de la chorégraphie, car le niveau de la composition se base sur les interactions atomiques. La chorégraphie est généralement modélisée à l'aide de Web Services Choreography Description Language (WSCDL) [47], qui précise les comportements de tous les participants engagés dans la collaboration lors de la création ou l'exécution des processus métier.

#### 3.4.3.2 Utilisation de WSMO

Considéré comme une des plateformes les plus complètes du web sémantique dans le domaine de composition de services, WSMO a été utilisé dans de nombreuses contributions décrites dans les sections qui suivent.

#### 3.4.3.3 IRS-III

IRS-III (Internet Reasoning Service, version III) est une plateforme que nous avons déjà décrite dans un paragraphe précédent (ré-utilisation de OWL-S). On retrouve cette plateforme dans cette section parce qu'elle s'inspire également de WSMO. En effet, l'architecture de IRS-III est composée de cinq composants mettant en œuvre la composition de services web sémantiques voir figure 3.18 : une bibliothèque de services web sémantiques (SWS Library), un interpréteur de chorégraphie (Choreography Interpreter), un interpréteur d'orchestration (Orchestration Interpreter), un gestionnaire de médiation (Mediation Handler) et un module d'invocation (Invoker).

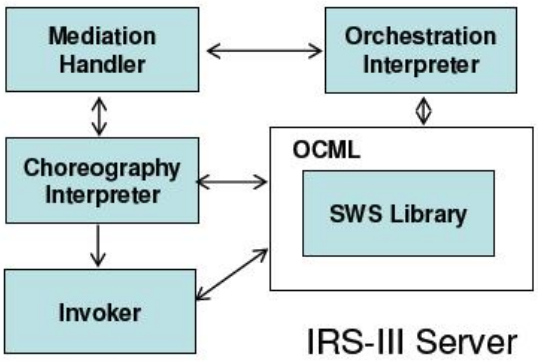


FIGURE 3.18 – Architecture de IRS-III [109]

47. http://www.w3.org/TR/ws-cdl-10/

- **La bibliothèque de services web sémantiques (SWS Library)**: Au cœur du serveur IRS-III, cette bibliothèque rassemble les descriptions sémantiques utilisées dans le cadre de la plateforme. Ces descriptions sont formalisées en OCML (Operational Conceptual Modeling Language, langage de représentation sémantique issu des travaux de l'institut KMI (Knowledge Media Institute) [136]. La bibliothèque est structurée en modèles de connaissances des objectifs, des services Web et des médiateurs. Elle rassemble aussi toutes les ontologies de domaine ainsi que la base de connaissances (toutes les instances). Ces modèles de connaissances sont utilisés afin de mettre en œuvre la résolution de problèmes, pour répondre à la requête de l'utilisateur.
- **L'interpréteur de chorégraphie (Choreography Interpreter):** Le rôle de la chorégraphie est de décrire les interactions entre la plateforme et un service web élémentaire et aussi entre les services web élémentaires [29] [47] [48]. Le composant Choreography Interpreter interprète le grounding et surveille les transitions lors de la description de la chorégraphie. Pour cela la chorégraphie est représentée, d'une part, par un ensemble de règles de chaînage avant (forward-chaining rule), et, d'autre part, par une déclaration d'accès au service exprimée en OCML. Une règle exécute des interactions entre la plate-forme et le service web si les conditions associées sont satisfaites. Ces interactions sont basées sur les primitives de communication suivantes : début de chorégraphie (init-choreography), envoi de message (send-message), réception de message (receive-message), réception d'erreur (received-error), et fin de chorégraphie (end-choreography).
- **L'interpréteur d'orchestration (Orchestration Interpreter):** Le rôle de ce composant est d'interpréter le workflow de la description de l'orchestration à la demande du gestionnaire de médiation. En effet, l'orchestration IRS-III est représentée par un modèle de workflow exprimé en OCML. La particularité d'utiliser OCML pour représenter un workflow est que l'unité de base d'une composition est un objectif. Le modèle fournit des constructions de flots de contrôle et de flots de données au-dessus de l'ensemble des objectifs. Les primitives de flots de contrôle disponibles dans IRS-III sont les suivantes : la liste des objectifs à invoquer de manière séquentielle (orch-sequence), une condition (orch-if), la répétition (orch-repeat), le résultat de la dernière invocation de l'objectif déclaré (orch-get-goal-value) et le résultat de l'exécution de l'objectif courant (orch-return).
- **Le gestionnaire de médiation (Mediation Handler)** :
  Les activités de courtage d'IRS-III y compris la sélection, la composition et l'invocation sont supportées par un composant spécifique de médiation au sein du gestionnaire de la médiation. Ces activités peuvent se rapporter à l'exécution d'un service de médiation à des règles de correspondance déclarées dans une description de médiateur. Ce composant associe à chaque objectif (représenté par une ontologie) un service web. Cette association est permise par des règles de correspondance (mapping rule). La médiation intervient aussi bien lors de l'exécution de la chorégraphie que lors de l'exécution de l'orchestration dans le but de sélectionner, d'invoquer et d'exécuter les services web adéquats.
- **Le module d'invocation (Invoker)**: Ce composant est l'interface de communication entre l'IRS-III et l'application réalisée à la suite de la requête du client. Il reçoit les entrées envoyées par le client et retourne les résultats de l'invocation au client.

#### 3.4.3.4 SOPHIE:

La plateforme SOPHIE (Semantic services web chOreograPHy servIcE), est une plateforme qui utilise en partie WSMO et les concepts de chorégraphie décrit dans WSMO [10]. Elle met en place un moyen pour surmonter l'hétérogénéité de la sémantique utilisée, des cardinalités et des formats et des structures utilisés par les services en interaction. Elle propose des solutions au manque d'indépendance technologique entre les plateformes à l'absence de modèles clairs qui séparent l'aspect structurel, comportemental et opérationnel, et qui permettent de surmonter l'hétérogénéité lors de l'échange de messages. Par exemple, les messages peuvent être envoyés / reçus dans un ordre différent de celui attendu, ou non-conforme au comportement attendu de l'autre partie (séquence et inadéquation des cardinalités), la structure ou le format des messages échangés peuvent être aussi incompatibles (incompatibilité structurelle), ou utiliser différentes conventions terminologiques pour représenter des concepts codés (incompatibilité sémantique) [9]. SOPHIE introduit aussi l'alliance commerciale [48] qui permet de minimiser les coût (en temps et argent) de diffusion des informations entre partenaires et clients et de présenter d'une façon plus sophistiquée et meilleure les services aux clients. L'objectif commun de ces approches sémantiques pour les alliances dans la couche métier est d'assurer l'interopérabilité entre les services qui fournissent et consomment diverses informations en se référant à leurs propres structures de connaissance.

#### 3.4.3.5 WSMO-MX :

Récemment Kaufer et Klusch [91] ont développé un Matchmaker appelé WSMO-MX, qui est un entremetteur hybride pour les services WSMO qui emploie une approche de matching de graphes combinée avec une mesure de similarité syntaxique.

#### 3.4.3.6 INFRAwebS :

INFRAwebS est un projet européen visant à développer un ensemble d'outils logiciels d'application basés sur la création, le maintien et l'exécution de services web sémantiques (SWS) décrits selon le modèle WSMO. Cette nouvelle génération d'outils et de systèmes permet aux fournisseurs de logiciels et de services de construire des plateformes de développement ouvertes et extensibles pour les applications de services web. Ces services utilisent des outils standards issue du web sémantique du W3C, comme BPEL4WS, WSMO, WSMX, WSML, SPARQL, RDF. En particulier, ils sont compatibles avec WSMO et WSML.

Les systèmes sont constitués de petits services faiblement couplés produits par la plateforme INFRAwebS. Chacune de ces unités fournit des outils et des composants adaptés au système. L'architecture INFRAwebS repose sur deux étapes différentes : la conception et l'exécution. Certains des composants interviennent dans chacune des deux étapes, car ils offrent des fonctionnalités qui vont être partagées dans les deux étapes. Au stade de la concep-

48. L'alliance commerciale est très similaire aux entreprises virtuelles qui se forment de plus en plus sur le Web. L'alliance commerciale est considérée comme un concept de gestion et de chaînage de services (de sSvcCM) qui permet aux organisations d'améliorer la satisfaction des clients et de réduire les coûts opérationnels grâce à la prévision, la gestion intelligente, l'optimisation, la planification et l'ordonnancement du workflow.

tion INFRAwebS offre un ensemble complet d'outils pour concevoir, développer et composer des services web sémantiques. La plupart des composants appartenant à ce stade proposent une interface graphique. Les fonctionnalités offertes par INFRAwebS sont les suivantes :

1. enregistrement des fichiers WSDL et BPEL : insertion manuelle des fichiers WSDL, BPEL et d'annotation ainsi que leurs catégorisations en fonction des spécificités des ontologies de services de domaine.

2. Conception des services : pour décrire les caractéristiques fonctionnelles et non fonctionnelles (QoS, SLA (Service Level Agreement, etc) d'un service web sémantique et assurer la conversion des fichiers WSDL existants en fichiers WSMO.

Lors de l'exécution, INFRAwebS offre une suite d'outils pour exécuter des services composites qui suivent un workflow orchestré, selon un processus collaboratif avec d'autres services. Lors de l'exécution d'un service composite, le moniteur QoS recueille et analyse les données d'exécution récupérées à partir du gestionnaire d'exécution. Les fonctionnalités offertes à ce niveau sont les suivantes:
**Découverte de service:** cette fonctionnalité fournit une liste des descriptions de services web sémantiques correspondant aux objectifs exprimant ce que l'utilisateur souhaite voir.
**Service de «testing»:**il permet l'exécution des tests unitaires et des tests de QoS.
**L'exécution SWS :** Elle permet l'exécution de services web sémantiques et par conséquent l'invocation du point de terminaison de service web.
**Surveillance de la qualité de service SWS:** Elle permet de surveiller l'exécution d'un service, de récupérer des informations de provenance, d'analyser ces informations et de générer une exception au cas où le contrôle révèle des incohérences.
**Le filtrage de sécurité :** évalue dynamiquement chaque SWS et agit comme un module intelligent de filtrage attachées aux applications infrawebs.

#### 3.4.3.7 BIOSENSE [21] :

Biosense est l'acronyme de (Semantic Enhancement for Sharing and Exploration in Bioinformatics). Il utilise WSMO et WSMX son environnement d'exécution, pour la composition, la découverte et l'exécution des services web en bioinformatique. Il est fondé sur trois couches différentes (i) couche de gestion de service (ii) couche ontologique et (iii) une couche de services web. La première est constituée de deux services qui aident à la composition des services web. La deuxième couche contient la description sémantique des services web à l'aide de WSMO et de l'ontologie de domaine. La dernière couche est la couche qui rassemble les différents services web et qui implémentent les fonctions d'accès aux bases de données. L'utilisateur construit son workflow et relie les entrées/sorties des services les unes aux autres. Le système BIOSENSE, lors de l'exécution du workflow vérifie la validité des paramètres d'entrées/sorties à l'aide des descriptions ontologiques.

### 3.4.4 METEOR-S

La plate-forme METEOR-S (Managing End-To-End OpeRations-Semantics) fait partie du projet METEOR-S [49] du laboratoire LSDIS [50] de l'Université de Georgia. Il s'agit d'une plateforme contenant un grand nombre de services web et gérant les processus de leur exécution. La mise en œuvre de la composition de services web avec la plate-forme METEOR-S [145] [1] repose sur quatre grandes phases illustrées dans la figure 3.19 : la conception du processus abstrait, la découverte des services, l'analyse de contraintes et l'exécution de la composition.

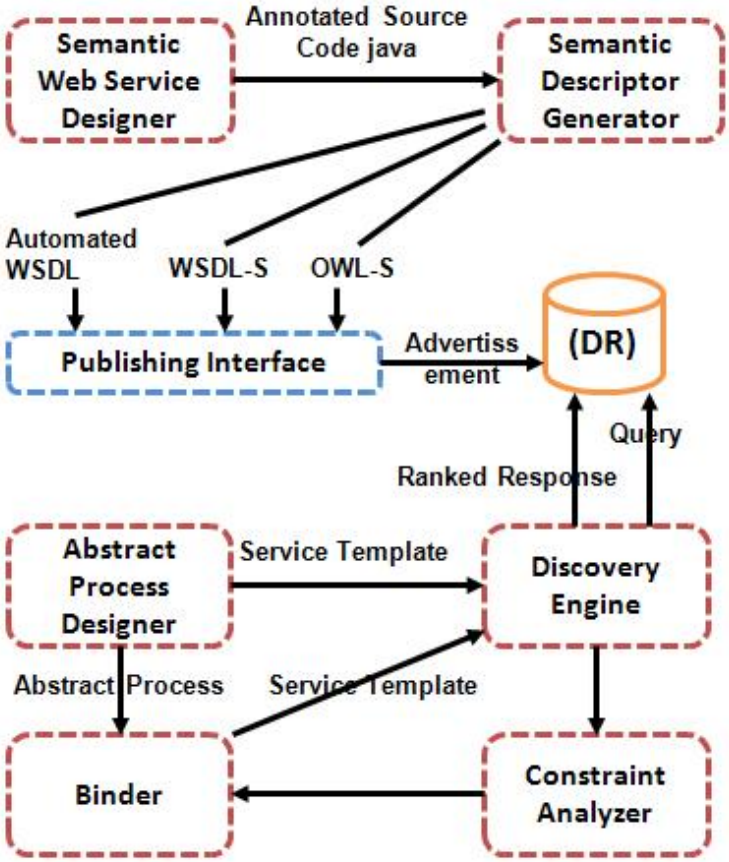


FIGURE 3.19 – Architecture de METEOR-S [162]

La composition de services web au sein de la plateforme METEOR-S repose sur un cadre commun appelé METEOR-S web service composition framework (MWSCF), fondé sur la notion de Semantic Process Template qui constituent des modèles sémantiques d'activités. Elle s'appuie sur quatre grandes composantes :

– Pour la description des flots de contrôle entre les services, la plate-forme utilise le langage de composition BPEL4WS. Ces templates contienent les pré-conditions, les

49. Projet METEOR-S: Semantic services web and Processes (http://lsdis.cs.uga.edu/projects/meteor-s/)
50. LSDIS : Large Scale Distributed Information Systems (http://lsdis.cs.uga.edu/)

contraintes (en termes de qualité de service, de localisation de serveur, etc.) et la sémantique du service. Le composant Abstract Process Designer transmet, d'une part, les Service Templates au moteur de découverte (Discovery Engine) afin de sélectionner les services, et, d'autre part, les processus abstraits (Abstract Process) au moteur d'exécution (Binder) pour orchestrer la composition.

- Le système de découverte de service fait appel au module Discovery Engine : en fonction du Service Template le moteur de découverte Discovery Engine transmet une requête au registre METEOR-S qui explore la possibilité d'ajouter de la sémantique aux WSDL et au UDDI [180]. Ce dernier retourne un ensemble de services correspondant au Template formalisé dans la requête lancée.
- Analyse de contraintes (réalisée par le composant Constraint Analyser): Le module Constraint Analyser filtre l'ensemble des services retournés par le registre UDDI, selon la faisabilité du processus (par exemple, en fonction des compatibilités entre les paramètres de sortie d'un service et les paramètres d'entrée du suivant) et son efficacité potentielle (par exemple, selon la localisation du serveur hébergeant les services).
- Exécution d'un enchaînement de services (réalisée par le composant Binder): Le module Binder permet la mise en œuvre de l'exécution de la composition de services à l'aide, d'une part, de la description du processus abstrait représenté en BPEL4WS, et, d'autre part, des services retenus par le module Constraint Analyser. Si le moteur d'exécution est confronté à un problème lors de l'exécution (tel qu'un service web non disponible), le moteur transmet les templates des services posant problème au composant Discovery Engine. Les étapes de découverte des services et d'analyse de contraintes sont alors répétées.

La sémantique est aussi gérée à l'aide d'une extension de WSDL pour les services web sémantiques (WSDL-S [162]) et du langage de description de services web sémantiques (OWL-S). Le premier langage permet d'utiliser l'approche par annotation sémantique alors que le second utilise un ensemble d'ontologies. Les descriptions sémantiques des services sont enregistrées dans une version améliorée de UDDI (interne à METEOR-S).

#### 3.4.4.1 Autre standards (WSDL-S et SA-WSDL)

Le standard WSDL des services web fonctionne à un niveau syntaxique simple car il dispose pas de la sémantique nécessaire pour représenter de façon signifiante les paramètres des services web. Ceci représente un problème récurrent pour les utilisateurs des services web. Pour apporter une première réponse à ce problème, le groupe de travail sur les annotations sémantiques du W3C pour WSDL et XML Schema (SAWSDL) a développé des mécanismes avec lesquels les annotations sémantiques peuvent être ajoutées à des composants WSDL.

Contrairement à OWL ou WSML, SAWSDL n'a pas été conçu pour remplacer une ontologie ou un langage d'ontologie mais par contre il prévoit des mécanismes par lesquels des concepts ontologiques définis en dehors des documents WSDL, peuvent être référencés pour annoter sémantiquement des description WSDL. SAWSDL s'appuie sur son prédécesseur WSDL-S [51] en cours de soumission au W3C. WSDL-S est donc un enrichissement sémantique

51. Créé dans METEOR-S:semantic web services and processes http://lsdis.cs.uga.edu/Projects/METEOR-S/

du standard WSDL, dans lequel les inputs et les outputs sont définis en OWL via l'ontologie RosettaNet et en XSD.
Les principes de conception fondamentaux pour SAWSDL sont les suivants :

a la spécification permet l'annotation sémantique de services web en utilisant et en s'appuyant sur l'extensibilité existante de WSDL

b il est agnostique à la sémantique (les ontologies et les langages de représentation)

c il permet d'ajouter des annotations sémantiques pour les services web, non seulement pour découvrir le services web, mais aussi pour enchainer leurs invocations.

Sur la base de ces principes de conception, SAWSDL définit trois nouveaux concepts pour l'annotation sémantique des services :

- Un attribut d'extension, nommé **ModelReference**, permet de préciser l'association entre une composante WSDL et un concept dans un certain modèle sémantique (ontologie de domaine). Cet attribut ModelReference est utilisé pour annoter des définitions de type simple ou des déclarations d'éléments ou d'attributs des interfaces WSDL comme les opérations et les espaces de nommage par défaut.
- Deux attributs d'extension nommés liftingSchemaMapping et loweringSchemaMapping, sont ajoutés aux déclarations des éléments de schéma XML de type complexe ou simple et assurent les mappings entre les concepts sémantique définis dans le domaine référencé par l'attribut ModelReference. Ces mappings peuvent être utilisés pendant l'invocation de service.

Un problème avec SAWSDL, c'est qu'il apparaît comme une simple extension syntaxique des WSDL, sans aucune sémantique formelle. Contrairement à OWL-S et (en partie) WSML, il ne fournit aucun grounding. Un autre problème avec SAWSDL est son support logiciel très limité. Des exceptions existent comme par exemple un service de découverte et de planification de la composition des services METEOR-S. Cependant, la soumission de SAWSDL comme une recommandation du W3C ne tient pas en compte seulement une évolution standardisée des plateformes de services web sémantique dans le consortium W3C (plutôt que d'être comme un commutateur de technologies révolutionnaire pour les technologies de pointe beaucoup comme OWL-S ou WSML), mais va pousser le développement de logiciels en tenant en compte cette proposition SAWSDL et renforcera la recherche sur la reformulation de ces solutions en y intégrant SAWSDL.

### 3.4.5 Synthèse

Pour pouvoir analyser des données biomédicales, il est primordial de véhiculer le sens de ces données. Actuellement, les descriptions de base des services restent syntaxiques, elles représentent un document technique qui décrit comment le service fonctionne et les types basiques des paramètres du service qui se limitent aux types de base. Il ne fournissent donc pas assez de sémantique pour pouvoir faire des interprétations ou des raisonnements. De la même façon, les registres de bases comme UDDI ne font que rassembler des détails techniques concernant les services web en se basant sur leurs WSDLs. L'intégration du sens et de la sémantique permet donc de faciliter les processus de sélection et de composition des services.

Les technologies du web sémantique permettent d'associer du sens aux données et aux traitements. Combinées aux technologies de composition de services web, elles permettent de remédier aux problèmes d'hétérogénéité des langages d'applications avec lesquels les plateformes ont été développées. Cependant, malgré les grandes avancées dans ce domaine, la composition de service reste toujours une tâche complexe [215]. Cette complexité résulte du fait qu'il existe plusieurs modèles de description de services web qui décrivent leurs paramètres et leurs fonctionnalités dans des domaines distincts. Ces modèles sont généralement peu compréhensibles par l'utilisateur, à cause de la spécificité des domaines et des terminologies qu'ils utilisent.

Les outils de composition de services proposés permettent de répondre aux besoins de constituer des modules logiciels complexes et permettant de véhiculer du sens lors de la circulation de l'information. Les standards de base comme les WSDL et les quelques solutions qui ont essayé de les étendre comme WSDL-S et SAWSDL présentent des solutions peu évoluées sémantiquement ; ce sont des solutions syntaxiques. Elles sont dites syntaxiques parce qu'elles ne permettent pas d'exprimer une sémantique large qui pourrait être exploitée par les humains et les agents logiciels dans les raisonnements. Elles sont standardisées par le W3C parce qu'elles sont issues de concepts très simples qui pourraient être adaptées à n'importe quelle plateforme logicielle.

D'autres solutions que l'on peut qualifier de "sémantiques" ajoutent du sens aux données et aux traitements. Elles peuvent articuler plusieurs ressources et rendre accessible une grande quantité de ressources et de connaissances. Certaines de ces solutions se basent sur des ontologies pour faciliter l'intégration et les aspects de raisonnement comme OWL-S, WSMO, etc. D'autres se basent sur des langages plus simples pour bénéficier d'une flexibilité lors de la réutilisation ou pour bénéficier de la légèreté de certain langages de description comme BPEL4WS.

WSMO propose une vision du monde dans laquelle les ontologies jouent un rôle essentiel pour permettre, par exemple, la découverte automatique, l'inter-fonctionnement et la composition des services web. Par conséquent, comme OWL-S et SWSO, WSMO est une ontologie qui décrit les divers aspects liés aux services web sémantiques. En outre, de la même manière que SWSF, WSMO fournit un langage expressif de programmation orienté web, WSML [99] fournit une syntaxe uniforme pour décrire tous les éléments décrits dans WSMO.
Comme dans OWL-S,les services web sont spécifiés dans WSMO par le biais de trois classes à savoir, la capacité de service, la chorégraphie du processus, son orchestration et le Grounding du service. WSMO permet la définition du corps du service à savoir, les entrées, sorties, conditions préalables, et les résultats associée aux services sa terminologie diffère de celle de OWL-S. Contrairement à OWL-S et SWSO, WSMO ne fournit pas de notation pour la construction des processus composites en termes de contrôle de flux de données. Il met l'accent sur la spécification de la chorégraphie interne et externe et l'orchestration en utilisant une approche basée sur des machines à états abstraits et des transitions. Le Grounding dans WSMO est définie de la même manière que celle définie dans la spécification OWL-S. Cette tâche est réalisée par le médiateur, qui est un concept clé dans WSMO.
Dans l'approche WSMO, les médiateurs effectuent des tâches telles que la gestion les correspondances entre les concepts des ontologies utilisées, ou entre les messages utilisés en entrée d'un autre service web lors d'une composition. La spécification WSMO contient une taxo-

nomie de médiateurs possibles qui permet de classer les différentes tâches qu'un médiateur est censé effectuer. La définition de médiateurs dans WSMO attire l'attention sur certaines tâches de traduction importantes associées à des services web. Il n'est pas surprenant que ces mêmes tâches de traduction soient nécessaires pour assurer des interactions avec services web décrit avec OWL-S comme le cas de la plateforme IRS-III. Certains systèmes utilisant la spécification OWL-S comme ceux présenté dans[89] [146] [147] utilisent également des composants de type médiateur. Toutefois, plutôt que d'exiger l'existence d'un type de données de l'entité dans la structure de la description des services web, OWL-S estime que les médiateurs sont à leur tour des services et en tant que tels, ces services de médiation peuvent utiliser les mécanismes prévus par OWL-S pour leur découverte, leur invocation et leur composition.
Selon moi, la spécification OWL-S en tant qu'ontologie de description de service est bien organisée pour être exploitée dans n'importe quel domaine d'application. C'est à mon avis le langage le mieux placé et le moyen le plus performant pour aborder la question de la composition des services (web ou non).

Dans nos travaux de thèse on a mis l'accent en premier lieu sur OWL-S, que nous avons détaillé dans le paragraphe 3.4.2. Nous notons que les acteurs du domaine de la composition de services web et du web sémantique ne sont pas encore à ce jour d'accord sur le rôle de OWL-S. Les acteurs du premier domaine pensent qu'il s'agit d'un langage de composition alors que les acteurs du domaine du web sémantique voient plutôt OWL-S comme une ontologie de description. En effet, Ankolekar et al [6], comparent OWL-S avec d'autres langages de composition des services et mettent OWL-S dans la catégorie des langages de description (tels que WSDL) plutôt que celles des langages de composition de services (tels que WS-CDL et BPEL4WS).

Les projets qui traitent les problèmes d'hétérogénéité des ressources en biomédecine sont nombreux, par exemple WS-BioZard, MOBY, BIRN, CaBIG etc. Toutefois, les solutions restent limitées vis à vis de l'aspect fonctionnel des plateformes et technologiques utilisées. Le nombre de workflows constitués de services web et de services (locaux) dépasse largement le nombre de workflows constitué de services web uniques. Ceci est du à l'hétérogénéité de l'implémentation des plateformes logicielles. Ceci implique que le nombre d'outils supportant plusieurs variétés de services est plus important que le nombre d'outils ne supportant que les services web. Qu'en est-il des technologies du web sémantique? Les plateformes à services web hétérogènes réussissent-elles à intégrer facilement les fonctionnalités du web sémantique ? Le paragraphe suivant détaille quelques uns de ces outils et aborde la question de l'utilisation de la sémantique au sein de ces outils.

## 3.5 Workflow

Par définition un « workflow » ou « flux de travail » est une succession logique d'étapes de traitement qui représentent des tâches ou des activités assurant un processus métier (Business Process). Un workflow représente la modélisation et la gestion informatique de l'ensemble des tâches à accomplir et des différents acteurs impliqués dans la réalisation d'un processus métier (aussi appelé processus opérationnel ou bien procédure d'entreprise). De façon plus pratique, le workflow décrit le circuit de validation, les tâches à accomplir par les différents acteurs d'un

processus, les délais, les modes de validation, et fournit à chacun des acteurs les informations nécessaires pour la réalisation de sa tâche. Il est doté généralement d'un système appelé « système de gestion de workflow » (voir figure 3.20) qui est un système d'information générique qui prend en charge la modélisation, l'exécution, la gestion et le suivi des flots de travail. Un tel système manipule une spécification des tâches, exprimées dans un langage approprié. La plupart du temps il s'agit d'une description du processus, qui décrit d'une façon détaillée les interactions entre les différents composants du workflow, y compris les paramètres fonctionnels (input output conditions d'exécutions) et non-fonctionnels (plateforme d'exécution).

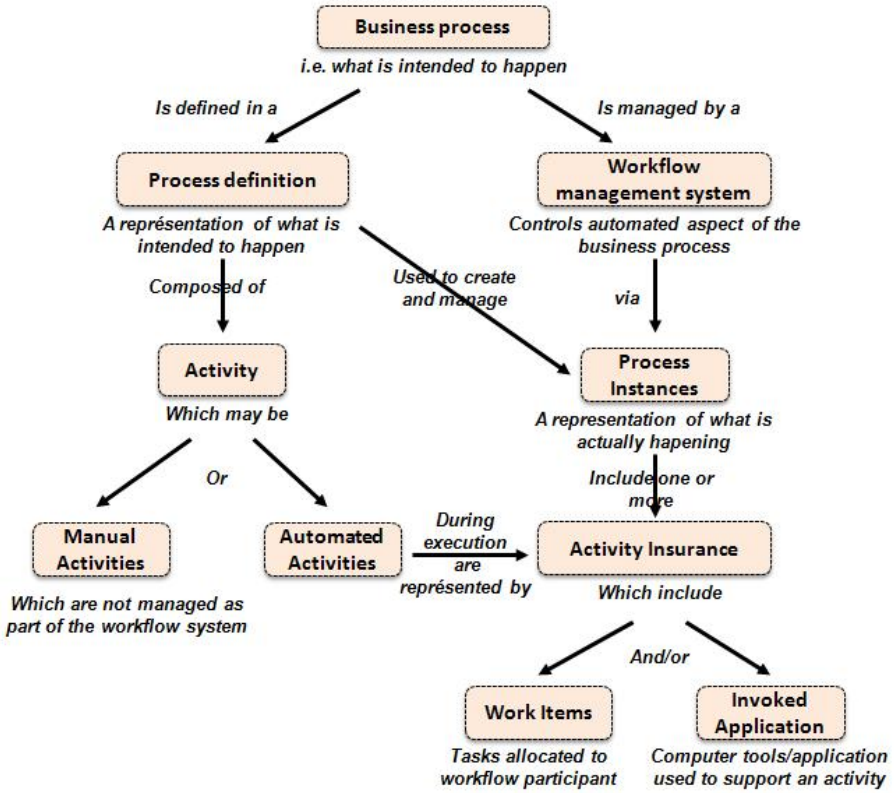


FIGURE 3.20 – Termes standards des systèmes de gestion de workflow [184]

En général on peut distinguer différents aspects dans la gestion des workflows [157]. Le premier aspect est l'aspect fonctionnel : il définit ce qui doit être fait. Il contient la définition des activités dans le workflow ainsi que la hiérarchie des workflows et des sous-workflows utilisés au sein du workflow père. Il permet de décrire aussi les paramètres fonctionnels du workflow comme les entrées sorties, les préconditions, les effets. Le deuxième aspect est l'aspect comportemental : le contrôle de flux ou le processus qui décrit les activités et permet de contrôler le flot d'exécution. Le troisième aspect est l'aspect informationnel : il est représenté par les données utilisées et le contexte d'exécution ou d'invocation des outils. Ceci suppose de disposer de données informationnelles (types des outputs, comportement de l'exécution des processus), et opérationnelles (type de traitement que la donnée doit subir) adéquates. Le quatrième aspect est l'aspect opérationnel : il est défini du point de vue des applications

utilisées dans le workflow. Généralement les opérations des workflows sont externes (services web, programmes d'application), elles peuvent être de nature très différente, mais au sein du workflow les détails techniques des programmes d'application ne doivent pas constituer des obstacles lors de l'exécution ou de l'invocation. Enfin, le cinquième et dernier aspect, est l'aspect organisationnel : il définit qui est responsable de quoi (en termes de tâches). En d'autres termes, cette perspective permet d'assigner des contraintes sur les activités qui déterminent les exigences sur l'agent (qu'il soit humain ou non humain) responsable de son exécution.

### 3.5.1 Composition des workflows

**BIRN [52](pour Biomedical Informatics Research Network) :** Le projet BIRN financé par le Centre national de recherche sur les ressources (NCRR), fournit une structure de partage de données et de services dans le domaine biomédical. L'architecture de BIRN permet aux groupes participants de déployer leurs propres moteurs de workflow tels que le LONI pipeline [112]. Il a été développé pour (i) faciliter la construction des workflows, (ii) vérifier leur consistance lors de la création et de l'exécution. Il propose une interface graphique pour la modélisation des workflows, et permet de faciliter la conception, le test et la visualisation des résultats. Il définit une spécification en XML pour l'intégration des workflows avec des outils comme les grilles de calculs.

**BioWorkflow:** Ce projet a été créé pour améliorer les techniques de mise à disposition des outils bioinformatiques. Le système de BioWorkflow est basé sur la combinaison de deux modèles (i) un modèle de workflow et (ii) un modèle de gestion de workflow. Le modèle de workflow permet la modélisation des services sous forme d'activités et l'autre modèle agit comme un système de gestion des données utilisées. Cet outil a été créé principalement pour les serveurs qui déploient leurs outils de traitement sous forme de services web comme playMoby [53] Soaplab [54] ou SRS [55].

**Soaplab :** Soaplab est un outil dédié à la bioinformatique. Il a été intégré comme support pour les programmes du projet EMBOSS, hébergé au sein de l'European Bioinformatics Institute EBI. Soaplab services web de l'EBI, comprend la plupart des applications EMBOSS qui peuvent être lancées par le biais de services web. Soaplab est une plateforme de services web spécialisée en bioinformatique avec interface WSDL et ligne de commande. Il comprend un module pour la manipulation de services web et fournit un support de base pour générer des applications Java sous forme de services web pour la gestion et la manipulation de ces services.

**Kepler :** est un projet collaboratif qui comprend plusieurs organisations qui développent des modules destinés à être utilisés dans des applications de workflows scientifiques. Il fournit un module autonome qui intègre toutes les fonctionnalités décrite au début de cette section. Kepler est un système autonome basé sur Java. Il utilise la notion «d'acteur» comme une abstraction pour les modules qui sont branchés dans un workflow. La principale limitation de ce système est qu'il ne s'adapte pas bien aux environnements distribués ou dans lesquels les

52. http://www.birncommunity.org/resources/share-your-tools/
53. http://lipm-bioinfo.toulouse.inra.fr/biomoby/playmoby
54. http://soaplab.sourceforge.net/soaplab2
55. http://wwww.biowisdom.com/navigation/srs/srs

traitements sont réalisés par des modules tiers. Il est considéré comme un système de gestion de flux de travail fournissant des fonctionnalités pour accéder à des modules réutilisables, pour pouvoir spécifier des workflows, et gérer leurs exécutions. Il implémente un système de gestion de workflow qui héberge des workflows présentant les caractéristiques suivantes:

- maintient des objectifs d'interopérabilité des architectures orienté services et calculs.

- permet aux utilisateurs ayant des connaissances techniques limitées de composer des workflows indépendant à des modules tiers.

- assure l'intégrité et l'efficacité des données générées à savoir leur insertion dans le bon endroit et leur sureté.

**Taverna [105] :** C'est un outil open source pour la conception et l'exécution des workflows, créé par le projet myGrid. Il permet aux utilisateurs d'intégrer de nombreux composants logiciels différents, y compris les services web SOAP ou REST, tels que ceux fournis par le National Center for Biotechnology Information, l'Institut européen de bioinformatique, la banque de données génétiques du Japon (DDBJ), SoapLab, BioMoby et EMBOSS.
Les utilisateurs peuvent importer des descriptions de nouveaux services dans le Workbench Taverna. Taverna permet aux utilisateurs finaux de connecter les services web dans un workflow/pipeline, qui peut être enregistré et invoqué à tout moment, en utilisant un ensemble de données d'entrée. Taverna fonctionne en grande partie sans surveillance (monitoring). Il est capable de détecter si un service a produit plusieurs outputs et le cas échéant transmettre ces résultats à leur destinataire. Il est capable de gérer les temps d'exécution, arrêter/initier une branche d'un workflow pour permettre à une autre branche de poursuivre son exécution (gestion des priorités des inputs et des outputs).
Des processeurs sont disponibles pour combiner les sorties des deux branches, en utilisant l'union (en enlevant la redondance), la différence ou intersection. Il détecte et récupère des erreurs. Les workflows complexes peuvent être chargés/reconçus/revisités, et réinitialisés par les utilisateurs, puis invoqués et exécutés, pendant des durée allant de quelques minutes à plusieurs. Les données de sortie et les résultats intermédiaires sont disponibles pour la navigation et/ou sauvegardés sur le disque lorsque le workflow est terminé, par exemple, sous forme de fichiers Excel pour une analyse ultérieures.

**TRIANA [191] :** offre une interface d'édition, d'exécution et de manipulation de workflows à travers une interface graphique pour l'accès aux services. La définition des workflows peut contenir des structures de contrôle et d'invocation parallèle des flux. C'est un outil très puissant permettant de gérer les flux parallèles répartis. Il a été intégré dans le projet GridLab et plus précisément dans l'outil Workflow Application Toolkit (TGAT) pour la manipulation des services à travers les grilles de calculs.
Plusieurs projets ont utilsé TRIANA: GridOneD [56], GEO 600 [57], DIPSO [58], FAEHIM [59], GEMMS [60], Data streaming between Web Services using Inferno [61], etc ... Ce sont des projets qui cherchent à mettre en place des outils exploitable sur les grilles de calcul en utilisant

56. http://www.gridoned.org
57. http://www.geo600.uni-hannover.de
58. http://www.wesc.ac.uk/projects/dipso/index.html
59. http://users.cs.cf.ac.uk/Ali.Shaikhali/faehim/index.htm
60. http://www.ccrl-nece.de/gemss/index.html
61. http://www.resc.rdg.ac.uk/projects.php

TRIANA comme outil de base. Ils opèrent généralement dans le domaine de santé.

### 3.5.2 Synthèse

A la différence d'autres techniques de composition de services, les plateformes de composition de services sous forme de workflows bruts sont plus simples à installer et à utiliser. Elles sont trop souvent considérées comme des plateformes d'intégration de services en prenant les workflow comme des boites noires représentant un seul service. étant donné que le but principal n'est pas de composer les services mais de les mettre à la disposition des utilisateurs. Elles sont majoritairement utilisées dans les milieux scientifiques. Elles sont généralement naïves, simples, restreintes et ne tolèrent aucune initiative intelligente. Comparées aux techniques du web sémantique, les techniques de Workflow ne disposent pas d'assez de sémantique pour construire des agents intelligents pour la découverte et la composition de services ni même la supervision des enchaînements des services lors de la construction de workflows ou lors de leur exécution. Ceci dit la comparaison ne tourne pas toujours au profit des outils sémantiques. En effet ce gain de fonctionnalités induit un surcroît de complexité de mise en œuvre. Contrairement à eux, les plateformes de workflow utilisant les descriptions XML sont simples à mettre en place et à adapter à n'importe quel domaine ou type d'application. Mais ceci ne représente pas le seul avantage par rapport aux techniques du web sémantique. En effet, les concepteurs de plateformes de workflows proposent des systèmes de contrôle de flux, qui est un concept spécifique à ce type de plateformes, comme, par exemple, la vitesse d'écoulement et le débit lors de la transmission des données (des tâches/ activités) d'une unité d'exécution à une autre. Les outils issues du monde du web sémantique ne se posent pas ces questions de performance mais focalise l'attention sur différents aspect comme l'interopérablité et les traitements intelligents réalisables par des agents logiciels.
En ce qui concerne nos travaux, on cherche à cumuler les avantages des deux approches à saisir, à composer des services de différents types, à maitriser ces services lors de leur composition et exécution et finalement à faciliter la tâche des utilisateurs en mettant à leurs disposition des outils intelligents et facilement gérables et maitrisables.

## 3.6 Bilan

Cette section présente une synthèse des travaux présentés dans l'état de l'art. Cette synthèse permet de mettre en évidence les éléments que nous devons apporter afin de faciliter la tâche des utilisateurs de la plateforme NeuroLOG lors de la composition des services de la plateforme.

Dans ce chapitre, nous nous sommes attachés à décrire les travaux les études réalisées portant sur les travaux concernant la description, la publication, la recherche et la composition des services sous forme de workflows. Dans un premier temps, on a évoqué les particularités des services web. Nous avons ensuite abordé le volet interopérabilité et sémantique. Enfin nous avons détaillé les outils sémantiques et non sémantique de composition de services.

### Représentation des services

Les services web sont généralement représentés à l'aide d'un ensemble d'informations destinées à faciliter les opération de publication de recherche et de sélection et de composition aux utilisateurs potentiels.
Pour que les clients puissent utiliser et réutiliser des services, le fournisseur doit présenter les aspects fonctionnels de ces services. Les aspects fonctionnel, tels qu'ils sont présentés dans les fichiers de descriptions de base des services, s'avèrent pas suffisants pour faciliter la tâche de recherche, sélection et composition à l'utilisateur. De nombreux travaux ont été menés pour remédier au problème du manque de sémantique des descripteurs de services. Certain de ces travaux enrichissent le contexte de publication du service (les informations sur le fournisseur du service), d'autres enrichissent les descriptions des services en rajoutant une description de la tâche qu'ils effectuent. Cette description s'intéresse généralement au domaine d'action du service et de la catégorie de la tâche ainsi que ses objectifs. Enfin, d'autres travaux s'intéressent plutôt à l'utilisation du service, des critères sur la qualité du service en spécifiant des critères sur la qualité de service et ses conditions d'exécution.

L'enrichissement de la description des services peut être réalisé de différents façons/-moyens. Il peut être fait soit par le biais de descriptions textuelles, soit à l'aide des langages semi-structurés comme XML, soit à l'aide des langage du web sémantique comme par exemple RDF et OWL. Le paragraphe 3.2.2 détaille les différents avantages du web sémantique en terme de capacité de représentation, de raisonnement et d'inférences logiques.

**Les aspects fonctionnels :** Les aspects fonctionnels sont généralement décrits à l'aide des langages semi-structurés. Un fichier XML est généralement utilisé pour décrire les types de données, les opérations, les messages, le protocole de communication et les ports d'accès aux services Web. Dans le cadre de nos études ce rôle est joué par les WSDLs, les descripteurs jGASW ou GWENDIA et les script shell, pyton, Java. D'autres langages comme SAWSDL et WSDL-S ajoutent des références vers des descriptions sémantiques sans pour autant les intégrer au seins de leurs plateformes, intégration qui peut néanmoins être réalisée si le fournisseurs propose le mécanisme d'intégration nécessaire.

**Les aspects liés au fournisseur :** La description des fournisseurs s'avère une aide essentielle pour les clients lors de la sélection des services. En effet, lors de la sélection et de l'invocation des services ils peuvent rencontrer des problèmes d'ordre technique ou fonctionnel qui peuvent être résolus à l'aide de la catégorisation des fournisseurs de services. Les informations telles que le site web, le mode de paiement, la sécurité de son site etc. sont importantes pour la sélection des services par les utilisateurs. Ce genre de description est généralement pris en compte par des registres de services UDDI, UDDI-M et BioRegistry. Seul OWL-S propose une ontologie de catégorisation des fournisseurs.

**Les aspects liés à la tâche du service Web :** L'information la plus pertinente à nos yeux pour la sélection et la composition des services est la description des tâches des services. En effet l'aspect fonctionnel et les aspects liés aux fournisseurs ne suffisent pas pour la sélection. Afin de sélectionner un service nous souhaitons connaître le domaine d'application du service, son environnement d'exécution et sa méthode de calcul (pour les services de recalage ou de segmentation d'image par exemple) ; on souhaiterait aussi avoir des informations sur la qualité de service lorsqu'il a été utilisé par des utilisateurs ayant un profil similaire au

notre. Tout ceci nécessite une description détaillée de la tâche et des utilisations faites dans un contexte (études) similaire au notre.
La catégorisation des services découle de leur objectif, de leur domaine d'action, de leur type, du type de données qu'ils acceptent ou selon la plateforme sur laquelle il tourne. Tous les outils qui intègrent le web sémantique WSMO, OWL-S, IRS-III etc intègrent ainsi la catégorisation des services.

