كلية العلوم و التقنيات فاس
ⵜⴰⵖⵡⵍⴰⵏⵜ ⵏ ⵜⵓⵙⵙⴰⵏⵉ ⴰ ⵜⵉⵙⵙⵉⵜⵉ
Faculté des Sciences et Techniques de Fès

جامعة سيدي محمد بن عبد الله
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵙⵉⴷⵉ ⵎⵓⵃⴰⵎⴻⴷ ⴱⵏ ⵄⴱⴷⵍⵍⴰⵀ
Université Sidi Mohamed Ben Abdellah

Centre d'Etudes Doctorales : Sciences et Techniques de l'Ingénieur

N° d'ordre 46 /2017

THESE DE DOCTORAT

Présentée par

Amina ADADI

Spécialité : **Informatique**

Sujet de la thèse

Élaboration d'une approche de composition dynamique et automatique des services Web: Application au domaine du gouvernement électronique

Thèse présentée et soutenue le 20 Décembre 2017 devant le jury composé de :

Nom Prénom	Titre	Etablissement	
M. Mostafa HARTI	PES	Faculté des Sciences Dhar Mahraz de Fès	Président
M. Ismail ESSAOUDI	PES	Faculté des Sciences de Meknès	Rapporteur
M. Said NAJAH	PH	Faculté des Sciences et Techniques de Fès	Rapporteur
M. Ahmed ZELLOU	PH	ENSIAS de Rabat	Rapporteur
Mme. Nour El Houda CHAOUI	PES	ENSA de Fès	Examinateur
M. Adil KENZI	PH	ENSA de Fès	Examinateur
M. Mohammed BERRADA	PH	ENSA de Fès	Directeur de thèse
M. Driss CHENOUNI	PES	Ecole Normale Supérieure de Fès	Directeur de thèse

Laboratoire d'accueil : Laboratoire d'informatique et de physique interdisciplinaire

Etablissement : École Normale Supérieure de Fès

Centre d'Etudes Doctorales : Sciences et Techniques de l'Ingénieur

Résumé de la thèse

La conjonction de la maturité des architectures logicielles à composants distribués, la rapidité de pénétration des technologies du Web et la forte pression économique, ont conduit à une irrésistible prolifération des services Web. L'une des facultés distinctives de ces entités autonomes distribués sur le Web, est leur capacité d'offrir des services composites à valeur ajoutée construits à partir de ces composants réutilisables, cette opération est connue sous le nom de « la composition de services Web ».

Dans ce travail nous nous intéressons à la composition dynamique et automatique de services Web. Nous proposons dans ce sens un cadre conceptuel et des outils techniques pour réaliser une composition particulièrement intelligente.

L'originalité de la solution proposée réside dans l'utilisation d'outils mixtes allant des systèmes multi-agents jusqu'au paradigme de l'Autonomic Computing en passant par des techniques de planification et des modèles sémantiques. L'architecture organisant ces éléments est soutenue par un ensemble de techniques de composition conçues en vue de permettre l'aspect dynamique et l'aspect automatique souhaités.

Les idées avancées dans cette thèse sont appliquées au domaine du gouvernement électronique et elles sont concrétisées par une preuve de concept illustrée par un scénario provenant de ce domaine.

Mots clés : Service Web, composition de services, systèmes multi-agents, techniques de planification, Web sémantique, Autonomic Computing, gouvernement électronique

À la mémoire de ma grand-mère Lalla Saâdia,
Comme une chandelle, tes bénédictions ont éclairé le début de ce chemin
dont je te dédie l'aboutissement.

Que Dieu, le miséricordieux, t'accueille dans son éternel paradis.

À maman et papa, je vous dois tout.

Remerciements

Je remercie Dieu, tout Puissant, de m'avoir permis de mener à terme cette thèse.

Ce travail a été réalisé au sein du Laboratoire d'Informatique et de Physique Interdisciplinaire (LIPI), sous la direction du Monsieur le Professeur Mohammed BERRADA et Monsieur le Professeur Driss CHENOUNI envers qui, je tiens à exprimer mes remerciements et ma gratitude.
Mes vifs remerciements sont destinés à Monsieur le Professeur M.BERRADA, professeur à l'École Nationale des Sciences Appliquées de Fès, pour sa confiance, sa disponibilité et son orientation. Ses qualités pédagogiques, scientifiques et humaines ont joué un rôle moteur dans l'accomplissement de cette œuvre.
Je tiens également à adresser mes remerciements à Monsieur le Professeur D.CHENOUNI, directeur de l'École Normale Supérieure de Fès, pour son appui généreux et ses conseils prodigués durant tout ce travail doctoral.

Je tiens à remercier chaleureusement les membres du jury d'avoir accepté de juger mes travaux.
Je remercie Monsieur le Professeur M.HARTI pour m'avoir fait l'honneur d'accepter de présider le jury de ma soutenance de thèse.
Mes remerciements vont également aux rapporteurs de cette thèse, Monsieur le Professeur I. ESSAOUDI, Monsieur le Professeur S. NAJAH et Monsieur le Professeur A. ZELLOU, pour l'intérêt qu'ils ont porté à mon travail.
J'adresse également mes remerciements à Madame le Professeur N.H. CHAOUI et Monsieur le Professeur A. KENZI d'avoir accepté d'être examinateurs de ce travail.

Je tiens aussi à mentionner le plaisir que j'ai eu de faire partie des étudiants doctorants du Centre d'Etudes Doctorales de la Faculté des Sciences et Techniques de Fès, je remercie tous les membres pour le suivi pédagogique et la qualité des formations dispensées.

A toute ma famille, mes amis et mes collègues de recherche et de travail, je leur adresse globalement un grand remerciement et je les prie d'accueillir ici mes sentiments de gratitude les plus profonds pour leur soutien et encouragement.

Enfin, parce que les mots les plus simples sont ceux les plus sincères, je dis tout simplement à mes parents, mon frère et ma sœur, MERCI pour votre amour, sacrifice et soutien inconditionnels et indéfectibles.

Résumé

La conjonction de la maturité des architectures logicielles à composants distribués, la rapidité de pénétration des technologies du Web et la forte pression économique, ont conduit à une irrésistible prolifération des services Web. L'une des facultés distinctives de ces entités autonomes distribués sur le Web, est leur capacité d'offrir des services composites à valeur ajoutée construits à partir de ces composants réutilisables, cette opération est connue sous le nom de « la composition de services Web ».

Dans ce travail nous nous intéressons à la composition dynamique et automatique de services Web. Nous proposons dans ce sens un cadre conceptuel et des outils techniques pour réaliser une composition particulièrement intelligente.

L'originalité de la solution proposée réside dans l'utilisation d'outils mixtes allant des systèmes multi-agents jusqu'au paradigme de l'Autonomic Computing en passant par des techniques de planification et des modèles sémantiques. L'architecture organisant ces éléments est soutenue par un ensemble de techniques de composition conçues en vue de permettre l'aspect dynamique et l'aspect automatique souhaités.

Les idées avancées dans cette thèse sont appliquées au domaine du gouvernement électronique et elles sont concrétisées par une preuve de concept illustrée par un scénario provenant de ce domaine.

Mots-clés: Service Web, composition de services, systèmes multi-agents, techniques de planification, Web sémantique, Autonomic Computing, gouvernement électronique

ملخص

أدى كل من نضج بنيان البرمجيات ذات المكونات الموزعة وسرعة اختراق تقنيات الويب وكذا الضغط الاقتصادي القوي إلى انتشارهائل لخدمات الويب. من مميزات هذه الخدمات المستقلة والموزعة على الويب، قدرتها على توفيرخدمات مركبة ذات قيمة مضافة انطلاقا من هذه المكونات القابلة لإعادة الاستعمال. هذه العملية تسمى "تركيب خدمات الويب".

في هذا العمل نهتم بالتركيب الديناميكي والتلقائي (أُوتُوماتِيكِيّ) لخدمات الويب. نقترح في هذا الإطار، التصميم والأدوات التقنية اللازمة لإنجاز تركيب ذكي.

الحل المقترح يتميز باستخدام مزيج من أدوات متنوعة: نظام الوكلاء المتعددة، تقنيات التخطيط، النماذج الدلالية، والحوسبة التلقائية الذاتية.

تدعم البنية المنظمة لهذه العناصر مجموعة من تقنيات التركيب التي أعدت من أجل الوصول إلى المستوى المرغوب من التلقائية والديناميكية.

الأفكار المقدمة في هذه الأطروحة تم تطبيقها في مجال الحكومة الإلكترونية، وتم تجسيد هذه الأفكار من خلال نموذج بدئي مبني على سيناريو ينتمي إلى هذا المجال.

المفاتيح: خدمات الويب، تركيب الخدمات، نظام الوكلاء المتعددة، تقنيات التخطيط، النماذج الدلالية، الحوسبة التلقائية الذاتية، الحكومة الإلكترونية.

Abstract

The conjunction of the maturity of software architectures with distributed components, the fast penetration of Web technologies and ultimately the high economic pressure, have led to an irresistible proliferation of Web services. One of the distinctive abilities of this autonomous entities distributed on the Web is its capacity of providing a value added composite services built based on this reusable components. This operation is known as “Web service composition”

In this work, we are interested in dynamic and autonomic web service composition. We propose a conceptual framework and a set of technical tools to carry a composition particularly intelligent.

The originality of the proposed solution lies in the use of a mix of diverse tools uniting multi-agents system, autonomic computing paradigm, planning techniques and semantic models. The architecture organizing these elements is supported by a set of composition techniques designed to allow the desired dynamic and automatic aspects.

The ideas set out in this thesis are applied to electronic government domain and are materialized by a proof of concept illustrated by a scenario coming from this domain.

Keywords: Web service, service composition, multi-agents systems, planning, semantic Web, Autonomic Computing, electronic governement

Table des matières

Liste des figures

Liste des tableaux

Liste des publications et des communications

Publications:

- A. Adadi, M.Berrada, D.Chenouni, B.Bounabat, A semantic Web service composition for e-Government services, Journal of Theoretical and Applied Information Technology (JATIT), Janvier 2015.
- A. Adadi, M.Berrada, D.Chenouni, A Multi-Agent Planning Architecture for Semantic Web Service Composition, International Review on Computers and Software (IRECOS), Février 2014.

Communications:

- A. Adadi, M.Berrada, D.Chenouni, Towards dynamic and automated composition of E-government Web services, 1ère journée Doctorale et Post Doctorale des Systèmes d'information et télécommunication (JDPD SIT'2016), Décembre 2016;
- A. Adadi, M.Berrada, D.Chenouni, A phased two-stage approach for dynamic Web service composition, The international conference on Information Technology for Organisation Development (IT4OD-2016), Avril 2016;
- A. Adadi, M.Berrada, D.Chenouni, B.Bounabat, Ontology based composition of e-Government services using AI Planning, the 10th International Conference on Inelligent Systems : Theories and Applications (SITA'2015), Octobre 2015;
- A. Adadi, M.Berrada, D.Chenouni, B.Bounabat, Towards a multi-agent planning based architecture for semantic Web Service Composition, The First International Workshop on Wireless Technologies and Distributed Systems (WITS 2014), Avril 2014;
- A. Adadi, M.Berrada, D.Chenouni, B.Bounabat, A multi-agent based architecture for automated dynamic Web Service Composition, Fifth workshop on information technologies and communication (WOTIC'2013), Décembre 2013.

Introduction Générale

Rien ne se crée, rien ne se perd, tout se transforme.
Antoine Lavoisier

Contexte général

Cela ne fait plus aucun doute, le monde est à l'heure du numérique. Ce qu'on a coutume d'appeler les TIC (Technologies de l'Information et de la Communication) font désormais partie intégrante de notre quotidien. Qu'il s'agisse d'individus, d'entreprises ou d'organisations publiques, la numérisation a des effets substantiels sur tout le corps social et économique au point d'engager une véritable mutation de l'ensemble des méthodes de travail de ceux-ci. Nous nous intéressons à l'un des traits caractérisant cette mutation numérique, particulièrement spectaculaire dès les années 2000: l'émergence des Services Web.
Entre évolution des techniques et révolution des pratiques, les services Web sont considérés comme une conséquence directe du renforcement de la compétitivité entre les entreprises pour assurer l'interopérabilité, l'une des clés de succès de développement et d'intégration des grands systèmes d'information. La notion de service Web désigne essentiellement une application mise à disposition sur Internet par un fournisseur de service, et accessible par des clients à travers des protocoles Internet standards. Les particularités des services Web résident dans le fait qu'ils offrent un modèle de composants à couplage faible en utilisant la technologie Internet comme infrastructure pour la communication.
Dès leur apparition, les services Web furent un succès immédiat grâce au haut degré d'interopérabilité fourni par les standards. Depuis, on témoigne une augmentation continue du nombre des services offerts sur le Web. L'acculturation de l'orientation client, qui fait que la satisfaction du client soit en grande partie dépendante du niveau de personnalisation et d'adaptation de l'offre selon le souhait du demandeur, n'a fait qu'accentuer la prolifération des services Web. Des questions sur l'efficacité et le sort d'un Web encombré par des services Web commencent alors à être posées.
C'est ainsi que la réflexion sur l'optimisation des services Web commence à prendre place. Selon cette réflexion plusieurs services peuvent interagir et échanger dynamiquement des informations afin d'accomplir un but complexe non concevable par un seul service Web. Cette agrégation des compétences pour réaliser un but commun est connue sous le nom de « composition des services Web ».
Entreprendre une composition à l'ère numérique exige, toutefois, de revenir sur quelques repères scientifiques pour bien appréhender le principe et la finalité de cette opération.
La composition comme un processus productif de la valeur est un principe fondamental qui se manifeste sous plusieurs formes dans la nature, un exemple simple mais significatif est celui de couleurs; il existe des milliers des couleurs dans la nature qui ne sont à la base que le résultat de la synthèse additive (ou soustractive) des trois couleurs primaires (rouge, bleu et jaune). Toutefois la métaphore la plus éloquente pour décrire le concept de composition étudié reste, sans doute, celle des réactions chimiques. Selon cette métaphore, des molécules flottent dans une solution, et réagissent selon des règles de réactions produisant de nouvelles

molécules. Ces réactions ont lieu de façon implicitement parallèle, dans un ordre indéterminé jusqu'à l'état d'inertie, dans lequel plus aucune réaction n'est possible. Formellement, les molécules forment un multi-ensemble réécrit par les règles représentant les réactions. Ce modèle représente une parfaite source d'inspiration pour une approche de composition dotée de qualité d'adaptation et d'autonomie comme on espère concevoir.
Montant vers l'abstraction, composition, synthèse, réaction, transformation, toutes ces notions ne sont finalement que l'expression d'une même volonté : créer en réutilisant un existant. Dans le contexte de services Web, l'objectif ultime de la composition est de transformer le Web en un dispositif distribué de calcul où les services peuvent interagir de manière intelligente en étant capables de se découvrir automatiquement, de négocier entre eux et de se composer dynamiquement en des services plus complexes. En d'autres termes, l'idée poursuivie avec les services Web, est de mieux exploiter les technologies d'Internet en substituant, autant que possible, les humains qui réalisent actuellement un certain nombre de tâches, par des machines en vue de permettre une composition automatique et dynamique des services sur le Web. L'automatisation et la dynamique sont donc des concepts clés qui doivent être pris en considération dans le processus de conception et de mise en œuvre de la composition de services Web.
Même si la composition des services Web connaît une phase très active, surtout avec l'évolution impressionnante de l'informatique en nuage (cloud computing), des services d'analyse des données (big datas analytics) et de l'internet des objets (Internet of things) qui contribuent encore plus à la profilération massives des services Web. Néanmoins, il n'existe pas encore un modèle standard pour composer automatiquement et dynamiquement les services Web. Loin de devoir se traduire par des solutions de repli, une profonde incertitude domine encore ce domaine de composition et contraint l'ensemble des acteurs économiques et scientifiques à placer au cœur de leur activité : la recherche, l'innovation et le laboratoire.