### Les systèmes de raisonnement pour la sélection, l'invocation et la composition des services

Cet aspect reste le plus difficile à mettre en œuvre car il pose des contraintes fortes liées au domaine d'application du service. Différents outils proposent des règles sémantiques pour inférer des décisions, d'autres proposent des mécanismes de vérification plus simples qui reposent sur la subsomption. Dans tout les cas la validation est difficile à faire car elle dépend aussi de l'ontologie choisie par les experts du domaine et de l'expressivité du langage utilisé.

**Application de règles d'inférences :** La sélection et la composition à l'aide des règles d'inférence est une des méthodes qui permet de composer des instances de services à la volée sans savoir d'avance qu'est ce que le service fait ou où est ce qu'il agit, mais en sachant ses objectifs. En effet, c'est un objectifs qui référence une liste de services. Plusieurs travaux ont été fait dans ce contexte [214] [213], ils visent à déterminer un objectif à l'aide d'une règle d'inférence qui correspond à l'objectif en cours puis ils sélectionnent le service qui va le mieux avec cet objectif.

**Optimisation, satisfaction de contraintes :** ces méthodes également issues du domaine de l'intelligence artificielle cherchent à trouver une solution parmi plusieurs possibles. Elles ont été intégrées avec les techniques du web sémantique pour répondre aux besoins de la composition de services.
En résumé, selon l'ensemble des travaux que nous avons étudiés en relatifs à la description, la publication, la recherche et la sélection de services web, cinq aspects doivent être pris en compte au niveau de la représentation des services en vue de faciliter leur implémentation, sélection et composition. Trois aspects ; fonctionnel, les aspects liés à la tâche du service et les et les aspects liés au fournisseur. Aucun des travaux étudiés ne couvre l'ensemble de ces aspects. Ceci nous conduit à préciser nos objectifs dans ce domaine.

### Améliorer les descriptions des services et des workflows

Afin d'améliorer la description sémantique des outils de traitement d'images considérés comme des services et des workflows nous souhaitons utiliser les techniques du web sémantique.

Nous nous appuierons sur le formalisme OWL-S qui nous semble le mieux adapté à notre contexte de travail. Ce formalisme est capable de décrire parfaitement les services, leurs propriétés, leurs capacités ainsi que leurs entrés/sorties. Cependant, il reste des points à améliorer et une réflexion à mener vis à vis d'autres outils présentés dans le chapitre précédent.
D'une part, nous devons étendre la spécification OWL-S pour l'adapter à nos besoins et d'autre part nous devons enrichir l'ontologie avec des ontologies de domaine pour pouvoir

ajouter des traitements relatifs aux outils de la plateforme NeuroLOG.
L'extension que l'on souhaite faire est vraiment relative à l'implémentation de l'API OWL-S qui ne correspond pas à la forme des services dont on dispose au niveau de la plateforme NeuroLOG.
Nous souhaitons en outre ajouter différent algorithmes de vérification et de validation sémantique qui s'appuieront sur les concepts et les relations de l'ontologie de domaine que nous utiliserons.

### Composition des outils de traitement d'images

Pour la composition nous souhaitons analyser les fonctionnalités des outils étudiés au niveau de l'état de l'art, et évaluer leur complémentarité avec les outils disponibles au sein de la plateforme NeuroLOG.

### Rajouter les métadonnées aux donnés produites

Il s'agit de nous appuyer sur l'état de l'art précédent pour ajouter de la sémantique (des métadonnées) aux données produites par les outils de traitement d'images. Ceci concerne le choix des raisonneurs capables d'exécuter des règles sémantiques et leurs intégration.

Chapitre 4

# Extension de OWL-S pour composer des services Web générés par des applications d'encapsulation de services

**Sommaire**


**4.1 Introduction . . . . . 93**
4.1.1 Situation par rapport au travail . . . . . 94
4.1.2 Problèmes des services jGASW . . . . . 95
4.1.3 Motivation pour la sélection et la réutilisation de OWL-S . . . . . 96
**4.2 Méthode . . . . . 96**
4.2.1 Réutilisation de OntoNeuroLOG . . . . . 98
4.2.2 Extension de OWL-S . . . . . 98
4.2.3 Ajout des mécanismes de raisonnement . . . . . 101
**4.3 Implémentation . . . . . 103**
**4.4 Discussion . . . . . 107**
**4.5 Conclusion . . . . . 109**


## 4.1 Introduction

Dans ce chapitre nous résolvons le problème de la composition de services web générés par des applications qui encapsulent des services déjà existants. En effet, ces applications génèrent des WSDLs non standard, conçues pour répondre à certains besoins du domaine d'application.

Dans ce chapitre, nous proposons une méthode pour la composition semi-automatique de services web dans le cadre du projet NeuroLOG. Dans ce projet, la réutilisation des outils de traitement d'images est réalisée en les encapsulant à l'aide de l'outil jGASW (java Generic Application Service Wrapper). Nous étendons la spécification OWL-S afin de la rendre compatible avec nos WSDLs non-standard et de faciliter la composition de services web générés à l'aide de jGASW. A cette extension nous ajoutons quelques mécanismes de vérification de compatibilité sémantique spécifiques à notre domaine d'application et au projet dans le cadre duquel nous travaillons.

Ce travail a été publié dans la conférence ICIW en 2012 [200].

### 4.1.1 Situation par rapport au travail

Nous travaillons dans le cadre du projet NeuroLOG qui vise à partager des ressources médicales (images du cerveau et outils de traitement d'images) [126]. Les outils de traitement d'images sont encapsulés comme des services Web en utilisant jGASW [13]. JGASW est une API java permettant l'encapsulation des applications scientifiques pour faciliter leur exécution dans un environnement SOA.
Afin de permettre le partage des ressources de neuroimagerie dans la plateforme NeuroLOG, une ontologie OntoNeuroLOG [111] a été conçue. Elle fournit une sémantique commune pour le partage de l'information dans le système NeuroLOG. Les ressources couvrent à la fois les données de neuroimagerie (comme les images IRM) et les outils de traitement d'images (par exemple: recalage, débruitage ou segmentation).

Nous souhaitons grâce à la sémantique mieux décrire et partager les images et les fonctionnalités des outils de traitement d'images dans le but de faciliter la réutilisation et l'invocation de ces outils par l'utilisateur. Pour cela, nous avons choisi d'ajouter de la sémantique aux services web jGASW pour (i) faciliter leur composition sous forme de workflows, (ii) automatiser leur invocation et (iii) pouvoir contrôler l'exécution du workflow entier. Cela aidera les utilisateurs qui n'ont pas participé au développement, à l'encapsulation et à la publication des outils à mieux comprendre et réutiliser ces outils.

Pour répondre à ces objectifs nous avons choisi d'utiliser la plateforme OWL-S. Elle représente une plateforme bien définie et répond bien à nos besoins. Nous devons faire face dans ce travail à quelques problèmes techniques qui entravent le processus de composition avec l'API OWL-S Nous leur apportons une solution et nous y ajoutons des contrôles de cohérence visant à : (i) assurer l'interopérabilité et la composition en vérifiant la compatibilité entre les entrées et les sorties des services web, (ii) vérifier la compatibilité entre les données sélectionnées par les utilisateurs et la définition sémantique des entrées du service, (iii ) vérifier la cohérence entre les fonctionnalités des outils de traitement d'images et la description des entrées/sorties, en respectant les normes définis par les spécialistes (les définitions formelles des actions conceptuelles) modélisées dans l'ontologie OntoNeuroLOG.

Nous avons utilisé l'ontologie OWL-S pour bénéficier de son modèle de description des services web et de sa grande expressivité en termes de description de paramètres et de l'aspect comportemental. Cependant, nous avons dû contourner quelques problèmes techniques concernant les WSDLs de la plateforme jGASW au moyen d'extensions de OWL-S pour permettre son utilisation dans notre cadre NeuroLOG.

Les contributions principales de ce travail sont : (1) une extension de la spécification OWL-S pour l'adapter à notre contexte d'utilisation de jGASW sans changer la structure de base des WSDL, (2) de nouvelles capacités de raisonnement pour effectuer des contrôles de cohérence concernant l'utilisation de nos services web annotés. La suite de ce chapitre est organisée comme suit : la section 4.1.2 fournit plus de détails sur les difficultés liées aux WSDLs des services générés par l'outil jGASW et la description sémantique de ces services faites à l'aide de l'ontologie OWL-S. Au niveau de la section 4.1.3 nous présentons notre solution basée sur l'extension de OWL-S pour remédier aux problèmes cités au niveau de la section 4.1.2 puis différents mécanismes de raisonnement associés qui valident les capacités de services, conformément à notre ontologie de domaine OntoNeuroLOG. La section 4.3

fournit quelques précisions sur l'implémentation. La section 4.4 résume notre contribution et la situe dans le contexte plus large des workflows sémantiques, et enfin, la section 4.5 fournit une conclusion et présente des perspectives pour les travaux futurs.

### 4.1.2 Problèmes des services jGASW

Dans cette section nous décrivons d'abord les fichiers WSDL générés automatiquement par jGASW, puis nous étudions la génération automatique de descriptions sémantiques avec OWL-S et finalement nous analysons le problème pour lui apporter une solution.

Tout d'abord, nous rappelons comment la plateforme jGASW fonctionne. C'est une interface graphique permettant à l'utilisateur : de télécharger les outils de traitement d'images (Shell), de définir les entrées/sorties nécessaires à leur exécution, de définir les arguments (constantes ou paramètres) qu'on associe à une ligne de commande et de définir les bibliothèques nécessaires à leur exécution. Ces éléments sont organisés selon un schéma XML, c'est le descripteur jGASW. La génération du service web consiste à transformer le descripteur jGASW en une interface de service web en générant le fichier WSDL et un schéma XSD (XML Schema Definition) qui détaille les entrées/sorties susceptibles d'être mises dans un fichier WSDL (jGASW a besoin du descripteur du service lors de l'exécution et de la récupération des résultats). Le schéma XSD détaille les entrées/sorties de chaque opération WSDL. En fait, tous les services web que jGASW crée ont des fichiers WSDL identiques, et des opérations identiques avec toujours une seule entrée et une seule sortie, et une sortie par défaut qui joue le rôle d'une exception au cas où l'exécution ne fonctionne pas. Seul le schéma XSD fait la différence entre les services Web jGASW. En effet, les entrées et les sorties sont détaillées dans le schéma XSD en fonction de chaque service.

Par exemple, dans la figure 4.1, la colonne 1 illustre la description d'une opération WSDL qui a pour nom « local » et qui est composée de `tns:local` en entrée, `tns:localResponse` en sortie, et `tns:SOAPException` comme message d'erreur (généré si l'exécution du service échoue).
La colonne 2 détaille l'entrée `tns:local` en la définissant comme un `complexType` contenant deux `xs:element` (i) (`simpleinput1` et `simpleinput2`: deux fichiers d'entrée) et la sortie `tns:localResponse` en définissant un autre `complexType` imbriqué dans un autre `complexType` appelé `jigsawOutputTest2111` (ii) (`jigsawOutputTest2111` : généré automatiquement à partir du nom du service web). Ce `complexType` `jigsawOutputTest2111` contient quatre `xs:element` représentant les différents fichiers de sortie (`stdout`, `stderr`, `simpleoutput1`, `simpleoutput2`).
La colonne 3 montre les enveloppes SOAP (appel/demande) utilisées au moment de l'exécution. En cas d'invocation d'un service, JGASW prépare l'enveloppe SOAP, appelle le service et retourne le résultat selon les séquences décrites dans le type complexe `jigsawOutputTest2111`. Comme on le voit dans la figure le nom de l'enveloppe SOAP reste toujours le même `ns1:localResult` alors que le `complexType` `jigaswOutputTest211` n'est pas considéré ici.
JGASW encapsule les exécutables dans les services web, et produit au moins deux fichiers standard (`std.out`: pour la sortie standard pour toute sortie du shell, `std.err`: message d'erreur généré si l'exécution échoue), et d'autres fichiers résultant de l'exécution des services

tels que les fichiers comme par exemple (`ex2output1.nii` et `ex2output2.nii`).
La colonne 4 montre la description du grounding de OWL-S générée automatiquement à partir de la description de la WSDL fournie par le service jGASW (en utilisant le convertisseur WSDL2OWLS [1]).
Sur la figure les flèches vertes montrent le grounding de chaque élément représentant une entrée individuelle tandis que la flèche rouge montre un seul Grounding global et unique pour les résultats, matérialisé par `localResut`. En fait, ici, nous avons perdu l'information selon laquelle cette sortie contient quatre fichiers et non un seul.

Comme on arrive pas à décrire sémantiquement les différents outputs du service, celà veut dire que la composition des services jGASW nous sera aussi impossible étant donné qu'on ne pourra pas avoir de liaison entre une output de service et un input du service suivant dans le workflow ou un output du workflow général. Ainsi, avec cette description sémantique ayant une seule sortie, la composition de services jGASW avec OWL-S n'est pas possible. Il faudra donc trouver un moyen pour rendre explicite ces outputs pour qu'on puisse les relier à d'autres inputs ou outputs dans le cas d'un workflow.

### 4.1.3 Motivation pour la sélection et la réutilisation de OWL-S

Tout d'abord, nous avons montré que les fichiers WSDL ne sont compris que partiellement par l'API OWL-S API. Deuxièmement, les structures de contrôle que l'API OWL-S propose nous seront utiles pour concevoir et exécuter nos workflows de traitement d'images. Troisièmement, OWL-S est une spécification OWL bien définie qui facilite la composition de services web et qui offre de nombreuses fonctionnalités qui ne sont pas traitées par d'autres spécifications. En outre, il est soumis au W3C, ce qui est un atout très important dans notre contexte de recherche collaborative en neuroimagerie. Ce choix de OWL-S est également compatible avec notre utilisation de l'ontologie OWL-Lite OntoNeuroLOG comme ontologie de domaine. En effet, OWL-S est une ontologie OWL-DL et il est plus judicieux d'utiliser des ontologies qui sont proches les unes des autres en terme d'expressivité et de raisonnement (i) les deux ontologies sont basées sur le langage OWL, ce qui facilite l'utilisation du même type de raisonnement et de raisonneurs (ii) si l'on utilisait par exemple WSMO nous devrions traduire OntoNeuroLOG en WSML. Enfin, OWL-S fournit l'expressivité appropriée pour représenter la sémantique de services web et répond parfaitement aux exigences de notre domaine application et des outils qui figurent dans notre plateforme. En effet, nous ne pouvons pas modifier la structure des fichiers WSDL, ni la description XML des entrées/sorties, car elles sont intrinsèques à la plateforme jGASW (sinon l'invocation ne fonctionnera pas convenablement).

## 4.2 Méthode

La seule couche qui ne peut pas être modifiée est la couche grounding. En revanche, on peut modifier les autres couches étant donné qu'elles ne sont pas directement reliées à la WSDL.

1. http://semwebcentral.org/projects/wsdl2owl-s/

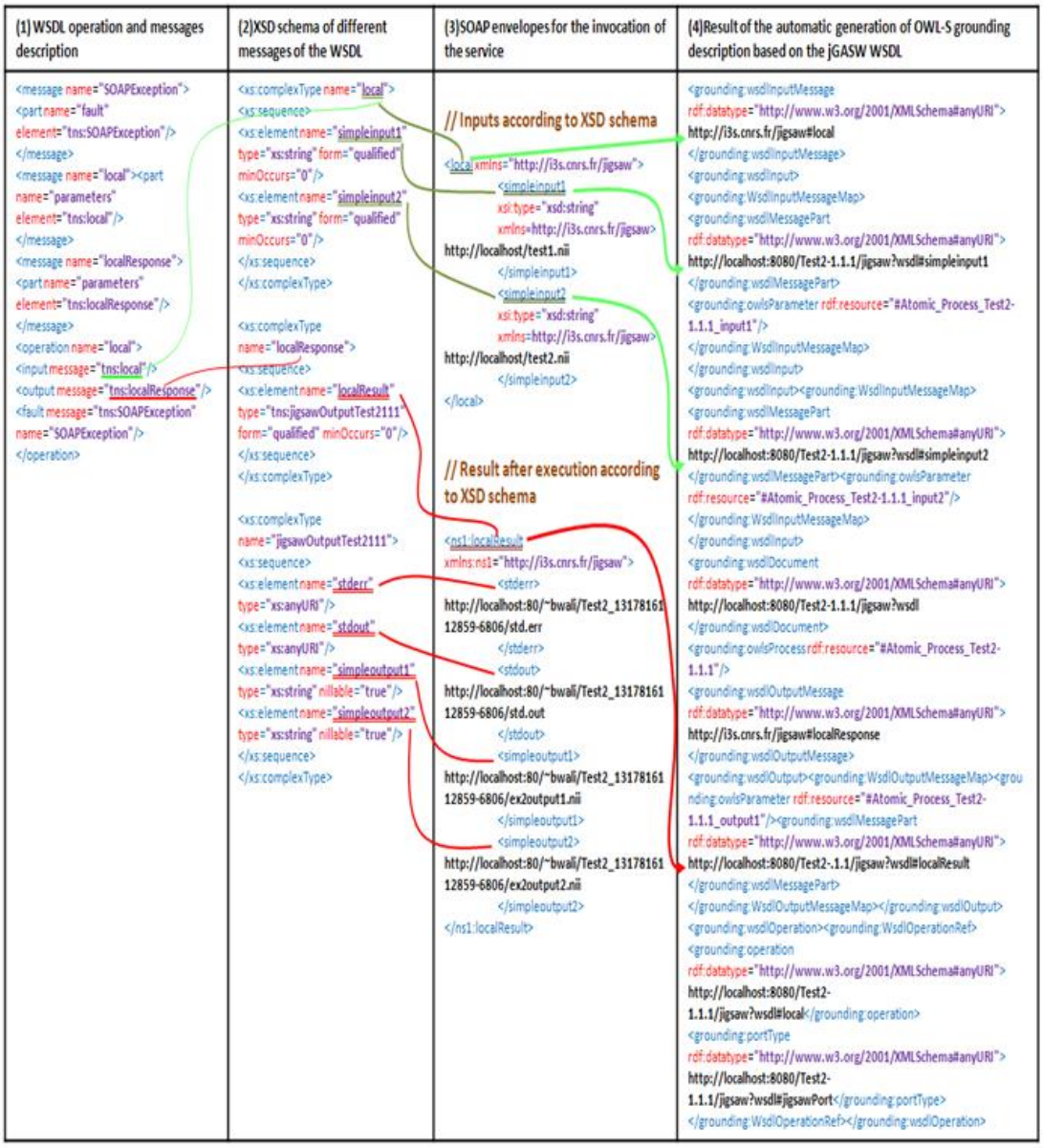


FIGURE 4.1 – Grounding automatique d'un service jGASW en utilisant l'API WSDL2OWLS

Cette figure montre le problème principal qui est posé quand on génère automatiquement la description sémantique d'un service jGASW selon la spécification OWL-S ; les colonnes 3 et 4 montrent comment les entrées sont mappées individuellement alors que les sorties sont mappées comme une seule sortie "locaResult". Par conséquent, le profile et le process OWL-S de ce service contiendront une seule sortie. Pourquoi OWL-S ne prend-il pas en compte les différentes sorties ? Parce qu'elles appartiennent à un type complexe "jigsawOutputTest2111" montré sur la colonne 2. Ces types de sortie générés automatiquement par le logiciel jGASW peuvent être encore plus complexes. En fait, ils peuvent contenir des imbrications multiples de complexType.

Pour ceci nous avons décidé de suivre les étapes suivantes: premièrement, nous étendons l'ontologie Profile de OWL-S pour l'adapter à OntoNeuroLOG. Deuxièmement, nous éten-

dons l'ontologie Process OWL-S pour permettre la description des services jGASW et enfin, nous ajoutons une couche logicielle qui implémente des services de raisonnement assurant la cohérence avec les connaissances modélisées dans l'ontologie OntoNeuroLOG.

### 4.2.1 Réutilisation de OntoNeuroLOG

L'ontologie OntoNeuroLOG contient deux ontologies qui nous seront utiles dans ce travail. La première est l'ontologie des datasets qui décrit les différents types de données IRM ainsi que les différents types d'input/output d'un service simple ou composite. La deuxième est l'ontologie des dataset processing qui décrit la fonctionnalité des outils de traitement d'images (e.g la segmentation, le recalage, le débruitage...). En occurrence, elle définit des classes d'action de traitement que l'outil réalise. Chaque classe de ces ontologies a des caractéristiques spécifiques définies à l'aide d'axiomes DL du langage OWL.

### 4.2.2 Extension de OWL-S

OWL-S est une ontologie OWL particulière. Elle permet la composition semi-automatique de services web. Elle est composée de trois couches. La couche Profile permet l'analyse, la publication et la découverte de services. Elle est utilisée par les fournisseurs de services ; elle leur permet de publier les services en utilisant un point d'accès, l'IRI du service. Elle représente le point de départ d'invocation des services pour les utilisateurs ; ils spécifient leurs besoins et en fonction de ces besoins la sélection des services se fait.

La couche serviceModel (les modèles des processus de OWL-S) est utilisée pour composer les services. Elle permet la modélisation des services web en tant que processus. Elle nous propose de créer trois types de process : les processus atomiques (AtomicProcess), simples (SimpleProcess) et composites (CompositeProcess). AtomicProcess représente la granularité la plus fine en termes de service. Les CompositeProcess sont décomposables en d'autres processus (qui peuvent être composites ou non). Leur enchaînement peut être spécifié à l'aide d'un ensemble de structures de contrôle telles que des séquences, Split, If-Then-Else, etc. SimpleProcess est utilisé pour fournir une vue d'un processus atomique ou une représentation simplifiée d'un processus composite.
La couche grounding décrit comment accéder au service et fournit la liaison entre les paramètres de la WSDL (entrées, sorties, opération, point d'invocation) et les paramètres sémantiques décris dans le formalisme OWL-S. Le but de ce mapping est de permettre la traduction des entrées sémantiques générées par un consommateur de services vers le prestataire de services, et la traduction des messages de sortie de ces services dans le format sémantique approprié (des descriptions OWL) pour que le consommateur puisse les interpréter convenablement.

#### 4.2.2.1 Extension du Process

Étant donné que la couche process n'est pas reliée directement à la WSDL on pourra la modifier selon nos besoins. Notre extension vise alors à décomposer la sortie représentée comme une seule entrée (localResult) au niveau du grounding OWL-S (décrit dans la figure 4.1) en ses différents éléments (par exemple, stderr et stdout, simpleoutput1, simpleoutput2).

Pour surmonter ce problème, plusieurs classes et propriétés ont été ajoutées au processModel de OWL-S :

- Une classe `NlogParameter` pour indiquer les paramètres qui sont incorporés dans un paramètre de nature composite (par exemple, `stderr`, `stdout`, `simpleoutput1`, `simpleoutput2`), elle est définie comme suit :

```
<owl:Class rdf:about="NlogParameter">
    <rdfs:subClassOf rdf:resource="#Parameter"/>
    <owl:disjointWith rdf:resource="#Output"/>
</owl:Class>
```

- Une propriété OWL de type objectProperty `nlogExpandsTo`, l'association d'un paramètre d'ordre composite à ses éléments essentiels selon le schéma XSD du service

```
<owl:ObjectProperty rdf:about="#nlogExpandsTo">
    <rdfs:domain rdf:resource="#Output"/>
    <rdfs:range rdf:resource="#NlogParameter"/>
</owl:ObjectProperty>
```

- Une propriété OWL de type dataProperty `hasID`, indiquant la balise qui représente l'élément au sein de l'enveloppe SOAP « la chaîne résultat après l'invocation de service »

```
<owl:DatatypeProperty rdf:about="#hasID">
    <rdfs:domain rdf:resource="#NlogParameter"/>
    <rdfs:range rdf:resource="&xsd;anyURI"/>
</owl:DatatypeProperty>
```

- Une propriété de type dataProperty `hasLabel`, qui est une propriété non-fonctionnelle fournissant une description informelle du paramètre

```
<owl:DatatypeProperty rdf:about="#hasLabel">
    <rdfs:domain rdf:resource="#NlogParameter"/>
    <rdfs:range rdf:resource="&xsd;anyURI"/>
</owl:DatatypeProperty>
```

La figure 4.2 donne un exemple de l'utilisation des extensions décrite ci-dessus.

#### 4.2.2.2 Extension du Profile

Comme décrit précédemment, la couche Profile de OWL-S donne des informations sur les capacités et le comportement du service. Nous l'enrichissons en lui ajoutant une référence vers la classe de traitement correspondante au niveau de l'ontologie NeuroLOG à l'aide de la propriété DataProperty `refers-to` qui appartient à l'ontologie OntoNeuroLOG.

```
<process:Output rdf:about="&tool;#Atomic_Process_Test2_output1">
    <process:nlogExpandsTo>
        <process:NlogParameter rdf:ID="Atomic_Process_Test2_output1_simpleoutput1">
            <process:hasLabel rdf:datatype=
            "http://www.w3.org/2001/XMLSchema#anyURI">
                This is the extension of the first parameter simpleoutput1
            </process:hasLabel>
            <process:hasID rdf:datatype=
            "http://www.w3.org/2001/XMLSchema#anyURI">
                simpleoutput1
            </process:hasID>
            <process:parameterType rdf:datatype=
            "http://www.w3.org/2001/XMLSchema#anyURI">
            http://localhost/dataset-owl-lite.owl#T1-weighted-MR-dataset
            </process:parameterType>
        </process:NlogParameter>
    </process:nlogExpandsTo>
</process:Output>
```

FIGURE 4.2 – Exemple d'utilisation de l'extension de la couche Process

Pour détecter la valeur de "simpleoutput1" nous sélectionnons d'abord son ID en utilisant la propriété hasID. En second lieu, nous analysons la chaîne résultat (enveloppe SOAP) à l'aide de l'ID récupéré correspondant à la balise "simpleoutput1". Le type de données de ce paramètre NlogParameter est donné par la propriété Process:ParameterType.

```
1  <owl:ObjectProperty rdf:about="&iec;refers-to">
2      <rdfs:domain rdf:resource="#Profile"/>
3      <rdfs:range rdf:resource=
4      "&data-processing-owl-lite;data-processing"/>
5  </owl:ObjectProperty>
```

#### 4.2.2.3 Composition de services en utilisant l'extension du Process

Comme mentionné plus haut, nous ne pouvons pas modifier le fichier WSDL sinon l'invocation de l'outil ne fonctionnera pas et par conséquent, nous ne pourrons pas non plus modifier directement le grounding du fichier WSDL. L'extension du ProcessModel est suffisante pour permettre la composition des services jGASW. Une fois que les sorties ont été liées aux NlogParameter correspondants selon le schéma XSD issu du traitement jGASW, nous sommes en mesure de composer ces services. Pour cela, nous avons introduit une autre objectProperty, qui fait le lien entre deux paramètres OWL-S (également adapté pour NlogParameter car c'est une sous classe de l'entité Parameter au niveau de l'ontologie OWL-S) :

```
1  <owl:ObjectProperty rdf:about="#links">
2      <rdfs:domain rdf:resource="#Parameter"/>
3      <rdfs:range rdf:resource="#Parameter"/>
4  </owl:ObjectProperty>
```

Ainsi les différents outputs du services sont explicites. Nous pourrons donc les relier avec les inputs des services suivants. La composition des services est faite ainsi, de la façon suivante (figure 4.3) :

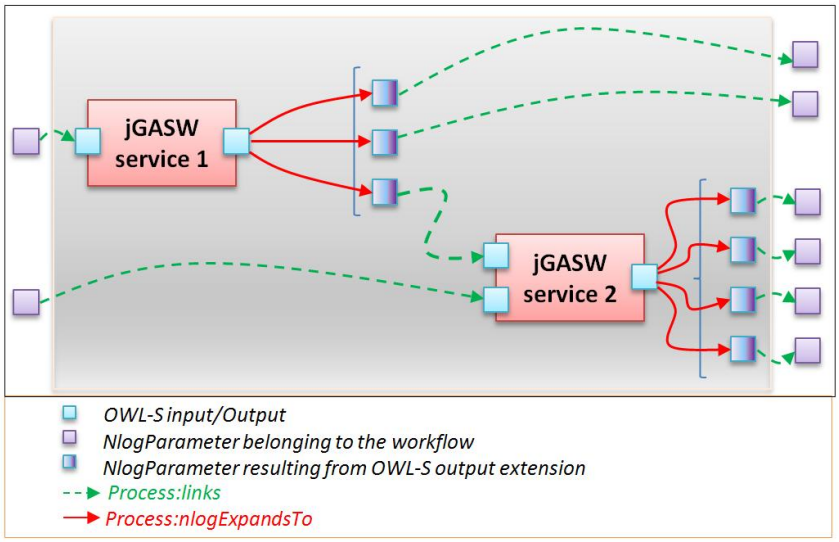


FIGURE 4.3 – Composition des services jGASW à l'aide de l'extension qui a été proposée et du lien ajouté entre les paramètres

Dans cet exemple, on compose deux services jGASW: le premier, (service1) présente une entrée et une sortie. La sortie est composée de trois sorties selon son schéma XSD, et le second (service 2) comporte 2 entrées et une sortie. La sortie est composée de quatre sorties selon son schéma XSD. Le profil du service intégrant l'ensemble du workflow possède deux entrées reliées respectivement aux services jGASW 1 et 2 et six sorties provenant des deux services composés. Un seul paramètre interne est transmis du service 1 au service 2.

### 4.2.3 Ajout des mécanismes de raisonnement

#### 4.2.3.1 Vérification de compatibilité entre le Profile et les classes de datasets processing

Ce service permet de garantir que la définition du profil est compatible avec la classe de traitement de données sélectionnée par l'utilisateur au moment de l'annotation.

Cette transformation se fait à l'aide d'un algorithme qui transforme le profile du service en une classe de traitement dont les axiomes expriment des contraintes sur les entrées et les sorties des traitements.

L'algorithme est le suivant: En premier lieu, nous créons une classe temporaire `tmp_Profile_data-processing` relative à l'opération en cours. Nous traduisons les relations entre le profile, les entrées et les sorties en axiomes et nous les ajoutons à la classe temporaire qu'on vient de créer. En particulier, nous comptons le nombre d'entrées groupées par type de classe `dataset` pour déterminer la cardinalité de l'axiome correspondant, par exemple:

```
1 proc1   process:hasInput                  i1
2 i1            process:parameterType       Mr-dataset
```

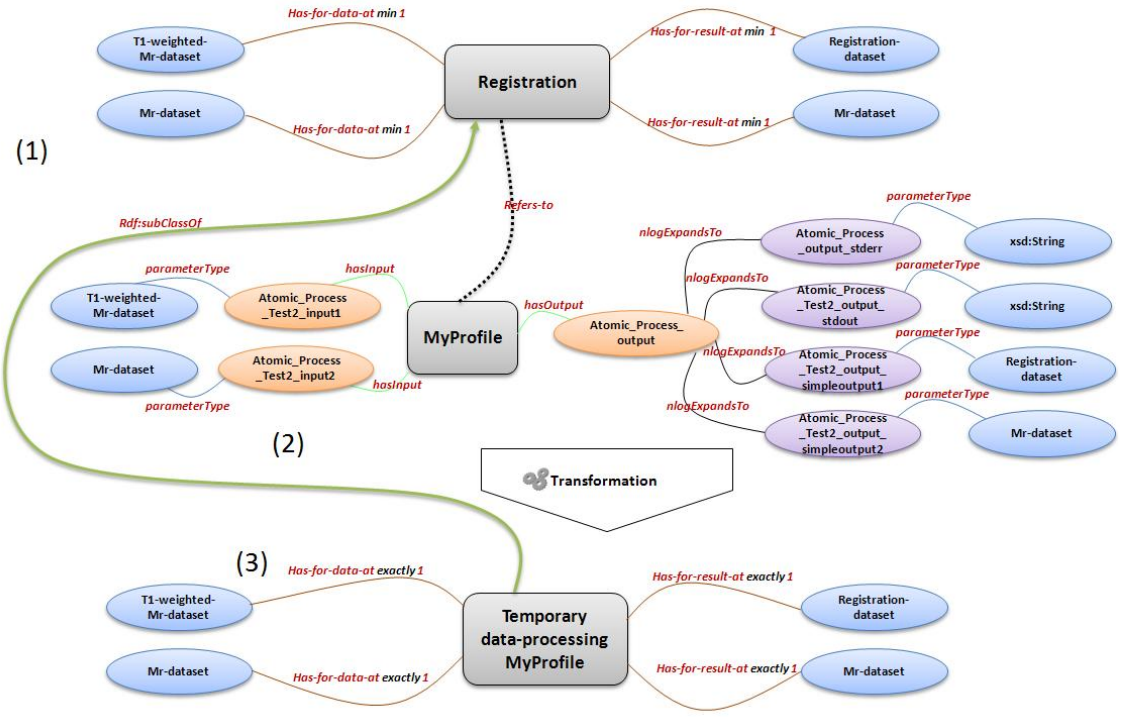


FIGURE 4.4 – Transformation d'un Profile en data-processing

Trois éléments composent cette figure (1) (2) (3) :
(1) représente la description de la classe de traitement Registration qui provient de l'ontologie NeuroLOG; (2) représente la description sémantique OWL-S de l'outil de recalage enrichi; (3) montre la transformation faite sur le profile pour le rendre équivalent à une classe de traitement.

```
proc1   process:hasInput                    i2
i2            process:parameterType      Mr-dataset
```

Ceci conduit à une cardinalité égale à 2 pour `Mr-dataset` (`Mr-dataset` représente un ensemble d'images acquis au moyen d'un appareil de résonance magnétique). La troisième étape consiste à sélectionner l'objectProperty approprié pour la construction de l'axiome (par exemple `process:hasInput` substitué par `has-for-data-at` et `process:hasOutput` substitué par `has-for-result-at`
Le résultat des trois étapes donne :

```
    proc1        process:hasInput                        i1
    i1                 process:parameterType             Mr-dataset
    proc1        process:hasInput                        i2
    i2                 process:ParameterType             Mr-dataset
    proc1       ws : has-for-data-at     exactly     2     Mr-dataset
```

La quatrième étape consiste à ajouter ces axiomes à la classe `tmp_Profile_data-processing`. Enfin la dernière étape consiste à ajouter la nouvelle classe `tmp_Profile_data-processing` avec ses axiomes comme sous-classe de

la classe de traitement visée par le Profile `MonProfil` sélectionnée par l'utilisateur (dans notre exemple `tmp_Profile_dataprocessing` subClassOf Registration) et à faire une classification à l'aide du raisonneur pour vérifier la cohérence. Si l'ontologie est consistante alors l'annotation est considérée comme valide ; sinon elle est considéré non valide. Ainsi sémantiquement, la fonctionnalité de l'outil est validée, c'est-à-dire que les ObjectProperty `has-for-data-at/has-for-result-at` sont compatibles avec les entrées/sorties spécifiées dans la classe de traitement correspondante dans l'ontologie OntoNeuroLOG. La figure 4.4 montre un exemple illustratif de l'algorithme.

#### 4.2.3.2 Vérification de compatibilité entre les sorties et les entrées des services lors de la composition

Ce service est appliqué lorsqu'un utilisateur crée un nouveau workflow. Le traitement vise à s'assurer qu'à chaque lien correspondant à un passage de paramètres les types correspondants sont compatibles. Ainsi, on distingue trois cas :

- types de données identiques : la sortie et l'entrée ont exactement le même type. La compatibilité est validée et la composition est acceptée.
- Lien vers un type de données plus spécifique : ils ne sont pas compatibles puisque la sortie est plus générique que l'entrée du service suivant.
- Lien vers un type de donnée plus général : le premier service va toujours retourner des résultats qui sont sémantiquement compatibles avec l'entrée du service suivant. La compatibilité est validée et la composition est acceptée.

N.B. Le workflow est valide si les paramètres ont le même type ou si le paramètre source est subsumé par le paramètre cible dans l'ontologie des datasets.

#### 4.2.3.3 Vérification de compatibilité entre les images sélectionnées par l'utilisateur et l'entrée du service

Ce service est appelé lorsqu'un service Web est invoqué. Il vérifie si les instances réelles sélectionnées par l'utilisateur (par exemple, un dataset) et affectées aux entrées/sorties des services répondent aux contraintes spécifiées dans les annotations sémantiques du service en question. En pratique, le service sémantique vérifie si la classe (ou le type de données) de cette instance est subsumée par le type de l'entrée du service.

## 4.3 Implémentation

L'annotation sémantique des services jGASW est générée automatiquement en utilisant l'API WSDL2OWLS [2] (voir annexe A.1). L'enrichissement de l'annotation sémantique se fait en utilisant la spécification OWL-S 1.2 [3] et OWL API 3.0 [4]. L'annotation sémantique des workflows est générée en utilisant la spécification OWL-S 1.2 et OWL API 3.0 (voir annexe A.2).

2. http://www.semwebcentral.org/projects/wsdl2owl-s/
3. http://www.ai.sri.com/daml/services/owl-s/1.2/
4. http://owlapi.sourceforge.net/documentation.html

La vérification de la cohérence entre le Profile et la classe de traitement des données est implémentée en utilisant l'API OWL 3.0, OWL-S API 3.0 [5] et le raisonneur HermiT [6].

L'invocation des services Web est faite à l'aide de l'API OWL-S, mais les résultats et les problèmes de composition utilisent le moteur de recherche sémantique CORESE [40] avec l'API OWL-S. CORESE est utilisé pour sélectionner les propriétés fonctionnelles qui concerne l'extension (paramètres liés, identifiants ...) des sorties standards des services en interrogeant l'entrepôt des données sémantiques des services. Nous ajoutons ici un exemple illustratif d'un workflow composé de deux services.

Tout d'abord, nous préparons l'enveloppe SOAP pour appeler le premier service jGASW : les balises représentées en vert montrent l'entrée de l'opération dans la WSDL (`tns:local`) et la balise bleue indique l'image concrète sélectionnée par l'utilisateur et que le service va utiliser.

```
<soapenv:Envelope>
<soapenv:Body>
<local xmlns="http://i3s. cnrs.fr/jigsaw">
<simpleinput xsi:type="xsd:string" xmlns=
"http://i3s.cnrs.fr/jigsaw">
http://localhost/test1.nii
</simpleinput>
</local>
</soapenv:Body>
</soapenv:Envelope>
```

La section suivante montre la sortie du service après invocation: la balise verte montre la sortie de l'opération de la WSDL (`tns:localResult`). Elle enveloppe trois balises bleues qui référencent trois fichiers générés par le service.

```
<ns1:localResult xmlns:ns1="http://i3s.cnrs.fr/jigsaw">
<stderr>
http://localhost:80/~bwali/Test1_1321350928548-9787/std.err
</stderr>
<stdout>
http://localhost:80/~bwali/Test1_1321350928548-9787/std.out
</stdout>
<simpleoutput>
http://localhost:80/~bwali/Test1_1321350928548-9787/testoutput.nii
</simpleoutput>
</ns1:localResult>
```

Les fichiers `stderr` et `stdout` sont acheminés en tant que sorties du workflow alors que `simpleoutput` doit être transmis au deuxième service jGASW. Avec CORESE nous interrogeons la description sémantique des services pour récupérer les différents outputs correspondant au `nlogParameter` qui étendent `localResult`. Ceci sert à décider pour chaque sortie : (1) comment l'extraire de son enveloppe SOAP (2) et vers quelle autre variable doit elle être transmise.

Ainsi, pour tout `nlogParameter` récupéré, on trouve la propriété `hasID` qui représente

5. http://on.cs.unibas.ch/owls-api/index.html
6. http://hermit-reasoner.com/

la balise qui la délimite au niveau de l'enveloppe SOAP `localResult` récupérée, la propriété `links` qui définit son lien avec un autre paramètre (passage de paramètres), et la propriété `parameterType` qui définit son type de données.
La requête SPARQL ci-dessous qui vise à identifier les différentes sorties qui étendent `tns:localResult` (correspondant sémantiquement à `ex001_output1`):

```
PREFIX p1: <http://localhost/kb/Test1_2.owl#>
PREFIX p2: <http://localhost/Process.owl#>
Select   ?nlogParameter  ?link  ?id  ?type
where  {
p1: ex001_output1      p2:nlogExpandsTo        ?nlogParameter
?nlogParameter           p2:links                 ?link
?nlogParameter  p2:hasID                  ?id
?nlogParameter  p2:parameterType        ?type
}
```

Les résultats retournés par la requête sont représentés ci-dessous :

```
?nlogParameter http://localhost/kb/extension-Test1_2.owl#ex001
_simpleoutput
?link    http://localhost/kb/Test1_2.owl#ex002_input2
?id    simpleoutput
?type http://localhost/dataset-owl-lite.owl
#T1-weighted-MR-dataset

?nlogParameter  http://localhost/kb/extension-Test1_2.owl
#ex001_stdout
?link  http://localhost/kb/extension-Test1_2.owl#WF_stdout
?id  stdout
?type  http://www.w3.org/2001/XMLSchema#string

?nlogParameter   http://localhost/kb/extension-Test1_2.owl#
ex001_stderr
?link  http://localhost/kb/extension-Test1_2.owl#WF_stderr
?id  stderr
?type  http://www.w3.org/2001/XMLSchema#string
```

Nous remarquons que le paramètre que nous extrayons est `simpleoutput`. Il doit être transmis à `ex002_input2` seconde entrée du second service jGASW.

La sortie `ex001_output1` est étendue à trois `nlogParameters` comme on le voit dans la Figure 4.5 (`ex001_stdout`, `ex001_stderr`, `ex001_simpleoutput`), correspondant respectivement à (`stdout`, `stderr`, `simpleoutput`) dans les résultats de la requête (la variable de la requête portant la désignation ?id). Ces ID sont les balises utilisées dans le résultat du `localResult`.
Les résultats montrent que les deux variables `ex001_stdout`, `ex001_stderr` sont liés à des sorties de workflow (`WF_stdout1`, `WF_std_err1`) comme montré dans la figure 4.5. Les résultats de la requête montrent que le paramètre `ex001_simpleoutput` est lié au paramètre `ex002_input2`.