On l'aura compris, l'ambition de ce travail de recherche est de proposer un cadre conceptuel et un outil technique pour la composition automatique et dynamique de services Web.

Intérêts et motivation

D'emblée, nous admettons le fait que nous nous sommes ni les premiers, ni les seuls à traiter la problématique de la composition dynamique et automatique des services Web. La valeur du présent travail ne réside pas dans l'originalité du problème, mais plutôt dans l'infaillibilité de l'approche de sa résolution. Son défi étant d'apporter du nouveau dans un sujet qui n'est pas nouveau. La littérature des approches de composition des services Web existantes montre que cette problématique a fait, de fait, l'objet de plusieurs travaux aussi bien dans la communauté de recherche qu'au niveau industriel.
La plupart des travaux s'entendent sur la nécessité de la prise en charge de l'aspect automatique et dynamique dans le processus de composition. En effet ces deux aspects sont essentiels pour intégrer les facteurs suivants :

- Passage à l'échelle : il faut être capable de traiter un nombre important de services Web (annuaire de services au niveau mondial) ;
- Forte réactivité dans un environnement hautement dynamique ;

- Réduction des coûts de développement et de maintenance des services Web;
- Forte adaptabilité : il est vraisemblable que vu, l'enjeu que représente leur réussite et de par leur orientation métier, les services Web créés doivent être flexibles pour s'adapter aux changements. En effet, les processus d'affaires subissent des changements continus afin de répondre aux exigences des entreprises qui veulent satisfaire les besoins des clients en leur offrant un service personnalisé et de qualité. Ce qui permet de les fidéliser.

Aussi prometteuse et bénéfique qu'elle soit, la composition dynamique et automatique des services Web souffre d'un manque de méthodes et d'outils. Puisque la plupart des travaux de composition issus du monde industriel possèdent une orientation statique et évitent, par conséquent, le problème de l'automatisation. La majorité des approches scientifiques, quant à elles, négligent le caractère holistique de la composition. Selon ces travaux, résoudre le problème de composition revient souvent à manipuler des services Web sémantiques ou à mettre en place un mécanisme d'intelligence artificielle, bref les approches scientifiques se limitent souvent à projeter la problématique de composition sur une seule autre thématique de recherche, ce qui risque de réduire la portée du processus de composition incluant différentes facettes (la sélection, l'exécution, la découverte, etc.) et de conduire à des solutions incomplètes de la problématique. Aussi, la plupart des résultats des approches scientifiques demeurent à un niveau abstrait et ils sont, par conséquent, difficilement applicables dans des conditions réelles.

Il ressort de ce qui précède, l'opportunité et les défis relevés par le traitement de la problématique de la composition dynamique et automatique des services Web.

En réalité, l'idée et la motivation derrière le choix de cette problématique est surtout son caractère composé et complexe, c'est en quelque sorte l'arbre qui cache la forêt. En effet, la composition de services Web est un processus complexe qui fait intervenir plusieurs activités telles que la découverte, la composabilité, l'ordonnancement, la sélection ou encore l'exécution des services Web. La résolution efficace du problème de la composition dépend fortement de la façon dont on gère chacune de ces activités. Ce thème de recherche a donc ceci de plaisant, qu'il suscite l'exploration de plusieurs autres disciplines à la recherche d'instruments pour fiabiliser la résolution de chaque fragment de la problématique. C'est essentiellement cette richesse qui est à l'origine de notre enthousiasme pour ce travail.

Contributions scientifiques

La thématique du travail de recherche que nous allons entreprendre gravite autour de l'idée suivante: *comment construire un service Web à valeur ajoutée à travers l'exploitation des services Web existants et ce, de la façon qui garantit le niveau d'automatisation et de dynamique le plus élevé possible*.
Deux principes guident notre démarche :

(1) Notre approche doit être construite au-dessus des approches existantes, les solutions qui ont démontré leur efficacité doivent être adoptées, adaptées et étendues en vue de consolider notre modèle;

(2) L'approche à proposer doit couvrir l'intégralité du processus de composition de bout en bout depuis l'expression du besoin par le client jusqu'à sa satisfaction.

Ainsi, les thématiques traditionnelles utilisées pour la composition vont être revisitées dans une perspective d'intégration avec d'autres thématiques qui vont être explorées afin d'étudier leur potentiel à contribuer à la résolution intégrale de la problématique traitée. Le résultat final formera alors ce qui pourrait considérer un patchwork de systèmes hétéroclites issus de différents horizons scientifiques dont le travail d'harmonisation à entretenir, leur donnera la faculté de pouvoir œuvrer en synergie.

Dans ce sens, les principales contributions de ce travail sont :

- Une étude approfondie des disciplines informatiques susceptibles de contribuer à la conception d'une approche de composition de services Web avec les qualités souhaitées;
- Un mécanisme de traduction en spécifications inter système des besoins de l'utilisateur ;
- Un algorithme de filtrage automatique des services qui peuvent potentiellement participer à la production d'un service Web composite;
- Un modèle de construction automatique des schémas de composition des services composants;
- Une stratégie de sélection des instances des services qui maximise la qualité du service composite produit;
- Un mécanisme d'exécution adaptable et dynamique du service composite;
- L'ensemble des techniques de composition précitées sont organisées dans une architecture pragmatique dont les composants et leurs interactions favorisent la synergie et garantissent la transparence du processus de la composition vis-à-vis de l'utilisateur;
- De par son caractère multidimensionnel, l'évaluation du résultat sur le plan fonctionnel et technique forme elle aussi un atout important de ce travail.

Organisation du document

Outre l'introduction, ce manuscrit comprend trois autres parties:

La première partie est consacrée à une étude de l'état de l'art autour de la composition de services Web et les disciplines y afférentes, elle est organisée en deux chapitres :

- le premier chapitre propose une revue littéraire des services Web et des directions de recherche actives dans ce domaine, notamment la composition de services Web,
- le deuxième chapitre poursuit l'état de l'art en présentant les quatre disciplines qui constituent le fond théorique de notre approche, il s'agit (i) du Web sémantique

comme une vision vers moins d'interventions humaines, (ii) des systèmes multi-agents comme un cadre architectural favorisant le caractère dynamique et décentralisé, (iii) des algorithmes de planification comme une méthode de raisonnement automatique et (iv) de l'informatique autonomique comme un générateur de comportement autonomique.

Puis, au niveau de la deuxième partie, nous réservons trois chapitres pour aborder successivement nos contributions en termes de composition dynamique et automatique de services Web :

- le troisième chapitre expose l'architecture proposée pour la composition dynamique et automatique de services Web. Meublée par des agents intelligents, des ontologies et une interface de communication, cette architecture sera détaillée en termes d'interactions entre ses composants et en termes de style architecturale sur lequel elle se base : il s'agit de la composition par phases à deux niveaux : abstrait et concret.
- le quatrième chapitre présente les techniques de composition conçues pour accomplir la composition abstraite. Il s'agit de (i) *l'Analyse des spécifications* : le mécanisme par lequel une demande exprimée par un utilisateur est transformée en une requête compréhensible par le système. Du (ii) *Mécanisme de découverte* : un processus de pré-planification qui permet de sélectionner les types de services qui ont le potentiel de participer à la composition. Et, du (iii) *Modèle de planification* : une sous-architecture et un ensemble de mécanismes utilisant les algorithmes de planification pour construire un plan de composition abstrait.
- le cinquième chapitre présente les techniques de composition conçues pour accomplir la composition concrète. Il s'agit de (i) la *Stratégie d'évaluation* : le mécanisme responsable de la sélection des instances adéquates des services Web pour répondre efficacement à la requête de l'utilisateur. Et, de (ii) *l'Exécution autonomique* : la technique qui gère d'une façon autonomique les comportements anormaux qui peuvent survenir en temps réel lors de l'exécution du service composite produit.

Les contributions présentées dans la deuxième partie, sont suivies d'une évaluation en trois niveaux :

- le sixième chapitre évalue l'approche sur le plan fonctionnel, il s'agit de vérifier la capacité de l'approche proposée à résoudre des problématiques réelles. Nous proposons alors une mise en application de notre modèle de composition dans un domaine assez particulier : le gouvernement électronique, tout en soulignant les retombées bénéfiques de l'adoption de notre approche pour faire face aux principaux défis techniques de ce domaine.
- le septième chapitre évalue l'approche sur le plan technique. Nous proposons alors au niveau de ce chapitre un prototype qui implémente les concepts et les techniques clés constituant notre approche.
- finalement, pour compléter la démarche d'évaluation, le huitième chapitre propose une évaluation dite qualitative qui consiste d'abord à s'assurer formellement que le résultat de nos contributions est conforme aux spécifications convenues au début de la thèse

(cf. chapitre 1), puis à positionner l'approche de composition proposée par rapports aux travaux connexes. L'objectif est d'identifier les points d'intérêt de notre approche et d'envisager les pistes d'amélioration.

Ces trois derniers chapitres sont regroupés dans une troisième partie nommée « Application et évaluation ».

La présentation de notre conclusion ainsi que des perspectives de nos travaux closent ce manuscrit.

En complément, des explications concernant les algorithmes de planification (annexe A), l'état des lieux de l'e-gouvernement (annexe B) et les outils de mise en place du prototype (annexe C), sont annexées à ce manuscrit.

Première partie

Etat de l'art

Chapitre 1

Les services Web et leur composition

Avant de penser, il faut étudier.
Seuls les philosophes pensent avant d'étudier.
De Gaston Bachelard

1.1. Introduction

L'interopérabilité entre systèmes a toujours constitué un frein aux échanges électroniques. A moins d'adhérer à une plateforme fédératrice spécifique, le dialogue entre applications hétérogènes nécessitait de longs et onéreux développements, qu'il fallait revoir et adapter à chaque fois qu'un nouveau partenaire se présente. L'avènement des services Web a marqué un tournant décisif dans l'approche du problème. Plutôt que de chercher à uniformiser les systèmes, les services Web s'adaptent à l'hétérogénéité de l'existant et favorisent une meilleure réactivité. L'infrastructure de base de services Web, reposant sur des technologies standards, permet d'intégrer des systèmes hétérogènes complexes et fortement distribués. Cependant, cette infrastructure ne permet pas encore aux services Web de tenir leur promesse d'une gestion largement automatisée et laisse plusieurs questions ouvertes. La composition des services Web, l'une de ces questions, se présente comme le résultat d'une évolution naturelle de l'architecture et des technologies sous-jacentes aux services Web. Son objectif est de construire de nouveaux services à valeur ajoutée à partir de services de base existants sur le Web.

Nous proposons dans ce chapitre de faire le point sur le sujet de services Web. Pour ce faire, nous présentons en premier temps la notion de service Web et les concepts associés, ensuite nous décrivons l'architecture orientée services et ses différentes caractéristiques, enfin nous détaillons les principales problématiques de recherche relatives au domaine de services Web.

Dans la deuxième partie de ce chapitre, nous exposons le problème de la composition des services Web et les différentes approches proposées dans la littérature pour le résoudre.

1.2. Services Web

1.2.1. Services Web : historique, définition et principes

Avant de nous pencher sur les différents concepts et technologies derrières les services Web, nous allons nous arrêter d'abord sur l'origine et la définition de ce modèle considéré, aujourd'hui, en vogue dans le domaine des nouvelles technologies.

1.2.1.1. Cadre historique

Les services Web (SW) ont émergé au début des années 2000 dans le contexte de la mise en œuvre d'Architectures Orientées Services, aussi récents que les services Web puissent paraitre, leur avènement est l'aboutissement d'une longue histoire qui prend ses racines dans les premiers temps de l'informatique. L'histoire commence en 1975 lorsque l'informatique souffrait encore de peu de standardisation, les innovations grandissantes qu'a vécues l'industrie de l'informatique à partir de cette époque ont rendu les environnements technologiques plus complexes et plus hétérogènes. Par conséquent, la question de standardisation est devenue plus critique du point de vue technique et économique. Les constructeurs et les éditeurs se rendent compte alors de la nécessité d'uniformiser les échanges de données et firent le vœu pieux de « l'interopérabilité » afin de standardiser la communication entre les applications au travers d'un réseau. C'est dans ce contexte qu'a vu le jour l'EDI (Electronic Data Interchange) [Eistert et al., 1995], l'ancêtre des services Web. De nombreuses technologies à composants distribués ont alors émergé depuis (DCOM, Corda, Java RMI) avec toujours l'objectif de connecter des logiques métiers au travers d'un réseau, mais aucune de ces technologies n'a réellement réussi à s'imposer comme standard universel, car souvent un manque de simplicité, de standards ou encore de neutralité technologique lutte contre la diffusion et le succès de ces technologies de « middleware ».

L'arrivée du Web a tout bouleversé, en s'appuyant sur des modèles académiques (modèle OSI) et en reprenant les clés du succès des protocoles du plus bas niveaux déjà adoptés, le Web a permis aux acteurs de la technologie de faire un grand pas vers l'ouverture de leurs systèmes informatiques traduit par l'émergence du commerce électronique (e-commerce). L'arrivée ensuite de XML, un métalangage de balisage générique, et la naissance de XML-RPC, technologie standard d'échange entre applications, ont complété la chaîne de l'interopérabilité et ont permis de donner naissance aux services Web.

1.2.1.2. Définitions et caractéristiques

Dans la littérature, il existe de nombreuses définitions des services Web mises en avant par différents auteurs :

"A software identified by a URI, whose interfaces and binding are capable of being defined, described, and discovered by XML artifacts and supports direct interactions with other software applications using XML-based messages via Internet-based protocols"

– **W3C [W3C, 2002]**

"Loosely coupled software components that interact one another dynamically via standard internet technologies"

– **Gartner group [Gartner, 2002]**

« Services Web consist of a group of standards intended to make it possible for diverse systems to communicate, without requiring a particular type of middleware, programming language or even operating system"

– **IBM [IBM, 2002]**

"*Un service Web est un programme informatique permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués. Il s'agit donc d'un ensemble de fonctionnalités exposées sur internet ou sur un intranet, par et pour des applications ou machines, sans intervention humaine, et en temps réel* "

–**Wikipédia [Wikipédia, 2016]**

Bien que publiées à différents stades de l'évolution des SW, et par des auteurs de différentes orientations, une étude approfondie des points communs partagés par les différentes définitions et par les usages qui sont faits des SW, permet de dégager quelques principes fondamentaux liés à la notion de service Web :

Caractéristique	**Description**
Une application logicielle définie par un URI	Un URI est la façon d'identifier un point de contenu sur le Web, que ce soit un fichier texte, audio ou vidéo. L'URI le plus connu est l'adresse d'une page Web. Le SW est donc accessible en spécifiant son URI
Capacité d'interagir avec d'autres composantes logicielles via des éléments XML en utilisant des protocoles Internet	L'une des bases des SW est l'utilisation de protocoles standards de l'Internet tels que HTTP (Hypertext Transfer Protocol), SMTP (Simple Mail Transfer Protocol, le protocole du courriel électronique) et XML (Extensible Markup Language). Par ailleurs, un SW est une application autosuffisante dans le sens qu'elle effectue une seule tâche et que ses composantes décrivent ses propres entrées et sorties de telle sorte que d'autres logiciels qui invoquent le SW puissent interpréter ce qu'il fait, comment invoquer sa fonctionnalité et à quel résultat cet autre service peut s'attendre.
Composantes logicielles légèrement couplées à interaction dynamique	Par « légèrement couplées », on veut dire que le SW et le programme (le consommateur de service Web) qui l'invoque peuvent être modifiés indépendamment l'un de l'autre. Cela veut aussi dire que, contrairement à une composante logicielle qui serait fortement couplée, il n'est pas nécessaire de connaître la machine, le langage, le système d'exploitation ou tout autre détail, pour qu'une communication puisse avoir lieu. Cela offre une flexibilité qui permet aux entreprises d'éviter les coûts engendrés par l'intégration via des communications fortement couplées, telles que requises par l'EDI, par exemple. L'interaction dynamique signifie que le consommateur de services Web peut localiser et invoquer ce dernier au moment de l'exécution du programme sans avoir à programmer cette habileté à l'avance. Cela présuppose aussi que l'on peut

	modifier le SW sans avoir à modifier le logiciel de chacun des utilisateurs potentiels.
Interface asynchrone utilisant des technologies indépendantes des plateformes	Le Web est par nature asynchrone, dans l'espace Web il est possible que deux entités se branchent (via une page Web par exemple) sans pour autant avoir une connaissance préalable l'un de l'autre. Dans le contexte des SW, cela veut dire que n'importe quel client peut avoir accès à un SW publié par n'importe qui pour autant que l'information à propos du service (le schéma) soit disponible et que le client XML soit capable de générer des messages conformes au schéma. Il est à noter que les SW supportent tous le paradigme de la communication synchrone [Newcomer, 2006]
Composantes réutilisables	L'une des caractéristiques particulièrement intéressantes des SW, est qu'une fois une fonctionnalité est développée sous forme de service Web, elle peut être réutilisée et combinée à d'autres fonctionnalités afin de composer de nouveaux services

Tableau 1.1*:* Caractéristiques d'un service Web

Tous les éléments présentés dans le Tableau 1.1 sont utils pour décrire les SW et chacun d'eux exprime une facette de ce qu'il est maintenant convenu d'appeler « un service Web ». Nous allons alors retenir la combinaison suivante:

Un service Web est une application logicielle, légèrement couplée, à interaction dynamique, identifiée par un URI, pouvant interagir avec d'autres composantes logicielles et dont les interfaces et associations (binding) ont la capacité d'être publiées, localisées et invoquées via XML et l'utilisation des protocoles Internet communs. Les SW sont les bases permettant de construire des systèmes distribués et ouverts sur Internet, grâce à leur interface asynchrone utilisant des technologies indépendantes des plateformes et de leurs composantes réutilisables.

1.2.2. Architecture orientée services

L'exposition et l'utilisation des services se fait dans un contexte particulier qui définit clairement les interactions entre le service et ses utilisateurs. Il s'agit d'une infrastructure standard appelée architecture orientée services (SOA). La notion d'architecture orientée services n'est pas nouvelle, elle est considérée comme une évolution normale de l'architecture classique « client-serveur » en conjonction avec le domaine des systèmes répartis. Là encore nous retrouvons plusieurs définitions, nous proposons celle établie dans le modèle de référence OASIS [MacKenzie et al., 2006], considérée l'une des définitions du SOA les plus courantes:

"*Service Oriented Architecture is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a*

uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations" [MacKenzie et al., 2006].

Nous retiendrons donc que la SOA est un paradigme d'organisation et d'utilisation des capacités distribuées qui peuvent être sous le contrôle de différents domaines propriétaires. Elle offre un moyen unique de découvrir et d'interagir avec les capacités pour produire des effets mesurables en accord avec les préconditions et les espérances.

Le choix d'une architecture SOA entre dans la perspective de transformer le Web en une énorme plateforme de composants faiblement couplés et automatiquement intégrables. Ce style architectural vise trois objectifs importants: (i) identification des composants fonctionnels, (ii) définition des relations entre ces composants et (iii) établissement d'un ensemble de contraintes sur chaque composant de manière à garantir les propriétés globales de l'architecture.

Les interactions entre les SW impliquent trois participants (acteurs) [Kellert et al., 2003]: le fournisseur de services (prestataire), le client (utilisateur du service) et l'annuaire de services (Fig.1. 1) :

(1) Le fournisseur de services correspond au propriétaire du service (i.e. entité responsable du SW). D'un point de vue technique, il est constitué par la plateforme d'accueil du service.
(2) Le client correspond au demandeur du service. D'un point de vue technique, il est constitué par l'entité de recherche et d'invocation d'un service.
(3) L'annuaire des services représente l'entité logicielle qui joue le rôle de l'intermédiaire entre les clients et les fournisseurs de services, il correspond à un registre de descriptions de services offrant des facilités de publication de services pour les fournisseurs ainsi que des facilités de recherche pour les clients et le moyen de localiser leurs besoins en termes de services.

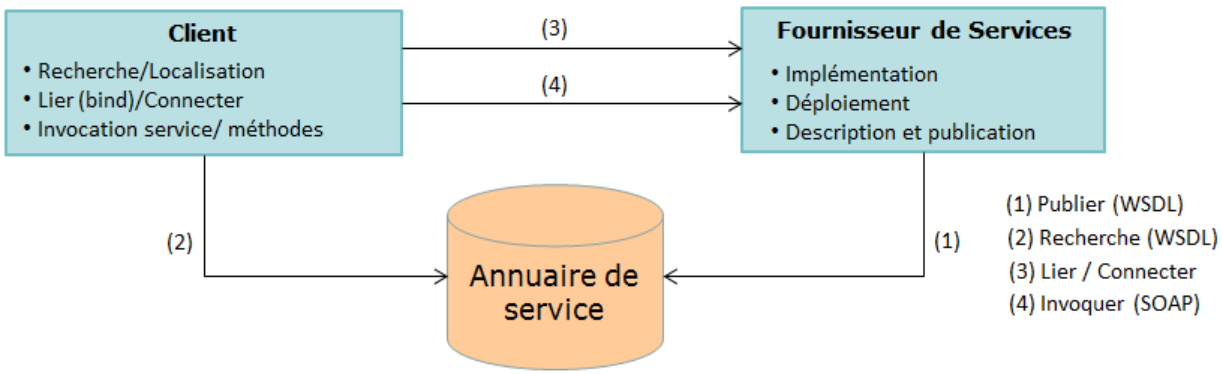


Fig.1. 1: Interactions entre les services Web [Kellert et al., 2003]

La dynamique de l'architecture se décompose ainsi : le fournisseur de services définit la description de son service et la publie dans un annuaire de service (dans des registres) en vue d'être localisé par des clients. Le client utilise les facilités de recherche disponibles au niveau

de l'annuaire pour retrouver et sélectionner un service donné. Il examine ensuite la description du service sélectionné (extrait de sa description disponible au niveau du registre) pour récupérer les informations nécessaires lui permettant de se connecter au fournisseur du service et d'interagir avec l'implémentation du service considéré (entreprend une interaction).

Le modèle de fonctionnement de l'architecture des services Web se base sur un cadre technologique qui constitue l'infrastructure de l'architecture par composants des services Web. Cette infrastructure offre les mécanismes nécessaires pour la réalisation des différentes étapes du cycle de vie d'un service Web.

1.2.3. Infrastructure des services Web

Dans sa première génération, le fonctionnement des services Web repose sur trois couches fondamentales présentées comme suit:

(1) **Invocation** : visant à établir la communication entre le client et le fournisseur en décrivant la structure des messages échangés.
(2) **Découverte**: permettant de localiser un service Web particulier dans un annuaire de services décrivant les fournisseurs ainsi que les services fournis.
(3) **Description**: décrivant les interfaces des services Web (paramètres des fonctions, types de données).

Pour assurer ces trois fonctionnalités, trois technologies de base en plus du métalangage XML ont été proposées et disposées en couches pour construire l'architecture de base des services Web, comme le montre le tableau ci-dessous:

UDDI	Découverte de services
WSDL	Description de services
SOAP	Communications
XML	

Tableau 1.2: Couches technologiques des services Web

Les couches XML (Extensible Markup Language) et SOAP (Simple Object Access Protocol) sont les couches de plus bas niveau, elles permettent le transport de l'information, alors que WSDL (Web Service Description Language) permet de décrire le service aux utilisateurs externes. Enfin la couche la plus haute UDDI (Universal Description Discovery and Integration) décrit ce que peut produire le service. Dans ce qui suit nous expliquons, en bref, chacun de ces standards.

1.2.3.1. XML

XML [W3C, 2008] constitue la technologie de base des architectures des SW, c'est un facteur important pour contourner les barrières techniques. XML est un standard qui permet de décrire des documents structurés transportables sur les protocoles d'Internet. En effet, il apporte à l'architecture des SW l'extensibilité et la neutralité vis à vis des plateformes et des langages de développement. La technologie des SW a été conçue pour fonctionner dans des environnements totalement hétérogènes. Cependant, l'interopérabilité entre les systèmes

hétérogènes demande des mécanismes puissants de correspondance et de gestion des types de données des messages entre les différents participants (clients et fournisseurs). C'est une tâche où les schémas de type de données XML s'avèrent bien adaptés. C'est pour cette raison que la technologie des SW est essentiellement basée sur XML ainsi que les différentes spécifications y afférentes (les espaces de nom, les schémas XML, et les schémas de Type).

1.2.3.2. SOAP

SOAP [Gudgin et al., 2003] est un protocole pour l'échange d'informations structurées avec les services Web, recommandé par le W3C (World Wide Web Consortium). SOAP est le successeur de XML-RPC. Il est basé sur XML pour le format des messages et il s'appuie généralement sur des protocoles de la couche application pour le transport des messages (RPC, HTTP).

SOAP forme la couche inférieure de la pile des protocoles des SW, fournissant un Framework pour l'échange de messages sur lequel les services Web peuvent se baser. Un message SOAP est structuré en trois parties: un entête (facultatif) et un corps à l'intérieur d'une enveloppe **(**Fig.1. 2).

SOAP présente l'avantage d'être suffisamment polyvalent pour utiliser différents protocoles de transport. En effet, bien que HTTP soit le protocole par défaut, il est tout à fait possible d'utiliser HTTPS (identique à HTTP mais avec chiffrement) ou SMTP (protocole de messagerie). En revanche, SOAP peut être considérablement plus lent que ses concurrents (e.g: CORBA) du fait de la verbosité de XML.

Fig.1. 2: Structure d'un message SOAP [Gudgin et al., 2003]

1.2.3.3. WSDL

WSDL [Christensen et al., 2001] est un langage basé sur XML pour la description de services Web. Un document WSDL contient quatre principales sections comme présenté dans la Fig.1. 3. La section interface définit les fonctions fournies par le service et qui sont accessibles publiquement. La section types définit en XML-schema les types de données utilisés pour chaque message de type requête ou réponse. La section binding indique le protocole de communication à utiliser, généralement SOAP. Enfin la section service définit le nom du service ainsi que l'adresse du service afin de pouvoir l'invoquer.

Le document WSDL représente un contrat entre le client et le service Web qui définit comment utiliser le service. Le WSDL présente l'intérêt d'être indépendant de tout langage de programmation ou plateforme puisqu'il est basé sur XML. En utilisant le WSDL, le client peut alors localiser le SW et invoquer une de ses fonctions publiquement accessible. Néanmoins, le manque de sémantique dans WSDL empêche la découverte automatique des SW et par conséquent leur invocation et composition automatiques.

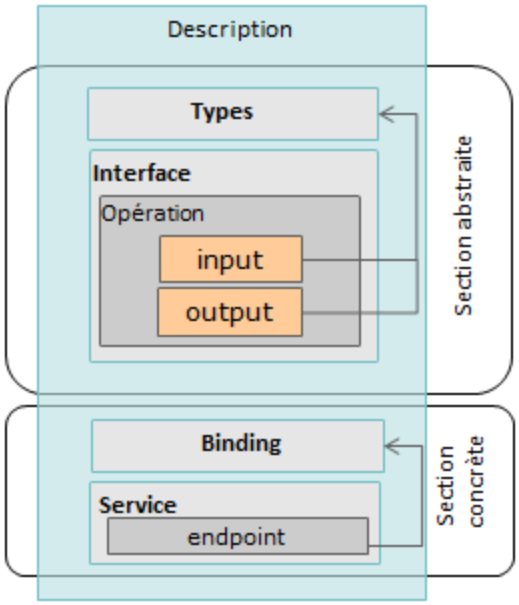


Fig.1. 3: Structure d'un message WSDL [Christensen et al., 2001]

1.2.3.4. UDDI

UDDI [Clement et al., 2004] est une spécification technique sponsorisée par OASIS pour la description, la découverte et l'intégration de services Web. Cette spécification constitue une partie importante dans la pile de technologies liées aux SW, elle permet aux personnes de publier et rechercher des SW. UDDI permet de mettre en œuvre le principe de découverte cautionné par le SOA.

UDDI est décomposé en deux parties. La première partie est une spécification technique pour construire un annuaire distribué de métiers et de services. Les données sont stockées dans un format XML spécifique et UDDI spécifie une interface de programmation (API) pour la recherche de données existantes ou la publication de nouvelles. La seconde partie est une implémentation entièrement fonctionnelle de la spécification UDDI.

Les données stockées dans l'annuaire sont structurées en trois catégories. Les pages blanches comprennent la liste des entreprises ainsi que des informations associées à ces dernières. Nous y retrouvons donc des informations comme le nom de l'entreprise, ses coordonnées, la description de l'entreprise mais également l'ensemble de ses identifiants. Les pages jaunes recensent les SW de chacune des entreprises sous le standard WSDL. Les pages vertes fournissent des informations techniques précises sur les services fournis. Typiquement on indiquera dans ces informations les adresses Web des services et les moyens d'y accéder.

Actuellement, SOAP, WSDL et UDDI sont les trois standards qui constituent l'architecture de base des services Web. Ensemble, ils résolvent les problèmes de l'hétérogénéité des systèmes pour l'intégration d'applications en ligne. Cependant, le triptyque SOAP/WSDL/UDDI n'est pas suffisant pour permettre une utilisation effective des services Web. Il existe de nombreux nouveaux besoins émergents (tels que les standards de sécurité, d'administration et du Web sémantique) dans le domaine des SW que cette architecture ne

peut pas aisément prendre en compte. Il s'avère donc nécessaire d'étendre l'architecture de base de services Web.