Ainsi, elle devrait être passée au deuxième service jGASW. Pour cela, la va-

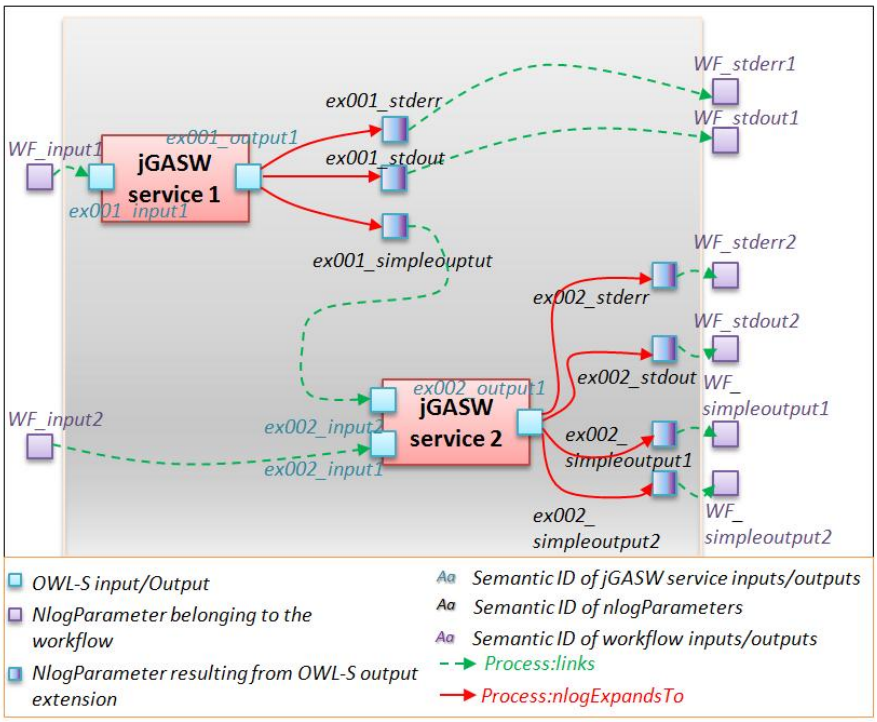


FIGURE 4.5 – Annotation sémantique des services jGASW utilisant l'extension OWL-S

leur de `ex001_simpleoutput` est extraite par le moteur de jGASW en donnant l'identifiant déjà sélectionné par la requête. Le résultat est le suivant: `http://localhost:80/ bwali/Test1_1321350928548-9787/testoutput.nii`. Une nouvelle enveloppe SOAP contenant deux entrées est préparée pour l'invocation du deuxième service jGASW.

```
<soapenv:Envelope>
<soapenv:Body>
<local xmlns="http:// i3s.cnrs.fr/jigsaw">
<simpleinput1 xsi:type=" xsd:string
"xmlns= "http://i3s.cnrs.fr/jigsaw">
http://localhost/test4.nii
</simpleinput1>
<simpleinput2 xsi:type= "xsd:string" xmlns=
"http://i3s.cnrs.fr/jigsaw">
http://localhost:80/~bwali/Test1_1321350928548-9787/testoutput.nii
</simpleinput2>
</local>
</soapenv:Body>
</soapenv:Envelope>
```

La valeur de la variable `simpleinput1` du deuxième service jGASW correspond au fichier sélectionné par l'utilisateur pour l'exécution du workflow `WF_input2`). Ce paramètre est passé à `ex002_input2`.
La valeur de la variable `simpleinput2` du deuxième service jGASW reçoit par contre le fichier extrait du résultat de l'exécution du premier service. Après exécution du deuxième

service jGASW, les fichiers résultats sont extraits de la même façon que celle décrite ci-dessus et ils sont transmis un par un en tant que sorties du workflow (voie annexe A.3).

## 4.4 Discussion

Plusieurs plateformes de composition de services web utilisant les technologies sémantiques ont été proposées et certaines d'entre elles soumises au W3C. Cependant, la composition de services web reste toujours un problème difficile à résoudre à cause de leurs hétérogénéités. Notre travail est une extension de OWL-S pour permettre la composition des services web non-standard.

Nous nous sommes appuyés sur la spécification OWL-S, car c'est une ontologie bien définie [117] basée sur des solutions antérieures multiples et elle est actuellement en cours de soumission au W3C. Elle représente aussi une plateforme sémantique qui fournit des spécifications plus complètes que d'autres plateformes. Elle est aussi représentée en OWL qui est un langage standardisé, facile à utiliser et qui a donné lieu au développement de nombreux outils qui facilite la tâche de composition et de raisonnement [206]. En outre, nous avons pu tirer parti de notre ontologie OntoNeuroLOG dans les raisonnements visant à effectuer différentes vérifications sémantiques que nous souhaitions implémenter.

OWL-S est une ontologie multi-couches ce qui fait d'elle une ontologie facile et souple à manipuler. Dans cette contribution, le fait d'étendre deux couches différentes pour deux besoins différents montre bien l'extensibilité [82] [50] de cette spécification. Le service gounding de OWL-S est conçu pour être adapté à n'importe quel type de grounding. Malheureusement dans notre cas, nos fichiers WSDL sont vraiment spécifiques et ne peuvent pas être encapsulés par la couche grounding. Le fait de représenter l'output du service comme une boîte unique et d'avoir les résultats dans une seule enveloppe SOAP rend la génération automatique du grounding concernant cette output ambiguë. Malgré tout, la spécification OWL-S a pu être étendue pour contourner le problème.

WSDL-S et SAWSDL définissent comment ajouter des annotations sémantiques à des WSDL. Au moyen de références à des concepts sémantiques via l'attribut ModelReference. Cet attribut a été conçu pour assigner à des types complexes à un ou plusieurs concepts sémantiques, via la propriété schemaMapping. Aussi, par l'intermédiaire des Conditions et Effects pourvu de faciliter la découverte de services et par l'intermédiaire serviceCategory pour faciliter la publication des services.
Contrairement à la spécification OWL-S, ce type de solution externalise le domaine d'application et les mécanismes de raisonnement. Ces solutions ne sont pas envisageables pour notre application pour plusieurs raisons :

- la forme de la WSDL (nombre de paramètres qu'elle comporte) change. Ceci conduirait à une incompatibilité avec la plateforme jGASW ;
- le grounding doit être interprété manuellement et la composition de services n'est pas explicite. Les solutions ne traitent pas le contexte d'exécution, ni l'aspect comportemental des services.

Nous avons préféré utiliser un langage plus sophistiqué et développé ayant des mécanismes de raisonnement spécifiques.

S'inspirant de ces deux contributions, YASA4WSDL [35] (Yet Another Semantic Annotation 4 WSDL) est une plateforme qui cherche à étendre le standard du W3C à l'aide d'une ontologie technique spécifique comportant des descriptions sémantiques des services et une ontologie de domaine spécifique à la couche métier. La contribution améliore la composition de services mais elle n'est pas applicable pour notre application car elle se base sur SAWSDL qui modifie la structure de base des WSDLs.

La composition de web services est toujours une tâche complexe. De nombreuses ressources bibliographiques sont relatives à la composition de services Web [164] [125] [51]. Dustdar et Schreiner [51] ont discuté la nécessité de la composition des services web et ils ont souligné l'importance du contexte dans la composition de services web. Celui-ci doit être représenté d'une manière adéquate pour une utilisation efficace par le prochain service à invoquer ou pour une utilisation efficace par le service de découverte. Dans notre travail, nous avons su faire face aux mêmes exigences concernant le problème de composition. L'enrichissement de OWL-S vise à représenter les sorties afin de les rendre adéquates pour le service suivant. La composition des services jGASW est le principal résultat de ce travail.

Rao et Su [164] ont étudié la composition automatique des services web et ont proposé une plateforme abstraite pour la composition automatique de services. Ils discutent l'impact de l'hétérogénéité des sources des services web sur le modèle abstrait des processus des workflows. Nous remarquons que la composition de services web devient plus difficile si nous nous écartons des standards pour passer à des plateformes spécifiques. Par exemple, la sélection automatique et la composition automatique fonctionne bien lorsque nous utilisons des standards du W3C. Elles sont par contre difficiles si nous travaillons dans un cadre spécifique en dehors des standards. Dans notre travail, les fichiers WSDL de jGASW sont différents des fichiers WSDL standard. Cela montre néanmoins la dépendance des APIs sémantiques des détails techniques les plus fins.

Casati et al. [32] utilisent la notion de template de processus pour modéliser la composition de services web composites. L'outil de composition rassemble les Templates dans une bibliothèque pour les appliquer ensuite sur les services web. Rao et al [164] et Dustdar et al. [51] distinguent la génération statique et dynamique de flux de travail lors de la composition des services : La génération statique est souvent relative aux tâches métiers alors que la génération dynamique est généralement relative aux instances concrètes des e-services. Les deux approches aident au suivi des exécutions des e-services. OWL-S ne fournit aucun support explicite pour le monitoring des services web lors de la composition ou l'exécution de workflows [197]. Le ServiceProfile de OWL-S est seulement une catégorisation des services, principalement faite pour étendre la description sémantique pour faciliter les processus de découverte et de sélection. Dans notre travail, nous ajoutons des mécanismes de vérification sémantique pour mieux assurer le suivi du workflow. Par exemple, lorsque les utilisateurs composent leurs workflows les algorithmes de vérification de cohérence sont là pour assurer la cohérence du worflow de données dans son ensemble.

Cardoso et Sheth [30] tentent de surmonter les problèmes de composition et d'interopérabilité des services. Ils utilisent une approche multidimensionnelle basée sur la médiation à l'aide des ontologies. Medjahed et al. [125] abordent la question de l'interopérabilité à l'aide de règles de composabilité. Actuellement, cette tâche est effectuée manuellement en recherchant les services grâce à un moteur de recherche, puis en les connectant à d'autres services

manuellement. Cependant, différents algorithmes de vérification ont été implémentés au sein de notre plateforme utilisant OWL-S et les informations nécessaires issues de l'ontologie OntoNeuroLOG. A ce stade, notre travail est encore basique et nécessite un effort significatif au niveau de la validation de consistance, la découverte automatique. En effet, ces processus doivent être complétés pour remédier à l'hétérogénéité des services web sémantiques.

Gannod et al. [63] présentent une approche générique pour le grounding des services utilisant la plateforme OWL-S. Les utilisateurs peuvent relier automatiquement ou manuellement le service à sa description WSDL de base. Même si c'est une approche générique ce type de grounding ne répond pas à nos besoins. En effet, ils considèrent que chaque point d'accès à la WSDL (ou autres) définit les sorties individuellement. Cependant, dans notre cas, les sorties sont intégrées dans une enveloppe SOAP unique et ne sont pas explicites pour l'API WSDL et ne sont compréhensibles que par notre moteur jGASW selon la complexité de l'output on pourrait utiliser l'éditeur proposé dans [63] pour générer le grounding de toute la plateforme jGASW; ceci supposerait d'éditer les paramètres de la WSDL et les paramètres du service décrits dans le descripteur.
Dans ce travail, nous avons adopté la solution la plus facile en termes d'implémentation. Nous n'avons pas gardé ces deux couches séparées, ce qui est une limitation de notre solution. En fait, la data property `process:hasID` qui a été ajoutée est le seul moyen d'accéder aux éléments WSDL (entrée / sortie), comme ceci a été expliqué dans la section implémentation. Ceci conduit à mélanger les fonctionnalités des deux couches. Ce qui est inévitable compte tenu de la façon dont jGASW récupère le résultat des traitements.

## 4.5 Conclusion

Dans ce travail, nous avons présenté une méthode consistant à étendre la spécification OWL-S pour contourner le problème de grounding des services jGASW. Les applications de wrapping comme jGASW fournissent des formes d'entrées/sorties variées, du fait des exigences fonctionnelles des domaines d'application. Dans notre cas, les solutions sémantiques standardisées ne sont pas suffisantes pour l'enrichissement sémantique et la composition.

Nous avons réussi à résoudre le problème de la composition de services web sémantiques et à ajouter des processus de validation sémantiques et des mécanismes de vérification. Cette solution traite de plusieurs questions relatives à la composition de services web dans le domaine de la neuroimagerie, mais n'a pas été suffisamment testée par les utilisateurs NeuroLOG. Pour que nous puissions évaluer la valeur ajoutée du point de vue de l'utilisateur final, il aurait fallut l'intégrer en entier dans la plateforme NeuroLOG et automatiser la composition des services.

Même si les services dont on dispose possèdent des spécificités techniques, l'idée proposée a un caractère de généralité et traite à la fois des problèmes techniques et sémantiques. Cette solution est complète pour les applications qui encapsulent des services en services web comme jGASW mais est encore susceptible d'améliorations dans la mesure où :

√ Sur le plan sémantique :

- elle ne permet pas la modélisation des rôles ;
- elle ne couvre pas tous les types de services dont on dispose ;

- elle nous limite aux fonctionnalités de API OWL-S alors que nous travaillons avec l'outil MOTEUR et on ne veut pas le substituer par l'outil OWL-S.

√ Au niveau implémentation :

- elle nous oblige à reconcevoir l'API selon le modèle NeuroLOG étant donné qu'elle ne supporte pas le système réparti fédéré (Client/serveur)
- elle ne traite pas une grande quantité de données comme ce qu'il y a dans NeuroLOG sachant que dans ce projet il y a déjà les différents outils choisie pour gérer une grande quantité de données ?

Chapitre 5

# Nouveau modèle pour le partage de services de traitement d'images en Neuro-imagerie

**Sommaire**


**5.1 Introduction . . . . . 111**
**5.2 Outils et contexte du travail . . . . . 112**
**5.3 Proposition d'un modèle de description de service . . . . . 114**
5.3.1 Ontologie des outils de traitement de services . . . . . 114
5.3.2 Articulation avec OntoNeuroLOG . . . . . 117
**5.4 Implémentation des services sémantiques . . . . . 120**
5.4.1 Annotation sémantique des services simples et composites . . . . . 121
5.4.2 Invocation sémantique des services simples et composites . . . . . 121
5.4.3 Génération et application de règles pour produire des métadonnées . . . 121
5.4.4 Vérification de la compatibilité entre le traitement et l'opération . . . . 122
**5.5 Aspect techniques de l'implémentation . . . . . 124**
5.5.1 Architecture de la plateforme et du module sémantique . . . . . 124
5.5.2 Annotation des services simples ou composites . . . . . 124
5.5.3 Annotation des pré/postconditions . . . . . 127
**5.6 Discussion . . . . . 131**
**5.7 Conclusion . . . . . 133**


## 5.1 Introduction

Nous avons vu dans le chapitre précédent que l'extension de OWL-S pour les services jGASW n'a pas rempli toutes les contraintes qu'impose la plateforme NeuroLOG. Ceci vaut également pour tout type de plateforme qui utilise des descripteurs de services non conformes aux standards ou ayant des outils de différents types qui ne se basent pas seulement sur le standard WSDL des services web . Le projet NeuroLOG rassemble un ensemble d'outils de traitement d'images en neuroimagerie, caractérisé par la diversité de ces outils et des plateformes sur lesquelles ils s'exécutent. Dans ce chapitre, nous remédions à ce problème en créant notre propre modèle de composition de services dans le cadre de l'ontologie NeuroLOG.

Dans le cadre de ce projet, l'ontologie OntoNeuroLOG est un élément clé de médiation qui assure le partage et l'interopérabilité des plateformes communicantes. Cette ontologie a

été étendue pour assurer le partage et l'interopérabilité des outils de traitement d'image en neuroimagerie, leur réutilisation et leur invocation dans le contexte de workflow de données. Ce travail a été publié dans la conférence WIMS en 2012 [201].

## 5.2 Outils et contexte du travail

La recherche en imagerie biomédicale, met en évidence une évolution des besoins concernant le partage des ressources dans le contexte d'études multicentriques. Ces ressources peuvent être (i) des données produites par les centres qui coopèrent, et (ii) des outils de traitement utilisés dans ces études.

Pour intégrer des données hétérogènes dans un système fédéré on a recours aux systèmes de médiation de données. Grâce à la médiation basée sur les ontologies [120], le partage de données se fait d'une façon plus facile. Cependant, quand il s'agit d'un domaine spécifique comme la neuroimagerie, le partage et la réutilisation des outils de traitement reste toujours difficile et nécessite de relever plusieurs défis.

Nous présentons dans ce chapitre notre travail concernant le partage des outils de traitement d'images en neuroimagerie dans le cadre du projet NeuroLOG.
Comme ceci a été expliqué aux chapitres précédents, le partage des outils de traitement et des modules d'analyse (workflow) est réalisé en les déployant sous forme de services réunis dans un annuaire de services spécifique au projet NeuroLOG. Leur exécution est assurée grâce à un outil, en occurrence par jGASW [13] qui encapsule les bibliothèques et les packages dédiés à l'exécution de ces outils. Le fonctionnement de jGASW a été décrit dans le chapitre 2, son but est de créer des descriptions XML des entrées, sorties et paramètres des outils de traitement d'images. Ces définitions sont enregistrées dans un descripteur XML, et sont utilisées pour construire la ligne de commande d'invocation d'outils de traitement d'image. En outre, la composition des services est géré par MOTEUR [65]. Ce logiciel représente un moteur de définition de stratégies et de politiques de composition de services à travers des descriptions XML pour combiner et invoquer des services hétérogènes sous forme de workflows de traitement. Ces deux outils ont été conçus et intégrés dans la plateforme pour répondre aux besoins spécifiques des utilisateurs de ces outils de traitement d'images.

L'ontologie OntoNeuroLOG s'appuie sur DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering), une ontologie fondamentale qui fournit à la fois les entités de base (la partie supérieure de la taxonomie des entités) et un cadre philosophique commun qui sous-tend la conceptualisation complète.
L'ontologie OntoNeuroLOG, définit des concepts relatifs à la neuroimagerie pour le partage:

- des études de cas scientifiques, dans une sous ontologie appelée l'ontologie des études de cas scientifiques ;
- des images médicales et notamment IRM, dans l'ontologie des datasets ;
- des sujets et des groupes de sujets dans l'ontologie des examens et des sujets ;
- Les traitements qui peuvent être réalisés par les outils de traitement d'images et les modules d'analyse, dans l'ontologie des datasets processing.

Les ontologies cités ci-dessus sont utilisées par le système ainsi que par les utilisateurs engagés dans les études. Elles permettent d'expliciter les contextes d'exécution des outils de traitement

en annotant, par exemple, les types des données images à mettre en entrée ou bien en sortie des outils de traitement d'images, ou pour annoter manuellement les données créées lors de l'invocation des outils de traitement d'images ou rattacher les résultats produits aux entités concernées (sujet, groupe de sujets, études). Cependant, ces concepts ne sont pas suffisants pour pouvoir partager et réutiliser les outils de traitement d'images et pour pouvoir fournir une description détaillée de l'outil ou du workflow, à partir de laquelle on pourrait raisonner sur les données produites et sur les fonctionnalités des outils et vérifier la consistance des données annotées.

Pour avoir une meilleure compréhension de ces outils et de ces données nous devons donc mettre en place une description sémantique plus détaillée et implémenter différentes vérifications sémantiques spécifiques en fonction des besoins du domaine. La sémantique aidera à mieux contrôler l'exécution et l'orchestration de ces services et à mieux répondre aux contraintes essentielles liées aux processus d'opérationnalisation. En outre, la réutilisation effective des données produites exige qu'elles soient correctement annotées avec des informations sémantiques se rapportant au domaine d'application. En effet, le workflow est un enchainement de plusieurs outils de traitement d'images permettant la création de nouveaux fichiers images. Ceci dépend parfois de plusieurs paramètres comme le type des entrés/sorties et le rôle qu'ils ont joué lors de l'invocation de l'outil. Ce type de fonctionnalité nécessite l'intervention d'outils purement sémantiques qui ont des capacités supplémentaires de raisonnement rendant l'ensemble de ces outils intelligent.

En d'autres termes, les outils de traitement d'images sont partagés à un niveau syntaxique basés sur des descriptions XML. Ce niveau est insuffisant pour permettre de vérifier la consistance du workflow, des traitements et des résultats. Par conséquent, l'information traitée est souvent ignorée car elle est souvent mal comprise ou analysée. L'information produite est mal analysée parce qu'elle manque de sémantique et en l'occurrence il peut y avoir des utilisations inadaptées de ces données de la part des utilisateurs. Par exemple nous souhaitons définir une catégorie de dataset pour les images produites après exécution, et nous souhaitons aussi expliciter dans quel contexte (traitement) ces données ont été produites. Nous devons donc pour cela ajouter des métadonnées associées aux services ainsi que leur contexte d'exécution afin d'assurer la conception et l'utilisation appropriées (invocation, composition) des services et des données résultant de leur exécution.

Si on fait le bilan, nous disposons d'une ontologie spécifique à notre domaine d'application. Elle définit les différentes entités pertinentes dans le domaine de la neuroimagerie. Nous avons également à notre disposition deux outils puissants pour la publication et le partage des outils de traitement d'images dans leur annuaire de services dédiés à la neuroimagerie et qui permettent leur invocation et exécution dans le cadre de la plateforme NeuroLOG. Des outils certes puissants mais qui manquent de sémantique. En comparant les fonctionnalités de ces outils et les attentes des utilisateurs de NeuroLOG ainsi que les objectifs fixés dans le chapitre contexte de travail, nous remarquons qu'elles ne sont pas suffisantes pour garantir une réutilisation efficace de ces outils, notamment à cause du manque d'information par rapport au contexte d'utilisation des outils de traitement d'images, par rapport à leur hétérogénéité et au manque de sémantique des descripteurs des outils utilisés. Ceci constitue un obstacle à leur enchaînement correcte, à la bonne gestion de leurs résultats et à leur utilisation ultérieure.

Dans le cadre de NeuroLOG et de l'approche proposée nous abordons trois aspects d'an-

notation sémantique des outils de traitement d'images pour :

1. assurer l'annotation sémantique selon les concepts et les relations de l'ontologie Onto-NeuroLOG et permettre à l'utilisateur de vérifier si une telle annotation répond bien à ses attentes et aux attentes du domaine de la neuroimagerie en mettant en œuvre des algorithmes de vérification spécifiques ;
2. vérifier si la composabilité de services est possible [51] [124] ;
3. rendre possible l'inférence de nouvelles connaissances sur les données exploitées dans la plateforme.

Ce dernier point est obtenu en ajoutant des règles d'enrichissement sémantique selon la nature de l'outil de traitement d'images. Les nouvelles métadonnées sont générées automatiquement à partir de règles, et viennent enrichir les données de la plateforme expérimentale avec de nouvelles informations précieuses pour les différents utilisateurs expert ou non-expert.

Dans ce chapitre, nous présentons notre ontologie des services partagés et son utilisation au sein de la plateforme NeuroLOG. L'ontologie a en effet été construite en suivant une approche bottom-up pour faciliter le partage, l'invocation et la réutilisation des services dans de nouveaux pipelines de traitement d'images [199].

La suite de ce chapitre est organisée comme suit. La section 5.3 décrit l'ontologie proposée, ainsi que les opérations de traitement sémantique qui produisent et exploitent les annotations sémantiques fondées sur cette ontologie. La section 5.3.2 décrit l'articulation du modèle avec l'ontologie NeuroLOG. La section 5.4 fournit plus de détail sur les fonctionnalités des services sémantiques proposés. La section 5.5 explique l'implémentation des services. La section 5.6 discute l'intérêt de l'approche proposée ainsi que sa valeur ajoutée, elle la situe brièvement par rapport aux travaux de modélisation similaires déjà présentés dans le chapitre état de l'art et finalement la section 5.7 fournit une conclusion.

## 5.3 Proposition d'un modèle de description de service

Le partage des outils de traitement dans un système fédéré nécessite de surmonter l'hétérogénéité de leur implémentation. Ceci comporte deux volets. Le premier est de nature syntaxique, et traite de manière pratique la sélection du logiciel correspondant (par exemple à partir d'un annuaire de services) et son invocation. La seconde est de nature sémantique et concerne la définition homogène et cohérente des classes de traitement dans un domaine d'application et des données mises en jeu au cours du traitement [143]. Dans NeuroLOG, le problème syntaxique est abordé par le biais de l'encapsulation des outils de traitement qui peuvent ensuite être invoqués d'une façon homogène dans tout le système fédéré. Ceci a été réalisé grâce à jGASW et MOTEUR.
La sémantique est un atout pour assurer la consistance et les vérifications relatives aux traitements et aux données utilisées ou produites.

### 5.3.1 Ontologie des outils de traitement de services

La définition de cette ontologie utilise le cadre commun de modélisation utilisé tout au long du projet NeuroLOG. Elle s'appuie sur l'ontologie fondamentale DOLCE et sur un en-

semble d'ontologies de base de modélisation d'entités clés qui sont compatibles avec différents domaines.
Notre modèle d'annotation de services met en évidence quelques notions classiquement impliquées, telles que les notions d'interface d'opération et d'autres concepts qui sont sélectionnés à partir de plusieurs formalismes de composition de services (WSMO, OWL-S, SAWSDL, BPEL voir figure 5.1). Par exemple il utilise les entités suivantes:
**Service :** pour définir un point d'accès au service, selon une approche analogue à celle de OWL-S avec son ontologie des services.
**Interface :** utilisé dans WSMO et facilitant l'extensibilité d'un service selon par exemple différentes classes d'utilisateurs; sur le plan fonctionnel elle ressemble aussi au ServiceProfile de OWL-S mais avec un champ sémantique plus large.
**Opération :** c'est l'entité qui se trouve dans quasiment toutes les spécifications sémantiques ; elle crée un corps au service même si cette notion n'apparait pas dans la spécification de base du service comme par exemple les scripts java traités par les servlets Tomcat ou d'autres scripts qui s'exécutent sur des plateformes spécifiques ; les opérations sont généralement décrites en termes d'entrées/sorties mais sans les représenter sémantiquement en tant qu'entité.
**Variables :** les input-variable et les output-variable, représentent les entrées/sorties des services. Elles peuvent être fonctionnelles comme celles définies dans plusieurs langages sémantiques et plateformes, ou non fonctionnelles (par exemple des constantes comme le centre de gravité du cerveau défini sur trois axes en 3D); il peut aussi s'agir de paramètres techniques d'invocation d'un service qui sont à la base des paramètres de scripts shell comme (-i input -o output qui définit une entrée et une sortie ayant pour options respectivement -i et -o).
Les autres entités viennent de l'ontologie OntoNeuroLOG, comme les dataset et les data-processing qui définissent respectivement les types et les rôles que peuvent jouer les images lors de l'invocation des outils et les fonctionnalités des classes de traitement que réalisent les outils.
**Pré-post conditions:** ont été définies pour enrichir les données créées avec des métadonnées pertinente pour l'analyse et la réutilisation ultérieure des images et des outils de traitement d'images. Un exemple est fourni au niveau de la section implémentation pour illustrer l'utilisation de ces conditions.

Le modèle suivant est spécifique pour l'annotation des orchestrations dans le cadre par exemple des workflows MOTEUR (voir figure 5.2). Pour cela de nouveaux concepts sont définis:
**Mapping :** spécialisé sous la forme de trois sous-types différents, cette entité spécifie l'acheminement des valeurs des variables; au niveau sémantique elle assure une compatibilité de type et de rôles lors de la composition de services. Par exemple un input2input mapping signifie un acheminement de la valeur de l'input d'un workflow à un input d'un service (composite ou non) interne au workflow. Le même comportement est retraduit pour un acheminement d'une valeur d'output d'un service interne vers un input d'un autre service interne via l'entité output2input et de la même façon pour ouptut2ouptut qui réalise un acheminement de valeur de variable entre un output d'un service interne vers un output final du workflow.
**Orchestration:** est un concept issu de l'ontologie WSMO. Il est associé à une opération d'un service de type composite pour indiquer qu'il s'agit d'un workflow MOTEUR. Une orchestration est rattachée aux mapping nécessaire pour assurer l'interopérabilité entre les services.

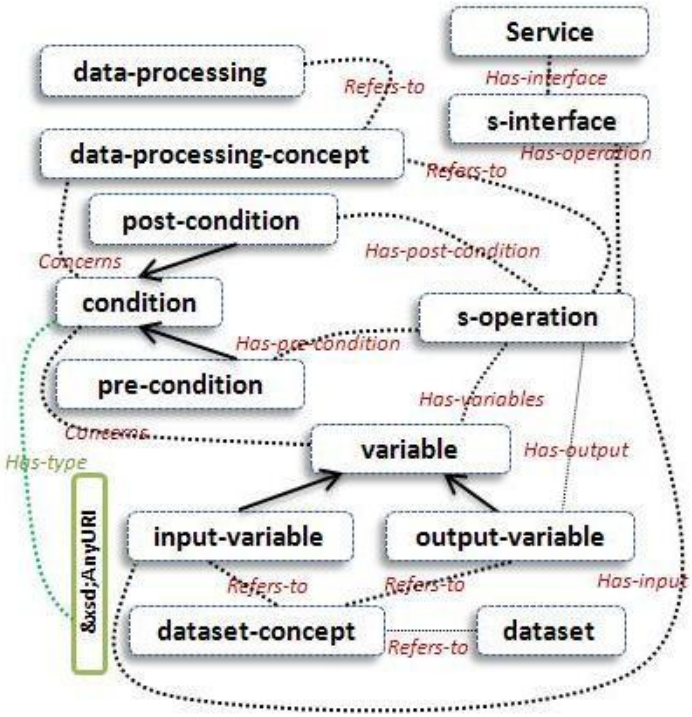


FIGURE 5.1 – Modèle pour la description sémantique des services (outils de traitement d'images dans NeuroLOG)

Un deuxième aspect de cette ontologie traite de l'exécution des opérations (voir figure 5.3) par l'intermédiaire des entités suivantes :
**execution :** cette entité permet de gérer les données de provenance, par exemple, relier à une opération particulière, réalisée au moyen d'un outil de traitement d'images, les instances d'images spécifiées par l'utilisateur, les paramètres utilisés ainsi que les résultats produits.
**variable-value :** elle représente l'instance sémantique affectée à une variable et qui référence par exemple l'image qu'un utilisateur sélectionne en entrée d'un traitement.

Pour enrichir les données de provenance par des informations supplémentaires et permettre à l'utilisateur final de spécifier les annotations à associer aux données en sortie des services, nous avons choisi de recourir à un mécanisme de règles d'inférence. Ce mécanisme n'est pas utilisé pour contrôler le comportement du workflow comme le font la majorité des spécifications sémantiques étudiées dans l'état de l'art mais pour enrichir l'annotation des inputs, outputs, des traitements et de tous les types de données susceptibles d'aider à l'analyse des résultats (voir figure 5.4).
**pré-post conditions :**sont deux entités destinées à être appliquées aux instances avant, lors ou après exécution de l'outil, en fonction de leurs relations (elles concernent quoi ? elles sont en relation avec qui : quel input? quel output?). L'utilisateur est censé définir ces règles lors de l'annotation d'un service ainsi que leurs modalités d'application.

La dernière partie du modèle relie la réalisation concrète du service (les descripteurs

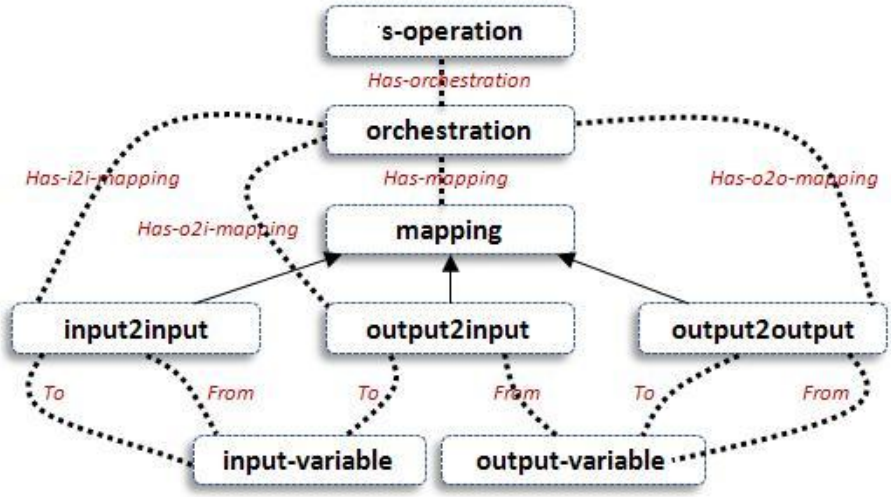


FIGURE 5.2 – Modèle pour la description sémantique des services (outils de traitement d'images dans NeuroLOG)

de base des services) avec les descriptions sémantiques : ceci correspond dans OWL-S au grounding ou dans WSMO aux liaisons vers les fichiers XML (voir figure 5.5).
**Base-document :** fait le lien avec les descripteurs XML des services.
**Input et output argument :** font le lien avec la description des inputs et outputs dans les descripteurs XML des services.

Encore une fois ce modèle est très générique pour pouvoir assurer l'extensibilité et la réutilisation; il ne comporte pas, par exemple, l'aspect opération contrairement au grounding de OWL-S ou de WSMO, pour donner la possibilité à l'utilisateur de faire le grounding des services qui ne possèdent pas une opération comme ceux proposés par la plateforme NeuroLOG.

### 5.3.2 Articulation avec OntoNeuroLOG

#### 5.3.2.1 Modélisation multi-couches et modularité

Lors de l'étude des définitions des ontologies et des différentes ontologies biomédicales nous avons constaté l'existence de plusieurs approches. L'approche que nous avons choisie pour construire nos ontologies est structurée par l'utilisation de différentes ontologies multi-couches ayant plusieurs niveaux d'abstraction, se posant sur une ontologie fondationnelle et des ontologies noyaux. Un aspect important est le caractère modulaire [46] [70] à savoir le dé-

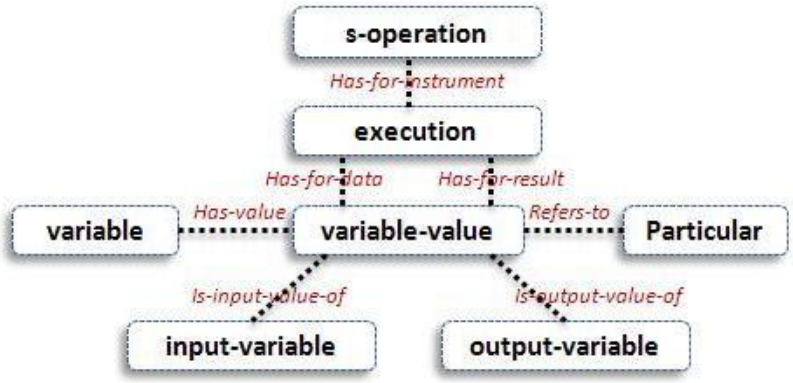


FIGURE 5.3 – Modèle de description d'exécution sémantique

coupage sous forme de modules distincts des différentes parties de l'ontologie, qui réutilisent chacune, une ou plusieurs ontologies noyaux, elles mêmes fondées sur l'ontologie fondationnelle (voir figure 5.6).

#### 5.3.2.2 Catégorisation des datasets et des traitements

Comme mentionné précédemment, les outils de traitement d'images peuvent être classés selon les types de données des entrées/sorties donc selon les types d'images (IRM, TEP, CT, etc) ou selon les rôles des images utilisées comme (image de référence, image flottante ...). La taxonomie des datasets représente les types et les rôles sont définis dans la même taxonomie. Les rôles des images doivent être réalisés physiquement par des Inscription(s) si l'on souhaite suivre le formalisme défini dans l'ontologie iec (Inscription-expression-conceptualisation). Nous avons envisagé de raffiner la taxonomie des datasets en la partageant en deux catégories types/rôles comme montré dans la figure 5.7; elle permet de mieux comprendre la problématique et comment elle a été résolue.

Comme décrit précédemment, nous faisons la distinction entre les instances physiques d'outils de traitement d'images que nous définissons comme des programmes qui font partie du logiciel (outil de recalage par exemple, outil de segmentation) et l'aspect fonctionnel des outils qui désigne l'action de traitement réalisé par les outils comme registration, segmentation etc.

Chaque opération doit effectuer un certain traitement et se réfère donc à la classe de

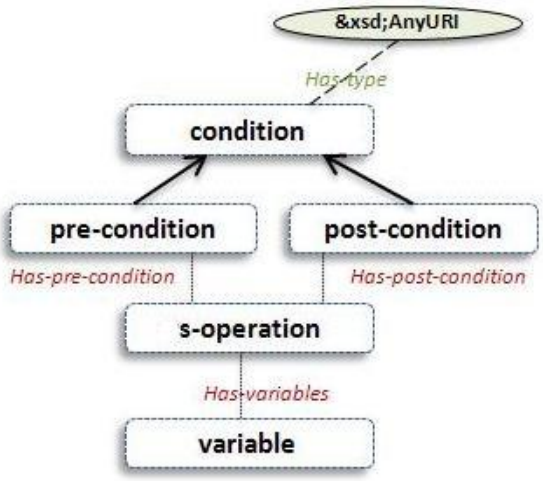


FIGURE 5.4 – Modèle d'enrichissement sémantique à l'aide des règles sémantiques

traitement appropriée. Selon la modularisation définie dans OntoNeuroLOG, l'opération is-an-instrument-of [1] son exécution. La relation Allows-to-carry-out [2] permet de lier un outil à la classe de traitement qu'il exécute.

Le concept "condition" est utilisé pour représenter une "pré-condition" ou "post-condition" qui va être généralement appliqué avant ou après l'exécution de l'outil de traitement d'images. Si on prend l'exemple suivant "condition allows-to-carry-out subjectmeta-data" (type de "condition" "de type" "reasoning"), cela signifie que chaque condition a sa propre sémantique et agit comme un outil d'ajout de métadonnées au data-processing et aux variables impliquées dans cette condition en utilisant la propriété concerns.

Chaque service ou condition a sa propre Inscription [3]. L'objectProperty "realize" indique le document de base "BaseDocument" représenté par sa localisation à l'aide de la dataProperty "hasLocation".

1. Has-for-instrument (inverse de la propriété is-an-instrument-of) est l'object property qui relie une action (un Perdurant) à un particulier qui est utilisé comme instrument pour réaliser cette action.
2. Allows-to-carry-out : défini pour distinguer pour chaque service / outil (logiciel) le type de traitement (raisonnement) qu'il pourra réaliser.
3. Inscription dans le formalisme de l'ontologie NeuroLOG représente les instances physiquement présentes sur un support physique comme les fichiers ou les données enregistrées sur un disque

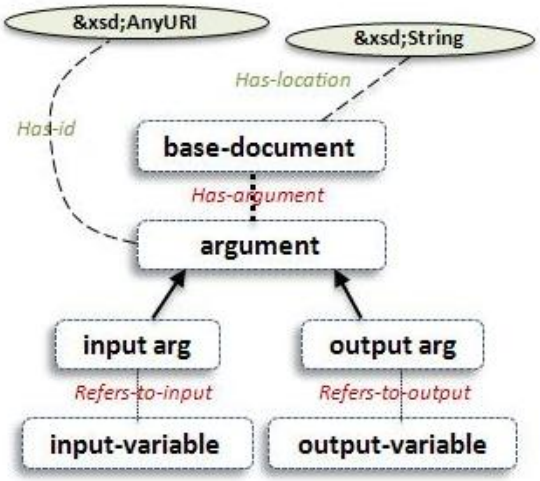


FIGURE 5.5 – Modèle de liaison entre les descripteurs de base des services et les descriptions sémantiques

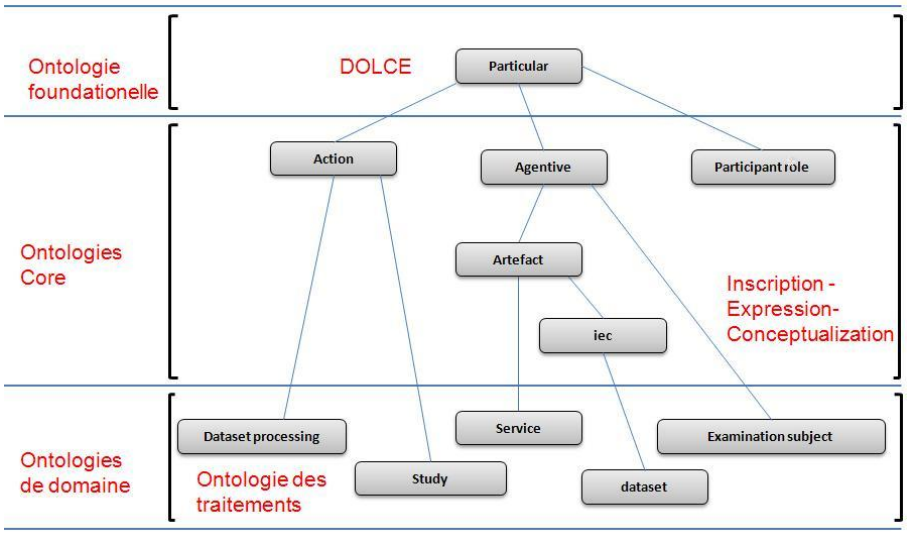


FIGURE 5.6 – Articulation de l'ontologie des services avec les axes des ontologies de DOLCE

## 5.4 Implémentation des services sémantiques

Dans cette section, nous décrivons l'ensemble des services sémantiques mis en œuvre dans le cadre de NeuroLOG.

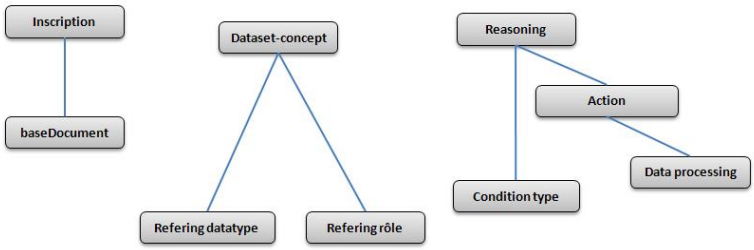


FIGURE 5.7 – Découpage de l'ontologie des dataset en datatype et rôle

### 5.4.1 Annotation sémantique des services simples et composites

Ce module est invoqué lorsqu'un utilisateur annote un service. Il permet d'une part de spécifier la classe de traitement réalisée par ce service et d'autre part les classes d'entités impliquées dans le traitement, comme entrées, sorties ou paramètres. L'algorithme en lui-même consiste à vérifier la cohérence de ces spécifications.
Le grounding des services est également créé au niveau de ce service. Les fichiers XML (les descripteurs jGASW ainsi que les descripteurs GWENDIA de workflow) contiennent les descriptions de base des services et sont liés aux descriptions sémantique à l'aide du grounding. Lors de l'annotation d'un workflow, une entité orchestration et des entités mappings sont créées selon le besoin. Le traitement sémantique correspond à la validation des mappings à l'aide d'un raisonneur.

Sur le plan technique, lors du chargement du fichier GWENDIA dans l'éditeur d'annotation le système va vérifier si tous les services élémentaires possèdent bien une description sémantique enregistrée dans la base de données sémantique. Ensuite, il va céder la main à l'utilisateur pour annoter les entrées/sorties et la classe de traitement et enfin il va vérifier la compatibilité des types de données par rapport aux descriptions sémantiques des services élémentaires.

### 5.4.2 Invocation sémantique des services simples et composites

Ce service est appelé lorsqu'un service (outil de traitement d'images) est invoqué. Il vérifie si les instances d'images sélectionnées par l'utilisateur (par exemple, un dataset) affectées en tant que valeurs de variables répondent bien aux contraintes spécifiées dans les annotations sémantiques du service (voir annexe B.3). En pratique, les vérifications sémantiques se résument en une vérification de subsomption des types de données utilisés entre type d'image sélectionnée par l'utilisateur et type de variable représentant une entrée dans un service.

### 5.4.3 Génération et application de règles pour produire des métadonnées

Les règles sont faites pour l'enrichissement sémantique des fichiers images résultant de l'exécution des services. Nous pouvons ajouter des règles aux services selon le modèle décrit ci-dessus. Au moment de l'exécution, des règles seront générées automatiquement à partir

de leurs annotations. Ainsi, les variables concernées seront remplacées par des valeurs réelles utilisées ou produites. Lorsque la règle est appliquée, les métadonnées sont ajoutées à la base de connaissances (voir annexe B.5).