1.2.4. Architecture étendue

Afin de répondre pleinement aux exigences des applications métiers, plusieurs technologies ont été proposées pour compléter les technologies de base. L'architecture étendue (avancée) est constituée de plusieurs couches se superposent les unes sur les autres, d'où le nom de pile des services Web. La Fig.1. 4 décrit cette pile.

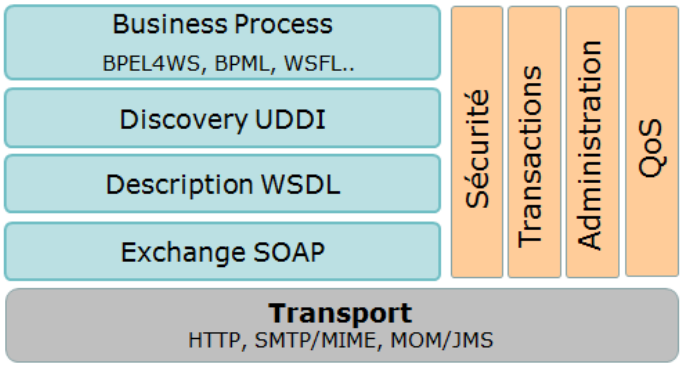


Fig.1. 4: Architecture en pile (étendue) [IBM,2004]

La pile est constituée de plusieurs couches, chaque couche s'appuyant sur un standard particulier. On retrouve, au-dessus de la couche de transport, les trois couches formant l'infrastructure de base décrite précédemment. Ces couches s'appuient sur les standards émergents SOAP, WSDL et UDDI. Comme mentionné précédemment, l'infrastructure de base définit les fondements techniques permettant de rendre les processus métiers accessibles à l'intérieur d'une entreprise et au-delà même des frontières d'une entreprise. Dans ce contexte deux types de couches permettent de la compléter : (i) les couches dites « *transversales* », [Gottschalk et al., 2002] (e.g: sécurité, administration, transactions et qualité de services (QoS)) rendent viable l'utilisation effective des SW dans le monde industriel ; (ii) une couche « *Business processus* » permet l'utilisation effective des SW dans le domaine d'e-business.

1.2.5. Services Web : problématiques et directions de recherche

Les nouvelles opportunités technologiques apportent toujours des nouvelles problématiques et conduisent à des nouvelles solutions. Les services Web basés sur l'architecture SOA ont dressés un portrait intéressant de ce que serais les futures applications d'entreprise. Néanmoins, l'infrastructure de base actuelle ne permet pas encore aux SW de tenir leur promesse d'une gestion largement automatisée fidèle à ce portrait. Le modèle des services Web présente quelques limitations notamment en matière de sémantique et de composition qui font l'objet des sujets de recherche actuels dans le domaine des services Web.

- **Problèmes de sécurité :**

Dans un environnement de services distribués et dynamiques, la sécurité ne doit pas se limiter à fournir des solutions technologiques mais à trouver une stratégie de sécurité prenant en compte les dimensions : métier, organisationnelle et technologique. En outre, la sécurité doit être appréhendée comme un processus continu qui vise l'optimisation des investissements de sécurité et assure la pérennité des mesures de sécurité mises en œuvre. Or les modèles et les architectures de référence du domaine des SW n'ont pas donné d'importance suffisante à la définition des besoins en termes de sécurité, les biens à protéger et l'identification des risques pesant sur ces biens. D'où la problématique de sécurité des services Web et des processus associés s'impose comme une dimension de recherche qui a motivé plusieurs travaux dans le domaine académique et industriel [Kuyoro et al., 2012].

- **Problèmes de sémantique:**

Tel que présentés précédemment, les services Web sont décrits syntaxiquement et ne permettent en aucun cas l'interaction entre services, leur découverte dynamique ou automatique, ou encore leur composition sans une intervention humaine. Pour le permettre il paraît alors nécessaire de se doter d'un mécanisme qui gère le problème de sémantique [Shalini et al., 2014].

- **Problèmes de composition de services Web:**

Les services Web, tels qu'ils sont définis actuellement, sont limités à des fonctionnalités relativement simples. Toutefois, pour certains types d'applications, il est nécessaire de combiner un ensemble de services Web simples en un service répondant à des exigences plus complexes [AlSedrani et al., 2016] .

Dans le cadre de ce travail nous nous intéressons à la problématique de composition de service Web, dont le traitement ne pourrait se faire indépendamment du traitement des autres problématiques de recherche précitées. Dans la suite de ce chapitre nous discutons les différentes facettes de cette problématique, aussi nous définissons les différents concepts liés à la composition et nous présentons les approches de composition de services existantes dans la littérature tout en soulignant les avantages et les inconvénients de chacune d'elles.

1.3. Composition de services Web

1.3.1. Problème de la composition de services Web

Si les services Web sont considérés comme « l'essor incontestable de l'utilisation du Web » alors la composition de services Web peut être définie comme étant « l'essor incontestable de l'utilisation des services Web ». En effet, l'utilisation intensive des standards et de l'architecture SOA, avec un effet d'interopérabilité à la clé, sont à l'origine du succès du modèle de service Web qui a marqué un nouveau point d'inflexion dans le monde des systèmes distribués et qui a révolutionné l'e-commerce, mais comme toute nouvelle opportunité technologique, la consommation de ce paradigme engendre de nouvelles problématiques et relève de nouveaux défis qui conduisent à des nouvelles idées, solutions et approches, permettant ainsi de dessiner une ligne continue d'évolution technologique avec

des points révolutionnaires. Nous nous intéressons, en particulier, à l'un des points de continuité : la composition dynamique et automatique de services Web.

Dans le contexte d'un environnement dynamique tel que le Web ou l'informatique pervasive, où les services changent volatilement et les besoins des utilisateurs varient imprévisiblement, imaginons un scénario où un client exprime une demande qui ne peut être directement et/ou totalement satisfaite par l'existant en termes de services. Face à ce genre de situations trois alternatives de réponses sont possibles:

(a) **Répondre avec l'existant qui ne satisfait que partiellement le souhait du client.** Cette solution conduira à creuser davantage le fossé entre ce que veut l'utilisateur (la cible) et ce que produit le Web (l'offre), et résultera donc en un faible niveau de satisfaction du client.

(b) **Répondre par la création d'un nouveau service pour chaque nouvelle requête.** Cette alternative n'est pas une meilleure solution : en plus d'être couteuse, une telle réponse n'optimise pas l'exploitation des ressources existantes, ce qui pousse à se poser des questions sur le sort et l'efficience d'un Web en perpétuelle croissance et qui risque d'exploser à tout moment.

(c) Décrits précédemment comme des composants « autosuffisants » et « légèrement couplés », les services Web sont donc capables d'être composés. La meilleure solution serait alors **de combiner les services existants susceptibles de contribuer à répondre à la requête, d'étudier leur complémentarité et de les assembler pour former une nouvelle entité plus complète satisfaisant la fonctionnalité cible**. Ce mécanisme est appelé la composition des services Web (CSW).

A partir de ce scénario illustratif nous pouvons former une première perception (modélisation) du problème de la composition: *étant donné une communauté de services et une requête client, nous réduisons le problème de la composition à l'existence d'un service objectif (composite) qui, pour satisfaire la requête client, effectue des tâches en sollicitant des fonctionnalités de la communauté de services.*
Formellement la composition de services Web peut être vue comme une fonction de composition F_c telle que :

$$\mathbf{F_c : I \times D \rightarrow O}$$

Où I est l'ensemble des requêtes possibles de l'utilisateur, O l'ensemble des sorties qu'obtient l'utilisateur et D l'ensemble des ensembles des descriptions de la communauté des services Web. Ce formalisme constitue la base du cadre théorique de l'approche que nous développerons au cours de ce manuscrit.

Au sens large la CSW se réfère donc au processus de créer de nouvelles fonctionnalités en combinant des fonctionnalités offertes par d'autres services existants, composés ou non, en vue d'apporter une valeur ajoutée en fonction d'une requête de l'utilisateur [Curbera et al., 2001], ce processus inclut un mécanisme de découverte, d'intégration et d'exécution de ces services dans un ordre bien défini afin de satisfaire un besoin bien déterminé.

Il est important de noter la différence entre le problème traité dans cette thèse : CSW et un autre concept homologue qui a gagné en popularité ces dernières années : le Mashup [Ankolekar et al., 2008]. Les deux concepts convergent vers la même philosophie qui vise à assembler des services métiers internes ou externes pour créer de la valeur. Le Mashup (application composite) par contre ne fait que mixer le contenu ou les services provenant de plusieurs applications plus ou moins hétérogènes et les présenter dans une seule interface graphique et ce, en se basant sur la programmation événementielle. La raison derrière la popularité de ce concept réside dans le fait que plusieurs grandes entreprises d'internet, comme Yahoo (www.yahoo.com), Google (www.google.com), et Amazon (www. amazon.com), ont ouvert leurs données pour être utilisées avec d'autres sources de données sans négociation de licence. Leur intérêt principal étant d'inciter les développeurs à répandre et à diffuser leur contenu.

Par ailleurs, il est également important de signaler que la composition des services n'est pas un nouveau concept, dans le monde industriel elle constitue un élément important dans la chaine de gestion des processus métiers (business process management). Une interaction métier complexe fait souvent intervenir une série de fonctionnalités de haut niveau qui continent des activités métiers basiques, si on considère que chacune de ces activités basiques est un service, alors le processus de création d'un service composite n'est autre que le processus de création d'un processus métier pour satisfaire un besoin métier donné. La perception de la composition de cet angle de vue a inspiré plusieurs approches de CSW (détaillées au niveau de la prochaine section) qui décrit notamment comment les services (activités) peuvent être composés pour former un nouveau service. Mais contrairement aux processus métiers « traditionnels » qui sont exécutés de manière prévisible et répétitive dans un environnement statique, les services **Web** composés s'exécutent dans un environnement versatile où le nombre de services disponibles évolue très rapidement. De plus, la forte compétition engendrée par la multitude de fournisseurs de services oblige les entreprises à adapter leurs services pour mieux répondre aux besoins des clients et ce à moindre coût.
Les défis qui rendent alors la CSW une tâche hautement complexe, malgré les efforts de recherche et de développement autour de cette problématique, peuvent être résumés en cinq points :

(i) la complexité liée à la conception de services Web : un SW n'est pas adaptable par nature. Il est passif jusqu'à ce qu'il soit invoqué. Il a des connaissances de lui-même mais pas de ses applications ou de ses utilisateurs clients;
(ii) le nombre des services Web disponibles est très grand, il est déjà au-delà des capacités humaines d'analyser manuellement ces entités en croissance continue;
(iii) puisque le nombre de services Web augmente jour après jour, il devient plus difficile de trouver automatiquement le SW qui peut effectuer la tâche à accomplir et encore plus difficile de composer un ensemble de services avec des approches classiques ;
(iv) les services Web peuvent être crées et réactualisés à la volée. Par conséquent, le système de composition doit détecter la mise à jour lors de l'exécution. Ainsi, le schéma de la composition devrait s'adapter en fonction des nouvelles informations ;
(v) les services Web sont généralement établis par des organisations différentes qui utilisent différents modèles conceptuels pour la présentation des caractéristiques des

services. Cela exige l'utilisation d'informations pertinentes pour faire correspondre (matching) et composer les services Web.

Ces facteurs imposent des contraintes fortes sur les systèmes qui délivrent des services composés. En conséquent, les modèles proposés pour la description des services composés devront intégrer d'emblée ces contraintes en exhibant des possibilités réelles d'adaptabilité à leur environnement. Dans ce contexte plusieurs travaux, traitant la problématique de CSW au travers différentes approches et techniques ont été proposés cette dernière décennie. Nous faisons le tour d'horizon de ces approches dans la section suivante.

1.3.2. Approches de la composition de services Web

Etant donné un ensemble de services Web disponibles et une requête d'un client, la composition de services Web consiste à synthétiser un nouveau service composé qui réalise la requête du client, donc de spécifier comment coordonner les services Web disponibles pour réaliser cette requête. Les techniques proposées pour réaliser une telle spécification peuvent être classées selon deux axes, comme illustré dans la Fig.1. 5: (1) en fonction du degré de participation de l'utilisateur dans la définition du schéma de composition, ces propositions sont manuelles, semi-automatiques ou automatiques. Et selon que, (2) la sélection des services Web et la gestion du flot soient faites ou non a priori, les techniques de cette dernière classe peuvent être regroupées en deux grandes familles, les techniques de composition dites statiques qui sont définies à l'aide de processus métier (orchestration et chorégraphie), et les techniques de composition dynamiques, c'est lorsque la composition de services Web tient compte des services disponibles, de leurs fonctionnalités et du but à atteindre que ce soit avant ou pendant l'exécution des services Web. Les techniques de composition dynamique peuvent être regroupées à leur tour en deux sous familles : les techniques utilisant une approche basée sur les Workflows (processus métier) et celles basées sur des techniques d'intelligence artificielle (IA). Nous soulignons que les approches statiques et manuelles de composition de services sont celles les plus adoptées par l'industrie. Nous présentons brièvement chacune de ces approches dans la suite de cette section.

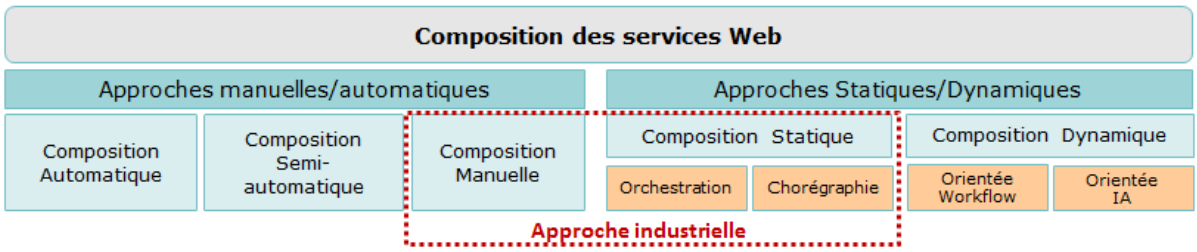


Fig.1. 5: Approches de composition de services Web

1.3.2.1. Composition manuelle/automatique

Les premiers travaux de CSW ont contribué essentiellement à des approches de composition de services manuelles et semi-automatiques, tandis que la plupart des recherches récentes se concentrent à automatiser les processus de composition de services Web.

La composition de services manuelle est basée sur l'intervention humaine. La plupart des approches de composition manuelle [Taylor et al., 2003] [Benatallah et al., 2003] laissent à la charge des utilisateurs la génération des Workflows spécifiques et exécutables pour représenter des services composites et décrire leur ordre d'exécution, soit graphiquement ou à travers des langages déclaratifs. La plupart de ces approches sont entièrement statiques puisque les services à composer sont préalablement sélectionnés, et le flot de composition défini a priori.