### 5.4.4 Vérification de la compatibilité entre le traitement et l'opération

Ce service permet aux utilisateurs de s'assurer que la classe de traitement sélectionnée par l'utilisateur et l'annotation du service en termes d'entrées/sorties sont compatibles.
L'algorithme ressemble beaucoup à celui présenté dans le chapitre précédent, la seule différence est qu'il prend en compte les rôles des variables. D'abord, nous créons une classe temporaire new_data_processing relative à l'opération en cours d'exécution, nous convertissons les relations utilisées pour relier l'opération à ses entrées/sorties en axiomes et nous les ajoutons à la classe new_data_processing. Ainsi, pour chaque relation has-input/has-output nous comptons le nombre d'entrées groupées par type de concept de dataset pour déterminer la cardinalité de l'axiome que nous essayons d'ajouter.
La définition du traitement des données diffère de la définition de l'opération et utilise les Object Properties différentes de has-input/has-output. Une troisième étape est alors nécessaire. Celle-ci consiste à sélectionner la propriété appropriée (Object Property) pour la construction de l'axiome selon le rôle exprimé par la deuxième annotation de l'entrée. Par exemple :
Un service MonService

```
(1) MonService has-input i1
i1 refers-to Mr-dataset-concept
Mr-dataset-concept refers-to Mr-dataset
```

Est remplacé par :

```
new_data_processing has-for-data-at Mr-dataset
```

Et

```
(2) MonService has-input i1
       i1 refers-to Foating-dataset-concept
       Floating-dataset-concept refers-to Floating-dataset
       et
       MonService has-input i1
       i1 refers-to Mr-dataset-concept
       Mr-dataset-concept refers-to Mr-dataset
```

Les deux sont remplacés par :

```
       new_data_processing has-for-floating Mr-dataset
```

Si la classe de concept (à laquelle l'input se réfère) se réfère à un dataset-concept qui correspond à un rôle dans la sous-ontologie des dataset, nous remplaçons alors l'Object Property has-for-data-at par la relation adéquate (correspondant à l'Object Property qui a la même sémantique que le concept auquel l'input se réfère). Une fois que l'axiome est créé, on l'ajoute

à la classe temporaire déjà créée.
Par exemple, floating-dataset est un rôle, il ne faut donc pas utiliser la propriété has-for-data-at mais has-for-floating-dataset. Ainsi, nous avons une nouvelle classe de traitement créée à partir de l'opération définie par l'utilisateur. Cette classe de traitement a les axiomes adéquats qui pourront représenter la fonction de l'opération décrite par l'utilisateur. Il reste à vérifier si l'utilisateur a annoté convenablement la fonctionnalité de l'opération de l'outil ce qui nous conduit à la dernière étape de cet algorithme.

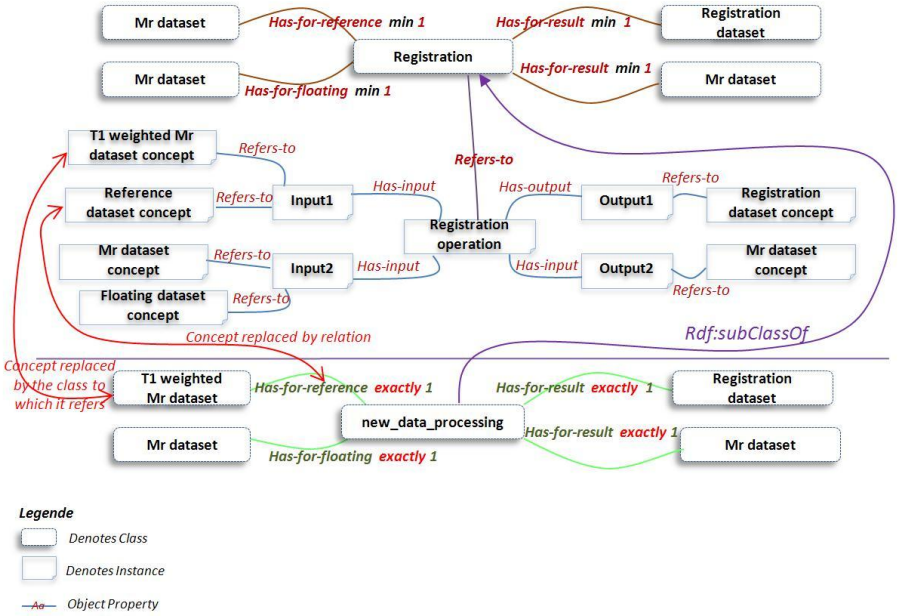


FIGURE 5.8 – Exemple illustratif de la transformation d'une opération en une classe de traitement en utilisant les rôles

Pour annote l'opération l'utilisateur rajoute les relations (refers-to) représentées en noir dans la figure (on remarque que chaque input par exemple est annoter par deux datasets le premier est fait pour le type de l'input l'autre est fait pour le rôle de la variable). Le module sémantique applique l'algorithme décrit en haut pour vérifier la compatibilité entre l'opération et la classe de traitement (Registration-operation, Registration).

Elle consiste à rajouter la classe new_data_processing avec ses nouveaux axiomes comme sous-classe de la classe de traitement sélectionnée par l'utilisateur, et ensuite, classifier l'ontologie et vérifier sa cohérence. Si l'ontologie est consistante alors l'annotation est valide sinon elle n'est pas valide et l'utilisateur a du mal annoter l'outil ou l'opération pour obtenir le résultat qu'il voulait. Sémantiquement, la fonctionnalité de l'outil est bien vérifiée, la figure 5.8 montre un exemple illustratif (voir annexe B.1).

## 5.5 Aspect techniques de l'implémentation

### 5.5.1 Architecture de la plateforme et du module sémantique

La figure 5.9 illustre l'architecture de la plateforme NeuroLOG. Le client peut accéder aux différents outils de packaging d'outils de traitement d'images en l'occurrence jGASW et MOTEUR. Une fois qu'il a créé ses services et les a partagés au sein de la plateforme, il pourra les annoter sémantiquement en utilisant le module sémantique.

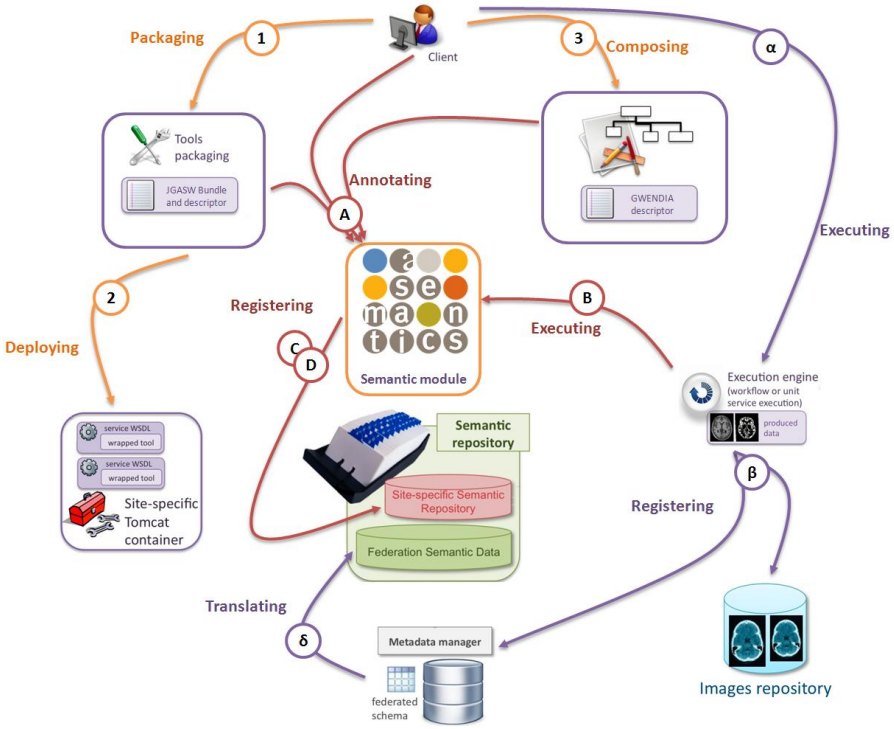


FIGURE 5.9 – Architecture de la plateforme NeuroLOG

La figure suivante 5.10 représente l'architecture interne du module sémantique, la liste des API utilisées ainsi que les différentes étapes d'annotation, de vérification et invocation. Le module sémantique utilise le raisonneur HermiT pour la vérification sémantique, le moteur de recherche sémantique CORESE pour la validation des pré/post conditions et l'API Jena pour la gestion des données sémantiques (enregistrer et lire les données OWL/RDF).
Après la création des instances sémantiques un module de conversion du sémantique en relationnel est utilisé pour enregistrer les données finales dans la base de données relationnelle.

### 5.5.2 Annotation des services simples ou composites

L'annotation sémantique d'un outil de traitement est accessible via l'interface graphique et permet à l'utilisateur de charger un descripteur jGASW ou GWENDIA. Elle met à la disposition de l'utilisateur la taxonomie des datasets et des classes de traitement de sorte que celui-ci puisse les sélectionner et les glisser/Déposer sur les espaces réservés leurs annotations.

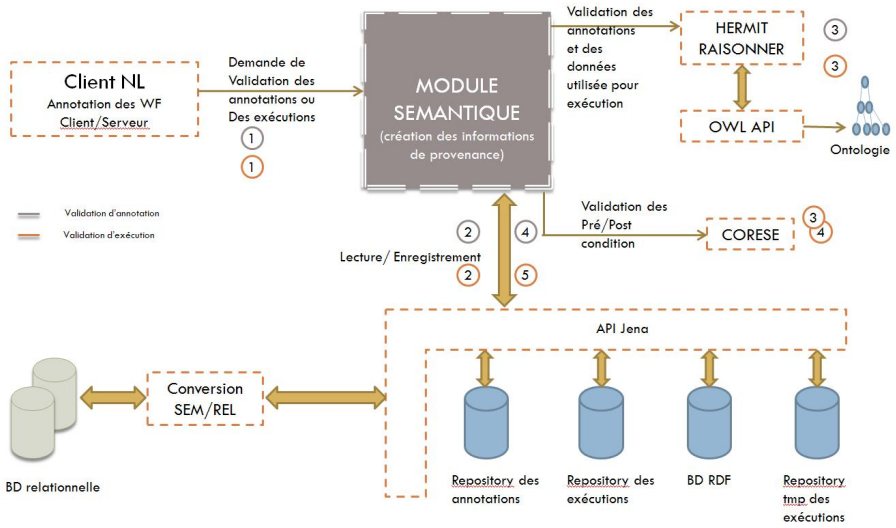


FIGURE 5.10 – Architecture du module sémantique

Ici nous présentons un ensemble de triplets RDF générés après l'annotation d'un fichier GWENDIA

```
<!-- &wf;baseDocument-wfEx12V01.gwendia_416 -->
<ws:BaseDocument rdf:about="&wf;baseDocument-wfEx12V01.gwendia_416">
<ws:has-location rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
/home/.../wfEx12V01.gwendia</ws:has-location>
<ws:has-argument rdf:resource="&wf;inputArgument-Ex1input1_417"/>
<ws:has-argument rdf:resource="&wf;inputArgument-Ex2input2_419"/>
 <ws:has-argument rdf:resource="&wf;outputArgument-output1Ex2_421"/>
 <ws:has-argument rdf:resource="&wf;outputArgument-output2Ex2_423"/>
</ws:BaseDocument>
    <!-- &wf;input-variable_Ex1input1_418 -->
<ws:Input-Variable rdf:about="&wf;input-variable_Ex1input1_418">
<iec:refers-to rdf:resource=
"&ws;#T1-weighted-MR-template-dataset-concept_ind"/>
<ws:is-involved-as-input rdf:resource="&wf;I2Imapping_428"/>
<ws:refers-to-argument rdf:resource=
"&wf;inputArgument-Ex1input1_417"/>
<ws:is-input-of rdf:resource="&wf;operation_wfEx123V01_427"/>
</ws:Input-Variable>
    <!-- &wf;inputArgument-Ex1input1_417 -->
<ws:InputArgument rdf:about="&wf;inputArgument-Ex1input1_417">
<ws:has-id rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
Ex1input1</ws:has-id>
<ws:is-argument-of rdf:resource=
"&wf;baseDocument-wfEx12V01.gwendia_416"/>
<ws:refers-to-variable rdf:resource=
"&wf;input-variable_Ex1input1_418"/>
</ws:InputArgument>
    <!-- &wf;operation_wfEx123V01_427 -->
<ws:ws-Operation rdf:about="&wf;operation_wfEx123V01_427">
<rdf:type rdf:resource="&ws;#ws-Operation"/>
<iec:refers-to rdf:resource="&ws;#de-noising-concept_ind"/>
<ws:has-input rdf:resource="&wf;input-variable_Ex1input1_418"/>
<ws:has-input rdf:resource="&wf;input-variable_Ex2input2_420"/>
```

```
<ws:has-orchestration rdf:resource=
"&wf;orchestration_wfEx123V01_433"/>
<ws:has-output rdf:resource="&wf;output-variable_output1Ex2_422"/>
<ws:has-output rdf:resource="&wf;output-variable_output2Ex2_424"/>
</ws:ws-Operation>
    <!-- &wf;orchestration_wfEx123V01_433 -->
<ws:Orchestration rdf:about="&wf;orchestration_wfEx123V01_433">
 <ws:uses rdf:resource="&ex1-1;operation-ex1-1.0.0_395"/>
<ws:uses rdf:resource="&ex2-1;operation-ex2-1.0.0_1239"/>
<ws:has-mapping rdf:resource="&wf;I2Imapping_428"/>
<ws:has-mapping rdf:resource="&wf;I2Imapping_430"/>
<ws:has-mapping rdf:resource="&wf;O2Imapping_429"/>
<ws:has-mapping rdf:resource="&wf;O2Omapping_431"/>
 <ws:has-mapping rdf:resource="&wf;O2Omapping_432"/>
 <ws:is-orchestration-of rdf:resource=
 "&wf;operation_wfEx123V01_427"/>
</ws:Orchestration>
```

Le bloc ci-dessus représente les annotations sémantiques d'un fichier GWENDIA décrivant un service composite utilisant deux services élémentaires. Le premier est un service de dé-bruitage et le second est un service de segmentation. Nous utilisons CORESE pour récupérer les annotations sémantiques des services internes de l'entrepôt sémantique.
On remarque ici qu'il n'y a pas de grounding d'opération, mais il y a des grounding relatifs aux entrées/sorties décrits dans le fichier GWENDIA (**BaseDocument**). Ce grounding est suffisant pour générer une orchestration (voir annexe B.2). En fait, les mappings sont détectés à partir du fichier GWENDIA. L'opération de validation d'une orchestration se fait en utilisant le raisonneur HermiT. La validation de la cohérence de l'opération avec sa classe de traitement est faite à l'aide de l'API OWL et du raisonneur HermiT.

L'invocation d'un service utilise le moteur de recherche sémantique CORESE pour récupérer les annotations sémantiques associées aux datasets (images) sélectionnés par l'utilisateur à partir des entrepôts de données sémantiques NeuroLOG. Puis il utilise le raisonneur HermiT pour vérifier si les classes de datasets auxquelles appartiennent les images sélectionnées sont subsumées par celles qui sont spécifiées dans les annotations du service (l'annotation des services est extraite par la même instance de CORESE qui a déjà extrait les annotations sémantiques associées aux datasets choisis par l'utilisateur).
Les annotations représentées ci-dessous sont des annotations générées par le module sémantique après l'invocation d'un workflow de données. Ces annotations sont ensuite enregistrées dans la base de données sémantiques à l'aide de l'API Jena.

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://www.irisa.fr/wfEx12V01.gwendia"
     xml:base="http://www.irisa.fr/wfEx12V01.gwendia"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:DBfile="http://www.irisa.fr/DBfile.rdf#"
     xmlns:wf="http://www.irisa.fr/wfEx12V01.gwendia#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:ws="http://www.irisa.fr/web-service-owl-lite.owl#"
     xmlns:iec="http://www.irisa.fr/iec-owl-lite.owl#"
<owl:Ontology rdf:about="&wf;"/>
    <!-- &wf;execution_wfEx123V01_427 -->
```

```
<ws:execution rdf:about="&wf;execution_wfEx123V01_427">
<ws:is-instrument-of rdf:resource="&wf;operation_wfEx123V01_427"/>
<ws:has-for-data-at rdf:resource="&wf;input-vv_Ex1input1_418"/>
<ws:has-for-data-at rdf:resource="&wf;input-vv_Ex2input2_420"/>
<ws:has-for-result-at rdf:resource="&wf;output-vv_output1Ex2_422"/>
<ws:has-for-result-at rdf:resource="&wf;output-vv_output2Ex2_424"/>
</ws:execution>
        <!-- &wf;input-vv_Ex1input1_418 -->
<ws:Variable-Value rdf:about="&wf;input-vv_Ex1input1_418">
<iec:refers-to rdf:resource="&DBfile;#Image1.nii"/>
<ws:is-input-value-of rdf:resource=
"&wf;input-variable_Ex1input1_418"/>
</ws:Variable-Value>
    <!-- &wf;input-vv_Ex2input2_420 -->
<ws:variable-value rdf:about="&wf;input-vv_Ex2input2_420">
<iec:refers-to rdf:resource="&DBfile;#Image2.nii"/>
<ws:is-input-value-of rdf:resource=
"&wf;input-variable_Ex2input2_420"/>
</ws:Variable-Value>
    <!-- &wf;output-vv_output1Ex2_422 -->
<ws:variable-value rdf:about="&wf;output-vv_output1Ex2_422">
<iec:refers-to rdf:resource="&DBfile;#registeredimage1.nii"/>
<ws:is-output-value-of rdf:resource=
"&wf;output-variable_output1Ex2_422"/>
</ws:Variable-Value >
    <!-- &wf;output-vv_output2Ex2_424 -->
<ws:variable-value rdf:about="&wf;output-vv_output2Ex2_424">
<iec:refers-to rdf:resource="&DBfile;#outputImage2.nii"/>
<ws:is-output-value-of rdf:resource=
"&wf;output-variable_output2Ex2_424"/>
</ws:Variable-Value>
</rdf:RDF>
```

### 5.5.3 Annotation des pré/postconditions

Etant donné que nous utilisons CORESE pour la récupération des annotations des images ainsi que les annotations des outils, nous avons choisi de représenter les pré-post conditions en format CORESE.

L'exemple suivant illustre comment les pré-postconditions sont annotées et générées en format CORESE:

Après l'invocation de l'outil de recalage (Registration) nous devons sauvegarder dans l'entrepôt de données sémantiques le fait que, l'image résultante enregistrée ***concerne*** le même sujet ou groupe de sujets que l'image flottante ***utilisée*** dans le recalage. La règle vise à fournir une sémantique plus riche aux données générées que celle que les outils de traitements peuvent fournir, car ils se focalisent exclusivement sur le traitement à effectuer.

Sémantiquement la règle a pour type "RefersTo" (ce sont les types prédéfinis dans la base de connaissances par les spécialistes); elle exprime que : "L'instance d'ouput1 doit se référer au même sujet ou groupe de sujets que celle de input1".

L'annotation sémantique de la règle "RefersTo" est la suivante :

**Définition des namespaces:**

outil:http://www.irisa.fr/registration.owl (description de l'outil)

s:http://www.irisa.fr/web-service-owl-lite.owl (ontologie de services Web)

dp:http://www.irisa.fr/data-processing-owl-lite.owl (ontologie de traitement des données)

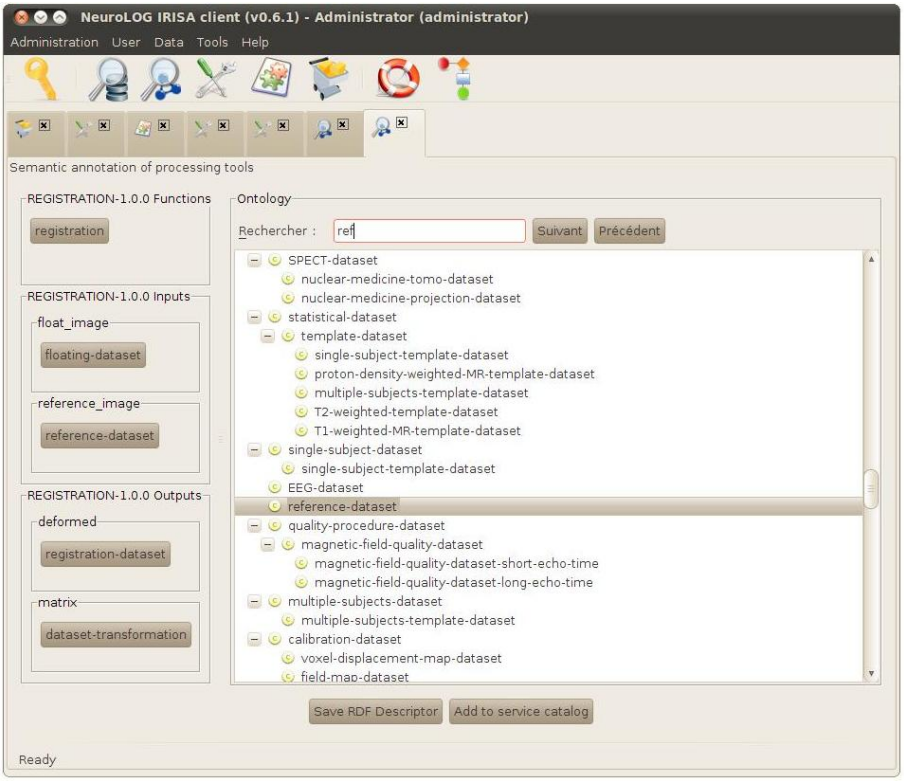


FIGURE 5.11 – Annotation des services élémentaires : annotation de l'outil de recalage : Registration

rs:http://www.irisa.fr/resource.owl (ressources temporaires extraites de la base de données sémantique (instances de concepts traitement des données ...))

**Définition des entités impliquées dans la condition**

outil:postC1 s:has-type "RefersTo"
outil:postC1 s:concerns dp:Registration
outil:postC1 s:concerns outil:input1
outil:postC1 s:concerns outil:output1

Les règles sont exprimées selon le format CORESE. Elles permettent d'ajouter des métadonnées aux données nouvellement créées, à travers un processus comprenant trois étapes:
(1) ajouter les triplets RDF concernés;
(2) générer la règle au format CORESE;
(3) appliquer la règle à l'ensemble de données figurant dans la mémoire CORESE. Les métadonnées qui en résultent peuvent être interrogées à l'aide d'une requête spécifique.
Le format des règles CORESE est très simple. Celles-ci présentent deux blocs (si-alors). Le premier contient une requête SPARQL pour sélectionner les triplets RDF concernés et le second génère des annotations et utilise un format spécifique à CORESE.
Par exemple, pour générer le bloc correspondant à la règle décrite ci-dessus, nous créons l'ensemble de triplets RDF pour sélectionner la valeur utilisée pour la variable concernée par la règle.

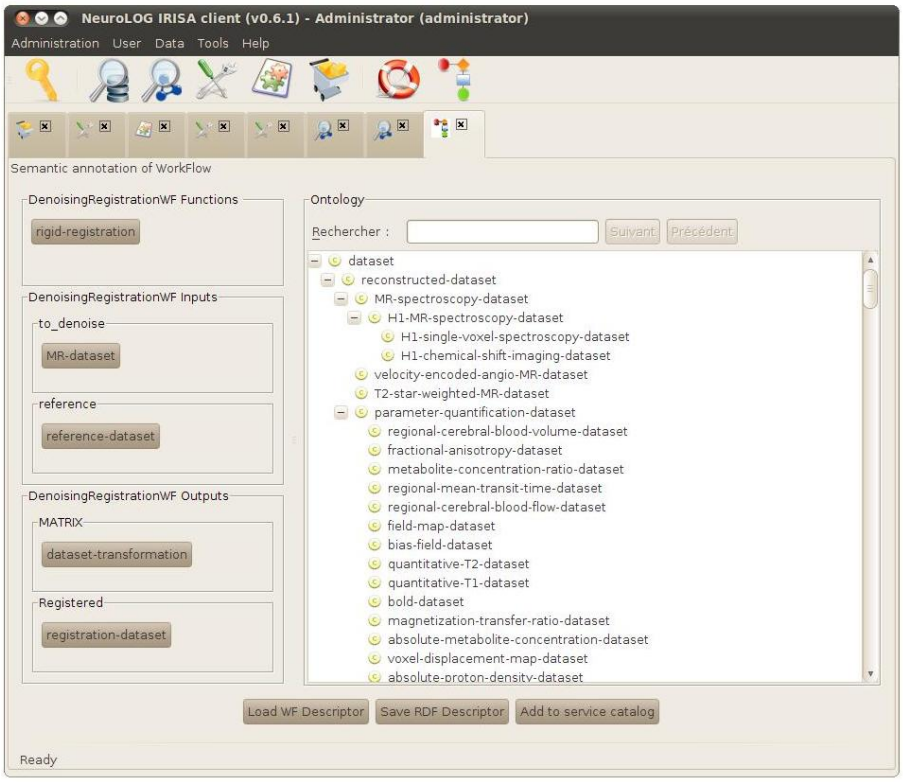


FIGURE 5.12 – Annotation d'un workflow de traitement : débruitage-recalage

Exemple : outil:postC1 s:concerns input1 veut dire que la valeur (qui est un dataset) de la variable input1 est concernée dans cette condition et que le sujet ou le groupe de sujets auquel cette image appartient est concerné aussi. On génère les triplets suivants :
outil:input1 s:has-value ?inputvalue1, pour sélectionner le dataset correspondant au moment de l'exécution. Nous ajoutons le triplet
?inputvalue1 iec:refers-to ?dsinputvalue1, pour sélectionner le sujet ou le groupe de sujets.
On ajoute ensuite le triplet suivant ?dsinputvalue1 iec:refers-to ?particulier1.
Ces triplets RDF sont générés automatiquement à partir de la description de la règle.
Le bloc suivant décrit la règle au format CORESE :

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE rdf:RDF [
<!ENTITY cos "http://www.inria.fr/acacia/corese#">
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
<!ENTITY owl "http://www.w3.org/2002/07/owl#">
<!ENTITY ws "http://www.irisa.fr/web-service-owl-lite.owl#">
<!ENTITY tool "http://www.irisa.fr/registration.owl#">
<!ENTITY iec "http://www.irisa.fr/iec-owl-lite.owl#"> ]>
<rdf:RDF xmlns:rdfs="&rdfs;"xmlns:rdf="&rdf;"xmlns:cos="&cos;"
xmlns:owl="&owl;"xmlns:xsd="&xsd;"xmlns:ws="&ws;
```

```
"xmlns:tool="&tool;"xmlns:iec="&iec;">
<cos:rule cos:name = '&tool;#postC1'>
<cos:if>
{
<!-- Selon l'input utilisée dans l'annotation des règles
(&tool;#postC1 ws:concern  tool:input1)
on sélectionne la valeur qui a été créer pour l'invocation
de l'outil ?inputvalue1 -->

tool :input1 ws:has-value ?inputvalue1

<!-- Pour cette valeur sélectionnée ?inputvalue1
nous sélectionnons l'image (dataset) sélectionné par
l'utilisateur dans la variable ?dsinputvalue1 (cette image
est déjà annotée et figure déjà dans l'entrepôt de
données sémantique) -->

?inputvalue1 iec:refers-to  ?dsinputvalue1

<!--  Pour le dataset séléctionné dsinputvalue1 on sélectionne
le particulier (sujet ou le groupe de sujets) auquel il se
réfère dans ?particular1-->

?dsinputvalue1 iec:refers-to ?particular1

<!-- http://www.irisa.fr/service_owl-lite#concern tool:output1
According to output concerned (output1) avec le triplet
d'annotation de l'output concerné dans la règle nous
sélectionnons la variable créée lors de l'execution
dans la variable ?outputvalue1-->

tool:output1   ws:has-value ?outputvalue1

<!--  pour cette outputvalue1 nous sélectionnons l'image
(dataset) crée après l'exécution de l'outil de
recalage dans ?dsoutputvalue1 -->
?outputvalue1 iec:refers-to  ?dsoutputvalue1
}
</cos:if>

<cos:then>
{
<!--  Selon la requête posée dans la clause if de la règle
CORESE on rajoute les métadonnées qui consiste en ;
l'image generée doit se référer au même sujet ou
groupe de sujet (particular1) auquel l'input1 se refère -->

?dsoutputvalue1 iec:refers-to ?particular1
}
</cos:then>

</cos:rule>

</rdf:RDF>
```

Lors de l'implémentation nous avons créé un processus qui permet de générer aussi la requête SPARQL spécifique à partir de l'annotation de la règle pour sélectionner les métadonnées ajoutées. Par exemple:

```
1
2  PREFIX reg: <http://www.irisa.fr/registration.owl#>
3  PREFIX ws: <http://www.irisa.fr/web-service-owl-lite.owl #>
4  PREFIX iec: < http://www.irisa.fr/iec-owl-lite.owl #>
5  Select * where {
6  reg:postC1  ws:concerns ?outputvariables
7  ?outputvariables rdf:type   ws:Output-Variable
8  ?outputvariables ws:has-value ?outputvariablevalues
9  ?outputvariablevalues iec:refers-to ?datasets
10 ?datasets iec:refers-to  ?particular
11 }
```

Dans le repository sémantique, il y a des triplets RDF qui associent les inputs image aux sujets, par exemple :
Au niveau de l'annotation de service:

```
1  tool:registration-operation  ws:has-input  tool:input1
2  tool:registration-operation  ws:has-output  tool:output1
```

Au niveau de l'exécution du service:

```
1  tool:input1 ws:has-value  rs:valuesintput1
2  rs:valuesintput1 ws:refers-to  rs:dataset1 (donnée par l'utilisateur)
```

Triplet RDF générés après l'exécution:

```
1  tool:output1  ws:has-value rs:valuesoutput1
2  rs:valuesoutput1 rs:refers-to rs:registered-Dataset1 (créée)
```

Dans la base de connaissance, nous disposons de :

```
1  rs:Dataset1   iec:refers-to rs:subject1
```

donc les métadonnées qui devraient être créées sont :

```
1  rs:registered-Dataset1 iec:refers-to  rs:subject1
```

## 5.6 Discussion

La réutilisation des outils de traitement d'images dans les systèmes fédérés est entravée par l'hétérogénéité de leur implémentation. Leur encapsulation sous forme de services permet de faciliter leur partage et d'homogénéiser leur invocation. L'hétérogénéité sémantique est un problème plus complexe. NeuroLOG la résout à travers la définition d'une ontologie qui fournit une définition explicite des services, des classes de traitement de données et des classes d'images appelée datasets. En outre, NeuroLOG utilise cette notion de service pour obtenir une meilleure description des fonctionnalités de ces services. Par exemple, l'outil de recalage possède plusieurs fonctionnalités qu'il exploite selon les types d'entrées qu'il utilise comme par

exemple, la normalisation anatomique sur un modèle (Template) ou le recalage rigide d'images appartenant à un même sujet. Cette approche permet donc de décliner l'utilisation d'un même outil sous la forme de différents services et donc d'avoir une interface de service plus simple et un contexte d'application mieux compris. La référence explicite (via les annotations sémantiques) à une classe de traitement, ainsi que la spécification des valeurs autorisées pour les données d'entrée fournissent des fonctionnalités supplémentaires afin de s'assurer que les contraintes essentielles pour une utilisation appropriée sont effectivement remplies. Cela devrait éviter les abus conceptuels et faciliter la réutilisation pour les non-spécialistes.

Notre modèle de service, présenté dans ce chapitre, partage certaines de ses conceptualisations avec d'autres modèles présentés par le W3C (OWL-S, et WSMO), mais il les devance au niveau des aspects conception et modularisation. Tout à fait en accord avec ce qui est décrit dans [141], notre objectif est de définir rigoureusement les concepts en les spécialisant à partir d'un ensemble de concepts déjà définis selon une logique ontologique structurée et suivant les règles de développement d'une ontologie fondationnelle, en l'occurrence DOLCE. De plus, ce modèle est intégré dans un modèle plus étendu comprenant un modèle de définition des traitements (data-processing), et un modèle de définition de données traitées (dataset) relatifs à la neuroimagerie, ce qui aidera à mieux tenir compte des informations de provenance.

OWL-S et WSMO n'ont pas atteint aujourd'hui la maturité. Ils sont encore au stade de soumission au W3C. Ils sont destinés à clarifier la sémantique des services web. OWL-S vise à améliorer l'expressivité des services web et à ajouter la capacité de raisonnement sémantique afin de découvrir, invoquer, composer et gérer des services web. WSMO tente de surmonter les problèmes de données hétérogènes avec les médiateurs. L'objectif principal de WSMO est d'automatiser la plupart des processus de découverte, de sélection, de composition, de médiation et d'exécution des services Web. WSMO y ajoute un découplage fort entre les différents composants et un rôle central de la médiation. L'un des principes fondamentaux de WSMO est la séparation totale entre les différents éléments participant à la composition de services Web. WSMO et OWL-S utilisent les séquences et les structures de contrôle pour gérer l'exécution des workflows de données. Ils sont principalement utilisés dans les systèmes B2B (Business to business). Actuellement telle qu'est implémentée la plateforme NeuroLOG, l'outil MOTEUR se charge partiellement de faire ce que ces outils font sur l'aspect fonctionnel, suivi des invocations et la structure de contrôle. Certainement le contrôle des flux à l'aide de la sémantique et des structures de contrôles serait plus efficace que celui fait par l'outil MOTEUR. Malheureusement, aujourd'hui on ne dispose pas encore de ces fonctionnalités au niveau du modèle sémantique. Des travaux à venir se penchent sur cette thématique pour mieux la discerner.

L'invocation des services sémantiques est déclenchée par MOTEUR comme l'indique le diagramme de séquence B.6; en effet, il possède un pointeur vers l'IRI de l'instance de l'opération du service (s-operation) ou du workflow qui va être invoqué. Les plateformes sémantiques soumises au W3C sont assez complexes et nous obligent à utiliser des modalités de contrôle dont nous n'avons pas besoin, comme les listes et les séquences. De même, WSMO est généralement utilisé pour la médiation. Nous avons essayé de garder un modèle très simple et une proposition souple facile à modifier, étendre et réutiliser. Notre modèle n'oblige pas, par exemple, la réutilisation des mécanismes de raisonnement proposés parce qu'ils sont externalisés, comme dans SAWSDL.

WSMO définit l'orchestration pour: (a) gérer l'aspect comportemental lors de la composition des services web, en les enchainant à l'aide de leurs chorégraphies, (b) faciliter la réutilisation des combinaisons de services, et (c) finalement permettre de vérifier/valider les contraintes posées par les utilisateurs des services [155].

Dans le domaine de la neuroimagerie, les services demandent à consommer des entrées/-sorties spécifiques, par exemple pour segmenter des lésions du cerveau, débruiter et réaligner les images IRM. Donc en se basant seulement sur les WSDL, les descripteurs jGASW ou les descripteurs GWENDIA, qui sont des documents XML pauvres sémantiquement, nous ne réussirons pas à enchaîner les outils d'une façon fiable. De même, l'architecture SOA représente un moyen d'intégration des services web, mais elle aussi manque de sémantique et n'assure pas la composition des services web. Notre proposition évite d'enrichir les services jGASW ou plus précisément les WSDL jGASW parce que comme démontré dans le chapitre précédent elles ont un format spécifique et ne sont compréhensibles que par jGASW.

Avec les règles, nous avons étendu l'expressivité, au niveau de l'annotation de services et de leur exécution. Contrairement à d'autres propositions qui utilisent des règles pour gérer le comportement des services, nous avons essayé d'ajouter plus de sémantique à la base de connaissances pour faciliter ensuite la réutilisation des données. Ces règles ajoutées à l'annotation sémantique des services peuvent toutefois entraîner des conséquences inattendues. Par exemple, l'ajout de différents rôles ou types à certaines images dans le cadre de deux workflows différents peut nuire à l'interprétation cohérente des données de provenance.

En termes de validation de la plateforme, beaucoup d'autres tests logiciels restent encore nécessaires pour évaluer la valeur ajoutée de cette approche. Nous pouvons exploiter le module sémantique pour interroger l'entrepôt de données sémantiques. Nous pouvons aussi valider nos résultats de traitements et nos annotations et métadonnées produites par visualisation directe par des spécialistes, qui pourront nous indiquer si les images résultant des traitements et l'information de provenance ajoutée sont cohérentes.
Sémantiquement, selon le modèle proposé, nous pouvons composer des workflows en les considérant comme un service unique ayant une opération composite. Malheureusement, l'outil MOTEUR ne prend pas en compte cette possibilité parce que lorsqu'on édite un workflow il est *aplati* à ses composants atomiques (services élémentaires). De cette caractéristique, il faudrait réajuster le mode de fonctionnement de l'outil MOTEUR au sein de la plateforme NeuroLOG.

## 5.7 Conclusion

En conclusion, nous sommes convaincus que l'annotation sémantique des outils de traitement partagés en tant que services est un facteur clé pour faciliter leur réutilisation appropriée et leur interopérabilité dans les systèmes fédérés. Ce travail présente un nouveau modèle pour l'annotation sémantique des outils de traitement d'images et quelques traitements sémantiques basés sur ces annotations pour la validation des services de la conception à l'exécution.
La motivation primordiale pour avoir opté pour un nouveau modèle est la nécessité de s'appuyer sur une ontologie de domaine et une ontologie fondamentale. La deuxième contribution

majeure de ce travail est la mise en œuvre de certains services sémantiques pour vérifier la cohérence des flux de travail au moment du design et de l'exécution ainsi que la création de métadonnées sémantiques associées aux résultats, grâce à des règles de production sémantiques.

Chapitre 6

# Discussion

Sommaire


**6.1 Évaluation du travail par rapport aux objectifs . . . . . 135**
6.1.1 Intégration de la sémantique dans le client NeuroLOG . . . . . 135
6.1.2 Intégration de la sémantique dans le middleware NeuroLOG . . . . . 136
6.1.3 Gestion des données sémantiques . . . . . 137
**6.2 Positionnement par rapport à l'état de l'art . . . . . 137**


## Discussion générale

Dans ce chapitre, nous présentons une évaluation des résultats par rapport aux objectifs posés. Ensuite, nous analysons brièvement les problèmes en suspens et nous discutons les solutions possibles en termes de recherche et développement. Enfin, nous exposant les orientations futures de la recherche sur le partage des services et l'apport des technologies sémantique pour leur description, leur découverte et leur composition.

## 6.1 Évaluation du travail par rapport aux objectifs

### 6.1.1 Intégration de la sémantique dans le client NeuroLOG

Nous avons créé un module client de service web sémantiques dans le quel figure toutes les fonctionnalités sémantiques que l'on souhaite mettre en place pour un utilisateur. Ce module accompagne les deux API principales de création de services web ainsi que de composition de services (jGASW et MOTEUR).

*bullet* **Annotation de services :**

- ♢ **Annotation des entrées/sorties :** nous avons mis en place l'annotation des services unitaires ainsi que l'annotation des services composites. Ces annotations sont relatives aux types de données et aux rôles des entrées/sorties des services.
- ♢ **Annotation des opérations de services :** on associe sémantiquement une opération à chaque service, qu'il soit simple ou composite ceci est également vrai lorsque la description de base des services ne fait pas référence à la notion d'opération comme c'est le cas pour les services jGASW ou les scripts(Python, Java etc).

- ♢ **Vérification des compatibilités des types :** nous avons implémenté un algorithme pour vérifier la compatibilité des types des entrées/sorties lors de la composition.
- ♢ **Vérification de la compatibilité entre l'opération et la classe de traitement :** Nous avons développé un algorithme de vérification entre l'opération qu'un service (simple ou composite) effectue et la classe de traitement à laquelle ce service fait référence.

- **Ajout de métadonnées aux informations de provenance :** cette partie a été modélisée d'une façon très simpliste. Le système de création de métadonnées mis en place permet de créer des métadonnées au début, pendant et à la fin de l'exécution des workflows. Il utilise les annotations sémantiques pour générer des règles de création de données sémantiques sous un format spécifique. Aujourd'hui, ce qui a été développé ne répond que partiellement aux exigences de la neuroimagerie.
  Nous n'avions pas conçu de templates de règles adaptables à n'importe quel type de règle SWRL. Ceci conduit à avoir des templates différents selon l'utilisation de la plateforme ou selon une configuration donnée par l'utilisateur.
  L'architecture client/serveur de la plateforme NeuroLOG nous a poussé à concevoir une implémentation sémantique sous forme de client/serveur. Comme le montre le diagramme de séquence (voir annexe B.6), le module sémantique implémenté au sein de la plateforme comporte une partie cliente et une partie serveur. La préparation du contexte d'exécution sémantique sollicitera des services du module sémantique du coté client. L'exécution d'un service elle sollicitera des services du module sémantique du coté serveur. Ceci facilitera la réutilisabilité de notre module sémantique par d'autre plateformes utilisant des architectures client/serveur. Toutefois, ce module nécessite l'adaptation avant son réutilisation.
- **Détail de la composition des services:** la composition de services est une fonctionnalité qui a été implémentée au niveau du client NeuroLOG pour être utilisée ensuite au niveau du middleware NeuroLOG. En réalité, nous ne cherchons pas à automatiser la composition, mais à l'assister et à la fiabiliser au moyen de la vérification sémantiques. En effet, la composition elle-même est réalisée par l'outil MOTEUR. Ceci constituait l'un des des objectifs important de la plateforme NeuroLOG, qui a été en grande partie atteint.
- **Partage des descriptions de services:** cette fonctionnalité est réalisée par les outils de publication et de composition des services. La sémantique qui a été ajouté sert à rendre plus explicite ce qui a été partagé. Ceci concerne notamment la description de la fonctionnalité du service et ses modalités d'usage, en particulier au travers de contraintes de types et de rôles des entrées/sorties.

### 6.1.2 Intégration de la sémantique dans le middleware NeuroLOG

Utilisant une architecture client/serveur qui a été adopté pour la plateforme NeuroLOG, nous avons utilisé les services web pour l'intégration de la sémantique dans le middleware NeuroLOG., sous forme d'un ensemble de plugins ajoutés aux interfaces des outils NeuroLOG.

La couche de communication utilisée est JAX-WS [1] de Sun qui offre aussi tout les protocoles de communication RPC en format SOAP.

### 6.1.3 Gestion des données sémantiques

Les données sémantiques sont traitées depuis le client ou le serveur et sont enregistrées dans des entrepôts de données sémantiques. Les entrepôts de données sémantique sont renforcé par une base de données MySQL dans laquelle on stocke tout les triplets de description et annotation des services.

## 6.2 Positionnement par rapport à l'état de l'art

Les récentes avancées en médecine ont permis non seulement la génération de très grandes quantités de données mais aussi des données de haute complexité. Pour pouvoir analyser ces données, des chaines de traitements complexes doivent être mise en œuvre, assurant par exemple que les prétraitements convenables soient réalisées pour garantir l'efficacité des traitements eux-même. Les méthodes classiques d'analyses de données et d'exploitations de données se trouvent incapables de répondre à l'ensemble des besoins des utilisateurs. Les limites des systèmes actuels concernent notamment la rapidité des traitements, de la facilité d'analyse des résultats et pourquoi pas d'un système qui aide à la vérification de la cohérence des traitements et des interprétations et analyses. Goble et al [67] soulignent le besoin d'avoir tout d'abord des entités et des identifiants partagés, le besoin d'une sémantique unique et commune des ressources partagées et le besoin d'avoir des mécanismes stables d'accès aux ressources. La standardisation est donc un point de passage obligé, et les standards du web sémantique constituent un élément déterminant des solutions à apporter.
Les plateformes qui ajoutent une description sémantique aux ressources partagées et qui reposent sur des standards du W3C facilitent certes l'interopérabilité [86] mais aussi le raisonnement sur ces ressources et par suite leur compréhension par l'humain, les logiciels et les machines. Elles proposent généralement une spécification commune standard pour représenter des connaissances, que les utilisateurs peuvent appliquer à leurs domaines spécifiques. Les standards présentent néanmoins des limites et s'avèrent parfois difficile à concilier avec des contraintes d'implémentation propre à l'application ou aux outils déjà en place. Ceci amène a proposer des adéquations du type de celle que nous avons déduite dans le chapitres 4 à propos de la spécification OWL-S.
Un autre aspect concerne le degré d'intégration souhaitable avec le domaine d'application pour bénéficier de modèle conceptualisant les entités et les processus métiers du domaine à couvrir. Les standards actuel comme SAWSDL sont de ce point de vue sont peu expressifs et ne répondent pas à tous nos besoins. Pour cette raison nous avons eu recourt aux ontologies fondationelles pour articuler convenablement l'ontologie des services et nos ontologies de domaine. B. Smith et W. Ceusters [181], initiateurs de la Fondation Open Biological and Biomedical Ontologies (OBO) expliquent que la principale finalité des ontologies fondationelles est d'apporter une solution au problème de la multiplicité des langages utilisés pour

1. http://jax-ws.java.net/

l'implémentation des ontologies et faciliter l'intégration de données issues de domaines diverses. Ces types de problèmes sont normalement résolus à l'aide des ontologies de domaine et d'algorithmes de détection de similarité entre les concepts [113].

**Environnement de composition de services :** l'idée du partage de ressources de traitement sous forme de service est une révolution aujourd'hui [196] [164]. Il reste alors à déployer les moyens nécessaires pour en bénéficier pleinement. Dans le cadre de notre travail, deux aspects sont abordés. La composition à l'aide des workflows et la composition à l'aide de la sémantique. Nous les avons combinés sous la forme d'une solution qui répond aux besoins du domaine de la neuroimagerie. En occurrence, nous avons gardé les fonctionnalités qu'offrent les systèmes à base de workflows et nous avons ajouté l'aspect sémantique pour mieux vérifier, partager et assurer l'exécution du flux. Le tableau suivant permet d'analyser les systèmes et plateformes (y compris le nôtre) qui composent des services selon plusieurs critères. Nous discutons ci-dessous chaque méthode, ses apports et ses limites. Bio-jETI et jORCA ne supportent que les services web, donc les services ayant une WSDL. Elles intègrent les services en leur ajoutant une description sémantique basée sur OWL qui décrit seulement les entrées/sorties des services. Bio-jETI cherche à résoudre les problèmes d'interopérabilité à l'aide de taxonomies définies en OWL [97]. Ces taxonomies sont représentées à l'aide d'une interface graphique intuitive qui classe les services en fonction de différents critères (fournisseur, types de données ...). La liste des services est raffinée en fonction du premier service sélectionné par l'utilisateur dans la liste. En effet, la nouvelle liste raffinée représente les services candidats sélectionnés suivant les types de données de leurs entrées/sorties compatibles avec le premier service sélectionné.

jORCA [168] une plateforme qui facilite l'intégration des services web. Elle uniformalise les descriptions des différentes ressources sur le web. Ses descriptions proviennent de différentes ontologies disponibles sur Internet comme l'ontologie BioMoby. D'autres outils rassemblent des outils de différents projets. Taverna [144], seahawk [69], MOWserv [163] et de nombreux autres outils ANNOTATOR [172], MOWserv [163], IWWEM [79], RIKEN [175], LONI Pipeline [112] permettent de construire des workflows en composant les services BioMoby [45] dans le domaine de la bioinformatique et sont suceptible de les appliquer dans le domaine de la neuroimagerie bien que le degré de complexité et d'hétérogénéité des données et des constructions de workflows dans les deux domaines ne sont pas les mêmes. Dans notre plateforme, nous ne voulons pas passer à côté des capacités et de la flexibilité des logiciels jGASW et MOTEUR. Ils sont dédiés à la neuroimagerie et représentent un élément clé dans la construction de workflows enchaînant des outils de traitement d'images. En plus parmi eux certains outils manquent de sémantique et ne répondent pas aux besoins du domaine de la neuroimagerie. Pour cette raison, nous avons choisi d'annoter sémantiquement les descripteurs génériques de jGASW et MOTEUR.

D'autres solutions se basent sur des formalismes existants de description logique pour la composition de services. Les solutions, se basant sur OWL-S (COCOA, IRS-III (voir chapitre état de l'art), vérifient la compatibilité des paramètres à l'aide de la logique de description (paramètre équivalent et vérification de la subsomption). Cependant, OWL-S-XPLAN, SHOP2 et MoSCoE ne vérifient que les paramètres équivalents.

| Plateforme-critères | Formalisme des services | Formalisme workflow | Formalisme traitement sémantique |
|---|---|---|---|
| **Bio-jETI [96]** | WSDL/OWL | BPEL ou JAVA | Compatibilité des E/S |
| **jORCA [168]** | WSDL/OWL | JAVA | Compatibilité des E/S |
| **OWL-S XPLAN [90]** | OWL-S | XPDDL | 1. Compatibilité des E/S<br>2. Monitoring des exécutions<br>3. Plusieurs stratégies d'exécution |
| **SHOP2 [179]** | OWL-S | XPDDL | 1. Compatibilité des E/S<br>2. Monitoring des exécutions<br>3. Plusieurs stratégies d'exécution |
| **MoSCoE [154]** | OWL-S | BPEL | 1. Compatibilité des E/S<br>2. Monitoring des exécutions<br>3. Plusieurs stratégies d'exécution |
| **WS-Biozard [202]** | OWL-S | SAWSDL | 1. compatibilité des E/S<br>2. monitoring des exécutions<br>3.Pplusieurs stratégies d'exécution |
| **COCOA [132]** | OWL-S | FSA:Finite State Automate | 1. compatibilité des E/S<br>2. monitoring des exécutions<br>3. plusieurs stratégies d'exécution |
| **WSMF [170]** | WSMO | WSML | 1. compatibilité des E/S<br>2. monitoring des exécutions<br>3. Utilisation des orchestrations |
| **TAVERNA** [105] | XSCUFL | WSDL/JAVA/ XML/Autres | 1. monitoring des exécutions<br>2. Utilisation des structures de contrôle et de différentes stratégies d'invocation |
| **BioMOBY** [45] | JAVA/PERL | JAVA/PERL/ SCUFL | 1. compatibilité des E/S<br>2. Semi composition des WFs<br>3. Annotation par des rôles<br>4. Ajout de mécanismes de raisonnement |
| **Extension de** OWL-S [200] | WSDL /OWL-S | JAVA | 1. compatibilité des E/S<br>2. Semi composition des WFs |
| **Notre modèle** [201] | WSDL/OWL JAVA/PERL PYTHON | JAVA | 1. compatibilité des E/S<br>2. Semi composition des WFs<br>3. Annotation par des rôles<br>4. Ajout de mécanismes de raisonnement<br>5. Utilisation des orchestrations |

TABLE 6.1 – Présentation d'exemples de systèmes de composition de services

D'autres solutions utilisent les capacités de SAWSDL pour composer les services. L'approche offre une flexibilité importante vis-à-vis de la sémantique et les mécanismes de raisonnement offerts étant donnée que les ontologies de domaine sont externalisées et représentent un module à part. Cependant, l'approche présente certaines limites concernant la capacité de l'orchestration et la composition de services, notamment parce qu'il n'existe pas de mécanismes de contrôle. WSMO définit l'entité orchestration pour: (a) s'assurer de l'aspect comportemental lors de la composition des services web, en les liant à leur chorégraphie, (b) faciliter la réutilisation des combinaisons de services, et (c) permettre aux contraintes du client à être contrôlées [156].

WSMO et OWL-S utilisent les séquences et les structures de contrôles pour gérer l'exécution des workflows et leurs ordonnancement. Dans notre plateforme, jGASW et MOTEUR s'occupent de cette tâche. Nous n'avons donc pas besoin de structures de contrôles sémantiques et des séquences dans notre modèle. Comme le montre le diagramme de séquences décrit dans l'annexe B, l'invocation sémantique est toujours déclenchée par MOTEUR, en effet, grâce à l'IRI de l'instance de l'opération qu'il récupère à partir de la base de données sémantiques lors de chaque exécution d'un service ou d'un workflow. Les différents standards du W3C sont complexes et nous obligent à utiliser une partie des détails techniques dont nous n'avons pas forcément toujours besoin, comme les listes et les séquences de OWL-S ou l'aspect médiation d'ontologies de WSMO. Cependant, le modèle proposé est très simple et flexible, il peut donc être réutilisé selon les besoins.

Les technologies actuelles présentent des approches ad-hoc pour surmonter les inadéquations entre la sémantique des modèles de description, les cardinalités et les formats et les structures que les services web hétérogènes utilisent. La plateforme SOPHIE (Semantic web services chOreograPHy servIcE) [10] définit une plateforme qui met en place une couche de médiation utilisant trois modèles différents entre les échanges de messages des services hétérogènes. Le modèle syntaxique détaille la syntaxe de la plateforme sur trois aspects complémentaires: structurel , comportemental et opérationnel. Le modèle structurel traite les provisions des collections d'entités réutilisables qui suivent des différents niveaux d'abstraction. Le modèle comportemental est ciblé pour la description de l'interaction dynamique parmi les entités définie dans le modèle structurel. Le modèle opérationnel, facilite l'interopérabilité avec les différents modèles comportementaux. Finalement, le modèle sémantique donne un moyen pour décrire le modèle structurel et comportemental [9]. Le grounding dans la plateforme SOPHIE prend en considération la nature découplé et distribuée des services web. Elle ne pose aucune restriction par rapport au point d invocation des services et le type de protocole de communication entre ses services. Elle permet aussi d'implémenter le modèle opérationnel comme une extension des descriptions des messages et des description des services ou comme un dépôt central de données sémantiques. La modélisation sémantique de SOPHIE fournit le mécanisme de mapping essentiel pour l'implémentation d un processus métier. Pour évaluer la plateforme SOPHIE, une étude de faisabilité a été faite pour montrer la viabilité de l'approche du point de vue du processus d'ingénierie.

**Notre solution :** rarissimes sont les solutions qui englobent tous les besoins d'un domaine d'application déterminé. Nous avons vu à travers l'analyse des outils et des plateformes dans l'état de l'art ainsi que dans la discussion précédente que la majorité des systèmes existants couvrent un besoin bien déterminé. Dans le travail que nous avons présenté, nous avons

essayé de couvrir les besoins de la neuroimagerie et la possibilité ouverte aux autres domaines équivalents de pouvoir s'approprier les outils et les méthodes s'ils leurs conviennent. Afin de prendre en compte les besoins de la neuroimagerie, nous avons focalisé notre solution sur l'utilisation des technologies du web sémantique. Nous basons notre travail sur deux grandes approches : (i) les ontologies fondationnelles comme élément clé pour articuler des ontologies couvrant des domaines différents et les ontologies de composition de services. Nous avons fait en sorte qu'à chaque étape de la conception il puisse y avoir un découplage entre les outils de base et la sémantique utilisée pour assurer une meilleure flexibilité de la plateforme. Dans le premier travail présenté dans le chapitre 4, la réutilisation des standards assure certes une meilleure interopérabilité mais limite l'utilisateur aux fonctionnalités offertes dans ces standards (dont certaines font éventuellement doublon dans des possibilités déjà existante) et conduit dans le cas échéant à devoir étendre la spécification pour couvrir les besoins spécifique.

La description sémantique proposée dans le chapitre 6 repose sur une ontologie fondationnelle. Ceci offre la possibilité d'articuler d'autres ontologies de domaine ou de réutiliser d'autres ontologies soit générique soit spécifiques aux domaines d'application visés. Cette approche n'est pas triviale car elle nécessite une bonne compréhension du cadre philosophique qui sous tend cette ontologie, mais une fois celle-ci comprise elle facilite la compréhension des entités du domaine. Elle permet aussi d'avoir un raisonnement plus solide et plus adéquat reposant sur un cadre lui même solide et bien pensé. Concernant le travail implémenté dans la plateforme NeuroLOG, on dégagera certes des aspects positifs mais aussi des limites et insuffisances. Les points les plus cruciaux sont les suivant :

- L'interopérabilité sémantique : celle-ci n'existait pas à la base, car le descripteur Gwendia de l'outil MOTEUR présente peu de sémantique. Cette interopérabilité ne porte pas seulement sur les types de données des entrées/sorties des outils lors de la composition mais aussi sur les rôles de ces outils. Ceci aidera les utilisateurs dans leurs réutilisations des outils.
- La vérification et la validation sémantique des outils et des traitements effectués : Ceci est fait à l'aide des algorithmes implémentés qui sollicitent les opérations de services présentés dans l'ontologie des services et les classent de traitement présentés dans l'ontologie des traitements. Celà donnera naissance à une meilleure qualité de workflow et à des analyses meilleures et plus consistantes d'un point de vue scientifique.
- L'ajout de connaissances à travers des règles d'inférence : là-dessus nous avons beaucoup hésité mais finalement nous avons retenu une implémentation utilisant l'API CORESE et le format CORESE. Ce format répond convenablement aux besoins. Les règles générées ajoutent les liens sémantiques nécessaires pour expliciter les contextes d'utilisation des outils de traitement. Elles permettent de constituer une base de données sémantique/relationnelle riche, consistantes et cohérente qui fournit toutes les données de provenance souhaitables sur les données produites au cours des traitements.

Les limites et insuffisances concernant les points suivants :

- Composition des workflows : Malheureusement aujourd'hui cette fonctionnalité ne fonctionne pas sur la plateforme NeuroLOG. Du fait de notre dépendance de l'outil MOTEUR qui, comme on l'a expliqué, "aplati" un workflow en ses services simples, il faut absolument ré-annoter l'ensemble du workflow. Toutefois, cette possibilité reste

valable à l'aide de l'ontologie des services que nous proposons. Une réutilisation de cette ontologie avec un autre outil de composition de services reste envisageable.
- Le système sémantique implémenté ne dispose pas d'un moteur d'invocation sémantique. Il ne fait qu'annoter les outils simples ou composites. Il dépendra toujours d'un système de composition de services.
- Le système de vérification de compatibilité entre une classe de traitement et une opération est efficace pour notre ontologie, mais il nécessiterait encore une généralisation, par exemple sous la forme d'un système de template.
- Le système de règles sémantiques qui sert à ajouter les connaissances aux données résultats des traitements reste spécifique au modèle proposé et nécessiterait la révision pour qu'il soit lui aussi généralisé.

Chapitre 7

# Conclusion

**Sommaire**


7.1 Problème général . . . . . 143
7.2 Travail effectué . . . . . 143
7.3 Apport de la démarche . . . . . 144
7.4 Perspectives . . . . . 145


## 7.1 Problème général

Le partage de services dans le domaine biomédical n'est pas une tâche triviale. Ces services diffèrent d'aspect technique ainsi que physique et sémantique. Le partage de ces services nécessite des technologies et des outils logiciels à la fois puissants et facile à intégrer. Malgré les efforts déployés pour le partage de ressources beaucoup de problèmes demeurent. L'interopérabilité n'est pas le moindre car les ressources de traitement conçues dans des environnements différents sont naturellement hétérogènes et ne peuvent donc être facilement partagées. La consistance des workflows sur le plan métier (la vérification de la compatibilité entres les entrées/sorties lors de l'enchaînement des outils) et la consistance des workflows en termes de structures et de fonctionnalités (la validation de la conformité entre les fonctionnalités recherchées par l'utilisateur et les outils enchaînés) représentent un moyen pour éviter les erreurs lors de l'exécution. De même la traçabilité et le suivie lors de l'annotation des outils simples ou composites et lors de l'annotation des informations de provenance, aident dans une étape ultérieure à mieux maitriser les informations quant à leurs enregistrement (dans le bon endroit dans la base de connaissance et la base de données) mais aussi à avoir une information suffisante pour décider du bon enchaînement des outils.

Des standards ont été développés pour remédier aux problèmes décrits, malgré tout, certains problèmes persistent. La standardisation est considérée comme un moyen prometteur pour favoriser et assurer l'interopérabilité mais elle reste incapable de répondre à certains besoins spécifiques des applications.

## 7.2 Travail effectué

La solution proposée dans le cadre de cette thèse consiste à enrichir les plateformes de partage de services actuelles en s'appuyant sur les technologies du web sémantique. Le principal avantage de ces technologies est qu'elles complètent les infrastructures existantes, sans les

remettre en cause, ce qui confère à la solution une grande flexibilité. Cette flexibilité couvre à la fois le niveau d'enrichissement sémantique apporté, qui découle de la richesse de l'ontologie des services, et le niveau d'exploitation de cette sémantique pour améliorer l'interopérabilité et la fiabilité des ressources ainsi partagées.
Nous avons élaboré une description sémantique des outils de traitement d'images qui utilise différents concepts issues de contributions antérieures. Cette description ontologique a été construite en s'appuyant sur une ontologie fondationnelle et d'autres ontologies représentant des concepts et des relations de base entre les concepts qui faciliteront son extension et son adaptabilité à différents domaines d'application.
Nous avons également pris en considération l'annotation des rôles des entrées/sorties des services pour pouvoir assister l'utilisateur et faciliter l'analyse des données exploitées ou produites. Le développement de mécanismes de vérification lors de la composition (en mode création des workflows) aide aussi à maintenir une consistance dans les résultats mémorisés dans des bases de données sémantiques et relationnelles.
Cependant, il reste beaucoup de choses à améliorer notamment sur la consistance des workflows comme par exemple des algorithmes d'analyses statistiques sur les résultat obtenus, les durées d'exécution et pour éviter les erreurs pendant le temps d'exécution [205].
Le dernier élément qui a été ajouté concerne les règles sémantiques pour l'enrichissement de la base de connaissance sémantique. Cette partie contribue aussi de façon significative au maintien de la cohérence des données produites suite à l'exécution des outils, lors de l'enregistrement des données et leurs données de provenance.
A ce stade, nous avons utilisé un format spécifique (le format CORESE) qui était le plus facile à mettre en place au sein de la plateforme NeuroLOG. Cependant, cette technique reste limitée par rapport à d'autres projets ou plateformes d'enrichissement sémantique. En effet, nous pourrions bénéficier d'autres travaux pour ajouter des règles qui améliorent le système de raisonnement ou, pour assister les utilisateurs à aligner leurs ontologies avec la nôtre [142].

## 7.3 Apport de la démarche

Le travail présenté ici montre la valeur ajoutée des descriptions sémantiques dans le partage de ressources de traitement. Il montre aussi l'apport des descriptions sémantiques dans la facilitation de la composition de services. L'association d'une description sémantique aux services permet leur caractérisation selon leurs fonctionnalités et leurs paramètres d'entrées/sorties. Ainsi l'acheminement des données via les services se trouve complété pour l'acheminement d'informations sémantiques relatives aux types/rôles des ressources utilisées.

Les algorithmes de vérification de consistance des traitements aident partiellement à avoir des workflows consistants. Cependant, ils s'avèrent efficace quand à la fonctionnalité globale sélectionnée du workflow. Ils ne participent pas pleinement à la vérification et la validation sémantique de l'enchaînement des services au sein du workflow. Cette fonctionnalité est traitée grâce aux algorithmes de vérification de types de données des entrées/sorties des services.

Cette approche est généralisable, et peut être appliquée à d'autres domaines d'application. Elle s'appuie sur les standards reconnus à savoir les standards du W3C RDF(S) et OWL.

## 7.4 Perspectives

Les travaux futurs devraient faire face à l'amélioration des mécanismes de raisonnement pour assurer la sélection automatique et la découverte des services. Ce travail devrait s'appuyer sur les algorithmes de vérification. Un outil interactif qui raffine les recherches sur les outils selon les critères de sélection des utilisateurs est également un composant important pour un déploiement applicatif significatif.

Nous souhaitons enfin intégrer des ontologies qui gèrent de façon plus complète les aspects logiciels (version, mise à jour) et différents contextes d'exécution. Ces éléments sont importants à la fois du point de vue de la traçabilité des données et de leur éventuelle reproductibilité et du point de vue de la gestion optimisée des ressources comme utilisation de grilles de calcul notamment.

# Annexe A

## A.1 Exemple de composition de services JGASW avec OWL-S

Cette section contient la description sémantique OWL-S de deux services jGASW simples nommés Service-ex001 et Service-ex002 (La description est générée automatiquement à l'aide de l'API WSDL2OWLS [1]) ainsi que leurs enchaînements sous forme d'un workflow.

**– Source Code: Test1_2.owl**

```
<?xml version="1.0"?>

<!DOCTYPE uridef [
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
<!ENTITY owl "http://www.w3.org/2002/07/owl">
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema">

<!ENTITY service "http://localhost/Service.owl">
<!ENTITY process "http://localhost/Process.owl">
<!ENTITY profile "http://localhost/Profile.owl">
<!ENTITY grounding "http://localhost/Grounding.owl">
<!ENTITY list "http://www.daml.org/services/owl-s/1.2/generic/ObjectList.owl">
<!ENTITY expr "http://www.daml.org/services/owl-s/1.2/generic/Expression.owl">
<!ENTITY shadow-rdf "http://www.daml.org/services/owl-s/1.2/generic/ObjectList.owl">
<!ENTITY DEFAULT "http://localhost/OWLSTest/Test1_2.owl">
]>
<rdf:RDF

xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:swrl="http://www.w3.org/2003/11/swrl#"
xmlns:expr="http://www.daml.org/services/owl-s/1.2/generic/Expression.owl#"
xmlns:service="http://localhost/Service.owl#"
xmlns:process="http://localhost/Process.owl#"
xmlns:list="http://www.daml.org/services/owl-s/1.2/generic/ObjectList.owl#"
xmlns:shadow-rdf="http://www.daml.org/services/owl-s/1.2/generic/ObjectList.owl#"
xmlns="http://localhost/OWLSTest/Test1_2.owl"
xmlns:grounding="http://localhost/Grounding.owl#"
xmlns:profile="http://localhost/Profile.owl#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base="http://localhost/OWLSTest/Test1_2.owl"
>
```

1. http://www.mindswap.org/2004/owl-s/wsdl2owl-s/

```
35
36 <owl:Ontology rdf:about="">
37
38 <owl:imports rdf:resource="&process;"/>
39 <owl:imports rdf:resource="http://localhost/dataset-owl-lite.owl"/>
40 <owl:imports rdf:resource="http://localhost/OWLSTest/extension-Test1_2.owl"↩
       />
41
42 </owl:Ontology>
43
44
45 <service:Service rdf:ID="Test1_2">
46
47 <!-- Reference to the Test1_2 Profile -->
48 <service:presents rdf:resource="#Test1_2_Profile"/>
49
50 <!-- Reference to the Test1_2 Process Model -->
51 <service:describedBy rdf:resource="#Test1_2_Process"/>
52
53 <!-- Reference to the Test1_2 Grounding -->
54 <service:supports rdf:resource="#Test1_2_Grounding"/>
55
56 </service:Service>
57
58 <!-- Reference to the Test1_2 Profile -->
59 <profile:Profile rdf:ID="Test1_2_Profile">
60 <service:presentedBy rdf:resource="#Test1_2"/>
61 <profile:refers-to rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
62 >http://localhost/dataset-processing-owl-lite.owl#registration</profile:↩
       refers-to>
63
64 <!-- Inputs -->
65
66 <profile:hasInput>
67
68 <process:Input rdf:ID="input1">
69 <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
       anyURI"
70 >http://localhost/dataset-owl-lite.owl#T1-weighted-MR-dataset</process:↩
       parameterType>
71 <rdfs:label>input1</rdfs:label>
72 </process:Input>
73 </profile:hasInput>
74 <profile:serviceName>local</profile:serviceName>
75
76 <profile:hasInput>
77 <process:Input rdf:ID="input2">
78 <rdfs:label>input2</rdfs:label>
79 <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
       anyURI"
80 >http://localhost/dataset-owl-lite.owl#T2-weighted-MR-dataset</process:↩
       parameterType>
81 </process:Input>
82 </profile:hasInput>
83
84
85 <!-- Outputs -->
86
87 <profile:hasOutput>
88 <process:Output rdf:ID="output1">
89 <rdfs:label>output1</rdfs:label>
90 <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
```

```
    anyURI"
>http://localhost/dataset-owl-lite.owl#T1-weighted-MR-dataset</process:↩
    parameterType>
</process:Output>
</profile:hasOutput>

<profile:hasOutput>
<process:Output rdf:ID="output2">
<rdfs:label>output2</rdfs:label>
<process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
    anyURI"
>http://localhost/dataset-owl-lite.owl#MR-dataset</process:parameterType>
</process:Output>
</profile:hasOutput>


<profile:hasResult rdf:resource="#HaveSeatResult"/>

<profile:textDescription> generated manually from /home/bwali/Bureau/↩
    wfEx123V01.gwendia </profile:textDescription>

</profile:Profile>


<!-- Reference to the Test1_2_Process Process Model -->


<process:CompositeProcess rdf:about="#Test1_2_Process">
<rdfs:label> This is the top level process for Test1_2_Processs</rdfs:label↩
    >
<rdfs:comment>
Test1_2_Processs is a composite process.

It is composed of a sequence whose components are 3 atomic
processes.
</rdfs:comment>
<process:invocable rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">↩
    true</process:invocable>
<service:describes rdf:resource="#Test1_2_Processs"/>

<process:hasInput rdf:resource="#input1"/>
<process:hasInput rdf:resource="#input2"/>
<process:hasOutput rdf:resource="#output1"/>
<process:hasOutput rdf:resource="#output2"/>

<process:hasResult>
<process:Result rdf:ID="HaveSeatResult">
<process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
<process:withOutput>
<process:OutputBinding>
<process:theParam rdf:resource="#output1"/>
                                                <process:valueSource>
  <process:ValueOf>
   <process:theVar rdf:resource="#Atomic_Process_ex001_output1"/>
   <process:fromProcess rdf:resource="#PerformAtomic_Process_ex001"/>
  </process:ValueOf>
</process:valueSource>
</process:OutputBinding>
</process:withOutput>
<process:withOutput>
<process:OutputBinding>
<process:theParam rdf:resource="#output2"/>
```

```
<process:valueSource>
          <process:ValueOf>
        <process:theVar rdf:resource="#Atomic_Process_ex002_output1"/>
        <process:fromProcess rdf:resource="#PerformAtomic_Process_ex002"/>
          </process:ValueOf>
</process:valueSource>
</process:OutputBinding>
</process:withOutput>
</process:Result>
</process:hasResult>
<process:composedOf>
<process:Sequence>
<process:components>
<process:ControlConstructList>
<list:first>
<process:Perform rdf:ID="PerformAtomic_Process_ex001">
<process:process rdf:resource="#Atomic_Process_ex001"/>
<process:hasDataFrom>
<process:InputBinding>
<process:theParam rdf:resource="#Atomic_Process_ex001_input1"/>
<process:valueSource>
<process:ValueOf>
<process:theVar rdf:resource="#input1"/>
<process:fromProcess rdf:resource="&process;#ThisPerform"/>
</process:ValueOf>
</process:valueSource>
</process:InputBinding>
</process:hasDataFrom>
</process:Perform>
</list:first>
<list:rest>
<process:ControlConstructList>
<list:first>
<process:Perform rdf:ID="PerformAtomic_Process_ex002">
<process:process rdf:resource="#Atomic_Process_ex002"/>
<process:hasDataFrom>
<process:InputBinding>
<process:theParam rdf:resource="#Atomic_Process_ex002_input1"/>
<process:valueSource>
<process:ValueOf>
<process:theVar rdf:resource="#input2"/>
<process:fromProcess rdf:resource="&process;#ThisPerform"/>
</process:ValueOf>
</process:valueSource>
</process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
<process:InputBinding>
<process:theParam rdf:resource="#Atomic_Process_ex002_input2"/>
<process:valueSource>
<process:ValueOf>
<process:theVar rdf:resource="#Atomic_Process_ex001_output1"/>
<process:fromProcess rdf:resource="#PerformAtomic_Process_ex001"/>
</process:ValueOf>
</process:valueSource>
</process:InputBinding>
</process:hasDataFrom>
</process:Perform>
</list:first>
<list:rest rdf:resource="&shadow-rdf;#nil"/>
</process:ControlConstructList>
</list:rest>
```

```
</process:ControlConstructList>
</process:components>
</process:Sequence>
</process:composedOf>
</process:CompositeProcess>


<!-- ############################################ -->

<!--                    Service-ex001 ↩
                                                # -->

<!-- ############################################ -->

<!-- ############################################-->
<!-- #                    Service_ex001                                  # -->

<service:Service rdf:ID="Service_ex001">

<service:supports>
<grounding:WsdlGrounding rdf:ID="WsdlGrounding_Process_ex001"/>
</service:supports>

<service:presents>
<profile:Profile rdf:ID="Profile_Service_ex001"/>
</service:presents>

<service:describedBy>
<process:AtomicProcess rdf:ID="Atomic_Process_ex001"/>
</service:describedBy>

</service:Service>

<profile:Profile rdf:about="#Profile_Service_ex001">

<service:presentedBy rdf:resource="#Service_ex001"/>

<profile:hasInput>
<process:Input rdf:ID="Atomic_Process_ex001_input1">
<rdfs:label>Atomic_Process_ex001_input1</rdfs:label>
<process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
    anyURI"
>http://www.w3.org/2001/XMLSchema#string</process:parameterType>
</process:Input>
</profile:hasInput>

<profile:hasOutput>
<process:Output rdf:ID="Atomic_Process_ex001_output1">
<rdfs:label>Atomic_Process_ex001_output1</rdfs:label>
<process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
    anyURI"
>http://www.w3.org/2001/XMLSchema#string</process:parameterType>
</process:Output>
</profile:hasOutput>
<profile:textDescription>Auto generated from http://localhost:8080/Test1↩
    -1.1.1/jigsaw?wsdl</profile:textDescription>
<profile:serviceName>local</profile:serviceName>

</profile:Profile>

<!-- ############################################ -->
<!-- # Atomic Process ex001                                          # -->
```

```
267
268 <process:AtomicProcess rdf:about="#Atomic_Process_ex001">
269 <service:describes rdf:resource="#Service_ex001"/>
270 <rdfs:label>Atomic_Process_ex001</rdfs:label>
271
272 <process:hasInput rdf:resource="#Atomic_Process_ex001_input1"/>
273
274 <process:hasOutput rdf:resource="#Atomic_Process_ex001_output1"/>
275
276 </process:AtomicProcess>
277
278 <!-- ########################################### -->
279 <!-- # Atomic Grounding ex001                                  # -->
280 <grounding:WsdlGrounding rdf:about="#WsdlGrounding_Process_ex001">
281 <service:supportedBy rdf:resource="#Service_ex001"/>
282
283 <grounding:hasAtomicProcessGrounding>
284 <grounding:WsdlAtomicProcessGrounding rdf:ID="↩
      WsdlGrounding_Atomic_Process_ex001"/>
285 </grounding:hasAtomicProcessGrounding>
286 </grounding:WsdlGrounding>
287
288 <grounding:WsdlAtomicProcessGrounding rdf:about="#↩
      WsdlGrounding_Atomic_Process_ex001">
289 <grounding:wsdlOutputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema↩
      #anyURI"
290 >http://i3s.cnrs.fr/jigsaw#localResponse</grounding:wsdlOutputMessage>
291 <grounding:wsdlOutput>
292 <grounding:WsdlOutputMessageMap>
293 <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
      anyURI"
294 >http://localhost:8080/Test1-1.1.1/jigsaw?wsdl#localResult</grounding:↩
      wsdlMessagePart>
295 <grounding:owlsParameter rdf:resource="#Atomic_Process_ex001_output1"/>
296 </grounding:WsdlOutputMessageMap>
297 </grounding:wsdlOutput>
298 <grounding:wsdlInput>
299 <grounding:WsdlInputMessageMap>
300 <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
      anyURI"
301 >http://localhost:8080/Test1-1.1.1/jigsaw?wsdl#simpleinput</grounding:↩
      wsdlMessagePart>
302 <grounding:owlsParameter rdf:resource="#Atomic_Process_ex001_input1"/>
303 </grounding:WsdlInputMessageMap>
304 </grounding:wsdlInput>
305 <grounding:wsdlInputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
      anyURI"
306 >http://i3s.cnrs.fr/jigsaw#local</grounding:wsdlInputMessage>
307 <grounding:owlsProcess rdf:resource="#Atomic_Process_ex001"/>
308 <grounding:wsdlOperation>
309 <grounding:WsdlOperationRef>
310 <grounding:operation rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
311 >http://localhost:8080/Test1-1.1.1/jigsaw?wsdl#local</grounding:operation>
312 <grounding:portType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
313 >http://localhost:8080/Test1-1.1.1/jigsaw?wsdl#jigsawPort</grounding:↩
      portType>
314 </grounding:WsdlOperationRef>
315 </grounding:wsdlOperation>
316 <grounding:wsdlDocument rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
      anyURI"
317 >http://localhost:8080/Test1-1.1.1/jigsaw?wsdl</grounding:wsdlDocument>
318 </grounding:WsdlAtomicProcessGrounding>
```

```

<!-- ############################################### -->

                <!--                        Service-ex002                #-->

<!-- ############################################### -->



<service:Service rdf:ID="Service_ex002">

<service:supports>
<grounding:WsdlGrounding rdf:ID="WsdlGrounding_Process_ex002"/>
</service:supports>

<service:presents>
<profile:Profile rdf:ID="Profile_Service_ex002"/>
</service:presents>

<service:describedBy>
<process:AtomicProcess rdf:ID="Atomic_Process_ex002"/>
</service:describedBy>

</service:Service>


<profile:Profile rdf:about="#Profile_Service_ex002">

<service:presentedBy rdf:resource="#Service_ex002"/>

<profile:hasInput>
<process:Input rdf:ID="Atomic_Process_ex002_input1">
<rdfs:label>Atomic_Process_ex002_input1</rdfs:label>
<process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
>http://www.w3.org/2001/XMLSchema#string</process:parameterType>
</process:Input>
</profile:hasInput>

<profile:hasInput>
<process:Input rdf:ID="Atomic_Process_ex002_input2">
<rdfs:label>Atomic_Process_ex002_input2</rdfs:label>
<process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
>http://localhost/dataset-owl-lite.owl#MR-dataset</process:parameterType>
</process:Input>
</profile:hasInput>

<profile:hasOutput>
<process:Output rdf:ID="Atomic_Process_ex002_output1">
<rdfs:label>Atomic_Process_ex002_output1</rdfs:label>
<process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
>http://www.w3.org/2001/XMLSchema#string</process:parameterType>
</process:Output>
</profile:hasOutput>
<profile:textDescription>Auto generated from http://localhost:8080/Test2-1.1.1/jigsaw?wsdl</profile:textDescription>
<profile:serviceName>Service_ex002</profile:serviceName>

</profile:Profile>
```

```
376
377 <!-- ################################### -->
378 <!-- # Atomic Process ex002                                    # -->
379
380 <process:AtomicProcess rdf:about="#Atomic_Process_ex002">
381
382 <service:describes rdf:resource="#Service_ex002"/>
383 <rdfs:label>Atomic_Process_ex002</rdfs:label>
384
385 <process:hasInput rdf:resource="#Atomic_Process_ex002_input1"/>
386 <process:hasInput rdf:resource="#Atomic_Process_ex002_input2"/>
387
388 <process:hasOutput rdf:resource="#Atomic_Process_ex002_output1"/>
389
390 </process:AtomicProcess>
391
392 <!-- ################################### -->
393 <!--                  # Atomic Grounding ex002                  # ↩
    -->
394
395 <grounding:WsdlGrounding rdf:about="#WsdlGrounding_Process_ex002">
396
397 <service:supportedBy rdf:resource="#Service_ex002"/>
398
399 <grounding:hasAtomicProcessGrounding>
400 <grounding:WsdlAtomicProcessGrounding rdf:ID="↩
    WsdlGrounding_Atomic_Process_ex002"/>
401 </grounding:hasAtomicProcessGrounding>
402
403 </grounding:WsdlGrounding>
404
405 <grounding:WsdlAtomicProcessGrounding rdf:about="#↩
    WsdlGrounding_Atomic_Process_ex002">
406
407 <grounding:wsdlOutputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema↩
    #anyURI"
408 >http://i3s.cnrs.fr/jigsaw#localResponse</grounding:wsdlOutputMessage>
409 <grounding:wsdlOutput>
410 <grounding:WsdlOutputMessageMap>
411 <grounding:owlsParameter rdf:resource="#Atomic_Process_ex002_output1"/>
412 <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
    anyURI"
413 >http://localhost:8080/Test2-1.1.1/jigsaw?wsdl#localResult</grounding:↩
    wsdlMessagePart>
414 </grounding:WsdlOutputMessageMap>
415 </grounding:wsdlOutput>
416 <grounding:wsdlInputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
    anyURI"
417 >http://i3s.cnrs.fr/jigsaw#local</grounding:wsdlInputMessage>
418 <grounding:wsdlInput>
419 <grounding:WsdlInputMessageMap>
420 <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
    anyURI"
421 >http://localhost:8080/Test2-1.1.1/jigsaw?wsdl#simpleinput2</grounding:↩
    wsdlMessagePart>
422 <grounding:owlsParameter rdf:resource="#Atomic_Process_ex002_input2"/>
423 </grounding:WsdlInputMessageMap>
424 </grounding:wsdlInput>
425 <grounding:wsdlDocument rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
    anyURI"
426 >http://localhost:8080/Test2-1.1.1/jigsaw?wsdl</grounding:wsdlDocument>
427 <grounding:owlsProcess rdf:resource="#Atomic_Process_ex002"/>
```

```
<grounding:wsdlInput>
<grounding:WsdlInputMessageMap>
<grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
>http://localhost:8080/Test2-1.1.1/jigsaw?wsdl#simpleinput1</grounding:wsdlMessagePart>
<grounding:owlsParameter rdf:resource="#Atomic_Process_ex002_input1"/>
</grounding:WsdlInputMessageMap>
</grounding:wsdlInput>
<grounding:wsdlOperation>
<grounding:WsdlOperationRef>
<grounding:operation rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
>http://localhost:8080/Test2-1.1.1/jigsaw?wsdl#local</grounding:operation>
<grounding:portType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
>http://localhost:8080/Test2-1.1.1/jigsaw?wsdl#jigsawPort</grounding:portType>
</grounding:WsdlOperationRef>
</grounding:wsdlOperation>

</grounding:WsdlAtomicProcessGrounding>


<!-- ################################## -->
<!-- # Instance Definition of Test1_2 Grounding                # -->
<!-- ################################## -->
<!-- ################################## -->
<!-- # service:ServiceGrounding                                  # -->
<grounding:WsdlGrounding rdf:ID="Test1_2_Grounding">
<service:supportedBy rdf:resource="#Test1_2"/>
<!-- Collecton of all the groundings specifications -->
<grounding:hasAtomicProcessGrounding rdf:resource="#WsdlGrounding_Atomic_Process_ex001"/>
<grounding:hasAtomicProcessGrounding rdf:resource="#WsdlGrounding_Atomic_Process_ex002"/>

</grounding:WsdlGrounding>

</rdf:RDF>
```

## A.2 Extension de la description sémantique OWL-S

Cette section contient l'extension sémantique faite en fonction de la spécification décrite dans le chapitre 4 pour pouvoir enchaîner les deux services jGASW décrit dans la section précédente A.1.
Elle montre que par exemple l'output "tool;Atomic_Process_ex001_output1" a été étendue en ses trois éléments essentiels "Atomic_Process_ex001_output1_stderr ", "Atomic_Process_ex001_output1_stdout" et Atomic_Process_ex001_output1_output". Chacune de ses outputs possède des paramètres qui l'identifie dans l'ontologie utilisant la description de l'extension.