Construire un service composite implique la définition d'un schéma de composition (ou modèle de processus) spécifiant le flux de contrôle et de données entre les activités relatives aux services composants. Le degré d'automatisation dans la création d'un tel schéma est une caractéristique importante d'une stratégie composition de services. Les méthodes de composition de service traditionnelles requièrent de l'utilisateur de définir le flux de données et le flux de contrôle manuellement, soit directement, soit par l'intermédiaire d'outils de conception. En considérant l'augmentation continue de services composants, sélectionner manuellement les services adéquats pourrait être une tâche fastidieuse. En outre, la création de flux de données est une tâche complexe et exige que le compositeur ait une connaissance approfondie sur les représentations et la signification des paramètres sous-jacents. Des stratégies de composition avancées soutiennent activement l'utilisateur dans la création du schéma de composition. Elles sont souvent considérées comme des méthodes de composition semi-automatiques. Elles permettent de faciliter et d'accélérer la création du schéma de composition en proposant des suggestions sémantiques pour aider et guider l'utilisateur dans le processus de composition. L'outil graphique IRS-III développé par [Domingue et al., 2004] et le Framework décrit dans [Chen et al., 2003] sont des exemples d'approches semi-automatiques.

Les approches de composition entièrement automatiques visent la génération d'un plan de composition de services sans intervention humaine. La plupart des méthodes de composition inspirées de l'intelligence artificielle et basées sur la logique formelle sont utilisées pour cette fin. Aussi, au moyen d'un algorithme sophistiqué de l'IA, un Workflow contenant des activités ou des services est généré automatiquement pour satisfaire les exigences établies par le demandeur. Dans ce cas, le moteur de composition y est toujours dynamique. SWORD [Ponnekanti et al., 2002] est un exemple d'outil qui automatise la tâche de composition en utilisant des descriptions de services basées sur les règles. L'utilisateur spécifie les faits d'état initial et d'état final. Le planificateur tente alors de construire une chaîne de services pouvant satisfaire ces besoins. La question de « jusqu'à quel point le processus de composition peut être automatisé ? » a été et demeure l'objet de nombreux travaux [Kim et al., 2009], [Rodriguez-Mier et al., 2011], [Feng et al., 2012].

1.3.2.2. Composition statique/dynamique

1.3.2.2.1. Composition statique de services Web

Dans cette approche, les SW à composer sont choisis et reliés ensemble, avant d'être compilés et déployés. Microsoft Biztalk et Bea Weblogic sont deux exemples de moteurs de composition statique de services web [Sun et al., 2003]. Ceci peut marcher autant que

l'environnement des services Web, les partenaires d'affaires, ainsi que lesdits composants changent peu ou pas du tout. Si les fournisseurs de services proposent d'autres services ou changent les anciens services, des incohérences peuvent être causées, ce qui demanderait un changement de l'architecture du logiciel, voire la définition du processus et créerait l'obligation de faire une nouvelle conception du système. Dans ce cas, la composition statique des services Web est considérée trop restrictive parce que les composants doivent s'adapter automatiquement aux changements imprévisibles. Deux approches existent dans la littérature pour organiser les interactions entre les services intervenants dans le processus de la composition : orchestration et chorégraphie.

❖ Orchestration :

L'orchestration de services Web exige de définir l'enchaînement des services Web selon un canevas prédéfini, et de les exécuter selon un script d'orchestration. Ces derniers (le canevas et le script) décrivent les interactions entre services Web en identifiant les messages, et en spécifiant la logique et les séquences d'invocation. Le module exécutant le script d'orchestration de services Web est appelé un moteur d'orchestration. Ce moteur d'orchestration est une entité logicielle qui joue le rôle d'intermédiaire entre les services en les appelant suivant le script d'orchestration. Ce type de composition permet de centraliser l'invocation des services Web composants.

La Fig.1. 6 illustre l'orchestration [VELASCO, 2008]. La requête du client (logiciel ou humain) est transmise au moteur d'exécution (*Moteur*). Ce dernier, d'après le processus préalablement défini, appelle les services Web (ici, SW_1, SW_2, SW_3 et SW_4) selon l'ordre d'exécution.

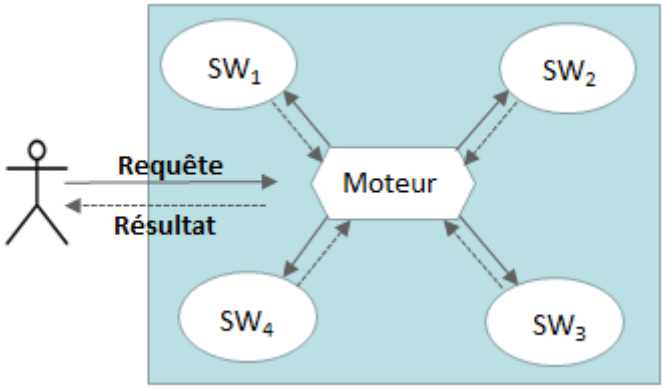


Fig.1. 6: **Vue générale de l'orchestration**

Les principaux langages d'orchestration de services Web [Peltz, 2003] sont : BPML (Business Process Modeling Language) [Arkin, 2002] et BPEL4WS (Business Process Execution Language for Web Services) [Curbera et al., 2003] qui est considéré également un langage de chorégraphie.

❖ Chorégraphie :

Contrairement à l'orchestration, la chorégraphie n'a pas un coordinateur central. Chaque service Web mêlé dans la chorégraphie connaît exactement quand ses opérations doivent être exécutées et avec qui l'interaction doit avoir lieu [Arenaza, 2006]. Elle est associée à l'échange

de messages entre services Web plutôt qu'à un processus métier exécuté par un seul partenaire.
La Fig.1. 7 permet d'illustrer une vue générale d'une composition de services Web de type chorégraphie [VELASCO, 2008]. Le client (logiciel ou humain) établit une requête qui est satisfaite par l'exécution automatique de quatre services Web (SW_1, SW_2, SW_3 et SW_4).

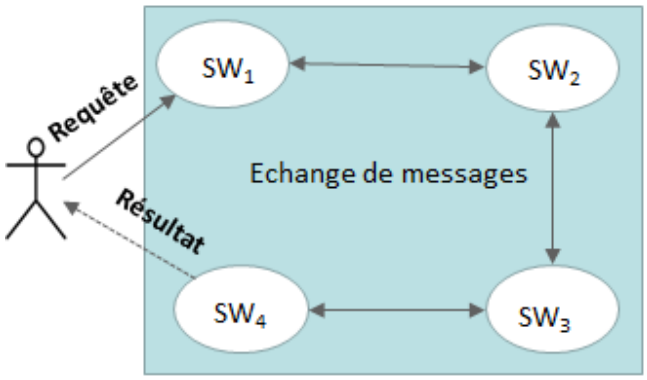


Fig.1. 7: **Vue générale de la chorégraphie**

Les principaux langages de chorégraphie de services Web [Peltz, 2003] sont : XLANG [Thatte, 2001] et WSCL (Web Service Conversation Language) [Banerji et al., 2002].

Qu'il s'agisse d'orchestration ou de chorégraphie, l'agrégation des différents services Web dans ce type de composition se fait de manière statique avant l'exécution. L'agrégation obtenue est rigide et donc difficilement modifiable. Ainsi si les fournisseurs de services proposent d'autres services ou changent les anciens services, des incohérences peuvent être causées, ce qui demanderait un changement de l'architecture du logiciel, voir la re-conception du système.

1.3.2.2.2. Composition dynamique de services Web

Dans cette approche, les services sont sélectionnés et composés à la volée en fonction des besoins formulés par l'utilisateur**.** Cette approche offre le potentiel de réaliser des applications flexibles et adaptables, en sélectionnant et en combinant les services de manière appropriée sur la base de la requête et du contexte de l'utilisateur. Ce type de composition peut engendrer de nombreuses applications utiles, qui n'ont pas été prévues à l'étape de conception. Par conséquent, la composition dynamique de services Web est propice dans un environnement, tel que le Web et l'informatique pervasive où les composants disponibles sont dynamiques et les attentes des utilisateurs variables et personnalisées.

Les différentes approches existantes pour la composition dynamique de services Web peuvent être regroupées en deux sous familles: (i) La première se base sur le fait qu'un service Web composite est défini par un ensemble de services atomiques et par la façon dont ils communiquent entre eux. Cette définition est de même type que la manière dont est défini un processus métier. Cette sous famille propose donc d'adapter les méthodes d'orchestration et de chorégraphie afin de les rendre dynamiques. (ii) Dans la seconde sous famille, la

composition est vue comme la génération automatique d'un plan d'exécution des services Web. Les approches envisagées sont des approches du domaine de l'intelligence artificielle et plus particulièrement de la planification.

❖ Approches orientées Workflow

D'une certaine façon, un service composite est similaire à un processus métier [Casati et .al, 2001]. Un service composite inclut un ensemble de services atomiques ainsi que les contrôles et les échanges de données entre ces services. De la même façon, un processus métier est composé d'un ensemble d'activités élémentaires structurées, ainsi que l'ordre d'exécution entre elles.

Parmi les travaux qui investissent dans ce type d'approches de composition on peut évoquer les travaux suivants :

- Casati présente Eflow [Casati et al., 2000], une plateforme pour la spécification, la création et la gestion de services composites en utilisant les méthodes de génération de workflow statiques mais représentés en interne sous la forme d'un graphe qui peut être modifié dynamiquement lors de l'exécution. Ce graphe contient, en plus des nœuds représentant les services, des nœuds de décision et d'événements qui permettent une plus grande robustesse du système : en cas de non disponibilité d'un service Web, celui-ci peut être automatiquement remplacé par un service Web équivalent. Khalaf [Khalaf et al., 2003] propose de considérer les chorégraphies créées avec BPEL4WS comme des services Web, c'est-à-dire créer des patrons de composition qui pourront être composés, formant ainsi de nouveaux patrons de composition. Cette solution facilite la composition par une intégration plus simple de patrons d'assemblage prédéfinis dans un nouveau patron mais ne règle pas le problème de la dynamique de la composition de services Web**.** Laukkanen [Laukkanen et al., 2003] identifie deux solutions possibles pour composer dynamiquement des processus métier : (i) remplacer un service Web dans un processus métier existant par un autre service ayant des fonctionnalités similaires, ou (ii) définir un nouveau Workflow à partir des services Web disponibles. Les auteurs proposent d'utiliser les descriptions sémantiques des services Web pour pouvoir comparer les fonctionnalités.Osman [Osman et al., 2005] propose de limiter les points négatifs des techniques de composition de type Workflow en utilisant les technologies du Web sémantique, il ne s'agit pas directement de composition automatique mais le but est de limiter le nombre de services Web disponibles afin de créer plus facilement des Workflows. Pour cela, les auteurs proposent de regrouper les services Web en domaines spécifiques (e.g., domaine des services Web d'hôtels, de transports, etc.) Ainsi le choix des services à utiliser est restreint au domaine des services ayant comme fonctionnalités les objectifs du Workflow.

❖ Approches orientées intelligence artificielle

La composition dynamique de services Web par des techniques d'intelligence artificielle, et plus particulièrement par des techniques de planification, est la voie qui semble la plus prometteuse. Dans ce qui suit, nous présentons quelques axes de recherche actuels concernant la composition par planification et par d'autres techniques d'intelligence artificielle.

- **Calcul situationnel (situation calculus**) : McIlraith [McIlraith et al ., 2002] propose d'adapter et d'étendre le langage Golog pour la construction automatique de services Web. Golog est un langage de programmation logique qui permet de faire du calcul situationnel (i.e., langage logique qui sert à représenter des changements ou des évolutions en termes de situations, d'actions et d'objets). Le problème de la composition de services Web est abordé de la façon suivante : la requête de l'utilisateur et les contraintes des services sont représentées en termes de prédicats du premier ordre dans le langage de calcul situationnel. Les services sont transformés en actions (primitives ou complexes) dans le même langage. Puis, à l'aide de règles de déduction et de contraintes, des modèles sont ainsi générés et instanciés à l'exécution à partir des préférences utilisateur. Ces recherches ont été étendues dans [Narayanan et al ., 2002] : les prédicats nécessaires aux calculs situationnels sont tirés de DAML-S (un langage sémantique) [Ankolekar et al. , 2002] et représentés sous la forme de réseaux de Petri.
- **Theorem proving :** Waltinger [Waltinger et al ., 2000] a élaboré une approche pour la composition de services par preuve de théorèmes. Cette approche est basée sur la déduction automatique et la synthèse de programmes. Les services disponibles et les requêtes de l'utilisateur sont traduits dans un langage du premier ordre. Puis des preuves sont produites à partir d'un prouveur de théorèmes. La composition est obtenue à partir de preuves particulières. Dans le même genre, Rao et al [Rao et al ., 2003] ont introduit une méthode de composition automatique de services Web sémantiques en utilisant des preuves de théorèmes logiques linéaires (Linear Logic Theorem Proving).
- **Système multi-agents :** Du fait de leur autonomie et leur hétérogénéité, les services Web peuvent être vus comme des agents. Ainsi Cheng [Cheng et al ., 2002] proposent le langage ASDL (Agent Service Description Language) qui a pour objectif de décrire le comportement externe des services Web. Une spécification ASDL décrit les messages compris par un service, ainsi que les protocoles d'interactions utilisés. Puis la composition est effectuée à l'aide du langage ASCL (Agent Service Composition Language) qui permet de décrire la logique avec laquelle un nouveau service est composé à partir de services existants. L'avantage de ces travaux est de mettre l'accent sur les interactions entre services au sein de la composition et d'adapter celle-ci en fonction des contraintes résultant des interactions. Cependant cette méthode n'est pas totalement dynamique. En effet, elle consiste à assembler et adapter au contexte d'exécution des patrons de composition définis a priori par le concepteur.
- **Planification :** Plusieurs travaux de recherche ont exploité les techniques de planification de l'intelligence artificielle pour résoudre le problème de la composition de services comme [Vrakas et al.,2010], [Tang et al., 2013], [Puttonena et al., 2015].L'intérêt croissant de la communauté de planification pour la composition de services Web peut être expliqué par la similarité entre les descriptions sémantique d'un service Web et les représentations en planification. De nombreux travaux ont été produits pendant la dernière décennie utilisant plusieurs techniques de planification, nous citons parmi ceux les plus connus :

- Les travaux de Sheshagiri [Sheshagiri et al ., 2003] qui proposent l'utilisation d'ontologies de domaines afin d'ajouter des contraintes pour optimiser la recherche dans un espace de plans.
- Ceux de McDermott [McDermott, 2002] proposent d'utiliser la planification par régression estimée (Estimated-Regression Planning) qui permet d'utiliser des heuristiques pour guider la recherche dans un espace de situations.
- Peer [Peer, 2004] propose une approche basée sur le langage PDDL (langage de planification) dans laquelle, après création du domaine de planification à partir de la description sémantique, un planificateur est choisi parmi plusieurs en fonction des instructions PDDL utilisées dans le domaine ou de la complexité du but à atteindre.
- Dans [Wu et al ., 2003], Wu préconise l'utilisation du planificateur SHOP2 pour la composition automatique de services Web à partir de leur description sémantique. SHOP2 est un planificateur HTN (Hierachical Task Network).
  D'après l'auteur, le principe de décomposition d'une tâche en sous-tâches dans la planification hiérarchique est très similaire au concept de décomposition de processus composites dans OWL-S (un langage sémantique) [Martin et al. ¸ 2004]. La planification HTN est très puissante pour les domaines où la connaissance est complète et détaillée, ce qui n'est pas, d'après Klush [Medjahed et al ., 2003], le cas avec les services Web.
- Medjahed [Medjahed et al ., 2003] présente une technique pour générer des services composites à partir de descriptions déclaratives de haut niveau. Cette méthode utilise des règles de composabilité pour déterminer dans quelle mesure deux services sont composables. L'approche proposée se déroule en quatre phases : premièrement, une phase de spécification offre une spécification de haut niveau de la composition désirée en utilisant le langage CSSL (Composite Service Specification Langage). Ensuite, la phase de correspondance utilise des règles de composabilité pour générer des plans conformes aux spécifications du service demandé. Si plus d'un plan est généré, une sélection est effectuée par rapport à des paramètres de qualité de la composition. La dernière phase est la phase de génération : Une description détaillée du service composite est automatiquement générée et présentée au demandeur. Les règles de composabilité considèrent les propriétés syntaxiques et sémantiques des services web.