**– Source Code: extension-Test1_2.owl**

```
<?xml version="1.0"?>

<!DOCTYPE uridef [
```

```
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
<!ENTITY owl "http://www.w3.org/2002/07/owl">
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema">

<!ENTITY service "http://localhost/Service.owl">
<!ENTITY process "http://localhost/Process.owl">
<!ENTITY profile "http://localhost/Profile.owl">
<!ENTITY grounding "http://localhost/Grounding.owl">
<!ENTITY list "http://www.daml.org/services/owl-s/1.2/generic/ObjectList.owl">
<!ENTITY expr "http://www.daml.org/services/owl-s/1.2/generic/Expression.owl">
<!ENTITY shadow-rdf "http://www.daml.org/services/owl-s/1.2/generic/ObjectList.owl">
<!ENTITY tool "http://localhost/OWLSTest/Test1_2.owl">

<!ENTITY DEFAULT "http://localhost/OWLSTest/extension-Test1_2.owl">
]>
<rdf:RDF

xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:swrl="http://www.w3.org/2003/11/swrl#"
xmlns:expr="http://www.daml.org/services/owl-s/1.2/generic/Expression.owl#"
xmlns:service="http://localhost/Service.owl#"
xmlns:process="http://localhost/Process.owl#"
xmlns:list="http://www.daml.org/services/owl-s/1.2/generic/ObjectList.owl#"
xmlns:shadow-rdf="http://www.daml.org/services/owl-s/1.2/generic/ObjectList.owl#"
xmlns:tool ="http://localhost/OWLSTest/Test1_2.owl#"
xmlns="http://localhost/OWLSTest/extension-Test1_2.owl"
xmlns:grounding="http://localhost/Grounding.owl#"
xmlns:profile="http://localhost/Profile.owl#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base="http://localhost/OWLSTest/extension-Test1_2.owl"
>

<owl:Ontology rdf:about="">

<owl:imports rdf:resource="&process;"/>
<owl:imports rdf:resource="&tool;"/>
<owl:imports rdf:resource="http://localhost/dataset-owl-lite.owl"/>

</owl:Ontology>
<process:Output rdf:about="&tool;#Atomic_Process_ex001_output1">
<process:nlogExpandsTo>
<process:NlogParameter rdf:ID="Atomic_Process_ex001_output1_stderr">
<process:hasLabel rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
>stderr</process:hasLabel>
<process:hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
>stderr</process:hasID>
<process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
>http://www.w3.org/2001/XMLSchema#string</process:parameterType>
<process:links rdf:resource="#output1_wfstderr"/>
</process:NlogParameter>
</process:nlogExpandsTo>
<process:nlogExpandsTo>
<process:NlogParameter rdf:ID="Atomic_Process_ex001_output1_stdout">
<process:hasLabel rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
```

```
61 >stdout</process:hasLabel>
62 <process:hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
63 >stdout</process:hasID>
64 <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
       anyURI"
65 >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
66 <process:links rdf:resource="#output1_wfstdout"/>
67 </process:NlogParameter>
68 </process:nlogExpandsTo>
69 <process:nlogExpandsTo>
70 <process:NlogParameter rdf:ID="Atomic_Process_ex001_output1_output">
71 <process:hasLabel rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
72 >simpleoutput</process:hasLabel>
73 <process:hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
74 >simpleoutput</process:hasID>
75 <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
       anyURI"
76 >http://localhost/dataset-owl-lite.owl#T1-weighted-MR-dataset</process:↩
       parameterType>
77 <process:links rdf:resource="&tool;#Atomic_Process_ex002_input2"/>
78 </process:NlogParameter>
79 </process:nlogExpandsTo>
80 </process:Output>
81 <process:Output rdf:about="&tool;#Atomic_Process_ex002_output2">
82 <process:nlogExpandsTo>
83 <process:NlogParameter rdf:ID="Atomic_Process_ex002_output2_stderr">
84 <process:hasLabel rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
85 >stderr</process:hasLabel>
86 <process:hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
87 >stderr</process:hasID>
88 <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
       anyURI"
89 >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
90 <process:links rdf:resource="#output2_wfstderr"/>
91 </process:NlogParameter>
92 </process:nlogExpandsTo>
93 <process:nlogExpandsTo>
94 <process:NlogParameter rdf:ID="Atomic_Process_ex002_output2_stdout">
95 <process:hasLabel rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
96 >stdout</process:hasLabel>
97 <process:hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
98 >stdout</process:hasID>
99 <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
       anyURI"
100 >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
101 <process:links rdf:resource="#output2_wfstdout"/>
102 </process:NlogParameter>
103 </process:nlogExpandsTo>
104 <process:nlogExpandsTo>
105 <process:NlogParameter rdf:ID="Atomic_Process_ex002_output2_output1">
106 <process:hasLabel rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
107 >simpleoutput1</process:hasLabel>
108 <process:hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
109 >simpleoutput1</process:hasID>
110 <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
       anyURI"
111 >http://www.irisa.fr/visages/team/farooq/ontologies/dataset-owl-lite.owl#↩
       MEG-dataset</process:parameterType>
112 <process:links rdf:resource="#output2_wfoutput1"/>
113 </process:NlogParameter>
114 </process:nlogExpandsTo>
115 <process:nlogExpandsTo>
```

```
116 <process:NlogParameter rdf:ID="Atomic_Process_ex002_output2_output2">
117 <process:hasLabel rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
118 >simpleoutput2</process:hasLabel>
119 <process:hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
120 >simpleoutput2</process:hasID>
121 <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
    anyURI"
122 >http://www.irisa.fr/visages/team/farooq/ontologies/dataset-owl-lite.owl#MR↩
    -dataset</process:parameterType>
123 <process:links rdf:resource="#output2_wfoutput2"/>
124 </process:NlogParameter>
125 </process:nlogExpandsTo>
126 </process:Output>
127 <process:Output rdf:about="&tool;#output1">
128 <process:nlogExpandsTo>
129 <process:NlogParameter rdf:ID="output1_wfstderr">
130 <process:hasLabel rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
131 >stderr</process:hasLabel>
132 <process:hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
133 >stderr</process:hasID>
134 <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
    anyURI"
135 >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
136 </process:NlogParameter>
137 </process:nlogExpandsTo>
138 <process:nlogExpandsTo>
139 <process:NlogParameter rdf:ID="output1_wfstdout">
140 <process:hasLabel rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
141 >stdout</process:hasLabel>
142 <process:hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
143 >stdout</process:hasID>
144 <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
    anyURI"
145 >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
146 </process:NlogParameter>
147 </process:nlogExpandsTo>
148 </process:Output>
149 <process:Output rdf:about="&tool;#output2">
150 <process:nlogExpandsTo>
151 <process:NlogParameter rdf:ID="output2_wfstderr">
152 <process:hasLabel rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
153 >stderr</process:hasLabel>
154 <process:hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
155 >stderr</process:hasID>
156 <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
    anyURI"
157 >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
158 </process:NlogParameter>
159 </process:nlogExpandsTo>
160 <process:nlogExpandsTo>
161 <process:NlogParameter rdf:ID="output2_wfstdout">
162 <process:hasLabel rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
163 >stdout</process:hasLabel>
164 <process:hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
165 >stdout</process:hasID>
166 <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#↩
    anyURI"
167 >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
168 </process:NlogParameter>
169 </process:nlogExpandsTo>
170 <process:nlogExpandsTo>
171 <process:NlogParameter rdf:ID="output2_wfoutput1">
```

```
<process:hasLabel rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
>simpleoutput1</process:hasLabel>
<process:hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
>simpleoutput1</process:hasID>
<process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
>http://www.irisa.fr/visages/team/farooq/ontologies/dataset-owl-lite.owl#MEG-dataset</process:parameterType>
</process:NlogParameter>
</process:nlogExpandsTo>
<process:nlogExpandsTo>
<process:NlogParameter rdf:ID="output2_wfoutput2">
<process:hasLabel rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
>simpleoutput2</process:hasLabel>
<process:hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
>simpleoutput2</process:hasID>
<process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
>http://www.irisa.fr/visages/team/farooq/ontologies/dataset-owl-lite.owl#Mr-dataset</process:parameterType>
</process:NlogParameter>
</process:nlogExpandsTo>
</process:Output>
</rdf:RDF>
```

## A.3 Code modifié dans l'API OWL-S pour la composition

Cette section présente le principal code JAVA qui a été rajouté à l'API OWL-S pour la composition des services jGASW.

**– Source Code: ProcessExecutionUtil.java**

```
/*
 * Created 26.12.2008
 *
 * (c) 2008 Thorsten Möller - University of Basel Switzerland
 *
 * The MIT License
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to
 * deal in the Software without restriction, including without limitation the
 * rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
 * sell copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
```

```
20  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
21  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
22  * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER ↩
        DEALINGS
23  * IN THE SOFTWARE.
24  */
25 package impl.owls.process.execution;
26
27 import util.MatrixResult;
28 import java.util.ArrayList;
29 import java.util.Collections;
30 import java.util.HashMap;
31 import java.util.List;
32 import java.util.Map;
33
34
35 import org.mindswap.exceptions.DataFlowException;
36 import org.mindswap.owl.OWLFactory;
37 import org.mindswap.owl.OWLKnowledgeBase;
38 import org.mindswap.owl.OWLValue;
39 import org.mindswap.owls.process.Perform;
40 import org.mindswap.owls.process.variable.Binding;
41 import org.mindswap.owls.process.variable.InputBinding;
42 import org.mindswap.owls.process.variable.ProcessVar;
43 import org.mindswap.owls.vocabulary.OWLS;
44 import org.mindswap.query.ValueMap;
45 import org.slf4j.Logger;
46 import org.slf4j.LoggerFactory;
47 import query.Query;
48 import semantics.DataProcessingCompatiblility;
49 import semantics.ValuesCompatiblities;
50
51
52 /**
53  *
54  * @author unascribed
55  * @version $Rev:$; $Author:$; $Date:$
56  */
57 public class ProcessExecutionUtil
58 {
59     private static final Logger logger = LoggerFactory.getLogger(↩
          ProcessExecutionUtil.class);
60
61     /**
62      * Create an unordered list of <tt>n</tt> integers whose smallest ↩
            number is
63      * <tt>0</tt> and whose largest number is <code>n - 1 + exclude.length↩
            </code>,
64      * that is, excluding all numbers given by <code>exclude</code>; of ↩
            course,
65      * only if they are within the interval [0,n). There are no duplicated
66      * numbers in the list.
67      *
68      * @param n Number of integers to create.
69      * @return A list of integers from <tt>0</tt> to <tt>n</tt>, excluding ↩
            numbers
70      *  specified by <code>exclude</code>.
71      */
72     public static final List<Integer> createRandomIntegers(final int n, ↩
          final int... exclude)
73     {
74         List<Integer> integers = new ArrayList<Integer>(n);
```

```
75          List<Integer> excludes = new ArrayList<Integer>(exclude.length);
76          for (int e : exclude)
77          {
78              excludes.add(Integer.valueOf(e));
79          }
80
81          int i = 0;
82          while (integers.size() < n)
83          {
84              Integer integer = Integer.valueOf(i++);
85              if (excludes.remove(integer)) continue;
86              integers.add(integer);
87          }
88          Collections.shuffle(integers);
89          return integers;
90      }
91
92      /**
93       * This method iterates through all bindings given and tries to put the
94       * bound process variable mapped to its actual value into the target ↩
             value
95       * map according to the data flow specification of each binding. The ↩
             given
96       * perform results map is the source from which we try to get the ↩
             values,
97       * but it must not contain a mapping for {@link OWLS.Process#↩
             ThisPerform}
98       * as key because the target value map represents all values in the ↩
             scope
99       * of ThisPerform.
100      *
101      * @param performResults The map of input and output values associated ↩
             to
102      *  their perform (which consumed respectively produced the values).
103      * @param targetPerform The target map to put parameter and their ↩
             corresponding
104      *  value to.
105      * @param bindings The list of bindings to iterate trough.
106      * @throws DataFlowException In case the data flow specification of ↩
             some
107      *  binding refers to an parameter or parameter values that could not ↩
             be
108      *  found.
109      */
110     public static final void processValues(final Map<Perform, ValueMap<↩
            ProcessVar, OWLValue>> performResults,
111         final ValueMap<ProcessVar, OWLValue> thisPerform,
112         final ValueMap<ProcessVar, OWLValue> targetPerform, final List<? ↩
            extends Binding<?>> bindings)
113         throws DataFlowException
114     {
115         ProcessVar procVar=null;
116         OWLValue paramValue=null;
117                 OWLValue prevValue;
118         final Map<Perform, ValueMap<ProcessVar, OWLValue>> allPerforms = ↩
            new HashMap<Perform, ValueMap<ProcessVar,OWLValue>>
119                     (performResults);
120
121                 /* Vérification de la compatiblité entre l'input et a ↩
                    valeur donnée par l'utilisateur */
122
123                 ValuesCompatiblities.verifyExecutionValues(thisPerform);
```

```
124                 allPerforms.put(OWLS.Process.ThisPerform, thisPerform);
125                 for (final Binding<?> binding : bindings)
126                 {
127                     int indice = 1;
128 
129 
130                     if ( binding instanceof InputBinding ) {
131                         InputBinding ib = (InputBinding)binding;
132                         procVar = ib.getProcessVar();
133 
134                         org.mindswap.owl.OWLIndividual ind;
135 
136                         ind = ib.getProperty(OWLS.Process.valueSource).↩
                                getProperty(OWLS.Process.theVar);
137 
138                         if ( !ind.isType(OWLS.Process.Output) )
139                             ind = null;
140                         paramValue = binding.getValue().↩
                                getValueFromPerformResults(allPerforms);
141                         MatrixResult r=null;
142                         if (ind != null) {
143                             r = Query.getExtentionListAndLinks(ind);
144                             if ( r != null) {
145                                     paramValue = binding.getValue().↩
                                            getValueFromPerformResults(↩
                                            allPerforms);
146                                     for ( int i = 1 ; i < r.getNBLignes↩
                                            (); i++) {
147                                         System.out.println(r.getValue(i↩
                                                , "?id") +" ///////////////↩
                                                 " + r.getValue(i, "?id").↩
                                                length() );
148                                         String value = paramValue.↩
                                                toString().substring(↩
                                                paramValue.toString().↩
                                                indexOf("<"+ r.getValue(i, ↩
                                                "?id") + ">") + r.getValue(↩
                                                i, "?id").length() + 2,↩
                                                paramValue.toString().↩
                                                indexOf("</"+r.getValue(i, ↩
                                                "?id")+">") );
149                 OWLKnowledgeBase kb = OWLFactory.createKB();
150                                         OWLValue newValue  = kb.↩
                                                createDataValue(value, r.↩
                                                getValue(i, "?type"));
151                                         prevValue = targetPerform.↩
                                                setValue ( procVar, ↩
                                                newValue );
152                                     }
153                                     indice =0;
154                                 }
155                         }
156                     }
157 
158                     if (indice == 1 )  {
159                         procVar = binding.getProcessVar();  // theParam
160                         paramValue = binding.getValue().↩
                                getValueFromPerformResults(allPerforms);
161                         prevValue = targetPerform.setValue ( procVar, ↩
                                paramValue );
162                     }
163                     else
```

```
164                         indice = 1;
165
166             if ( prevValue != null )
167             {
168                 if ( procVar instanceof Loc )
169                 {
170                     logger.debug("Local variable {} reassigned with new ↩
                            value. Previous value disposed.", procVar);
171                 }
172                 else
173                 {
174                     throw new DataFlowException ("Attempt to reassign ↩
                            process variable " + procVar +
175                         ", which is forbidden for write-once variables such↩
                                as Link and other non-local " +
176                         "variables. This is likely caused by an invalid ↩
                            data flow specification.");
177                 }
178             }
179         }
180     }
181 }
```

# Annexe B

## B.1 Validation d'une opération et d'une classe de traitement

Cette section renferme le code source qui fait la validation de l'opération par rapport à la classe de traitement à la quelle elle se réfère.

**– Source Code: CtOpValidator.java**

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package fr.anr.techlog.neurolog.semantics.validators;
7
8  import fr.anr.techlog.neurolog.semantics.entities.Service;
9  import fr.anr.techlog.neurolog.semantics.entities.Operation;
10 import fr.anr.techlog.neurolog.semantics.entities.Input;
11 import fr.anr.techlog.neurolog.semantics.entities.Output;
12 import fr.anr.techlog.neurolog.semantics.myfonctionspack.ContextLoader;
13 import fr.anr.techlog.neurolog.semantics.myfonctionspack.Prefixs;
14 import fr.anr.techlog.neurolog.semantics.util.Util;
15 import java.io.File;
16 import java.util.ArrayList;
17 import java.util.HashSet;
18 import java.util.Set;
19 import org.semanticweb.owlapi.model.IRI;
20 import org.semanticweb.owlapi.model.OWLClass;
21 import org.apache.log4j.Logger;
22 import org.semanticweb.owlapi.apibinding.OWLManager;
23 import org.semanticweb.owlapi.model.AddAxiom;
24 import org.semanticweb.owlapi.model.AddImport;
25 import org.semanticweb.owlapi.model.OWLAxiom;
26 import org.semanticweb.owlapi.model.OWLDataFactory;
27 import org.semanticweb.owlapi.model.OWLImportsDeclaration;
28 import org.semanticweb.owlapi.model.OWLObjectExactCardinality;
29 import org.semanticweb.owlapi.model.OWLObjectProperty;
30 import org.semanticweb.owlapi.model.OWLOntology;
31 import org.semanticweb.owlapi.model.OWLOntologyManager;
32
33 /**
34  *
35  * @author bwali
36  */
37 public class CtOpValidator {
38
39     private static final Logger logger = Logger.getLogger(CtOpValidator.↩
        class);
40     private Service webservice=null;
41     private Operation operation=null;
```

```
42      private static OWLOntology ontology = null;
43      private static OWLClass dsp_cpy = null;
44
45      /**
46       *
47       * @param webservice
48       * @param operation
49       */
50      public CtOpValidator ( Service webservice, Operation operation ) {
51          this.webservice = webservice;
52          this.operation=operation;
53      }
54
55      public CtOpValidator ( Service webservice, String operationIRI ) {
56
57          this.webservice = webservice;
58          this.operation=webservice.getWebserviceinterface().↩
                getOperationByIRI(operationIRI);
59      }
60
61      public CtOpValidator ( Service webservice, IRI operationIRI ) {
62
63          this.webservice = webservice;
64          this.operation=webservice.getWebserviceinterface().↩
                getOperationByIRI(operationIRI);
65      }
66
67
68      public boolean verifyCompatibilityDSP_OP () {
69
70          webServiceOperationInstance2DataProcessingClass(operation);
71
72          return testConsistency();
73      }
74
75
76      /**
77       * Convertie une instance de web service operation en une classe de ↩
             traitement temporaire
78       * @param operation : c'est l'operation qui va etre convertie
79       * @return retourne une entology des dataset processing qui importe l'↩
             ontologie des datasets inclus dedans la
80       * nouvelle classe de traitement (en copie "_cpy") elle les mets dans ↩
             Temp
81       */
82
83          public  static void webServiceOperationInstance2DataProcessingClass↩
                ( Operation operation ) {
84
85              String dir = ContextLoader.getOntoDirIRI();
86              OWLOntologyManager manager=OWLManager.createOWLOntologyManager↩
                    ();
87              OWLDataFactory factory = manager.getOWLDataFactory();
88
89              OWLOntology ontoCpy=null;
90              OWLOntology ontoDataset=null;
91              try {
92
93                  try {
94
95                      ontoCpy = manager.loadOntologyFromOntologyDocument ( ↩
                            IRI.create ( dir + "/dataset-processing-owl-lite.↩
```

```
                    owl") );
96                  ontoDataset = manager.loadOntologyFromOntologyDocument (
                        IRI.create ( dir + "/dataset-owl-lite.owl") );
97                  //importing result
98                  OWLImportsDeclaration id =  factory.
                        getOWLImportsDeclaration(IRI.create ( "http://www.
                        irisa.fr/visages/team/farooq/ontologies/dataset-owl
                        -lite.owl" ));
99                  AddImport addimport = new AddImport(ontoCpy, id);
100                 manager.applyChange(addimport);
101                 manager.saveOntology(ontoCpy);
102                 logger.info ( ontoCpy.getDirectImports() );
103             }
104             catch (Exception e) {
105
106             }
107                 // get the operation and convert it to dataset processing
108
109                 OWLClass dataprocessingcpy = factory.getOWLClass(IRI.create
                        (operation.
110                         getDataprocessingconcept().getReferredClass().
                            toStringID() + "_cpy"));
111                 OWLClass dataprocessing = operation.
                        getDataprocessingconcept().getReferredClass();
112
113                 OWLObjectProperty hasForDataAt = factory.
                        getOWLObjectProperty(
114                         IRI.create (Prefixs.HAS_FOR_DATA_AT));
115                 OWLObjectProperty hasForResultAt = factory.
                        getOWLObjectProperty(
116                         IRI.create (Prefixs.HAS_FOR_RESULT_AT));
117                 logger.info("classe de traitement et classe copie ==" +
                        operation.
118                         getDataprocessingconcept().getReferredClass() + "
                            ___copie ___" + dataprocessingcpy );
119                 ArrayList <AddAxiom> l = new ArrayList<AddAxiom>();
120
121                 // sélectionner les différent types d'input qui existe
122                 Set<OWLClass> classes = new HashSet<OWLClass> ();
123                 for ( Input input : operation.getInputs() ) {
124                     // créer l'individu inputDataset
125                     OWLClass ref = input.getParticularConcept().
                            getreferencedClass();
126                     boolean exist = false;
127                     for (OWLClass c : classes ) {
128                         if (c.toStringID().equals(ref.toStringID())) {
129                             exist = true;
130                             break;
131                         }
132                     }
133                     if (exist == false) {
134                         classes.add(ref);
135                     }
136                 }
137                 OWLObjectExactCardinality exactcardinality = null;
138                 OWLAxiom axiom = null;
139                 AddAxiom addaxiom = null;
140
141                 //créer le nombre par type
142                 for (OWLClass c : classes ) {
143                     int number = 0;
144                     logger.info( "passage pour nombre : classes ==" + c );
```

```
145                     for ( Input input : operation.getInputs() ) {
146                         // créer l'individu inputDataset
147                         OWLClass ref = input.getParticularConcept().↩
                                getreferencedClass();
148                         if ( c.toStringID().equals( ref.toStringID() ) ) {
149                             number++;
150                         }
151                     }
152                     // créer l'axiom qui va avec
153                     exactcardinality = factory.getOWLObjectExactCardinality↩
                            (number, hasForDataAt, c);
154                     axiom = factory.getOWLSubClassOfAxiom( ↩
                            dataprocessingcpy, exactcardinality );
155                     addaxiom = new AddAxiom(ontoCpy, axiom);
156                     l.add(addaxiom);
157                 }
158 
159 
160              // faire pareil pour has-for-result-at
161 
162                 classes = new HashSet<OWLClass> ();
163                 for (Output output : operation.getOutputs()) {
164                     // créer l'individu inputDataset
165                     OWLClass ref = output.getParticularConcept().↩
                            getreferencedClass();
166                     boolean exist = false;
167                     for (OWLClass c : classes ) {
168                         if (c.toStringID().equals(ref.toStringID())) {
169                             exist = true;
170                             break;
171                         }
172                     }
173                     if (exist == false) {
174                         classes.add(ref);
175                     }
176                 }
177             //créer le nombre par type
178                 for (OWLClass c : classes ) {
179                     int number = 0;
180                     logger.info( " passage pour nombre : classes == " + c )↩
                            ;
181                     for ( Output output : operation.getOutputs() ) {
182                         // créer l'individu inputDataset
183                         OWLClass ref = output.getParticularConcept().↩
                                getreferencedClass();
184                         if ( c.toStringID().equals( ref.toStringID() ) ) {
185                             number++;
186                         }
187                     }
188                     // créer l'axiom qui va avec
189                     exactcardinality = factory.getOWLObjectExactCardinality↩
                            (number, hasForResultAt, c);
190                     axiom = factory.getOWLSubClassOfAxiom( ↩
                            dataprocessingcpy, exactcardinality );
191                     addaxiom = new AddAxiom(ontoCpy, axiom);
192                     l.add(addaxiom);
193                 }
194                 // une fois la création des axioms , classe de traitement ↩
                        est fini je met la classe créé sous la classe de ↩
                        traitement
195                 //choisie par l'utiilisateur
196                 axiom = factory.getOWLSubClassOfAxiom( dataprocessingcpy, ↩
```

```
                    dataprocessing );
197                 addaxiom = new AddAxiom(ontoCpy, axiom);
198                 l.add(addaxiom);
199                 manager.applyChanges(l);
200                 File file = new File(Util.getTemporaryDirectory() + "/↩
                        muOnto.owl");
201                 manager.saveOntology(ontoCpy, IRI.create(file.toURI()));
202                 // On les stocke dans Temp
203                 ontology=ontoCpy;
204                 dsp_cpy=dataprocessingcpy;
205             }
206             catch (Exception e) {
207                 e.printStackTrace();
208             }
209     }
210
211         /**
212          * Test
213          * @return
214          */
215
216     public boolean testConsistency() {
217
218         boolean b = false;
219         try {
220
221             logger.info ( " get onto on reasoner  " );
222             org.semanticweb.HermiT.Reasoner hermit = new  org.semanticweb.↩
                    HermiT.Reasoner(ontology);
223
224             logger.info ( "End  get onto on reasoner  " );
225             hermit.prepareReasoner();
226             logger.info ( " start to test consistency  " );
227             if ( hermit.isConsistent() )
228             {
229                 logger.info ( " is consistent " );
230                 logger.info ( " start to classify  " );
231                 hermit.classify();
232                 if (hermit.isDefined(dsp_cpy))
233                 {
234                     logger.info ( " is defined  " );
235                 }
236                 else
237                 {
238                     b = false;
239                     logger.info ( " not defined class " );
240                 }
241                 if (hermit.isSatisfiable(dsp_cpy))
242                 {
243                     logger.info ( " is satisfiable   " );
244                     b = true;
245                 }
246                 else
247                 {
248                      logger.info(" not satisfiable ");
249                      b = false;
250                 }
251             }
252             else {
253                 b = false;
254                 logger.info ( "is not consistent  " );
255             }
```

```
256         }
257         catch (Exception e) {
258             e.printStackTrace();
259         }
260 
261         return b;
262     }
263 }
```

## B.2 Validation d'une orchestration

Cette section renferme le code qui fait la validation d'une orchestration (vérification de compatibilité entre les inputs et les outputs).

**– Source Code: OrchestrationValidator.java**

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5  
6  package fr.anr.techlog.neurolog.semantics.validators;
7  
8  import fr.anr.techlog.neurolog.semantics.entities.Input2Input;
9  import fr.anr.techlog.neurolog.semantics.entities.Orchestration;
10 import fr.anr.techlog.neurolog.semantics.entities.Output2Input;
11 import fr.anr.techlog.neurolog.semantics.entities.Output2Output;
12 import fr.anr.techlog.neurolog.semantics.entities.Operation;
13 import fr.anr.techlog.neurolog.semantics.entities.Input;
14 import fr.anr.techlog.neurolog.semantics.entities.Output;
15 import fr.anr.techlog.neurolog.semantics.myfonctionspack.SubSumption;
16 import java.util.Iterator;
17 import java.util.Set;
18 import org.apache.log4j.Logger;
19 
20 
21 /**
22  *
23  * @author bwali
24  */
25 public class OrchestrationValidator {
26 
27     private static final Logger logger = Logger.getLogger(↩
        OrchestrationValidator.class);
28 
29     public OrchestrationValidator () {
30 
31     }
32 
33     /**
34      * ********************      VOLET DE VALIDATION DE LA DESCRIPTION D UN↩
          WS      *********************
35      *      Valider les mappings et les PRECONDITIOND
36      */
37 
38     public boolean verifierMappings (Operation wsop) {
39 
40         boolean validorchestration = true;
```

```
41
42          if (wsop.isComposite())
43          {
44              Orchestration orchestration = wsop.getOrchestration();
45              Set<Input2Input> i2imappings = orchestration.getI2Imappings();
46              Iterator iti2imappings = i2imappings.iterator();
47              while (iti2imappings.hasNext())
48              {
49                  Input2Input i2i = (Input2Input)iti2imappings.next();
50                  Input inputfrom = i2i.getInputfrom();
51                  for (Input wsopin : wsop.getInputs())
52                              if (wsopin.toStringID().equals(inputfrom.↩
                                    toStringID()))
53                                  inputfrom = wsopin;
54                  Input inputto = i2i.getInputto();
55                      for (Operation op : orchestration.getUsingOperations())
56                          for (Input in : op.getInputs())
57                              if (in.toStringID().equals(inputto.toStringID()↩
                                    ))
58                              {
59                                  logger.info("------> VERIFICATION DE INPUT ↩
                                        TO INPUT MAPPING" + i2i.toStringID());
60                                  if (SubSumption.isSubsumedBy(inputfrom, in)↩
                                        )
61                                  {
62                                      logger.info("-- VALID INPUT TO INPUT ↩
                                            MAPPING WITH " + inputfrom.↩
                                            toStringID() + "," + in.toStringID↩
                                            ());
63                                      validorchestration = true;
64                                  }
65                                  else
66                                  {
67                                      logger.info("-- NOT VALID INPUT TO ↩
                                            INPUT MAPPING WITH " + inputfrom.↩
                                            toStringID() + "," + in.toStringID↩
                                            ());
68                                      validorchestration = false;
69                                  }
70                               }
71              }
72              // verifiaction des input to input mapping
73              Set<Output2Input> o2imappings = orchestration.getO2Imappings();
74              Iterator ito2imappings = o2imappings.iterator();
75              while (ito2imappings.hasNext())
76              {
77                  Output2Input o2i = (Output2Input)ito2imappings.next();
78            //1Â°) chercher l'input sur le ws courant donc dans la variable ↩
                    webservice..
79                  Output outputfrom = o2i.getOutputfrom();
80                  Input inputto = o2i.getInputto();
81
82
83                          for (Operation opr : orchestration.↩
                                getUsingOperations())
84                              for (Output wsopout : opr.getOutputs())
85                                  if (wsopout.toStringID().equals(outputfrom.↩
                                        toStringID()))
86                                      outputfrom = wsopout;
87              //************************ MAPPING DES VARIABLES DE L'↩
                    ORCHESTRATION ET DES WEB SERVICES ↩
                    ************************
```

```
88                  // l'inputto devrait etre d'un web service composant ↩
                        donc il faut aller la chercher!
89
90                  for (Operation op : orchestration.getUsingOperations())
91                      for (Input in : op.getInputs())
92                          if (in.toStringID().equals(inputto.toStringID()↩
                                ))
93                              inputto = in;
94                  logger.info("------> VERIFICATION DE OUTPUT TO INPUT ↩
                        MAPPING " + o2i.toStringID());
95                  if (SubSumption.isSubsumedBy(outputfrom, inputto))
96                  {
97                      logger.info("-- VALID OUTPUT TO INPUT MAPPING WITH ↩
                            " + outputfrom.toStringID() + "," + inputto.↩
                            toStringID());
98                      if (validorchestration == true)
99                          validorchestration = true;
100                 }
101                 else
102                 {
103                     logger.info("-- NOT VALID OUTPUT TO INPUT MAPPING ↩
                            WITH " + outputfrom.toStringID() + "," + ↩
                            inputto.toStringID());
104                     validorchestration = false;
105                 }
106             }
107             // verifiaction des input to input mapping
108             Set<Output2Output> o2omappings = orchestration.getO2Omappings()↩
                    ;
109             Iterator ito2omappings = o2omappings.iterator();
110             while (ito2omappings.hasNext())
111             {
112                 Output2Output o2o = (Output2Output)ito2omappings.next();
113                 //1Â°) chercher l'input sur le ws courant donc dans la ↩
                        variable webservice..
114                 Output outputfrom = o2o.getOutputfrom();
115                 Output outputto = o2o.getOutputto();
116                 for (Operation opr : orchestration.getUsingOperations())
117                     for (Output wsopout : opr.getOutputs())
118                         if (wsopout.toStringID().equals(outputfrom.↩
                                toStringID()))
119                             outputfrom = wsopout;
120                 for (Output out : wsop.getOutputs())
121                     if (out.toStringID().equals(outputto.toStringID()))
122                         outputto = out;
123                 logger.info("------> VERIFICATION DE OUTPUT TO OUTPUT ↩
                        MAPPING " + o2o.toStringID());
124                 if (SubSumption.isSubsumedBy(outputfrom, outputto)) {
125                     logger.info("-- VALID OUTPUT TO OUTPUT MAPPING WITH " +↩
                             outputfrom.toStringID() + "," + outputto.↩
                            toStringID());
126                     if (validorchestration == true)
127                         validorchestration = true;
128                 }
129                 else
130                 {
131                     logger.info("-- NOT VALID OUTPUT TO OUTPUT MAPPING WITH↩
                            " + outputfrom.toStringID() + "," + outputto.↩
                            toStringID());
132                     validorchestration = false;
133                 }
134             }
```

```
135             if (validorchestration == true)
136                logger.info("--********************************* VALID ↩
                       ORCHESTRATION *********************************--");
137             else
138                logger.info("--********************************* NOT VALID ↩
                       ORCHESTRATION *********************************--");
139         }
140         else
141         {
142             logger.info("-- THERE IS NO MAPPING TO VERIFY ::> THIS WEB ↩
                    SERVICE IS NOT COMPOSITE");
143             validorchestration = false;
144         }
145         return validorchestration;
146     }
147 }
```

## B.3 Vérification du type de l'input sélectionnée par l'utilisateur et le type de la variable à la quelle l'input est assignée

Cette section renferme le code source qui décrit la validation du type d'une image sélectionnée par l'utilisateur et assignée à une entrée (variable input)

**– Source Code: ValueInputValidator.java**

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package fr.anr.techlog.neurolog.semantics.validators;
7
8  import fr.anr.techlog.neurolog.semantics.entities.Input;
9  import fr.anr.techlog.neurolog.semantics.entities.Operation;
10 import fr.anr.techlog.neurolog.semantics.entities.Execution;
11 import fr.anr.techlog.neurolog.semantics.entities.VariableValue;
12 import fr.anr.techlog.neurolog.semantics.myfonctionspack.SubSumption;
13 import org.apache.log4j.Logger;
14
15 /**
16  *
17  * @author bwali
18  */
19 public class ValueInputValidator {
20
21     private static final Logger logger = Logger.getLogger(↩
           ValueInputValidator.class);
22
23
24     public ValueInputValidator () {
25
26     }
27
28     /**
29      * Vérifie la compatibilité entre les types de variables et les types ↩
            des inputs
30      * utilisées dans le cadre d'une exécution
```

```
31      * @param wsop l'operation qui va être exécuté
32      * @param wsope l'exécution en cours relative à l'opération en cours
33      * @return retourne vrai ou faux selon si tout les types sont ↩
            compatibles ou pas
34      * et retourne le rapport de fautes d'incompatibilitées dans quelques ↩
            types
35      */
36
37     public boolean verifyExecutionValues ( Operation wsop, Execution wsope ↩
           ) {
38
39         /** elle est à vrai si l'exécution est relative à l'opération et ↩
               visversa   */
40         boolean validExecution = false;
41         boolean validMapping = true;
42         for (Execution ope : wsop.getExecutions())
43             if (ope.toStringID().equals(wsope.toStringID()))
44                 validExecution = true;
45
46         if ( validExecution == true )
47         {
48             // veirfication des inputs
49             for ( VariableValue value : wsope.getInputvariablevalues() )
50                 for ( Input input : wsop.getInputs() )
51                     if ( ((Input)value.getVariable()).toStringID().equals(↩
                           input.toStringID()) )
52                     {
53                         logger.info("------> VERIFICATION DE VALEUR ↩
                               AFFECTEE A UN INPUT DANS LE CADRE DE L'↩
                               EXECUTION ::>" + wsope.toStringID());
54
55                         if ( SubSumption.isSubsumedBy(value, input) ) {
56                             logger.info("--- VALID VALUE/INPUT MAPPING WITH↩
                                   " + value.toStringID() + "," + input.↩
                                   toStringID());
57                             if (validMapping == true)
58                                 validMapping = true;
59                         }
60                         else
61                         {
62                             logger.info("--- NOT VALID VALUE/INPUT MAPPING ↩
                                   WITH " + value.toStringID() + "," + input.↩
                                   toStringID());
63                             validMapping = false;
64                         }
65                     }
66         }
67
68         return validMapping;
69     }
70 }
```

## B.4 Utilisation du raisonneur HermiT pour la vérification de types pour les inputs et les outputs

Cette section renferme les différentes fonctions qui vérifient les compatibilités entre les entrées/sorties et les valeurs/variables

**– Source Code: SubSumption.java**

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package fr.anr.techlog.neurolog.semantics.myfonctionspack;
7
8  import fr.anr.techlog.neurolog.semantics.entities.Dataset;
9  import fr.anr.techlog.neurolog.semantics.entities.VariableValue;
10 import fr.anr.techlog.neurolog.semantics.entities.Input;
11 import fr.anr.techlog.neurolog.semantics.entities.Output;
12 import org.semanticweb.HermiT.Reasoner;
13 import org.semanticweb.HermiT.hierarchy.AtomicConceptSubsumptionCache;
14 import org.semanticweb.HermiT.model.AtomicConcept;
15 import org.semanticweb.owlapi.model.IRI;
16 import org.semanticweb.owlapi.model.OWLOntology;
17
18 /**
19  *
20  * @author BWALI
21  */
22 public class SubSumption {
23
24
25     // subsomption à partir des IRIs
26     /**
27      * elle compare deux classes pour voir si l'une subsume l'autre
28      * @return elle retourne vrai si superAcIRI subsume subAvcIRI sinon ↩
           elle retourne faux
29      */
30
31     public static boolean isSubsumedBy ( String rootAcIRI, String ↩
           superAcIRI, String subAcIRI, String ontologyIRI ){
32
33
34         OWLOntology o = ContextLoader.getOWLOntologyManager().getOntology(↩
               IRI.create(ontologyIRI));
35
36         Reasoner hermit = new Reasoner(o);
37         hermit.classify();
38         AtomicConcept rootAc = AtomicConcept.create(rootAcIRI);
39         AtomicConceptSubsumptionCache acsb = new ↩
               AtomicConceptSubsumptionCache(hermit.getTableau());
40         AtomicConcept superAc = AtomicConcept.create(superAcIRI);
41         AtomicConcept subAc = AtomicConcept.create(subAcIRI);
42         boolean subsumption = false;
43         if (acsb.isSubsumedBy(superAc, rootAc))
44             subsumption=acsb.isSubsumedBy(subAc, superAc);
45
46         return subsumption;
47     }
48
49     public static boolean isSubsumedBy ( String rootAcIRI, String ↩
           superAcIRI, String subAcIRI, OWLOntology o ){
50
51         Reasoner hermit = new Reasoner(o);
52         hermit.classify();
53         AtomicConcept rootAc = AtomicConcept.create(rootAcIRI);
54         AtomicConceptSubsumptionCache acsb = new ↩
               AtomicConceptSubsumptionCache(hermit.getTableau());
55         AtomicConcept superAc = AtomicConcept.create(superAcIRI);
```

```
56          AtomicConcept subAc = AtomicConcept.create(subAcIRI);
57          boolean subsumption = false;
58          if (acsb.isSubsumedBy(superAc, rootAc))
59              subsumption=acsb.isSubsumedBy(subAc, superAc);
60
61          return subsumption;
62      }
63
64      // subsomption input to input
65
66      public static boolean isSubsumedBy ( Input inputfrom, Input inputto ){
67
68           Reasoner hermit = new Reasoner(ContextLoader.getNLOntology());
69           hermit.prepareReasoner();
70           hermit.classify();
71           AtomicConceptSubsumptionCache acsb = new ↩
                 AtomicConceptSubsumptionCache(hermit.getTableau());
72           System.out.println ("--- the inputs  :" + inputfrom.toStringID() +↩
                  "," + inputto.toStringID());
73           System.out.println ("--- the ref input from " + inputfrom.↩
                 getParticularConcept().getreferencedClass().toStringID() );
74           System.out.println ("--- the ref input  to " +  inputto.↩
                 getParticularConcept().getreferencedClass().toStringID());
75           AtomicConcept superAc = AtomicConcept.create(inputto.↩
                 getParticularConcept().getreferencedClass().toStringID());
76           AtomicConcept subAc = AtomicConcept.create(inputfrom.↩
                 getParticularConcept().getreferencedClass().toStringID());
77
78           return acsb.isSubsumedBy(subAc, superAc);
79      }
80
81      public static boolean isSubsumedBy ( Output outputfrom, Input inputto){
82
83
84           Reasoner hermit = new Reasoner(ContextLoader.getNLOntology());
85           hermit.prepareReasoner();
86           hermit.classify();
87           AtomicConceptSubsumptionCache acsb = new ↩
                 AtomicConceptSubsumptionCache(hermit.getTableau());
88           System.out.println ("--- the output/input : " + outputfrom.↩
                 toStringID() + "," + inputto.toStringID());
89           System.out.println ("--- the ref output from " + outputfrom.↩
                 getParticularConcept().getreferencedClass().toStringID() );
90           System.out.println ("--- the ref input  to " +  inputto.↩
                 getParticularConcept().getreferencedClass().toStringID());
91           AtomicConcept superAc = AtomicConcept.create(inputto.↩
                 getParticularConcept().getreferencedClass().toStringID());
92           AtomicConcept subAc = AtomicConcept.create(outputfrom.↩
                 getParticularConcept().getreferencedClass().toStringID());
93
94           return acsb.isSubsumedBy(subAc, superAc);
95      }
96
97      public static boolean isSubsumedBy ( Output outputfrom, Output outputto↩
            ){
98
99
100          Reasoner hermit = new Reasoner(ContextLoader.getNLOntology());
101          hermit.prepareReasoner();
102          hermit.classify();
103          AtomicConceptSubsumptionCache acsb = new ↩
                 AtomicConceptSubsumptionCache(hermit.getTableau());
```

```
104             System.out.println ("--- the outputs : " + outputfrom.toStringID()↩
                    + "," + outputto.toStringID());
105             System.out.println ("--- the ref output from " + outputfrom.↩
                    getParticularConcept().getreferencedClass().toStringID() );
106             System.out.println ("--- the ref input  to " +  outputto.↩
                    getParticularConcept().getreferencedClass().toStringID());
107             AtomicConcept superAc = AtomicConcept.create(outputto.↩
                    getParticularConcept().getreferencedClass().toStringID());
108             AtomicConcept subAc = AtomicConcept.create(outputfrom.↩
                    getParticularConcept().getreferencedClass().toStringID());
109
110             return acsb.isSubsumedBy(subAc, superAc);
111         }
112
113         public static boolean isSubsumedBy ( VariableValue value, Input input )↩
                {
114
115
116             Reasoner hermit = new Reasoner(ContextLoader.getNLOntology());
117             hermit.prepareReasoner();
118             hermit.classify();
119             AtomicConceptSubsumptionCache acsb = new ↩
                    AtomicConceptSubsumptionCache(hermit.getTableau());
120             System.out.println ("--- the value Mapped on input is  " + value.↩
                    toStringID() + ",\n "  + input.toStringID());
121
122             AtomicConcept superAc = AtomicConcept.create(input.↩
                    getParticularConcept().getreferencedClass().toStringID());
123
124             String s = ((Dataset)value.getParticular()).getType().toStringID()↩
                    ;
125             System.out.println (s);
126             AtomicConcept subAc = AtomicConcept.create(s);
127
128             return acsb.isSubsumedBy(subAc, superAc);
129         }
130
131         /*
132          * simple subsumption entre les classes
133          */
134         public static boolean isSubsumedBy (String ssuperAC, String ssubAc ) ↩
                {
135
136             Reasoner hermit = new Reasoner( ContextLoader.getNLOntology() );
137             hermit.prepareReasoner();
138             hermit.classify();
139             AtomicConceptSubsumptionCache acsb = new ↩
                    AtomicConceptSubsumptionCache(hermit.getTableau());
140
141
142             AtomicConcept superAc = AtomicConcept.create(ssuperAC);
143             AtomicConcept subAc = AtomicConcept.create(ssubAc);
144
145             return acsb.isSubsumedBy(subAc, superAc);
146         }
147 }
```

## B.5 Génération d'une règle CORESE à partir de son annotation

Cette section renferme le code source qui permet de générer un pré ou post condition sous le format CORESE et de générer la requête spécifique qui permet de récupérer les métadonnées créer suite à l'application de cette requête.

**– Source Code: PostConditionTypeSameDataset.java**

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package fr.anr.techlog.neurolog.semantics.rules;
7
8  import fr.anr.techlog.neurolog.semantics.entities.Input;
9  import fr.anr.techlog.neurolog.semantics.entities.Output;
10 import fr.anr.techlog.neurolog.semantics.myfonctionspack.Prefixs;
11 import java.io.File;
12 import java.io.PrintStream;
13 import java.util.ArrayList;
14 import java.util.HashSet;
15 import java.util.Set;
16 import org.semanticweb.owlapi.model.AddAxiom;
17 import org.semanticweb.owlapi.model.IRI;
18 import org.semanticweb.owlapi.model.OWLClass;
19 import org.semanticweb.owlapi.model.OWLClassAssertionAxiom;
20 import org.semanticweb.owlapi.model.OWLDataFactory;
21 import org.semanticweb.owlapi.model.OWLNamedIndividual;
22 import org.semanticweb.owlapi.model.OWLObjectProperty;
23 import org.semanticweb.owlapi.model.OWLObjectPropertyAssertionAxiom;
24 import org.semanticweb.owlapi.model.OWLOntology;
25 import org.semanticweb.owlapi.model.OWLOntologyManager;
26
27 /**
28  *
29  * @author bwali
30  */
31 public class PostConditionTypeSameDataset extends Condition {
32
33     //Inputs mis en jeux
34     Input input = null;
35
36
37     //les outputs mis en jeux
38     Set<Output> outputs = null;
39
40     public PostConditionTypeSameDataset(OWLNamedIndividual condition) {
41         super(condition);
42     }
43
44     public PostConditionTypeSameDataset ( OWLNamedIndividual pcrt
45     , Set<Output> outputs,
46       Input input
47      ) {
48
49         super (pcrt);
50         this.outputs = outputs;
```

```
51          this.input = input;
52      }
53
54      public Input getInput() {
55          return input;
56      }
57
58      public void setInput(Input input) {
59          this.input = input;
60      }
61
62      public Set<Output> getOutputs() {
63          return outputs;
64      }
65
66      public void setOutputs(Set<Output> outputs) {
67          this.outputs = outputs;
68      }
69      // pour créer le gabarit de la
70      public void create () {
71
72
73          Set<String> datasetoutputvalues = new HashSet<String>();
74
75           /**
76           *  creation de la partie if de la condition
77           */
78           TripleSet triplesetif = new TripleSet();
79            TripleSet triplesetelse = new TripleSet();
80           int i = 1;
81
82               /**
83               *  creation des triplets qui selectionnent la variable value ↩
                     de chaque variable (output ou input)
84               */
85
86               Var subject = new Var(input.getInput().getIRI().toString());
87               Var predicate = new Var("http://www.irisa.fr/visages/team/↩
                     farooq/ontologies/web-service-owl-lite.owl#has-value");
88               Var object = new Var("?inputvalue" + Integer.toString(i));
89               Triple triple = new Triple(subject, predicate, object);
90               triplesetif.addTriple(triple);
91
92               /**
93               *  creation des triplets qui selectionnent les datasets auquel↩
                      se rÃ©fÃ¨rent les variablevalue
94               * des inputvalue et des outputsvalue
95               */
96
97               subject = new Var(object.getVar());
98               predicate = new Var("http://www.irisa.fr/visages/team/farooq/↩
                     ontologies/iec-owl-lite.owl#refers-to");
99               object = new Var("?dsinputvalue" + Integer.toString(i));
100              triple = new Triple(subject, predicate, object);
101              triplesetif.addTriple(triple);
102
103              /**
104              *  selection des types de ces inputs
105              */
106
107              subject = new Var("?dsinputvalue" + Integer.toString(i));
108              predicate = new Var("rdf:type");
```

```
109             object = new Var("?dsinputvalueType" + Integer.toString(i));
110             triple = new Triple(subject, predicate, object);
111             triplesetif.addTriple(triple);
112
113
114             int j = 1;
115
116             for (Output output : outputs ) {
117
118                 /**
119                 *  creation de la close if
120                 */
121
122                     /**
123                     *  creation des triplets qui selectionnent la variable↩
                            value de chaque variable (output ou input)
124                     */
125
126                     Var subject1 = new Var(output.getOutput().getIRI().↩
                        toString());
127                     Var predicate1 = new Var("http://www.irisa.fr/visages/↩
                        team/farooq/ontologies/web-service-owl-lite.owl#↩
                        has-value");
128                     Var object1 = new Var("?outputvalue" + Integer.↩
                        toString(j));
129                     triple = new Triple(subject1, predicate1, object1);
130                     triplesetif.addTriple(triple);
131
132
133                     /**
134                     *  creation des triplets qui selectionnent les ↩
                            datasets auquel se rÃ©fÃ¨rent les variablevalue
135                     * des inputvalue et des outputsvalue
136                     */
137
138                     subject1 = object1;
139                     predicate1 = new Var("http://www.irisa.fr/visages/team↩
                        /farooq/ontologies/iec-owl-lite.owl#refers-to");
140                     object1 = new Var("?outputvaluedataset" + Integer.↩
                        toString(j));
141                     triple = new Triple(subject1, predicate1, object1);
142                     triplesetif.addTriple(triple);
143
144                 /**
145                 *  creation de la partie then de la condition
146                 */
147
148                     /**
149                     *  Changer le type des datasets de sorties selon le ↩
                            type de l'input concerné dans la condition
150                     *  niveu d'action Corese cette condition ne rajoute un↩
                            type pour l'output dans corese
151                     *  donc c'est au niveau de la requete qu'on va ↩
                            sélectionner le nouveu type rajouté et qu'on va
152                     *  mettre en évidence dans la création des triplets d'↩
                            exécution
153                     */
154
155                     subject1 = object1;
156                     predicate1 = new Var("rdf:type");
157                     object1 = object;
158                     triple = new Triple(subject1, predicate1, object1);
```

```
159                     triplesetelse.addTriple(triple);
160
161                 j++;
162             }
163
164
165             /**
166             *  affecter la partie if et else de la condition corese
167             */
168
169             super.setCloseIf(new If(triplesetif));
170             super.setCloseThen(new Else(triplesetelse));
171         }
172          /** A revoir */
173         public void WriteSpecificQuery (String path) {
174
175             try
176             {
177                 PrintStream pstream = new PrintStream(new File(path));
178
179                 String query =  "select ?dataset where { ";
180                 for (Output output : outputs ) {
181                 query += "<" + output.toStringID() + ">" +
182                 " <http://www.irisa.fr/visages/team/farooq/ontologies/web-↩
                        service-owl-lite.owl#has-value>    ?outputvariablevalues" +
183                 " ?outputvariablevalues <http://www.irisa.fr/visages/team/↩
                        farooq/ontologies/iec-owl-lite.owl#refers-to>    ?datasets ↩
                        ";
184                 }
185                 query += "}";
186
187                 pstream.println(query);
188                 pstream.close();
189             }
190             catch (Exception e) {
191                 e.printStackTrace();
192             }
193
194         }
195
196         /** A revoir */
197         /**
198          * Elle retourne la requete qui va avec la condition crée--> Elle ↩
                selectionnes a richesse sémantique apporté
199          * par l'aplication de la condition dans corese.
200          * @return Une chaine caractère représentant la requête qui doit être ↩
                posée à Corese pour
201          * détecter l'enrichissement sémantique résultant de l'application de ↩
                la regle Corese
202          */
203         public String getSpecificQuery () {
204
205                 String query ="PREFIX my: <http://www.irisa.fr/visages/team/↩
                        farooq/ontologies/dataset-owl-lite.owl#> " +
206                 "Select ?outputvariable ?outputvariablevalues ?dataset ?type ↩
                        where { \n" +
207                 "<"+ super.getCondition().toStringID()+">   <"+ Prefixs.CONCERN↩
                        +">   ?outputvariable \n" +
208                 "  ?outputvariable rdf:type <"+ Prefixs.OUTPUT_VARIABLE +"> \n"↩
                        +
209                 " ?outputvariable <http://www.irisa.fr/visages/team/farooq/↩
                        ontologies/web-service-owl-lite.owl#has-value>    ?↩
```

```
                outputvariablevalues \n" +
210             " ?outputvariablevalues <http://www.irisa.fr/visages/team/farooq/ontologies/iec-owl-lite.owl#refers-to>    ?dataset \n" +
211             " <"+ super.getCondition().toStringID()+">   <"+ Prefixs.CONCERN +">   ?inputvariable \n" +
212             "  ?inputvariable rdf:type <"+ Prefixs.INPUT_VARIABLE +"> \n" +
213             " ?inputvariable <http://www.irisa.fr/visages/team/farooq/ontologies/web-service-owl-lite.owl#has-value>    ?inputvariablevalues \n" +
214             " ?inputvariablevalues <http://www.irisa.fr/visages/team/farooq/ontologies/iec-owl-lite.owl#refers-to>    ?dataset1 \n" +
215             " ?dataset1 rdf:type ?type " +
216             " filter(?type ^ my:)" +
217             " } ";
218
219         return query;
220     }
221
222
223
224     /**
225      * Cette méthode met dans une ontologie OWL les annotations concernant l'individu WebService
226      * @param service :L'ontologie ou les annotations vont etre stockées
227      * @param manager : Le OWLOntologyManager qui gère l'ontologie service
228      * @param dataFactory : La machine de création des classes des ontologies et des individus OWL liée au manager
229      * @param baseIri l'iri de base de l'ontologie service à créer
230      * @return retourne une ontologie de type OWLOntology contenant les annotations
231      *         de la description du service en cours
232      */
233
234     @Override
235     public OWLOntology writeCondition (OWLOntology service,  OWLOntologyManager manager , OWLDataFactory dataFactory , String baseIri) {
236
237         try {
238
239             OWLObjectProperty objectproperty = null;
240             OWLObjectPropertyAssertionAxiom objectpropertyassertionaxiom = null;
241
242             ArrayList <AddAxiom> list = new  ArrayList<AddAxiom>();
243             OWLClass webserviceClass = dataFactory.getOWLClass(IRI.create(Prefixs.POST_CONDITION_TYPE_SAME_DATASET_CLASS));
244
245             // Type de l'entité web service ==> en occurence web-service
246             OWLClassAssertionAxiom classAssertion = dataFactory.getOWLClassAssertionAxiom(webserviceClass, super.getParticular());
247             AddAxiom addAxiom = new AddAxiom(service, classAssertion);
248             list.add(addAxiom);
249
250             // L'input concerné par cette postcondition
251             if (input != null) {
252                 objectproperty = dataFactory.getOWLObjectProperty(IRI.create(Prefixs.CONCERN));
253                 objectpropertyassertionaxiom = dataFactory.getOWLObjectPropertyAssertionAxiom(objectproperty, 
```