1.3.2.3. Analyse des approches dynamiques de composition de services Web

Etant intéressés dans le cadre de ce travail de recherche par la composition « dynamico - automatique» des services Web, un focus sur ce type de composition est effectué en confrontant les deux orientations appartenant à cette classe (orientée Workflow et orientée intelligence artificielle) dans le tableau comparatif ci-après :

Critère	Approches orientée Workflow (AOW)	Approches orientée IA (AoIA)	Analyse
Niveau d'automatisation	Seules la recherche et la liaison aux services concrets sont automatiques (moyen)	Pratiquement tout le processus se fait de manière automatique (élevé)	- Généralement les AoIA présente un niveau d'automatisation plus élevé
Formulation de la requête par l'utilisateur	Bas niveau : l'utilisateur doit mettre en place le modèle de processus en utilisant un langage de flux	Définit l'état initial, l'état final et certaines contraintes avec un langage de très bas niveau (les langages logiques)	- L'expression du besoin s'effectue avec un langage de bas niveau surtout pour les approches AoIA, ce qui est contraignant
Niveau de granularité des plans de composition générés	- Un Workflow abstrait - Et un workflow exécutable	Un seul plan (plan exécutable)	- Ceci rend la réutilisabilité très réduite pour les AoIA. - Mais il faut noter que les workflows abstraits de l'AOW sont définis d'une façon manuelle
Support sémantique	La majorité utilise des descriptions syntaxiques (WSDL, BPEL4WS, WSFL…)	Elles utilisent les formalismes logiques ou sémantiques	- Seules les AoIA supportent la composition sémantique ; d'ailleurs c'est l'une des raisons qui les rend plus automatiques que celles orientées workflow
Recherche des services	La majorité utilise le moteur de recherche UDDI	Recherche des services dans les profils DAML-S ou OWL-S	
Niveau de Scalabilité	Moyen	Moyen	- Le passage à l'échelle représente l'un des défis des approches de CSW
Niveau de décentralisation	Faible	Faible	- Le caractère distribué intrinsèque des services Web est peu considéré dans les approches de la CSW (sauf pour ceux qui se basent sur les systèmes multi-agents)

Tableau 1. 3: Tableau comparatif des approches dynamiques de CSW

Comme illustré dans le Tableau 1. 3, chacune des deux approches de composition présente des points forts dans certains aspects de la composition, et des limites sur d'autres. Pourtant, aucune de ces approches ne peut être considérée comme étant pleinement automatique et dynamique. En effet des efforts doivent être fournis pour améliorer : (i) le niveau de décentralisation des architectures de composition utilisées, (ii) l'intuitivité d'expression de la requête par l'utilisateur, (iii) le niveau de support sémantique et (iv) le niveau de granularité des plans générés.

Pour travailler ces aspects il est nécessaire, à notre avis, d'adopter une perception ouverte du problème de CSW, qui ne le réduit pas à la génération d'un schéma ou d'un plan et ne se limite pas à un package d'outils offert par une discipline donnée (génie logiciel, planification…).

Nous considérons plutôt la CSW comme un processus continu de manipulation de Workflows complexes, qui nécessite la résolution de problèmes de synthèse, d'optimisation, de découverte et de contrôle d'exécution. Nous allons alors faire appel à différentes techniques issues de différentes disciplines technologiques (Système multi-agents, Planification, ontologies et l'informatique autonomique) en vue de proposer une approche efficace de composition automatique et dynamique de services Web. Cette façon d'aborder la CSW est exprimée dans le chapitre suivant sous forme d'un principe qu'on nomme l'hypothèse des quatre forces (H4F).

1.4. Conclusion

Bâtis sur des standards (SOAP, WSDL, UDDI, XML) et une architecture (SOA), les services Web ont permis la naissance d'une nouvelle génération de systèmes caractérisés par une meilleure intégration d'applications hétérogènes et une meilleure communication entre les différents composants de ces applications. La recherche dans ce domaine est très active et s'intéresse notamment à la composition des services Web. En effet, plusieurs travaux ont été proposés dans la littérature pour répondre à ce problème, aucun ne fait l'objet d'un standard, leur étude permet, par contre, de dégager deux grandes familles de compositions : les compositions manuelles/automatiques et les compositions statiques/dynamiques, nous nous intéressons dans le cadre de cette thèse à la composition automatique et dynamique des services Web. De ce fait, nous allons explorer dans le prochain chapitre les disciplines technologiques susceptibles de nous aider à construire un système de composition dynamique et automatique de services Web efficace.

Chapitre 2

Hypothèse des quatre forces

Creativity is just connecting things.
Steve Jobs

2.1. Introduction

Le présent chapitre poursuit l'état de l'art en exposant les aspects relatifs aux quatre domaines de recherche qui constituent la base de laquelle notre approche puisera son fond théorique. Nous découvrons également la motivation derrière ce choix et la synergie escomptée de cette combinaison.

2.2. Hypothèse des quatre forces : introduction

Une remarque s'impose lors de l'analyse des approches de composition présentées dans le chapitre précédent: la plupart des contributions actuelles ne sont, en réalité, qu'une manifestation des disciplines auxquelles leurs fondateurs appartiennent. Les industriels, par exemple, perçoivent la composition comme étant un processus métier et proposent des approches orientées Workflow pour répondre au problème CSW. Alors que les chercheurs dans l'intelligence artificielle (IA) affirment qu'il s'agit plutôt d'un comportement intelligent et utilisent des techniques d'IA pour résoudre le problème (Calcul situationnel, algorithme génétique...). Nous estimons toutefois que cette façon de percevoir le problème de la CSW aura inéluctablement un effet limitatif et restrictif sur la solution conçue.

En effet, fort est de constater qu'en pratique, attacher une problématique à une seule autre discipline permet, peut-être, de résoudre la projection du problème sur la discipline mais pas forcément la problématique initiale. C'est pareil pour la CSW, adopter une vision qui se limite à calquer le problème de la composition sur un seul autre problème, résulte souvent en des solutions jugées incomplètes, voir inadaptées, et c'est justement à cause du cadre théorique qui finit souvent par dévier du problème original (par exemple, l'équivalence exclusive entre la composition et la planification est susceptible d'apporter des réponses aux problèmes relatifs à l'ordonnancement des services élémentaires formant le service composite, mais cette équivalence n'offre aucune solution pour les autres questions liées au problème de la CSW comme la recherche, la sélection et la sémantique des services Web).

Partant de cette prémisse, nous proposons dans le cadre de ce travail une orientation plutôt ouverte, qui consiste à exploiter les méthodes et les outils de plusieurs directions de recherche en vue de converger vers une réponse crédible et efficace à la problématique.

Par ailleurs et au-delà de la composition des services, le présent travail défend la composition comme un concept général, La composition représente l'objectif et l'outil de cette thèse :

- C'est l'objectif dans la mesure où nous cherchons un moyen pour **composer** dynamiquement et automatiquement des services Web.
- Et c'est l'outil dans le sens où pour atteindre cet objectif, nous allons **décomposer** le problème CSW en sous problèmes et nous allons utiliser une **combinaison** de méthodes et d'outils issus de différents horizons technologiques pour le résoudre.

Ainsi :

- Nous adoptons une perception holistique du problème favorisant l'ouverture sur tout moyen susceptible de contribuer à la résolution du problème, quelques soit son appartenance disciplinaire (*Ouverture*) ;
- En même temps, une rupture avec les approches existantes n'est absolument pas envisageable, nous essayons au contraire de capitaliser sur les méthodes qui ont prouvé leur potentiel à résoudre, même partiellement, la problématique (*Continuité*).
- Ces deux principes convergent vers l'idée centrale suivante: Pour apporter une réponse holistique au problème, la combinaison d'outils et de méthodes issus de différents champs de recherches, ayant chacun un potentiel de contribuer à la résolution partielle du problème peut former un tout indivisible ayant le potentiel de résoudre complétement et efficacement la problématique.

Ce préambule nous amène à formuler l'hypothèse fondamentale de travail sur lesquelles est basée l'approche proposée.

Etant donnée la problématique de la CSW traitée dans ce travail:

(1) **Si** pour composer automatiquement des services Web ces derniers doivent se comprendre mutuellement, **alors** une technologie sémantique est nécessaire ;
(2) **Si** pour composer dynamiquement des services Web appartenant à des organisations différentes, dispersées et distribuées ces derniers doivent coordonner, **alors** une architecture multi-agents est à utiliser.
(3) **Si** pour composer automatiquement des services Web, chaque service doit être en mesure de raisonner d'une façon automatique, **alors** les algorithmes de planification peuvent être adoptés.
(4) **Si** pour composer efficacement des services Web, un comportement autonomique doit être généré par le processus global de composition, **alors** l'application des principes du paradigme d'Autonomic Computing est envisageable.
(5) **Si** (1) et (2) et (3) et (4) **alors** la synergie entre quatre champs de recherche, à savoir : le Web sémantique (WS) [Berners-Lee et al. ,2001], les Systèmes Multi-Agents (SMA) [Ferber.,1999], l'Autonomic Computing (AC) [IBM, 2003] et la planification de l'intelligence artificielle (PIA) [D.S Weld, 1999] peut conduire au développement d'une plateforme complète qui supporte l'intégration, la personnalisation et l'interopérabilité des services Web.

Nous appelons cette vision l'hypothèse des quatre forces (H4F).

Dans ce qui suit nous donnons un aperçu sur chacune des quatre forces afin d'identifier les aspects sur lesquels nous travaillons pour générer la synergie entre les forces.

2.3. Le Web Sémantique : une vision d'un Web intelligent

2.3.1. Vision du Web sémantique

Le Web tel que nous le connaissons aujourd'hui est encore syntaxique, dans le sens que la structure des données (ou les ressources au sens large) est bien définie, mais que son contenu reste quasi inaccessible aux traitements machines. Seuls les humains peuvent interpréter leurs contenus. Proclamée « technologie du futur », le Web sémantique est la nouvelle vision du Web qui promet de lever cette difficulté en permettant de s'ouvrir sur des nouvelles possibilités d'automatisation dans le Web et d'assurer une meilleure collaboration entre les humains et les machines.

Le terme de Web sémantique (WS) a été proposé la première fois en 2001 par Tim Berners Lee, l'inventeur du World Wide Web [Berners-Lee et al. ,2001] pour désigner une évolution du Web qui permettrait aux données disponibles d'être plus facilement utilisables et interprétables aussi bien par l'homme que par la machine, et ce grâce à la représentation sémantique de leurs contenus. Le Web sémantique, appelé également Web des données ou encore Web 3.0, étend le réseau de pages Web actuel en y ajoutant des métadonnées qui sont compréhensibles par la machine et qui créent des liens entre les contenus des différentes pages, ce qui permet à des agents (logiciels ou humains) d'accéder au Web de façon plus intelligente et d'effectuer des tâches d'une façon toute à fait automatique. Pour permettre cette évolution, le Web a besoin d'être équipé de technologies nécessaires à la création d'un Web de données (Web of Data).

2.3.2. Architecture du Web sémantique

Les technologies du WS, développées essentiellement par le W3C, complètent le Web actuel avec des outils sémantiques. Il ne s'agit donc pas de créer un nouveau Web ou un Web séparé de l'existant : le Web de données repose entièrement sur les technologies et les concepts qui ont fait le succès du Web tel que nous le connaissons aujourd'hui. Cette infrastructure de briques technologiques se matérialise par une représentation graphique, le « layer cake » [W3C, 2007], qui montre l'agencement des différentes briques technologiques du Web sémantique (Fig.2. 1).

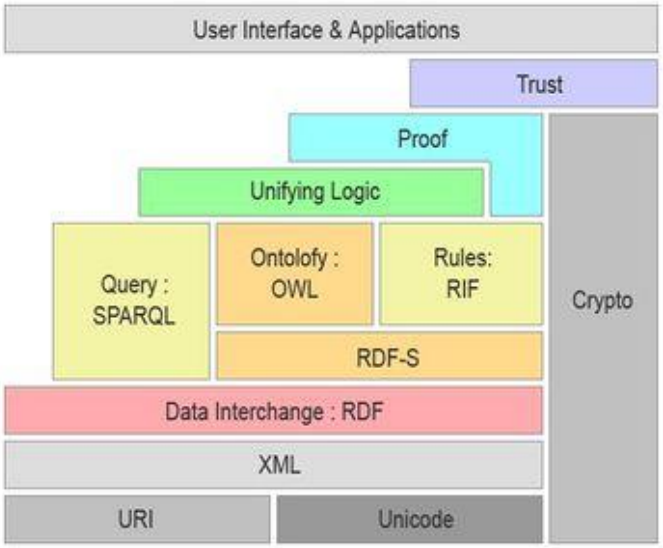


Fig.2. 1: L'architecture du Web sémantique [W3C, 2007]

Comme illustré dans la Fig.2. 1, la proposition du W3C s'appuie au départ sur une pyramide de technologies dont seulement les couches basses sont aujourd'hui relativement stabilisées. Chaque technologie se base sur les couches inférieures et est utilisée par les couches supérieures et chaque nouvelle couche est plus expressive et plus riche que le substrat sur lequel elle repose [W3C, 2007] :

i. Les couches les plus basses assurent l'interopérabilité syntaxique, à ce niveau d'architecture, on n'est toujours pas au point d'affecter une sémantique à l'information, c'est-à-dire de la décrire et de lui donner un sens. Les technologies qui appartiennent à ces couches existent donc indépendamment du WS :
   - La notion d'URI (Uniform Ressource Identifier, identifiant uniforme de ressource) fournit un adressage standard universel permettant d'identifier des ressources;
   - Unicode est un encodage textuel universel pour échanger des symboles ;
   - XML, comme défini dans le chapitre précédent, est un langage proposant une syntaxe de description de la structure des ressources, mais aussi de création et de manipulation de ces ressources. Cependant, XML n'impose aucune contrainte sémantique à la signification de ces ressources. XML ne suffit donc pas pour assurer aux logiciels la compréhension du contenu des données incluses dans la ressource: l'interopérabilité syntaxique n'est pas suffisante.

ii. Les couches RDF [W3C, 2004] et RDF Schéma sont considérées comme les premières fondations de l'interopérabilité sémantique. Leur rôle essentiel est de décrire des lois de classification de concepts ou de propriétés sur des données :
   - RDF est le premier langage fournissant un moyen d'insérer de la sémantique dans un document. Tandis que le schéma RDFS décrit les hiérarchies des concepts et des relations entre les concepts.

iii. La couche Ontologie apporte une évolution car elle assure la description de sources d'information hétérogènes. Ces sources peuvent d'ailleurs formaliser une conceptualisation partagée par plusieurs personnes, voire par toute une communauté. Le rôle de l'ontologie est donc d'aider l'humain et la machine à communiquer, en

priorisant sur l'échange de sémantique des informations plutôt que la syntaxe et en utilisant des règles précises.

- Le langage OWL (Ontology Web Langage) [W3C, 2003] est le langage dédié à la définition d'ontologie. Inspiré des logiques de descriptions (successeur de DAML+OIL [Harmelen et al. , 2001]), OWL fournit un grand nombre des règles et des contraintes logiques entre les triplets définis en RDF de façon à permettre d'assurer un mécanisme d'interconnexion entre chaque élément, et ce dans une grammaire également XML.
  Selon le niveau d'expressivité qu'on veut exprimer, OWL peut être défini en trois sous langages complémentaires OWL Lite, OWL DL et OWL Full [W3C2, 2004].OWL constitue le ciment du Web sémantique car il donne la cohérence nécessaire aux imbrications formalisées en RDF en proposant les inférences et des relations de transitivité. C'est aussi le langage sur lequel repose l'Ontologie un concept clé du Web sémantique qui sera détaillé dans la section suivante.

iv. Pour mettre en œuvre le Web sémantique de nombreuses étapes sont encore à imaginer, clarifier et spécifier. Tim Berners Lee résume ces étapes à travers les couches du « haut » de l'édifice, notamment :

- La couche Rules (règles) offre les moyens de l'intégration, de la dérivation, et de la transformation de données provenant de sources multiples ;
- La couche Logique se trouve au-dessus de la couche Ontologie, certains la considèrent comme étant au même niveau. En général, elle est utilisée pour exprimer les règles d'inférences ;
- Les couches Proof (Preuve) et Trust (Confiance) permettent de vérifier des déclarations effectuées dans le WS.

2.3.3. L'ontologie, pierre angulaire du Web sémantique

Un concept fondamental du Web sémantique est celui de l'ontologie, qui produit une signification bien définie des informations contenues dans le Web.

Le terme ontologie est initialement dédié à la philosophie et signifie l'explication systématique de l'existence, c'est-à-dire la doctrine ou la théorie de l'être [GRAF, 1996]. Cette notion a été reprise par les chercheurs dans le domaine de l'intelligence artificielle et utilisée dans le cadre de construction des systèmes à base de connaissances. L'idée était de séparer, d'un côté, la modélisation des connaissances d'un domaine, et d'un autre côté, l'utilisation de ces connaissances (i.e. le raisonnement). Ainsi, les ontologies décrivent généralement des individus (considérés comme les objets de base du domaine étudié), les classes (ensembles ou types d'objets), les attributs (propriétés que les objets possèdent et peuvent partager avec d'autres objets), les relations (liens entre objets) et enfin les évènements (changements subis par des attributs ou des relations entre objets).

Dans ce contexte, Il existe plusieurs définitions pour le terme « Ontologie». L'une des plus simples et populaires est celle de Tom Gruber [Gruber, 1993] «**Une ontologie est une**

spécification formelle explicite d'une conceptualisation partagée ». Les attributs "formelle" et "explicite" signifient qu'une ontologie permet une interprétation automatisée de la conceptualisation par une machine.

Nous retiendrons donc qu'une ontologie traduit un consensus explicite sur la formalisation des connaissances d'un domaine afin de faciliter le partage et la réutilisation de ces connaissances par les membres d'une communauté ou par des agents logiciels. C'est dans cette optique que les ontologies se présentent comme un pilier du Web sémantique, car elles permettent de faire communiquer les hommes et les machines en utilisant la sémantique partagée par les différents acteurs du Web et en décrivant ses ressources.

De nombreux langages informatiques basés sur la logique de descriptions, plus ou moins récents, spécialisés dans la création et la manipulation des ontologies sont apparus à partir des années 1990, parmi eux on trouve ; RDF, OWL, SHOE [Heflin et al., 1999], DAML+OIL.

En fonction de leur usage, on distingue classiquement cinq catégories d'ontologies [Van Heijst et al., 1997] : génériques, de domaine, d'application, de représentation et de méthodes (Tableau 2. 1)

Catégorie d'ontologie	**Description**	**Exemple**
Ontologie générique	Décrit des concepts généraux, indépendants d'un domaine ou d'un problème particulier. Il s'agit par exemple des concepts de temps, d'espaces, d'événement. Ce type d'ontologies sont prévues pour être utilisées dans des situations diverses, et pour servir une large communauté d'utilisateurs. Les ontologies génériques sont aussi appelées Modèles de haut niveau ou ontologies TOP	Sowa [Sowa, 1999], CYC [Lenat et al., 1990]
Ontologie de domaine	Spécifie un point de vue sur un domaine particulier. Le vocabulaire est généralement lié à un domaine de connaissances générique comme la médecine ou la loi. Les concepts d'une ontologie de domaine sont souvent définis comme une spécialisation des concepts des ontologies génériques	Enterprise est un exemple d'ontologie décrivant le domaine de l'entreprise [Entreprise, 2015]
Ontologie d'application	Une ontologie d'application décrit la structure des connaissances nécessaires à la réalisation d'une tâche particulière [Van Heijst et al., 1997]. Elle permet aux experts du domaine d'utiliser le même langage que celui de l'application	
Ontologie de représentation	Spécifie un formalisme de description qui fournit une structure de représentation et des primitives pour décrire les concepts des ontologies de domaine et des ontologies génériques	La Frame Ontology [Gruber, 1992]

Ontologie de méthodes, de tâches et de résolution de problèmes	Décrit le processus de raisonnement d'une façon indépendante d'un domaine et d'une implémentation donnée. Ce type d'ontologies spécifie des entités qui relèvent de la résolution d'un problème telles que l'abduction, la déduction ou l'observation [Chandrasekaran et al., 1998]. Elles fournissent les définitions des concepts et relations utilisés pour spécifier un processus de raisonnement lors de la réalisation d'une tâche particulière	OntoKADS [Bruaux et al. 2014]

Tableau 2. 1 : Catégories d'ontologies

2.3.4. Service Web Sémantique, la nouvelle génération des services Web

2.3.4.1. Service Web sémantique

La convergence du Web sémantique avec les technologies de services Web se manifeste par la notion de service Web sémantique (SWS) [W3C, 2005]. Les services Web sémantiques sont des services Web dont la description WSDL est augmentée par des descriptions sémantiques supplémentaires. Ces descriptions sont explicitées dans des ontologies à part, afin que d'autres applications ou d'autres services Web puissent comprendre la sémantique de ces derniers et puissent accomplir des comportements automatiques. Les services Web sémantiques résultent donc des efforts menés sur l'axe de la dynamique des services sur internet et l'axe de la sémantique des ressources Web et énoncent l'émergence d'une nouvelle génération de services Web caractérisés par leur intelligence (Fig.2. 2)

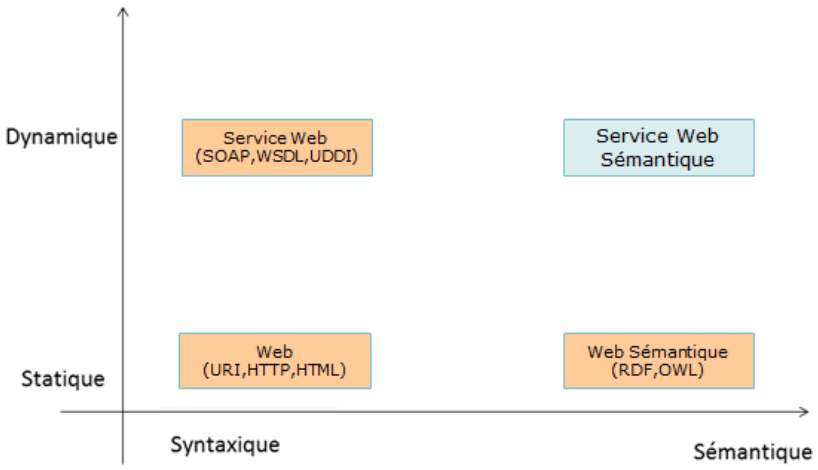


Fig.2. 2: Emergence du Service Web Sémantique

Cette nouvelle génération des services Web s'ouvre sur une vision où les services vont bénéficier d'une description riche grâce au Web sémantique. Ce qui fait que le rôle clé de l'intermédiaire entre le client et le fournisseur dans l'architecture SOA peut disparaitre. Il peut garder sa fonction de recherche de services Web mais il perdra son rôle central puisque tout le monde peut publier des descriptions sémantiques et les agents intelligents peuvent les retrouver. Par ailleurs les agents logiciels vont prendre le rôle du demandeur de service

(client) à la place de l'agent humain. Dorénavant, c'est une application ou un autre service qui va demander le service puisqu'il comprend son offre. Un dernier apport important de cette nouvelle vision est les approches de modélisation des services Web (section 2.3.4.2) qui vont permettre la description d'un type particulier de service qui fait appel, dans le cadre d'un processus, à d'autres services : un service composé.

Dimension	Vison Traditionnelle	Nouvelle vision
Service	Simple	Composé
Demandeur du service	Humain	Machine
Fournisseur	Inscription	Inscription non nécessaire
Annuaire	Joueur principal	Facilitateur
Service description	taxinomie	Ontologie
Echange des données	Basé sur la syntaxe	Basée sur la sémantique

Tableau 2. 2 : Comparaison entre la vision traditionnelle et sémantique des services Web

2.3.4.2. Modélisation des services Web sémantiques

Une condition sine qua non pour mettre en œuvre les SWS est de pouvoir les décrire de manière formelle et standard. WSDL se limite à une description syntaxique des services Web, il n'est donc plus suffisant pour décrire les SWS. Dans ce contexte, la notion d'ontologie en tant que conceptualisation formelle et consensuelle d'un domaine donné, peut jouer un rôle important pour associer une sémantique formelle à la description d'un Web service.

Plusieurs approches de modélisation des SWS ont été proposées, elle se divisent en deux catégories, la première catégorie consiste à développer un langage complet qui décrit le service Web et sa description sémantique dans un seul bloc, on peut citer dans cette catégorie les approches OWL-S [Martin et al., 2004] et WSMO [Roman et al., 2005]. La deuxième catégorie consiste à annoter les langages existants avec la description sémantique, parmi les approches qui s'inscrivent dans cette catégorie WSDL-S [Akkiraju et al., 2005] et SAWSDL [Farrell et al. , 2007]. Nous présentons dans la suite les quatre approches les plus utilisées, à savoir : OWL-S, WSMO, WSDL-S et SAWSDL. Au fur et à mesure du développement du sujet, des critères d'évaluation de ces approches émergeront et nous aideront à les comparer afin de choisir celle la plus adaptée (cf. chapitre 4).

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

OWL-S est une extension du langage OWL qui se présente comme le successeur de DAML-S (DARPA agent markup language for services) [Ankolekar et al., 2002], l'un des premiers langages de description sémantiques de services utilisant le modèle des logiques de descriptions (plus précisément DAML+OIL). Il a pour objectif de fournir une plus grande expressivité en permettant la description des caractéristiques des services afin de pouvoir raisonner dessus dans le but de découvrir, invoquer, composer et gérer les services Web de façon la plus automatisée possible.

Ainsi un service dans OWL-S est décrit à travers les quatre zones conceptuelles suivantes :

- Le *service profile*: contient la réponse à la question: le service exige-t-il quoi d'un utilisateur ou d'un autre agent, et lui en fournit quoi ? Ainsi, la classe service *présente* un ServiceProfile.
- Le *Service Process Model*: contient la réponse à la question: Comment fonctionne le service ? Ainsi la classe service est décrite par un ServiceModel.
- Le *Service Grounding*: contient la réponse à la question: de quelle façon le service doit-il être utilisé ? Ainsi, la classe service supporte un ServiceGrounding.

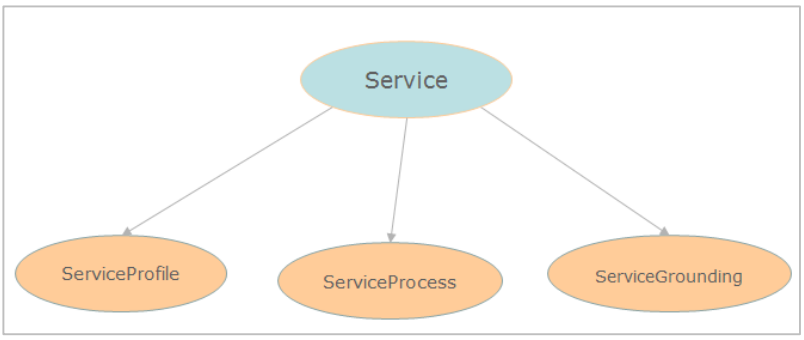


Fig.2. 3: Eléments d'une description OWL-S

❖ WSMO (Web Service Modeling Ontology)

WSMO (Web Service Modeling Ontology) est une architecture conceptuelle visant à expliciter la sémantique des services Web. Elle est organisée en quatre éléments principaux :

- Les services Web: sont définis comme des entités qui fournissent une fonctionnalité. Une description est associée à chaque service, dans le but de décrire sa fonctionnalité, son interface, et ses détails internes.
- Les objectifs: servent à décrire les souhaits des utilisateurs en termes de fonctionnalités requises. Les objectifs sont une vue orientée utilisateur du processus d'utilisation des services Web, ils sont une entité à part entière dans le modèle WSMO. Un objectif décrit la fonctionnalité, les entrées, les sorties, les préconditions et les postconditions d'un service Web.
- Les médiateurs: sont utilisés pour résoudre de nombreuses incompatibilités, telles que les incompatibilités de données dans le cas où les services Web utilisent différents terminologies, les incompatibilités de processus dans le cas de la combinaison de services Web, et les incompatibilités de protocoles lors de l'établissement des communications.
- Les ontologies: fournissent la terminologie de référence aux autres éléments des WSMO, afin de spécifier le vocabulaire du domaine de connaissance d'une manière interopérable par les machines.

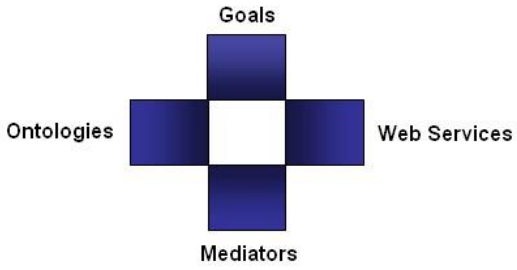


Fig.2. 4: Eléments de WSMO [Roman et al., 2005]

❖ **WSDL-S (Web Service Description Language-Semantic)**

WSDL-S (Web Service Description Language-Semantic) est un langage WSDL augmenté de sémantique, cette sémantique est ajoutée en deux étapes : (i) la première consiste à faire référence, dans la partie définition WSDL à une ontologie dédiée au service à publier; (ii) la deuxième consiste à annoter les opérations de définition WSDL de sémantique.