```
                    super.getParticular(), input.getParticular());
254                 addAxiom = new AddAxiom(service, ↩
                        objectpropertyassertionaxiom);
255                 list.add(addAxiom);
256                 objectproperty = dataFactory.getOWLObjectProperty(IRI.↩
                        create(Prefixs.CONCERNED_BY));
257                 objectpropertyassertionaxiom = dataFactory.↩
                        getOWLObjectPropertyAssertionAxiom(objectproperty, ↩
                        input.getParticular(), super.getParticular());
258                 addAxiom = new AddAxiom(service, ↩
                        objectpropertyassertionaxiom);
259                 list.add(addAxiom);
260             }
261
262             // Les inputs concerné par cette postcondition
263             if (outputs != null) {
264                 for ( Output output : outputs ) {
265                     objectproperty = dataFactory.getOWLObjectProperty(IRI.↩
                            create(Prefixs.CONCERN));
266                     objectpropertyassertionaxiom = dataFactory.↩
                            getOWLObjectPropertyAssertionAxiom(objectproperty, ↩
                            super.getParticular(), output.getParticular());
267                     addAxiom = new AddAxiom(service, ↩
                            objectpropertyassertionaxiom);
268                     list.add(addAxiom);
269                     objectproperty = dataFactory.getOWLObjectProperty(IRI.↩
                            create(Prefixs.CONCERNED_BY));
270                     objectpropertyassertionaxiom = dataFactory.↩
                            getOWLObjectPropertyAssertionAxiom(objectproperty, ↩
                            output.getParticular(), super.getParticular());
271                     addAxiom = new AddAxiom(service, ↩
                            objectpropertyassertionaxiom);
272                     list.add(addAxiom);
273                 }
274             }
275
276             manager.applyChanges(list);
277         }
278         catch (Exception e) {
279             System.out.println("Could not add axioms to ontology: " + e.↩
                    getMessage());
280         }
281
282         return service;
283     }
284
285
286 }
```

## B.6 Diagramme de séquence

Cette section représente le diagramme de séquence qui a été conçu pour la mise en place de cette solution. Il représente les différentes interactions entre les différentes entités de la plateforme NeuroLOG. Il détaille donc l'utilisation des outils sémantiques depuis la préparation de l'enveloppe sémantique jusqu'a l'exécution des règles et la création et l'enregistrement des métadonnées.

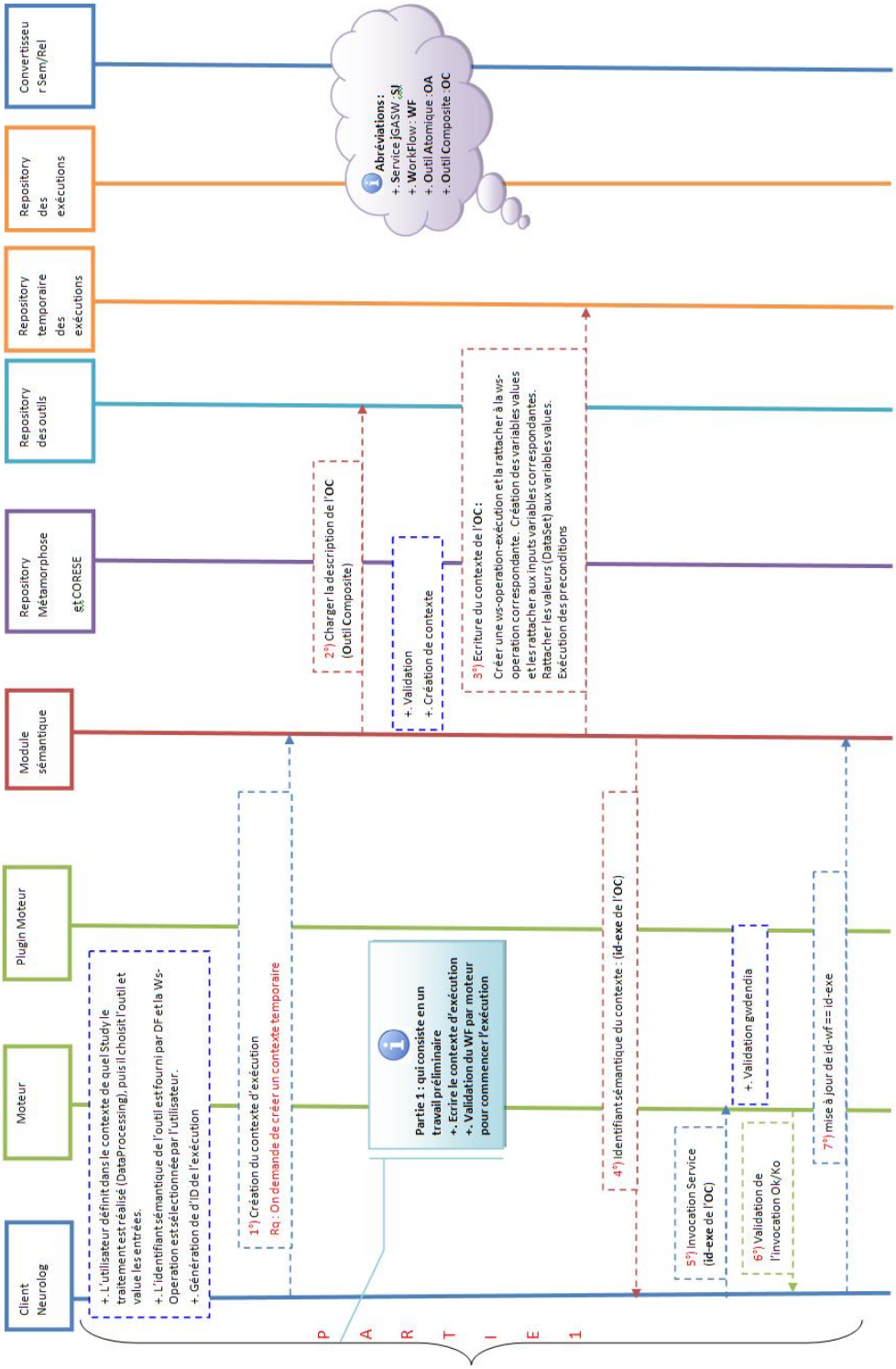


FIGURE B.1 – Diagramme de séquence expliquant les phases d'implémentations : partie 1

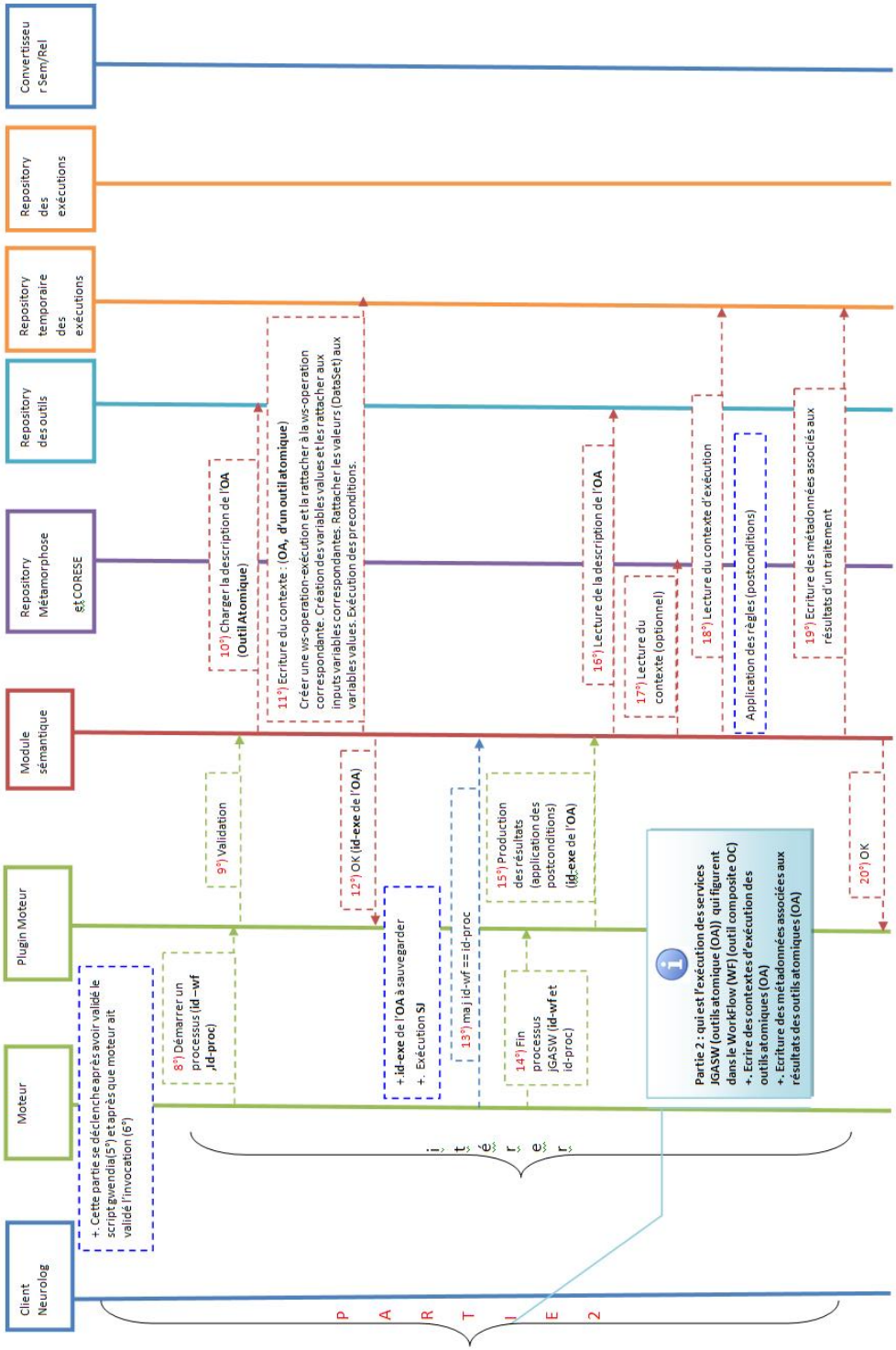


FIGURE B.2 – Diagramme de séquence expliquant les phases d'implémentations : partie 2

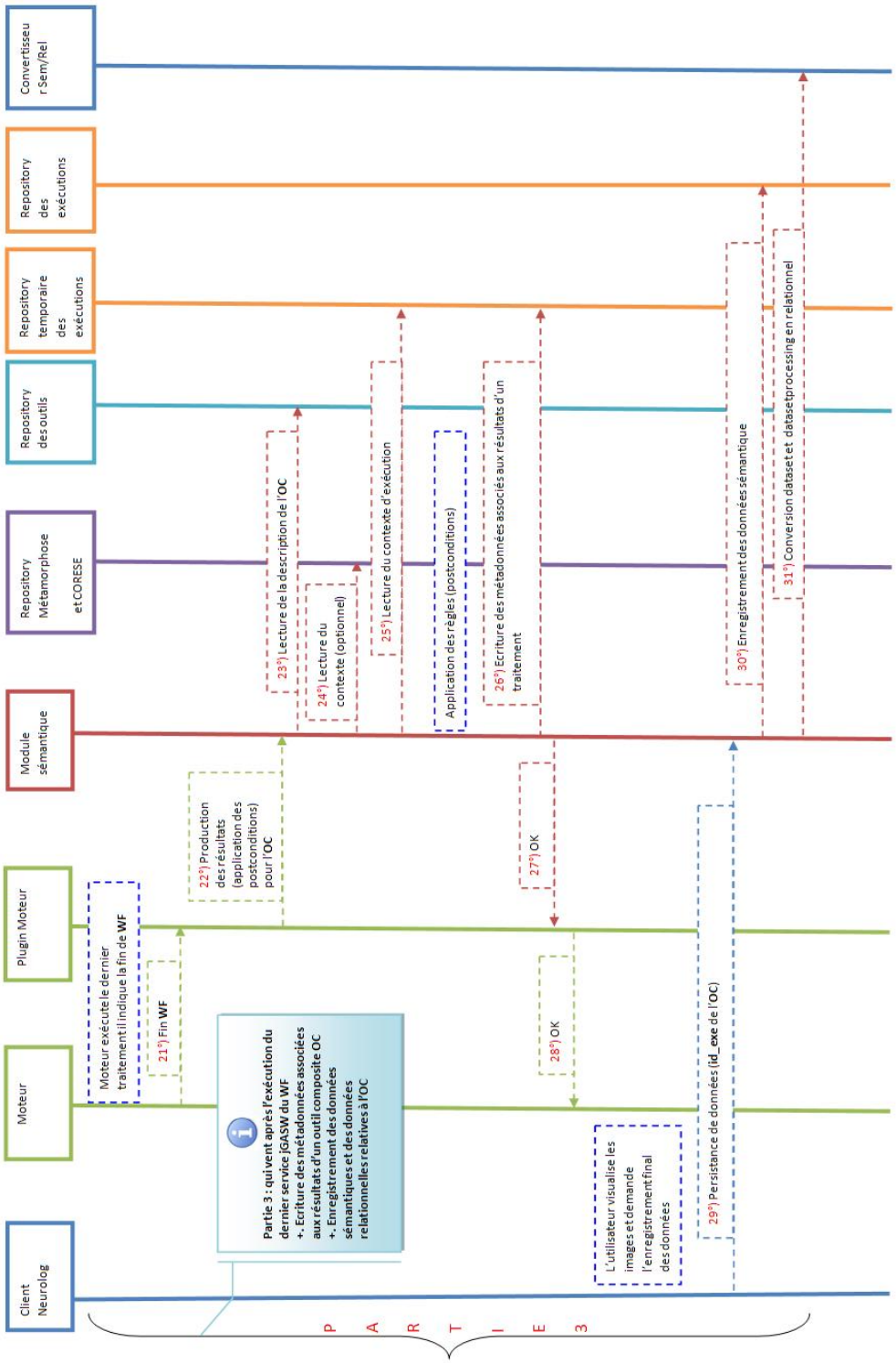


FIGURE B.3 – Diagramme de séquence expliquant les phases d'implémentations : partie 3

# Bibliographie


[1] Rohit Aggarwal et al. "Constraint Driven Web Service Composition in METEOR-S". Dans : *Proceedings of the 2004 IEEE International Conference on Services Computing*. SCC '04. Washington, DC, USA : IEEE Computer Society, 2004, p. 23–30. ISBN : 0-7695-2225-4. URL : http://dl.acm.org/citation.cfm?id=1025130.1026125.

[2] A. Ahtisham, Sören Auer et J. Shen. "From BPEL4WS Process Model to Full OWL-S Ontology". Dans : *Proceedings of Posters and Demos, 3rd European Semantic Web Conference (ESWC 2006) in June 11-14*. Budva, Montenegro : IEEE Computer Society, 2006, 61–fffd62.

[3] Jonas S. Almeida, Helena F. Deus et Wolfgang Maass. "S3DB core : a framework for RDF generation and management in bioinformatics infrastructures". Dans : *BMC Bioinformatics* 11 (2010), p. 387.

[4] Gustavo Alonso et al. *Web Services : Concepts, Architectures and Applications*. Berlin : Springer, 2004. ISBN : 978-3-540-44008-6.

[5] Tony Andrews et al. *BPEL4WS, Business Process Execution Language for Web Services Version 1.1*. IBM. 2003. URL : http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel/ws-bpel.pdf.

[6] Anupriya Ankolekar et al. *OWL-S' Relationship to Selected Other Technologies*. Rap. tech. DAML, 2004.

[7] Rainer Anzböck et Schahram Dustdar. "Modeling and implementing medical web services". Dans : *Data Knowl. Eng.* 55.2 (nov. 2005), p. 203–236. ISSN : 0169-023X. DOI : 10.1016/j.datak.2005.03.009. URL : http://dx.doi.org/10.1016/j.datak.2005.03.009.

[8] "Architecture CaBIG approche opfffdrationnelle". Dans : (fév. 2013). URL : http://deainfo.nci.nih.gov/advisory/ncab/155_0910/presentations/1115%20Buetow%20NCAB_caBIG%2009.08.10.pdf.

[9] Sinuhe Arroyo et Miguel-Angel Sicilia. "SOPHIE : Use case and evaluation". Dans : *Inf. Softw. Technol.* 50.12 (nov. 2008), p. 1266–1280. ISSN : 0950-5849. DOI : 10.1016/j.infsof.2008.01.001. URL : http://dx.doi.org/10.1016/j.infsof.2008.01.001.

[10] Sinuhe Arroyo et al. "A model-driven choreography conceptual framework". Dans : *Comput. Stand. Interfaces* 29.3 (mar. 2007), p. 325–334. ISSN : 0920-5489. DOI : 10.1016/j.csi.2006.05.004. URL : http://dx.doi.org/10.1016/j.csi.2006.05.004.

[11] Franz Baader, Ian Horrocks et Ulrike Sattler. "Description Logics for the Semantic Web". Dans : *IEEE Data Engineering Bulletin* 16.25 (2001), p. 4–9.

[12] Ziv Baida, Jaap Gordijn et Borys Omelayenko. "A shared service terminology for online service provisioning". Dans : *Proceedings of the 6th international conference on Electronic commerce*. ICEC '04. New York, NY, USA : ACM, 2004, p. 1–10. ISBN : 1-58113-930-6. DOI : 10.1145/1052220.1052222. URL : http://doi.acm.org/10.1145/1052220.1052222.

[13] Javier Rojas Balderrama, Johan Montagnat et Diane Lingrand. "jGASW : A Service-Oriented Framework Supporting HTC and Non-functional Concerns". Dans : *Proceedings of the 2010 IEEE International Conference on Web Services*. ICWS '10. Washington, DC, USA : IEEE Computer Society, 2010, p. 691–694. ISBN : 978-0-7695-4128-0. DOI : http://dx.doi.org/10.1109/ICWS.2010.59. URL : http://dx.doi.org/10.1109/ICWS.2010.59.

[14] Christian Barillot et al. "Federating distributed and heterogeneous information sources in neuroimaging : the NeuroBase Project." Anglais. Dans : *Studies in health technology and informatics* 120 (2006), p. 3–13. ISSN : 0926-9630. URL : http://www.hal.inserm.fr/inserm-00141685.

[15] S. Batra et S. Bawa. "Semantic categorization of Web services". Dans : *International Journal of Recent Trends in Engineering* v2 (2009), p. 19–23.

[16] Rainer Berbner, Oliver Heckmann et Ralf Steinmetz. "An Architecture for a QoS driven composition of Web Service based Workflows". Dans : *Networking and Electronic Commerce Research Conference (NAEC 2005), Riva Del Garda*. 2005. URL : http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.87.2435.

[17] Rainer Berbner et al. "Heuristics for QoS-aware Web Service Composition". Dans : *ICWS '06 : Proceedings of the IEEE International Conference on Web Services*. Washington, DC, USA : IEEE Computer Society, 2006, p. 72–82. ISBN : 0769526691. DOI : 10.1109/ICWS.2006.69. URL : http://dx.doi.org/10.1109/ICWS.2006.69.

[18] Tim Berners-Lee, James Hendler et Ora Lassila. "The Semantic Web (Berners-Lee et. al 2001)". 17 mai 2001. URL : http://www.sciam.com/print_version.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21.

[19] Alain Bidault, Christine Froidevaux et Brigitte Safar. "Repairing Queries in a Mediator Approach." Dans : *ECAI*. Sous la dir. de Werner Horn. IOS Press, 30 juin 2003, p. 406–410. URL : http://dblp.uni-trier.de/db/conf/ecai/ecai2000.html#BidaultFS00.

[20] Laura Bocchi et al. "An OWL-S based approach to express grid services coordination". Dans : *Proceedings of the 2005 ACM symposium on Applied computing*. SAC '05. Santa Fe, New Mexico : ACM, 2005, p. 1661–1667. ISBN : 1-58113-964-0. DOI : 10.1145/1066677.1067054. URL : http://doi.acm.org/10.1145/1066677.1067054.

[21] Athman Bouguettaya et al. "Bio-Sense : A System for Supporting Sharing and Exploration in Bioinformatics Using Semantic Web Services". Dans : *Proceedings of the 2008 Fourth IEEE International Conference on eScience*. ESCIENCE '08. Washington, DC, USA : IEEE Computer Society, 2008, p. 452–453. ISBN : 978-0-7695-3535-7.

DOI : 10.1109/eScience.2008.95. URL : http://dx.doi.org/10.1109/eScience.2008.95.

[22] Andrey BOVYKIN et Evgeny ZOLIN. "Deciding semantic matching of stateless services". Dans : *In Proceedings of the 21st National Conference on Artificial Intelligence (AAAI*. AAAI Press, 2006.

[23] Tim BRAY. *What Is RDF*. http ://www.xml.com/lpt/a/2001/01/24/rdf.html. Jan. 2001. URL : http://www.xml.com/lpt/a/2001/01/24/rdf.html.

[24] Dan BRICKLEY et R.V. GUHA. *RDF Vocabulary Description Language 1.0 : RDF Schema*. http ://www.w3.org/TR/rdf-schema/. Fév. 2004. URL : http://www.w3.org/TR/rdf-schema/.

[25] Jeen BROEKSTRA, Arjohn KAMPMAN et Frank van HARMELEN. "Sesame : A Generic Architecture for Storing and Querying RDF and RDF Schema". Dans : *Proceedings of the First International Semantic Web Conference on The Semantic Web*. ISWC '02. London, UK, UK : Springer-Verlag, 2002, p. 54–68. ISBN : 3-540-43760-6. URL : http://dl.acm.org/citation.cfm?id=646996.711426.

[26] Sabine BRUAUX, Gilles KASSEL et Gilles MOREL. "An ontological approach to the construction of problem-solving models". Anglais. LRR 2005-03 LRR 2005-03. URL : http://hal.archives-ouvertes.fr/hal-00005019.

[27] Jos de BRUIJN et al. "Web Service Modeling Ontology (WSMO)". Dans : (Juin 2005). URL : http://www.w3.org/Submission/WSMO/.

[28] Mark BURSTEIN et al. "A Semantic Web Services Architecture". Dans : *IEEE Internet Computing* 09 (oct. 2005), p. 72–81. DOI : http://doi.ieeecomputersociety.org/10.1109/MIC.2005.96. URL : http://dx.doi.org/http://doi.ieeecomputersociety.org/10.1109/MIC.2005.96.

[29] Liliana CABRAL et al. "IRS-III : A Broker for Semantic Web Services based Applications". Dans : *In proceedings of the 5 th International Semantic Web Conference (ISWC 2006*. 2006, p. 201–214.

[30] Jorge CARDOSO et Amit SHETH. "Semantic E-Workflow Composition". Dans : *J. Intell. Inf. Syst.* 21.3 (nov. 2003), p. 191–225. ISSN : 0925-9902. DOI : 10.1023/A:1025542915514. URL : http://dx.doi.org/10.1023/A:1025542915514.

[31] Jeremy J. CARROLL et Graham KLYNE. *Resource Description Framework (RDF) : Concepts and Abstract Syntax*. W3C Recommendation. http ://www.w3.org/TR/2004/REC-rdf-concepts-20040210/. Fév. 2004.

[32] Fabio CASATI et al. "Adaptive and Dynamic Service Composition in eFlow". Dans : *Proceedings of the 12th International Conference on Advanced Information Systems Engineering*. CAiSE '00. London, UK, UK : Springer-Verlag, 2000, p. 13–31. ISBN : 3-540-67630-9. URL : http://dl.acm.org/citation.cfm?id=646088.679914.

[33] Mercedes Argüello CASTELEIRO et Jose Julio DES DIZ. "Clinical practice guidelines : A case study of combining OWL-S, OWL, and SWRL". Dans : *Know.-Based Syst.* 21.3 (avr. 2008), p. 247–255. ISSN : 0950-7051. DOI : 10.1016/j.knosys.2007.11.008. URL : http://dx.doi.org/10.1016/j.knosys.2007.11.008.

[34] Ethan Cerami. *Web Services Essentials*. Sous la dir. de Simon St.Laurent. 1st. Sebastopol, CA, USA : O'Reilly & Associates, Inc., 2002. isbn : 0596002246.

[35] Yassin Chabeb et Samir Tata. "Yet another semantic annotation for WSDL". Dans : *IADIS International Conference WWW/Internet 2008, October 13-15, Freiburg, Germany*. INF - Dept. Informatique (Institut Mines-Télécom-Télécom SudParis), SAMOVAR - SAMOVAR UMR INT-CNRS 5157 (Institut Mines-Télécom-Télécom SudParis). 2008.

[36] J. Charlet et al. "Interopffffdrabilitffffd en mffffddecine : quand le contenu interroge le contenant et l'organisation". Dans : *Information - Interaction - Intelligence* 2.2 (2002), p. 37–62.

[37] Kei-Hoi Cheung et al. "Approaches to neuroscience data integration". Dans : *Brief Bioinform* 10.4 (1er juil. 2009), p. 345–353. doi : 10.1093/bib/bbp029. url : http://dx.doi.org/10.1093/bib/bbp029.

[38] Philipp Cimiano. "ORAKEL : A Natural Language Interface to an F-Logic Knowledge Base." Dans : *NLDB*. Sous la dir. de Farid Meziane et Elisabeth Métais. T. 3136. Lecture Notes in Computer Science. Springer, 2004, p. 401–406. isbn : 3-540-22564-1. doi : conf/nldb/2004. url : http://dblp.uni-trier.de/db/conf/nldb/nldb2004.html#Cimiano04.

[39] Ion Constantinescu, Boi Faltings et Walter Binder. "Type based service composition". Dans : *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*. WWW Alt. '04. New York, NY, USA : ACM, 2004, p. 268–269. isbn : 1-58113-912-8. doi : 10.1145/1013367.1013429. url : http://doi.acm.org/10.1145/1013367.1013429.

[40] Olivier Corby, Rose Dieng-Kuntz et Catherine Faron-Zucker. "Querying the Semantic Web with Corese Search Engine". Dans : *ECAI'04*. 2004, p. 705–709.

[41] "Couches d'un entrepffffdt de donnfffdes". Dans : (fév. 2013). url : http://www.1keydata.com/datawarehousing/data-warehouse-architecture.html.

[42] Jiangbo Dang et al. "An ontological knowledge framework for adaptive medical workflow". Dans : *J. of Biomedical Informatics* 41.5 (oct. 2008), p. 829–836. issn : 1532-0464. doi : 10.1016/j.jbi.2008.05.012. url : http://dx.doi.org/10.1016/j.jbi.2008.05.012.

[43] Stefan Decker et al. "Ontobroker : Ontology based Access to Distributed and Semi-Structured Information". Dans : *Database Semantics : Semantic Issues in Multimedia Systems*. Kluwer Academic Publisher, 1998, p. 351–369.

[44] P Degoulet et al. "Rationale and design considerations for a semantic mediator in health information systems". Dans : *Methods of Information in Medicine* 37 (nov. 1998), p. 518–526. issn : 0026-1270. url : http://www.ncbi.nlm.nih.gov/pubmed/9865050.

[45] Michael DiBernardo, Rachel Pottinger et Mark Wilkinson. "Semi-automatic web service composition for the life sciences using the BioMoby semantic web framework". Dans : *J. of Biomedical Informatics* 41.5 (oct. 2008), p. 837–847. ISSN : 1532-0464. DOI : 10.1016/j.jbi.2008.02.005. URL : http://dx.doi.org/10.1016/j.jbi.2008.02.005.

[46] "Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006". Dans : *KR*. Sous la dir. de Patrick Doherty, John Mylopoulos et Christopher A. Welty. AAAI Press, 2006. ISBN : 978-1-57735-271-6.

[47] John Domingue, Stefania Galizia et Liliana Cabral. "The Choreography Model for IRS-III". Dans : *In Proc. of the 39th Hawaiian International Conference On System Sciences (HICSS-39), Kauai Island*. 2006.

[48] John Domingue et al. "IRS-III : A broker-based approach to semantic Web services". Dans : *Web Semantics : Science, Services and Agents on the World Wide Web* 6.2 (2011). ISSN : 1570-8268. URL : http://www.websemanticsjournal.org/index.php/ps/article/view/140.

[49] Hai Dong, Farookh Khadeer Hussain et Elizabeth Chang. "A survey in semantic search technologies". Dans : *Digital Ecosystems and Technologies, 2008. DEST 2008. 2nd IEEE International Conference on* (fév. 2008), p. 403–408. DOI : 10.1109/DEST.2008.4635202. URL : http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4635202&isnumber=4635078.

[50] Jing Dong, Yongtao Sun et Sheng Yang. "OWL-S Ontology Framework Extension for Dynamic Web Service Composition". Dans : *Proceedings of the Eighteenth International Conference on Software Engineering & Knowledge Engineering (SEKE 2006), San Francisco, CA, USA, July 5-7, 2006*. Sous la dir. de Kang Zhang, George Spanoudakis et Giuseppe Visaggio. 2006, p. 544–549. ISBN : 1-891706-18-7.

[51] Schahram Dustdar et Wolfgang Schreiner. "A survey on web services composition". Dans : *Int. J. Web Grid Serv.* 1.1 (août 2005), p. 1–30. ISSN : 1741-1106. DOI : 10.1504/IJWGS.2005.007545. URL : http://dx.doi.org/10.1504/IJWGS.2005.007545.

[52] David Eichmann et Miguel E. Ruiz. "Cross-Language Information Retrieval with the UMLS Metathesaurus". Dans : *In : Proc. of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1998, p. 72–80.

[53] Daniel Elenius et al. "The OWL-S editor &#8211 ; a development tool for semantic web services". Dans : ESWC'05 (2005), p. 78–92. DOI : 10.1007/11431053_6. URL : http://dx.doi.org/10.1007/11431053_6.

[54] O. Erling et I. Mikhailov. "RDF Support in the Virtuoso DBMS". Dans : *Proceedings of the 1st Conference on Social Semantic Web CSSW*. T. 221. Springer. 2007.

[55] Thomas Erl. *Service-Oriented Architecture : A Field Guide to Integrating XML and Web Services*. Upper Saddle River, NJ, USA : Prentice Hall PTR, 2004. ISBN : 0131428985.

[56] Thomas Erl. *Service-Oriented Architecture : Concepts, Technology, and Design*. Upper Saddle River, NJ, USA : Prentice Hall PTR, 2005. isbn : 0131858580.

[57] Dieter Fensel et al. *On2broker : Semantic-Based Access to Information Sources at the WWW*. 1999.

[58] Dieter Fensel et al. *Ontobroker : The Very High Idea*. 1998.

[59] G Fortier J.-Y. & Kassel. "Managing knowledge at the information level : an ontological approach". Anglais. Dans : Dordrecht, Kluwer, 2004, p. 39–45.

[60] Joe Futrelle et al. "Semantic middleware for e-Science knowledge spaces". Dans : *Concurr. Comput. : Pract. Exper.* 23.17 (déc. 2011), p. 2107–2117. issn : 1532-0626. doi : 10.1002/cpe.1705. url : http://dx.doi.org/10.1002/cpe.1705.

[61] L. Gandon Fabien. *Ontology Engineering : a Survey and a Return on Experience*. Anglais. Rap. tech. RR-4396. INRIA, mar. 2002. url : http://hal.inria.fr/inria-00072192.

[62] "Core Ontologies in Ontology Engineering 2004. (Un)Successful cases and best practices for ontology engineering : reusing well-founded ontologies for domain content specification." Dans : Proceedings of the EKAW*04 Workshop on Core Ontologies in Ontology Engineering (oct. 2004). Sous la dir. d'Aldo Gangemi et Stefano Borgo. url : http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-118/.

[63] Gerald C. Gannod, Raynette J. Brodie et John T. E. Timm. "An Interactive Approach for Specifying OWL-S Groundings". Dans : *Proceedings of the Ninth IEEE International EDOC Enterprise Computing Conference*. EDOC '05. Washington, DC, USA : IEEE Computer Society, 2005, p. 251–260. isbn : 0-7695-2441-9. doi : 10.1109/EDOC.2005.7. url : http://dx.doi.org/10.1109/EDOC.2005.7.

[64] Daniel Gardner et al. "The Neuroscience Information Framework : A Data and Knowledge Environment for Neuroscience". Dans : *Neuroinformatics* (2008), p. 149–160.

[65] Tristan Glatard et al. "Flexible and Efficient Workflow Deployment of Data-Intensive Applications On Grids With MOTEUR". Dans : *IJHPCA* 22.3 (2008), p. 347–360.

[66] Francois Goasdoue, Veronique Lattes et Marie-Christine Rousset. "The Use of CARIN Language and Algorithms for Information Integration : The PICSEL System". Dans : *International Journal of Cooperative Information Systems* 9.4 (2000), p. 383–401. url : citeseer.nj.nec.com/goasdoue00use.html.

[67] Carole Goble et Robert Stevens. "State of the nation in data integration for bioinformatics". Dans : *J. of Biomedical Informatics* 41 (5 oct. 2008), p. 687–693. issn : 1532-0464. doi : 10.1016/j.jbi.2008.01.008. url : http://dl.acm.org/citation.cfm?id=1435008.1435208.

[68] Benjamin M. Good et Mark D. Wilkinson. "The Life Sciences Semantic Web is Full of Creeps !" Dans : *Briefings in Bioinformatics* 7.3 (2006), p. 275–286.

[69] Paul M. K. Gordon et Christoph W. Sensen. "Seahawk : moving beyond HTML in Web-based bioinformatics analysis". Dans : *BMC Bioinformatics* 8 (18 juin 2007), p. 208+. issn : 1471-2105. doi : 10.1186/1471-2105-8-208. url : http://dx.doi.org/10.1186/1471-2105-8-208.

[70] "Proceedings of the 2nd International Workshop on Modular Ontologies, WoMO 2007, Whistler, Canada, October 28, 2007". Dans : *WoMO*. Sous la dir. de Bernardo Cuenca Grau et al. T. 315. CEUR Workshop Proceedings. CEUR-WS.org, 2008.

[71] N. Guarino et P. Giaretta. "Ontologies and Knowledge Bases : Towards a Terminological Clarification". Dans : (1995). Sous la dir. de N. J. I. Mars, p. 25–32.

[72] R. Guha, Rob Mccool et Eric M. "Semantic Search". Dans : *INTERNATIONAL WORLD WIDE WEB CONFERENCE*. ACM, 2003, p. 700–709.

[73] Mohand-Said Hacid et Chantal Reynaud. "L'intégration de sources de données". fr. Dans : *Revue Information - Interaction - Intelligence (I3) Une Revue en Sciences du Traitement de l'I* 4.2 (juin 2004). Numéro hors série. url : http://liris.cnrs.fr/publis/?id=1979.

[74] Farshad Hakimpour et al. "Integration of OWL-S into IRS-III". Dans : *University, Milton Keynes, UK*. 2004.

[75] Alon Halevy, Anand Rajaraman et Joann Ordille. "Data integration : the teenage years". Dans : *VLDB '06 : Proceedings of the 32nd international conference on Very large data bases*. Seoul, Korea : VLDB Endowment, 2006, p. 9–16. url : http://portal.acm.org/citation.cfm?id=1182635.1164130.

[76] Jeff Heflin et James Hendler. *Searching the Web with SHOE*. 2000.

[77] J. Hendler et D. L. McGuinness. "The DARPA Agent Markup Language". Dans : *IEEE Intelligent Systems* 15.6 (2000), p. 67–73. url : http://www.ksl.stanford.edu/people/dlm/papers/ieee-trends-daml-final-version.html.

[78] Ian Horrocks, Ulrike Sattler et Stephan Tobies. "Practical Reasoning for Expressive Description Logics". Dans : LPAR '99 (1999), p. 161–180. url : http://dl.acm.org/citation.cfm?id=645709.664314.

[79] "Interactive Web Workflow Enactor & Manager". Dans : (). url : http://ubio.bioinfo.cnio.es/biotools/IWWEM/.

[80] M. A. ISMAIL, M. YAAKOB et S. A KAREEM, éds. *Semantic Search Engine in Institutional Repository : An Ontological Approach*. T. 3485. Lecture Notes in Computer Science. Kuala Lumpur-Malaysia : Information et Society ICoLIS 2007, 2007.

[81] Narendranadh Jabisetti et Yugyung Lee. "OWL-S Based Autonomic Services for Grid Computing". Dans : *Proceedings of the IEEE International Conference on Web Services*. ICWS '05. Washington, DC, USA : IEEE Computer Society, 2005, p. 825–826. isbn : 0-7695-2409-5. doi : 10.1109/ICWS.2005.89. url : http://dx.doi.org/10.1109/ICWS.2005.89.

[82] Stéphane JEAN et al. "Prise en compte des standards de qualité et des préférences utilisateurs pour la modélisation des propriétés non fonctionnelles dans OWL-S". Dans : *Technique et Science Informatiques* 31 (2012).

[83] Nicolai JOSUTTIS. *Soa in Practice : The Art of Distributed System Design*. O'Reilly Media, Inc., 2007. ISBN : 0596529554.

[84] Gilles KASSEL. "Integration of the DOLCE top-level ontology into the OntoSpec methodology". Dans : t. abs/cs/0510050. 2005.

[85] G. KASSEL et al. "Des Artefacts aux Programmes". Dans : *1ères Journées Francophones sur les Ontologies (JFO'2007)*. Sousse, Tunisie, oct. 2007, p. 281–300.

[86] Toshiaki KATAYAMA et al. "The DBCLS BioHackathon : standardization and interoperability for bioinformatics web services and workflows." Dans : *Journal of Biomedical Semantics* 1.1 (2010), p. 8. ISSN : 2041-1480. DOI : 10.1186/2041-1480-1-8. URL : http://www.jbiomedsem.com/content/1/1/8.

[87] Michael KIFER et al. "A realistic architecture for the semantic web". Dans : *In RuleML*. Springer, 2005, p. 17–29.

[88] Atanas KIRYAKOV, Damyan OGNYANOV et Dimitar MANOV. "OWLIM – a pragmatic semantic repository for OWL". Dans : *Proceedings of the 2005 international conference on Web Information Systems Engineering*. WISE'05. New York, NY : Springer-Verlag, 2005, p. 182–192. ISBN : 3-540-30018-X, 978-3-540-30018-2. DOI : 10.1007/11581116_19. URL : http://dx.doi.org/10.1007/11581116_19.

[89] Matthias KLUSCH, Benedikt FRIES et Mahboob KHALID. "Owls-mx : Hybrid owl-s service matchmaking". Dans : *In Proceedings of 1st Intl. AAAI Fall Symposium on Agents and the Semantic Web*. 2005.

[90] Matthias KLUSCH et Andreas GERBER. "Evaluation of Service Composition Planning with OWLS-XPlan". Dans : *Proceedings of the 2006 IEEE/WIC/ACM international conference on Web Intelligence and Intelligent Agent Technology*. WI-IATW '06. Washington, DC, USA : IEEE Computer Society, 2006, p. 117–120. ISBN : 0-7695-2749-3. DOI : 10.1109/WI-IATW.2006.68. URL : http://dx.doi.org/10.1109/WI-IATW.2006.68.

[91] Matthias KLUSCH et Andreas GERBER. "Semantic web service composition planning with OWLS-XPlan". Dans : *In Proceedings of the 1st Int. AAAI Fall Symposium on Agents and the Semantic Web*. 2005, p. 55–62.

[92] Holger KNUBLAUCH. *An AI tool for the real world Knowledge modeling with Prot&eacute ;g&eacute :* 20 juin 2003.

[93] Holger KNUBLAUCH, Olivier DAMERON et Mark A MUSEN. "Weaving the biomedical Semantic Web with the Protégé OWL Plugin". Dans : *KR-MED*. 2004, p. 39–47. DOI : DBLP:conf/krmed/2004.

[94] Dirk KRAFZIG, Karl BANKE et Dirk SLAMA. *Enterprise SOA : Service-Oriented Architecture Best Practices (The Coad Series)*. Upper Saddle River, NJ, USA : Prentice Hall PTR, 2004. ISBN : 0131465759. URL : http://fparreiras/books/EnterpriseSOAServiceOrien.chm.

[95] H. KREGER. "Web Services Conceptual Architecture (WSCA 1.0)". Dans : (2001).

[96] Anna-Lena LAMPRECHT, Tiziana MARGARIA et Bernhard STEFFEN. "Bio-jETI : a framework for semantics-based service composition." Dans : *BMC Bioinformatics* 10.S-10 (1er juil. 2010), p. 8. URL : http://dblp.uni-trier.de/db/journals/bmcbi/bmcbi10S.html#LamprechtMS09.

[97] Anna L. LAMPRECHT, Tiziana MARGARIA et Bernhard STEFFEN. "Supporting Process Development in Bio-jETI by Model Checking and Synthesis". Dans : *SWAT4LS Proceedings*. Edinburgh, Scotland, UK : CEUR-WS.org Workshop Proceedings, nov. 2008.

[98] Pascal LANDO et al. "Premiers pas vers une ontologie générale des programmes informatiques." Dans : *Actes d'IC*. Sous la dir. de Francky TRICHET. Cepadues, 6 juin 2009, p. 25–37. URL : http://dblp.uni-trier.de/db/conf/f-ic/ic2007.html#LandoFKL07.

[99] Holger LAUSEN et al. "WSML - a Language Framework for Semantic Web Services". Dans : *W3C Workshop on Rule Languages for Interoperability*. 2005.

[100] Nicolas LEBRETON et al. "Verification of parameters semantic compatibility for semi-automatic web service composition : a generic case study". Dans : *iiWAS*. 2010, p. 845–848. DOI : DBLP:conf/iiwas/2010.

[101] Jinsoo LEE et al. "Processing SPARQL queries with regular expressions in RDF databases". Dans : *BMC Bioinformatics* 12 (2011), p. 1–11.

[102] Henry H. LEITNER et Michael U. FREEMAN. "Structured inheritance networks and natural language understanding". Dans : (1979), p. 525–530. URL : http://dl.acm.org/citation.cfm?id=1624861.1624981.

[103] Yuangui LEI, Victoria S. UREN et Enrico MOTTA. "SemSearch : A Search Engine for the Semantic Web." Dans : *EKAW*. Sous la dir. de Steffen STAAB et Vojtech SVFFFDTEK. T. 4248. Lecture Notes in Computer Science. Springer, 24 oct. 2006, p. 238–245. ISBN : 3-540-46363-1. DOI : conf/ekaw/2006. URL : http://dblp.uni-trier.de/db/conf/ekaw/ekaw2006.html#LeiUM06.

[104] Maurizio LENZERINI. "Data integration : a theoretical perspective". Dans : *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. PODS '02. Madison, Wisconsin : ACM, 2002, p. 233–246. ISBN : 1-58113-507-6. DOI : 10.1145/543613.543644. URL : http://doi.acm.org/10.1145/543613.543644.

[105] Salayandia LEONARDO et Gates ANN Q. "Towards a Workflow Management System for Service Oriented Modules". Dans : *In International Journal of Simulation and Process Modeling* 3.2 (2007), p. 1–2.

[106] James LINDSAY. *XML Databases and Biomedical Informatics*. 2008.

[107] Shih-Hsi LIU et al. "A Semantics and Data-Driven Biomedical Multimedia Software System." Dans : *Journal of Multimedia* 5.4 (2010), p. 352–360. URL : http://dblp.uni-trier.de/db/journals/jmm2/jmm5.html#LiuCLKST10.

[108] Vanessa Lopez, Michele Pasin et Enrico Motta. "AquaLog : An Ontology-Portable Question Answering System for the Semantic Web." Dans : *ESWC*. Sous la dir. d'Asunción Gómez-Pérez et Jérôme Euzenat. T. 3532. Lecture Notes in Computer Science. Springer, 24 mai 2005, p. 546–562. isbn : 3-540-26124-9. doi : conf/esws/2005. url : http://dblp.uni-trier.de/db/conf/esws/eswc2005.html#LopezPM05.

[109] Céline Lopez-Velasco. "Sélection et composition de services Web pour la génération d'applications adaptées au contexte d'utilisation". Français. THESE. Université Joseph-Fourier - Grenoble I, nov. 2008. url : http://tel.archives-ouvertes.fr/tel-00388991.

[110] Phillip Lord et al. "Feta : A light-weight architecture for user oriented semantic service discovery". Dans : *Proceedings of the European Semantic Web Conference 2005*. Sous la dir. d'Asunción G. Pérez et Jérôme Euzenat. T. 3532. Lecture Notes in Computer Science. Springer-Verlag, 2005, p. 17–31. doi : 10.1007/11431053_2. url : http://homepages.cs.ncl.ac.uk/phillip.lord/download/publications/european_semantic_web2005_feta.pdf.

[111] Temal Lynda et al. "Towards an ontology for sharing medical images and regions of interest in neuroimaging". Dans : *J. of Biomedical Informatics* 41.5 (oct. 2008), p. 766–778. issn : 1532-0464. doi : 10.1016/j.jbi.2008.03.002. url : http://dx.doi.org/10.1016/j.jbi.2008.03.002.

[112] Allan J. Mackenzie-Graham et al. "Provenance and Annotation of Data and Processes". Dans : sous la dir. de Juliana Freire, David Koop et Luc Moreau. Berlin, Heidelberg : Springer-Verlag, 2008. Chap. Neuroimaging Data Provenance Using the LONI Pipeline Workflow Environment, p. 208–220. isbn : 978-3-540-89964-8. doi : 10.1007/978-3-540-89965-5_22. url : http://dx.doi.org/10.1007/978-3-540-89965-5_22.

[113] Ana G. Maguitman et al. "Algorithmic detection of semantic similarity". Dans : WWW '05 (2005), p. 107–116. doi : 10.1145/1060745.1060765. url : http://doi.acm.org/10.1145/1060745.1060765.

[114] Christoph Mangold. "A survey and classification of semantic search approaches". Dans : *Int. J. Metadata Semant. Ontologies* 2.1 (sept. 2007), p. 23–34. issn : 1744-2621. doi : 10.1504/IJMSO.2007.015073. url : http://dx.doi.org/10.1504/IJMSO.2007.015073.

[115] Frank Manola et Eric Miller. *RDF Primer*. http ://www.w3.org/TR/2004/REC-rdf-primer-20040210/. Fév. 2004. url : http://www.w3.org/TR/2004/REC-rdf-primer-20040210/.

[116] Devignes Marie-Dominique et al. "BioRegistry : automatic extraction of metadata for biological database retrieval and discovery". Dans : *iiWAS*. 2008, p. 456–461.

[117] David Martin et al. "Bringing Semantics to Web Services with OWL-S". Dans : *World Wide Web* 10.3 (sept. 2007), p. 243–277. issn : 1386-145X. doi : 10.1007/s11280-007-0033-x. url : http://dx.doi.org/10.1007/s11280-007-0033-x.

[118] David Martin et al. "OWL-S : Semantic Markup for Web Services". Dans : (nov. 2004). URL : http://www.w3.org/Submission/OWL-S/.

[119] Erick Martínez et Yves Lespérance. "Web Service Composition as a Planning Task : Experiments using Knowledge-Based Planning". Dans : *Proceedings of the Workshop on Planning and Scheduling for Web and Grid Services (ICAPS-2004)*. Whistler, juin 2004, p. 62–69.

[120] Maryann E. Martone, Amarnath Gupta et Mark H. Ellisman. "e-Neuroscience : challenges and triumphs in integrating distributed data from molecules to brains". Dans : *Nature Neuroscience* 7.5 (27 avr. 2004), p. 467–472. ISSN : 1097-6256. DOI : 10.1038/nn1229. URL : http://dx.doi.org/10.1038/nn1229.

[121] Claudio Masolo et al. *The WonderWeb Library of Foundational Ontologies and the DOLCE ontology*. Rap. tech. 2002. URL : http://www.loa-cnr.it/Papers/DOLCE2.1-FOL.pdf.

[122] Brian McBride. "Jena : A Semantic Web Toolkit". Dans : *IEEE Internet Computing* 6.6 (2002), p. 55–59.

[123] Ammar Mechouche et al. "A Hybrid System Using Symbolic and Numeric Knowledge for the Semantic Annotation of Sulco-Gyral Anatomy in Brain MRI Images". Dans : *IEEE Trans. Med. Imaging* 28.8 (2009), p. 1165–1178.

[124] Brahim Medjahed. "Semantic web enabled composition of web services". AAI3123728. Thèse de doct. 2004.

[125] Brahim Medjahed, Athman Bouguettaya et Ahmed K. Elmagarmid. "Composing Web services on the Semantic Web". Dans : *The VLDB Journal* 12.4 (nov. 2003), p. 333–351. ISSN : 1066-8888. DOI : 10.1007/s00778-003-0101-5. URL : http://dx.doi.org/10.1007/s00778-003-0101-5.

[126] F. Michel et al. "Grid-wide neuroimaging data federation in the context of the NeuroLOG project". Dans : *8th HealthGrid Conference (HG'10)*. Orsay, France : IOS Press, 2010, p. 112–123.

[127] Anton Michlmayr et al. "End-to-End Support for QoS-Aware Service Selection, Binding, and Mediation in VRESCo". Dans : *IEEE Trans. Serv. Comput.* 3.3 (juil. 2010), p. 193–205. ISSN : 1939-1374. DOI : 10.1109/TSC.2010.20. URL : http://dx.doi.org/10.1109/TSC.2010.20.

[128] Peter Mika, Daniel Oberle et Aldo Gangemi. "Foundations for Service Ontologies : Aligning OWL-S to DOLCE". Dans : ACM, 2004, p. 563–572.

[129] S. Miles et al. "Personalised grid service discovery". Dans : *IEE Proceedings Software : Special Issue on Performance Engineering*. 2003, p. 131–140.

[130] M.Mesiti et al. *Integration and Opportunities in Biological XML Data Management*. Hershey : Information Science Publishing, 2009, p. 263 –286. ISBN : 9781605663081.

[131] Sabah Mohammed, Jinan Fiaidhi et Marshal Hahn. "A UDDI Search Engine for SVG Federated Medical Imaging Web Services". Dans : (2006). URL : http://www.scipub.org/fulltext/jcs/jcs24303-313.pdf.

[132] Sonia Ben MOKHTAR, Nikolaos GEORGANTAS et Valérie ISSARNY. "Cocoa : Conversationbased service composition in pervasive computing environments". Dans : *In Proceedings of the IEEE International Conference on Pervasive Services (ICPS xfffd06*. 2006.

[133] Johan MONTAGNAT et al. "A data-driven workflow language for grids based on array programming principles". Dans : *Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science*. WORKS '09. Portland, Oregon : ACM, 2009, 7 :1–7 :10. ISBN : 978-1-60558-717-2. DOI : 10.1145/1645164.1645171. URL : http://doi.acm.org/10.1145/1645164.1645171.

[134] Johan MONTAGNAT et al. "NeuroLOG : a community-driven middleware design". Anglais. Dans : *HealthGrid*. Chicago, États-Unis : IOS Press, juin 2008, p. 49–58. URL : http://hal.archives-ouvertes.fr/hal-00461611.

[135] Luc MOREAU. "Provenance in myGrid and beyond". Dans : (2004). URL : twiki.pasoa.ecs.soton.ac.uk/pub/PASOA/PresentationStore/iam04.ppt.

[136] E. MOTTA. *Reusable Components for Knowledge Modelling : Case Studies in Parametric Design Problem Solving*. 1st. Amsterdam, The Netherlands, The Netherlands : IOS Press, 1999. ISBN : 1586030035.

[137] Daniele NARDI et Ronald J. BRACHMAN. *An Introduction to Description Logics*. Sous la dir. de F. BAADER et al. Cambridge University Press, 2003, p. 1–40.

[138] NICOLA. "An Overview of OntoClean". Dans : *Handbook on Ontologies*. Sous la dir. de STEFFEN. Springer, 2004. Chap. 8, p. 151–172.

[139] OASIS. "Using WSDL in a UDDI Registry, Version 1.08". Dans : (jan. 2013). URL : http://www.oasis-open.org/committees/uddi-spec/doc/bp/uddi-spec-tc-bp-using-wsdl-v108-20021110.htm.

[140] *Reference Model for Service Oriented Architecture 1.0*. Website. Août 2006. URL : http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf.

[141] Daniel OBERLE et al. "Towards ontologies for formalizing modularization and communication in large software systems". Dans : *Appl. Ontol.* 1.2 (avr. 2006), p. 163–202. ISSN : 1570-5838. URL : http://dl.acm.org/citation.cfm?id=1412362.1412366.

[142] Martin O'CONNOR et al. "Supporting Rule System Interoperability on the Semantic Web with SWRL". Dans : 2005, p. 974–986. DOI : 10.1007/11574620_69. URL : http://dx.doi.org/10.1007/11574620_69.

[143] Seog-chan OH et al. "Semantic web-service discovery and composition using flexible parameter matching". Dans : *In Proceedings of IEEE Joint Conference (CEC/EEE 2007) on E-Commerce Technology (9th CEC xfffd07) and Enterprise Computing, E-Commerce and E-Services (4th EEE xfffd07*. 2007, p. 533–536.

[144] Tom OINN et al. "Taverna : lessons in creating a workflow environment for the life sciences : Research Articles". Dans : *Concurr. Comput. : Pract. Exper.* 18.10 (août 2006), p. 1067–1100. ISSN : 1532-0626. DOI : 10.1002/cpe.v18:10. URL : http://dx.doi.org/10.1002/cpe.v18:10.

[145] Nicole Oldham et al. "METEOR-S web service annotation framework with machine learning classification". Dans : *Proceedings of the First international conference on Semantic Web Services and Web Process Composition*. SWSWPC'04. San Diego, CA : Springer-Verlag, 2005, p. 137–146. ISBN : 3-540-24328-3, 978-3-540-24328-1. DOI : 10.1007/978-3-540-30581-1_12. URL : http://dx.doi.org/10.1007/978-3-540-30581-1_12.

[146] Massimo Paolucci, Naveen Srinivasan et Katia Sycara. "Expressing WSMO Mediators in OWLS". Dans : *In Proceedings of the workshop on Semantic Web Services : Preparing to Meet the World of Business Applications held at the 3rd International Semantic Web Conference (ISWC 2004)*. 2004.

[147] Massimo Paolucci, Katia Sycara et Takahiro Kawamura. "Delivering Semantic Web Services". Dans : *IEEE Internet Computing*. 2003, p. 34–41.

[148] Michael P. Papazoglou. *Web Services : Principles and Technology*. Pearson, Prentice Hall, 2008.

[149] Michael P. Papazoglou et al. *SERVICE-ORIENTED COMPUTING : A Research Roadmap*. 2008.

[150] Michael P. Papazoglou et al. "Service-Oriented Computing : State of the Art and Research Challenges". Dans : *Computer* 40 (2007), p. 38–45. ISSN : 0018-9162. DOI : http://doi.ieeecomputersociety.org/10.1109/MC.2007.400.

[151] Mike P. Papazoglou. "Service -Oriented Computing : Concepts, Characteristics and Directions". Dans : *Web Information Systems Engineering, International Conference on* (2003), p. 3–12. DOI : 10.1109/WISE.2003.1254461. URL : http://dx.doi.org/10.1109/WISE.2003.1254461.

[152] Mike P. Papazoglou. "The Challenges of Service Evolution". Dans : *CAiSE*. 2008, p. 1–15.

[153] Jinsoo Park et Sudha Ram. "Information systems interoperability : What lies beneath ?" Dans : *ACM Trans. Inf. Syst.* (2004), p. 595–632.

[154] Jyotishman Pathak, Samik Basu Robyn Lutz et Vasant Honavar. "MoSCoE : A Framework for Modeling Web Service Composition and Execution". Dans : *In IEEE 22nd Intl. Conference on Data Engineering Ph.D. Workshop*. IEEE CS Press, 2006, p. 143.

[155] Carlos Pedrinaci, John Domingue et AmitP. Sheth. "Semantic Web Services". Dans : *Handbook of Semantic Web Technologies*. Sous la dir. de John Domingue, Dieter Fensel et JamesA. Hendler. Springer Berlin Heidelberg, 2011, p. 977–1035. ISBN : 978-3-540-92912-3. DOI : 10.1007/978-3-540-92913-0_22. URL : http://dx.doi.org/10.1007/978-3-540-92913-0_22.

[156] C. Pedrinaci, J. Domingue et A. Sheth. *Semantic Web Services*. T. 29. Springer, preprint, 2010.

[157] Simeon Petkov, Eyal Oren et Armin Haller. *Aspects in Workflow Management*. Rap. tech. DERI-TR-2005-04-10. Digital Enterprise Research Institute (DERI), 2005.

[158] Marco PISTORE, Pierluigi ROBERTI et Paolo TRAVERSO. "Process-Level composition of executable web services : ”on-the-fly” versus ”once-for-all” composition". Dans : *Proceedings of the Second European conference on The Semantic Web : research and Applications*. ESWC'05. Heraklion, Greece : Springer-Verlag, 2005, p. 62–77. ISBN : 3-540-26124-9, 978-3-540-26124-7. DOI : 10.1007/11431053_5. URL : http://dx.doi.org/10.1007/11431053_5.

[159] M. PISTORE et al. "Planning and Monitoring Web Service Composition". Dans : *Artificial Intelligence : Methodology, Systems, and Applications*. Sous la dir. de Christoph BUSSLER et Dieter FENSEL. T. 3192. Lecture Notes in Computer Science. Berlin, Heidelberg : Springer Berlin / Heidelberg, 2004. Chap. 11, p. 106–115. ISBN : 978-3-540-22959-9. DOI : 10.1007/978-3-540-30106-6_11. URL : http://dx.doi.org/10.1007/978-3-540-30106-6_11.

[160] J. B. POLINE et Et AL. "Data sharing in neuroimaging research". Dans : *Frontiers in Neuroinformatics* (2012). URL : http://www.frontiersin.org/Neuroinformatics/10.3389/fninf.2012.00009/full.

[161] Eric PRUD'HOMMEAUX et Benjamin GROSOF. *RDF Query Survey*. Technical Report. http ://www.w3.org/2001/11/13-RDF-Query-Rules/. Avr. 2004. URL : http://www.w3.org/2001/11/13-RDF-Query-Rules/.

[162] Preeda RAJASEKARAN et al. "Enhancing web services description and discovery to facilitate composition". Dans : *Proceedings of the First international conference on Semantic Web Services and Web Process Composition*. SWSWPC'04. San Diego, CA : Springer-Verlag, 2005, p. 55–68. ISBN : 3-540-24328-3, 978-3-540-24328-1. DOI : 10.1007/978-3-540-30581-1_6. URL : http://dx.doi.org/10.1007/978-3-540-30581-1_6.

[163] Sergio RAMÍREZ et al. "MOWServ : a web client for integration of bioinformatic resources". Dans : *Nucleic Acids Research* 38.Web-Server-Issue (2010), p. 671–676.

[164] Jinghai RAO et Xiaomeng SU. "A Survey of Automated Web Service Composition Methods". Dans : *In Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition, SWSWPC 2004*. 2004, p. 43–54.

[165] Alan L. RECTOR. "Modularisation of domain ontologies implemented in description logics and related formalisms including OWL". Dans : *Proceedings of the 2nd international conference on Knowledge capture*. K-CAP '03. Sanibel Island, FL, USA : ACM, 2003, p. 121–128. ISBN : 1-58113-583-1. DOI : 10.1145/945645.945664. URL : http://doi.acm.org/10.1145/945645.945664.

[166] Walter RENTERIA-AGUALIMPIA et al. "Exploring the Advances in Semantic Search Engines". Dans : *DCAI*. 2010, p. 613–620. DOI : DBLP:conf/dcai/2010.

[167] Nicolas REPP et al. "On distributed SLA monitoring and enforcement in service-oriented systems". Anglais. Dans : *International Journal On Advances in Systems and Measurements* 2.1 (juin 2009), p. 33–43. ISSN : 1942-261x. URL : ftp://ftp.kom.tu-darmstadt.de/papers/RSS+09.pdf.

[168] Victoria Martin Requena et al. "jORCA : easily integrating bioinformatics Web Services". Dans : *Bioinformatics* (2010), p. 553–559.

[169] Cristiano Rocha, Daniel Schwabe et Marcus Poggi Aragao. "A hybrid approach for searching in the semantic web". Dans : *Proceedings of the 13th international conference on World Wide Web*. WWW '04. New York, NY, USA : ACM, 2004, p. 374–383. ISBN : 1-58113-844-X. DOI : 10.1145/988672.988723. URL : http://doi.acm.org/10.1145/988672.988723.

[170] Dumitru Roman et al. "Web Service Modeling Ontology". Dans : *Appl. Ontol.* 1.1 (jan. 2005), p. 77–106. ISSN : 1570-5838. URL : http://dl.acm.org/citation.cfm?id=1412350.1412357.

[171] Alan Ruttenberg et al. "Advancing translational research with the Semantic Web." Dans : *BMC Bioinformatics* 8.S-3 (2007). URL : http://dblp.uni-trier.de/db/journals/bmcbi/bmcbi8S.html#RuttenbergCBSBCDFGKKLMORSWWZHNHC07.

[172] G Schneider et al. "Integrated Tools for Biomolecular Sequence-Based Function Prediction as Exemplified by the ANNOTATOR Software Environment". Dans : *Data Mining Techniques for the Life Sciences* 609 (2010), p. 257–267. DOI : 10.1007/978-1-60327-241-4_15.

[173] Christian Schönbach, P. Kowalski-Saunders et Vladimir Brusic. "Data Warehousing in Molecular Biology." Dans : *Briefings in Bioinformatics* 1.1 (2000), p. 190–198. URL : http://dblp.uni-trier.de/db/journals/bib/bib1.html#SchonbachKB00.

[174] Roy W Schulte et Yefim V Natis. "Service Oriented Architectures : Part 1". Dans : (1996), p. 2. URL : http://www.gartner.com/DisplayDocument?id=302868.

[175] J Severin et al. "FANTOM4 EdgeExpressDB : an integrated database of promoters, genes, microRNAs, expression dynamics and regulatory interactions". Dans : *Genome biology* 10 (2009), R39. DOI : 10.1186/gb-2009-10-4-r39.

[176] Nigel Shadbolt, Tim Berners-Lee et Wendy Hall. "The Semantic Web Revisited". Dans : *IEEE Intelligent Systems* 21.3 (2006), p. 96–101.

[177] Jun Shen et al. "From BPEL4WS to OWL-S : Integrating E-Business Process Descriptions". Dans : *Proceedings of the 2005 IEEE International Conference on Services Computing - Volume 01*. SCC '05. Washington, DC, USA : IEEE Computer Society, 2005, p. 181–190. ISBN : 0-7695-2408-7-01. DOI : 10.1109/SCC.2005.54. URL : http://dx.doi.org/10.1109/SCC.2005.54.

[178] Evren Sirin. "Combining description logic reasoning with ai planning for composition of web services". AAI3241437. Thèse de doct. College Park, MD, USA, 2006. ISBN : 978-0-542-96121-2.

[179] Evren Sirin et al. "HTN planning for Web Service composition using SHOP2". Dans : *Web Semant.* 1.4 (oct. 2004), p. 377–396. ISSN : 1570-8268. DOI : 10.1016/j.websem.2004.06.005. URL : http://dx.doi.org/10.1016/j.websem.2004.06.005.

[180] K. Sivashanmugam et al. *Adding semantics to web services standards*. 2003. URL : http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.14.6260.

[181] Barry SMITH et Werner CEUSTERS. "Ontological realism : A methodology for coordinated evolution of scientific ontologies". Dans : *Appl. Ontol.* 5.3-4 (août 2010), p. 139–188. ISSN : 1570-5838. URL : http://dl.acm.org/citation.cfm?id=1889917.1889919.

[182] J. F. SOWA. *Conceptual Structures : Information Processing in Mind and Machine*. Reading : Addison-Wesley, 1984.

[183] "Specification of the NeuroLOG architecture components". Dans : (jan. 2007). URL : http://neurolog.i3s.unice.fr/_media/public_namespace/neurolog-l3.pdf.

[184] Workflow Management Coalition SPECIFICATION. *Workflow Management Coalition, Terminology & Glossary (Document No. WFMC-TC-1011)*. Workflow Management Coalition Specification, fév. 1999. URL : http://www.wfmc.org/Download-document/WFMC-TC-1011-Ver-3-Terminology-and-Glossary-English.html.

[185] Ralf STEINMETZ et Klaus WEHRLE, éds. *Peer-to-Peer Systems and Applications*. T. 3485. Lecture Notes in Computer Science. Springer, 2005. ISBN : 3-540-29192-X.

[186] H. STUCKENSCHMIDT. *Ontologien : Konzepte, Technologien Und Anwendungen*. Informatik Im Fokus. Springer, 2011. ISBN : 9783642054037. URL : http://books.google.fr/books?id=F1HFirsck58C.

[187] SUN. "Service-Oriented Architecture (SOA)". Dans : (jan. 2013). URL : http://192.9.172.90/products/soa/benefits.jsp.

[188] Martin SVIHLA et Ivan JELINEK. "Two Layer Mapping from Database to RDF". Dans : *Proceedings of the Sixth International Scientific Conference Electronic Computers and Informatics ECI 2004*. 2004, p. 270–275.

[189] Katia P. SYCARA et al. "Automated discovery, interaction and composition of Semantic Web services". Dans : *J. Web Sem.* 1.1 (2003), p. 27–46.

[190] Katia SYCARA et al. "Larks : Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace". Dans : *Autonomous Agents and Multi-Agent Systems* 5.2 (juin 2002), p. 173–203. ISSN : 1387-2532. DOI : 10.1023/A:1014897210525. URL : http://dx.doi.org/10.1023/A:1014897210525.

[191] Ian TAYLOR et al. *Triana WorkFlow Specification*. Rap. tech. GridLab.

[192] "Ontologie de partage de donnfffdes et d'outils de traitement dans le domaine de la neuroimagerie". Thfffdse de doctorat : Mfffddecine. Vie-agro-santfffd. Mention gfffdnie biologique et mfffddical. Thèse de doct. Rennes : Rennes 1, 2008. URL : http://opac.inria.fr/record=b1129587.

[193] Lynda TEMAL et al. "OntoNeuroBase : a multi-layered application ontology in neuroimaging". Anglais. Dans : *Second Workshop : Formal Ontologies Meet Industry (FOMI 2006)*. trento, Italie, déc. 2006. URL : http://www.hal.inserm.fr/inserm-00140523.

[194] L. TEMAL et al. "OntoNeuroLOG : une ontologie modulaire et multi-niveaux pour gfffdrer l'hfffdtfffdrogfffdnfffditfffd sfffdmantique des mfffdtadonnfffdes". Dans : *Journfffde thfffdmatique "Ontologies et gestion de l'Hfffdtfffdrogfffdnfffditfffd Sfffdmantique" du GDR I3*. Grenoble, France, juil. 2007.

[195] Aditya THATTE. "Service Oriented Computing : Publish , Find , Bind". Dans : (jan. 2013). URL : http://bethorough.wordpress.com/.

[196] Aphrodite TSALGATIDOU et Thomi PILIOURA. "An Overview of Standards and Related Technology in Web Services". Dans : *Distrib. Parallel Databases* 12.2-3 (sept. 2002), p. 135–162. ISSN : 0926-8782. DOI : 10.1023/A:1016599017660. URL : http://dx.doi.org/10.1023/A:1016599017660.

[197] Roman VACULIN et Katia SYCARA. "Semantic Web Services Monitoring : An OWL-S based Approach". Dans : *Hawaii International Conference on System Sciences*. IEEE Computer Society, jan. 2008.

[198] Pierpaolo VITTORINI, Monica MICHETTI et Ferdinando di ORIO. "A SOA statistical engine for biomedical data". Dans : *Comput. Methods Prog. Biomed.* 92.1 (oct. 2008), p. 144–153. ISSN : 0169-2607. DOI : 10.1016/j.cmpb.2008.06.006. URL : http://dx.doi.org/10.1016/j.cmpb.2008.06.006.

[199] Tomas VITVAR et al. "WSMO-lite annotations for web services". Dans : *Proceedings of the 5th European semantic web conference on The semantic web : research and applications*. ESWC'08. Tenerife, Canary Islands, Spain : Springer-Verlag, 2008, p. 674–689. ISBN : 3-540-68233-3, 978-3-540-68233-2. URL : http://dl.acm.org/citation.cfm?id=1789394.1789455.

[200] Bacem WALI et Bernard GIBAUD. "Extending OWL-S for the Composition of Web Services Generated With a Legacy Application Wrapper". Anglais. Dans : *ICIW2012 The Seventh International Conference on Internet and Web Applications and Services*. Stuttgart, Allemagne, mai 2012, p. 97 –105. URL : http://hal.archives-ouvertes.fr/hal-00736765.

[201] Bacem WALI et Bernard GIBAUD. "Semantic annotation of image processing tools". Dans : *Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics*. WIMS '12. Craiova, Romania : ACM, 2012, 29 :1–29 :12. ISBN : 978-1-4503-0915-8. DOI : 10.1145/2254129.2254165. URL : http://doi.acm.org/10.1145/2254129.2254165.

[202] Zhiming WANG et al. "WS-BioZard : A Wizard for Composing Bioinformatics Web Services". Dans : *Proceedings of the 2008 IEEE Congress on Services - Part I*. SERVICES '08. Washington, DC, USA : IEEE Computer Society, 2008, p. 437–444. ISBN : 978-0-7695-3286-8. DOI : 10.1109/SERVICES-1.2008.69. URL : http://dx.doi.org/10.1109/SERVICES-1.2008.69.

[203] Peter WEGNER. "Interoperability". Dans : *ACM Comput. Surv.* 28.1 (mar. 1996), p. 285–287. ISSN : 0360-0300. DOI : 10.1145/234313.234424. URL : http://doi.acm.org/10.1145/234313.234424.

[204] Li WEIHUA et Li SHIXIAN. "Improve the semantic interoperability of information". Dans : *Proceedings. 2004 2nd International IEEE Conference*. IEEE Computer Society, 2004, p. 591–594. ISBN : 0-7803-8278-1. DOI : 10.1109/IS.2004.1344818.

[205] Gabriele Weiler, Arnd Poetzsch-Heffter et Stephan Kiefer. "Consistency Checking for Workflows with an Ontology-Based Data Perspective". Dans : *Proceedings of the 20th International Conference on Database and Expert Systems Applications*. DEXA '09. Linz, Austria : Springer-Verlag, 2009, p. 98–113. ISBN : 978-3-642-03572-2. DOI : 10.1007/978-3-642-03573-9_8. URL : http://dx.doi.org/10.1007/978-3-642-03573-9_8.

[206] Timo Weithöner et al. "Real-World Reasoning with OWL". Dans : *Proceedings of the 4th European conference on The Semantic Web : Research and Applications*. ESWC '07. Innsbruck, Austria : Springer-Verlag, 2007, p. 296–310. ISBN : 978-3-540-72666-1. DOI : 10.1007/978-3-540-72667-8_22. URL : http://dx.doi.org/10.1007/978-3-540-72667-8_22.

[207] W Wei, P. M. Barnaghi et A Bargiela. *The Anatomy and Design of A Semantic Search Engine*. Rap. tech. East Lansing, Michigan : Department of Computer Science, School of Computer Science, University of Nottingham Malaysia Campus, 2007.

[208] Gio Wiederhold. "Mediators in the Architecture of Future Information Systems". Dans : *Computer* 25.3 (mar. 1992), p. 38–49. ISSN : 0018-9162. DOI : 10.1109/2.121508. URL : http://dx.doi.org/10.1109/2.121508.

[209] Wikipedia. "Ontology". Dans : (jan. 2012). URL : http://fr.wikipedia.org/wiki/Ontologie.

[210] Wikipedia. "Taxonomy". Dans : (jan. 2012). URL : http://en.wikipedia.org/wiki/Taxonomy.

[211] Mark D. Wilkinson et Matthew Links. "BioMOBY : An Open Source Biological Web Services Proposal". Dans : *Briefings in Bioinformatics* 3.4 (2002), p. 331–341.

[212] K. Wolstencroft et al. "The myGrid ontology : bioinformatics service discovery". Dans : *Int. J. Bioinformatics Res. Appl.* 3.3 (sept. 2007), p. 303–325. ISSN : 1744-5485. DOI : 10.1504/IJBRA.2007.015005. URL : http://dx.doi.org/10.1504/IJBRA.2007.015005.

[213] Yujie Yao et Haopeng Chen. "A Rule-Based Web Service Composition Approach". Dans : *Proceedings of the 2010 Sixth International Conference on Autonomic and Autonomous Systems*. ICAS '10. Washington, DC, USA : IEEE Computer Society, 2010, p. 150–155. ISBN : 978-0-7695-3970-6. DOI : 10.1109/ICAS.2010.29. URL : http://dx.doi.org/10.1109/ICAS.2010.29.

[214] Ehtesham Zahoor, Olivier Perrin et Claude Godart. "Rule-Based Semi Automatic Web Services Composition". Dans : *Proceedings of the 2009 Congress on Services - I*. SERVICES '09. Washington, DC, USA : IEEE Computer Society, 2009, p. 805–812. ISBN : 978-0-7695-3708-5. DOI : 10.1109/SERVICES-I.2009.77. URL : http://dx.doi.org/10.1109/SERVICES-I.2009.77.

[215] Xinyu ZHANG et Nian Long LUO. "A Web Service System with Workflow Modeling and Scheduling". Dans : *Proceedings of the 2009 Sixth International Conference on Information Technology : New Generations*. ITNG '09. Washington, DC, USA : IEEE Computer Society, 2009, p. 1625–1626. ISBN : 978-0-7695-3596-8. DOI : 10.1109/ITNG.2009.69. URL : http://dx.doi.org/10.1109/ITNG.2009.69.

[216] Xia ZHAO et al. "SOA-based digital library services and composition in biomedical applications". Dans : *Comput. Methods Prog. Biomed.* 106.3 (juin 2012), p. 219–233. ISSN : 0169-2607. DOI : 10.1016/j.cmpb.2010.08.009. URL : http://dx.doi.org/10.1016/j.cmpb.2010.08.009.

**Résumé :**

Le domaine de la recherche en neuroimagerie nécessite de pouvoir partager, réutiliser et comparer les outils de traitement d'images des différents laboratoires. Cependant la tâche de partage de traitement sous forme de services et leur composition sous forme de workflow reste une tâche difficile et trop souvent complexe. Ceci est dû dans la plupart des cas à l'hétérogénéité des services et des plateformes qui diffèrent au niveau de leurs conceptions et de leurs implémentations.

Nous travaillons dans le cadre du projet NeuroLOG, une initiative cherchant à construire un système fédéré pour le partage de données et d'outils de traitement dans le domaine de la neuroimagerie. Il adopte une approche ontologique pour assurer la médiation et le partage de ressources entre les différents collaborateurs. Notre travail de thèse vise à compléter la médiation pour assurer le partage et la composition des outils de traitement d'images et à fournir aux utilisateurs spécialistes et non-spécialistes du domaine de la neuroimagerie une plateforme de composition de service ergonomique et facile à utiliser.

Nous utilisons pour cela les techniques du web sémantique afin de remédier aux différents problèmes d'interopérabilité et de cohérence de ressources utilisées et produites.

La première solution proposée se fonde sur une extension de la plateforme OWL-S. Elle a été adaptée aux différents services web de la plateforme de neuroimagerie. On a déduit que finalement les outils qui ne possèdent pas le format de services web et une description conforme au standard WSDL ne peuvent pas être enchaînés sous forme de workflow. A partir de là, nous avons proposé une autre approche pour effectuer la composition de services de traitement d'images. Elle se se fonde sur un nouveau modèle ontologique de composition de services qui répond aux exigences de la neuroimagerie, qui s'articule bien avec l'ontologie de domaine OntoNeuroLOG et qui pourra remédier aux différents problèmes rencontrés lors de l'élaboration de la première approche.

Ce travail a permit de remédier à la fois aux problèmes d'hétérogénéité des descripteurs des services et à l'interopérabilité des services selon les contraintes de la neuroimagerie au sein de la plateforme NeuroLOG.

**Mots clés :** Architecture Orientée Service, Web sémantique, Composition des services Web, Interopérabilité, Ontologies, Vérification sémantique.

**Abstract:**

The field of neuroimaging research requires the ability to share, reuse and compare image processing tools coming from different laboratories. However, sharing treatment as services and composing them as workflows, is usually difficult and a complex task. This is due in most cases to the heterogeneity of services and platforms with regards to their conception and their implementation.

We work within the NeuroLOG project, which aims at developing a middleware to federate data repositories and to facilitate the sharing and reuse of processing tools to analyze the shared images. It adopts an ontological approach for data and tools mediation and for sharing resources. This work aims to provide tools mediation to enhance the sharing and composition of image processing tools and provide non-specialist and expert users of neuroimaging field with an ergonomic and easy to use composition platform.

We have chosen to use the Semantic Web techniques to address the various problems of resource interoperability and consistency.

The first proposed solution is based on an extension of the OWL-S framework. It has been adapted to the various web services of our neuroimaging platform. We finally concluded that services that haven't the WSDL standard as descriptor could not be chained as workflow. So, we have proposed a new approach to compose image processing tools. It is based on a new ontological model for service composition that meets the requirements of the neuroimaging domain and the constraints of our domain ontology OntoNeuroLOG and addresses the various problems encountered in the development of the first approach.

This work led to solve the two major problems in the composition of services; the heterogeneity of services descriptors and the interoperability of services according to the constraints within the NeuroLOG platform.

**Keywords:** Service-Oriented-Architecture, Semantic Web, Web Services Composition, Interoperability, Ontologies, Semantic Verification and Validation.