WSDL-S distingue quatre modèles sémantiques à savoir:

- InputSemantic: le sens des paramètres d'entrée.
- OutputSemantic: le sens des paramètres de sortie.
- Precondition: un ensemble d'états sémantiques qui doivent être vrais afin d'invoquer une opération avec succès.
- Effect: un ensemble d'états sémantiques qui doivent être vrais après qu'une opération réalise son exécution

❖ **SAWSDL (Semantic Annotations for WSDL and XML Schema)**

SAWSDL (Semantic Annotations for WSDL and XML Schema), est la suite de WSDL-S. Il a été conçu pour spécifier les éléments de WSDL à l'aide des ontologies de manière plus standardisée. Le mécanisme d'annotation de SAWSDL est décrit de manière indépendante de tout langage de représentation d'ontologies.

L'approche SAWSDL permet d'ajouter facilement une description sémantique au langage WSDL. L'avantage de cette initiative par rapport au WSDL-S est la standardisation ainsi qu'elle permet d'annoter les WSDL existants sans trop alourdir le fichier de description initiale.

✦ Le web sémantique est une vision vers l'automatisation de la gestion des ressources Web. Son intersection avec la technologie des services Web donne naissance aux services Web sémantiques, une nouvelle génération de services Web caractérisés par leur intelligence. Nous nous intéressons alors à la composition de ce type évolué de service, en utilisant notamment les langages de modélisation sémantiques (OWL-S, WSMO…) et les ontologies.

2.4. Les Systèmes Multi-Agents : vers une intelligence collective

L'approche multi-agents se situe au carrefour de nombreuses disciplines, notamment l'intelligence artificielle, les systèmes informatiques distribués et le génie logiciel. En effet, les SMA ont pris avec le temps une place importante dans le domaine de l'IA collective ou encore de l'IAD : intelligence artificielle distribuée [Bond et al., 1988]. Issus de la recherche

sur les mécanismes d'auto-organisation et sur la réalisation d'artefacts distribués capables de résoudre un problème complexe par interaction et coopération, les SMA ont su imposer leur nouvelle approche : des activités complexes peuvent être réalisées par la concrétisation de l'interaction d'entités autonomes.
Dans la suite nous exposons les principales notions liées à cette approche à savoir : les agents leurs classes et les types d'interactions dans un SMA.

2.4.1. Définition d'un agent

Les SMA sont caractérisés par l'autonomie et le comportement, plus ou moins, intelligent de leurs composants : les agents.

De nombreuses définitions existent pour définir un agent de par la disparité des domaines de recherche dans lesquels il est employé. Une revue littéraire de ces définitions permet, cependant, de dégager des caractéristiques, dites « primaires », qui semblent être indispensables pour pouvoir qualifier un agent d' « intelligent », une définition générale commune peut être, approximativement, la suivante [Ferber, 95]:

On appelle agent, une entité physique ou abstraite :

- *qui est capable d'agir dans un environnement ;*
- *qui peut communiquer directement avec d'autres agents ;*
- *qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser) ;*
- *qui possède des ressources propres ;*
- *qui est capable de percevoir (mais de manière limitée) son environnement ;*
- *qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune) ;*
- *qui possède des compétences et offre des services ;*
- *qui peut éventuellement se reproduire ;*
- *et dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit.*

Suite à cette définition, la caractéristique fondamentale d'un agent apparaît : l'autonomie. Cela signifie que les agents ne sont pas dirigés par des commandes venant de l'utilisateur (ou d'un autre agent), mais par un ensemble de buts individuels à satisfaire. L'agent est ainsi une sorte « d'organisme vivant » dont le comportement, qui se résume à communiquer à agir et, éventuellement, à se reproduire, vise à la satisfaction de ses besoins et de ses objectifs à partir de tous les autres éléments (perceptions, représentations, actions, communications et ressources) dont il dispose.

2.4.2. Classification des agents

Les agents peuvent être classés en deux catégories principales selon leur comportement et leur granularité. Cette notion de granularité est évidemment subjective, elle exprime la complexité de raisonnement d'un agent afin de séparer les agents dits intelligents des agents moins intelligents. On parle d'agents cognitifs et d'agents réactifs. Il s'agit là de deux écoles de pensée et d'un réel clivage. La première, celle des agents cognitifs, considère qu'un agrégat de ce type d'agents peut être quasiment assimilé à des systèmes experts distribués. La seconde école, celle des agents réactifs, ne suit pas le paradigme cognitiviste et considère qu'un agent de ce type, ne possède pas de vision globale du système mais il répond par des actions à des stimuli.

- **Agents Cognitifs**

Les systèmes d'agents cognitifs sont fondés sur la coopération d'agents capables d'effectuer des opérations complexes. Un système cognitif comprend un petit nombre d'agents qui disposent d'une capacité de raisonnement sur une base de connaissances, d'une aptitude à traiter des informations diverses liées au domaine d'application, et d'informations relatives à la gestion des interactions avec les autres agents et l'environnement. Un agent cognitif se distingue par une représentation explicite de l'environnement, possède des croyances sur les autres, des intentions (i.e des buts et des plans explicites lui permettant d'accomplir leurs buts) et des compétences. Les tâches demandées à des agents cognitifs relèvent, a priori, d'un raisonnement plus compliqué que celui des agents réactifs.

- **Agents Réactifs**

Les raisonnements réflexes ou réactifs caractérisent une activité de l'agent dans laquelle les actions sont exécutées rapidement. Un agent de ce type évolue parmi un nombre important de ses semblables et c'est le résultat des interactions entre leurs activités qui donne une impression de comportement global intelligent. Les agents réactifs sont de plus bas niveau, ils ne disposent que d'un protocole et d'un langage de communication réduit, leurs capacités répondent uniquement à la loi « stimulus/action ». Ils ne possèdent pas de représentations de leur environnement. De ce fait toutes les informations relatives à leur comportement se trouvent dans l'environnement et leurs réactions dépendent uniquement de la perception qu'ils peuvent en avoir. Ils opèrent dans l'ici et le maintenant, c'est à dire qu'ils ne mémorisent pas les événements passés et ne peuvent pas non plus anticiper sur le futur. Parmi les agents qui s'adaptent plus avec cette école est l'agent ARD [Berrada et al. , 2006].

La distinction que l'on peut faire entre agent cognitif et un autre réactif tient essentiellement de la représentation du monde dont dispose l'agent. Si l'individu est doté d'une représentation symbolique du monde à partir de laquelle il est capable de formuler des raisonnements, on parlera d'agent cognitif tandis que s'il ne dispose que d'une représentation symbolique, c'est-à-dire limitée à ses perceptions, on parlera d'agent réactif. Comme déjà cité, cette distinction cognitif/réactif correspond à deux écoles de pensée des systèmes multi-agents. La première soutient une approche de famille d'agents intelligents devant collaborer, avec une perspective plus sociologique. La deuxième étudie la possibilité de l'émergence d'un comportement

intelligent d'un ensemble d'agents non-intelligents (type fourmis). Le Tableau 2. 3 résume les différences entre les deux classes d'agents

Agent cognitif	**Agent réactif**
Représentation explicite de l 'environnement	Pas de représentation explicite
Peut tenir compte de son passé	Pas de mémoire de son historique
Agent complexe	Fonctionnement stimulus/réponse
Petit nombre d'agents	Grand nombre d'agents
Planification des réactions	Pas de planification de réactions

Tableau 2. 3: Comparaison entre agent cognitif et agent réactif [Berrada, 2008]

❖ **Agents hybrides**

Chacune des classes précédentes est appropriée à un certain type de problème. Cependant, pour la majorité des problèmes, ni une architecture complètement réactive, ni une architecture complètement cognitive n'est appropriée. Dans ce cas, un agent conciliant à la fois des aspects réactifs et cognitifs est requis. On parle alors d'agents hybrides.

2.4.3. Interaction entre agents

Un système multi-agents est un système distribué composé d'un ensemble d'agents qui interagissent entre eux dans un environnement commun selon des modes d'interaction [Jarras et al., 2002].

Selon [Ferber, 1995] l'interaction se définit ainsi « une interaction est la mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques». Les interactions entre agents peuvent varier selon les situations dont se trouvent ces agents : coexistence, compétition ou coopération.

- S'ils ne font que **coexister**, alors chaque agent ne considère les autres agents que comme des composants de l'environnement. Il n'y a aucune communication directe entre les agents.
- S'ils sont en **compétition**, alors le but de chaque agent est de maximiser sa propre satisfaction. La compétition entre agents peut avoir plusieurs sources : Les buts des agents peuvent être incompatibles ou les ressources peuvent être insuffisantes.
- S'ils sont en **coopération**, alors le but des agents n'est plus seulement de maximiser leurs propres satisfactions mais aussi de contribuer à la réussite du groupe. Les agents travaillent ensemble à la résolution d'un problème commun. La coopération est nécessaire quand un agent ne peut pas atteindre ses buts sans l'aide des autres agents.

Les formes d'interactions sont :

(a) Coordination: Il y a interaction entre les agents soit parce qu'ils coopèrent, soit parce qu'ils sont en compétition. Dans les deux cas, une coordination peut être nécessaire pour améliorer le fonctionnement global du système.

Lorsque plusieurs agents travaillent sur le même lieu, utilisent les mêmes ressources, où résolvent des sous problèmes qui ne sont pas complètement indépendants (conception d'un

objet complexe par exemple), ils doivent accomplir, en plus des tâches liées directement au problème traité, des tâches de coordination. Ces tâches ne sont pas directement productives mais elles améliorent les tâches productives.

Les tâches de coordination peuvent être accomplies directement par les agents concernés quand elles sont relativement rares et qu'elles n'engagent pas un grand nombre d'agents en même temps. Sinon, elles sont prises en charge par des agents spécialisés qui recueillent les demandes et fixent les ordres de priorité ou d'autres contraintes.

(b) Négociation: Comme nous avons vu précédemment, en interagissant dans un environnement partagé, les agents doivent coordonner leurs actions et avoir des mécanismes pour la résolution des conflits. Le mécanisme favori pour la résolution des conflits et la coordination, inspiré du modèle des humains, est **la négociation**.

Dans les systèmes multi-agents, la négociation est une composante de base de l'interaction surtout parce que les agents sont autonomes ; il n'y a pas de solution imposée à l'avance et les agents doivent arriver à trouver des solutions dynamiquement, pendant qu'ils résolvent les problèmes.

(c) Communication: Les communications, dans les systèmes multi-agents comme chez les humains, sont à la base des interactions et de l'organisation sociale. Sans communication, l'agent n'est qu'un individu isolé, renfermé sur sa boucle perception-délibération action. C'est parce que les agents communiquent qu'ils peuvent coopérer, coordonner leurs actions, réaliser des tâches en commun et devenir ainsi de véritables êtres sociaux.

Dans les SMA deux stratégies principales sont utilisées pour supporter la communication entre agents:

- **Communication par transfert de messages :** Dans cette approche, les agents échangent de messages entre eux directement (pas de mémoire partagée).
- **Communication par l'utilisation d'un tableau noir :** Un tableau noir est utilisé pour spécifier une mémoire partagée par divers systèmes. Dans un SMA utilisant un tableau noir, les agents peuvent écrire des messages, insérer des résultats partiels de leurs calculs et obtenir de l'information dans et à partir de ce tableau.

Pour coordonner l'activité d'un ensemble hétérogène d'agents autonomes, il faut que les agents communiquent dans un langage compréhensible par tous les autres. Les deux langages les plus utilisés sont : **KQML** [Patil et al., 1992] et **FIPA-ACL** [Poslad,2006].

✦ La discipline qui étudie les SMA s'intéresse en particulier aux comportements collectifs produits par les **interactions** de plusieurs entités autonomes et flexibles appelées **agents**.

Dans le cadre du traitement de notre problématique, les caractéristiques du SMA rendent ce type d'organisation particulièrement intéressant, principalement pour les raisons suivantes [Jarras et al., 2002]:

- dans un SMA, chaque agent possède des informations ou des capacités de résolution de problèmes limitées, ainsi chaque agent a un point de vue partiel ;
- les données sont décentralisées ;
- le calcul est asynchrone.

- certains domaines requièrent l'utilisation de plusieurs entités, par exemple, il y a des systèmes qui sont géographiquement distribués. Les SMA procurent une façon facile et efficace de les modéliser.
- une autre situation, où ils sont requis, est lorsque les différents systèmes et les données qui s'y rattachent appartiennent à des organisations indépendantes qui veulent garder leurs informations privées et sécurisées pour des raisons concurrentielles.
- les SMA possèdent également les avantages traditionnels de la résolution distribuée et concurrente de problèmes:
  - la *modularité*, permet de rendre la programmation plus simple ;
  - la *vitesse*, est due principalement au parallélisme ;
  - la *fiabilité*, qui peut être également atteinte, dans la mesure où le contrôle et les responsabilités étant partagés entre les différents agents, le système peut tolérer la défaillance d'un ou de plusieurs agents.

2.5. Planification : raisonner avant d'agir

La **planification** est une discipline de l'intelligence artificielle ayant une grande importance dans plusieurs domaines et surtout celui de la robotique. Elle vise le développement d'algorithmes pour produire des plans, typiquement pour l'exécution par un robot ou tout autre agent. Les logiciels de planification qui incorporent ces algorithmes se nomment **planificateurs**. Il existe plusieurs types de planificateurs implémentant des algorithmes différents, cette section expose l'état de l'art de la planification et de ses algorithmes les plus utilisés.

2.5.1. Définition et aspects

Planifier consiste à adopter un ensemble de dispositions en vue de la réalisation d'un projet. Il s'agit de raisonner avant d'agir. Pour un robot, planifier revient à synthétiser une trajectoire dans un espace de recherche abstrait - l'espace des états accessibles dans une certaine représentation - en faisant des prédictions sur des évolutions possibles, en choisissant et en organisant des actions en vue de réaliser un but fixé ou de satisfaire au mieux une fonction d'utilité donnée [Ghallab et al., 2004]. Cette approche diffère, mais elle est complémentaire, avec les approches d'apprentissage pour lesquelles le comportement des agents est appris automatiquement. Elle diffère aussi des approches de programmation habituelles pour lesquelles le comportement est codé d'avance.
La planification s'appuie sur :

- des modèles de l'environnement et du robot, en particulier de ses moyens d'action,
- la spécification des buts ou des fonctions d'utilité que l'on veut satisfaire,
- les entrées fournies en ligne par les capteurs et les canaux de communications.

La planification en robotique prend plusieurs formes [Helmert, 2008], le tableau ci-après décrit les caractéristiques des différents types de planification.

Forme de planification	Caractéristiques
Planification du mouvement	Il s'agit de synthétiser un chemin et une trajectoire dans l'espace des configurations du robot, en prenant en compte principalement la géométrie et la cinématique.
Planification sensorielle	Dans ce type de planification on veut répondre à des questions du type : quelle information est nécessaire pour la tâche en cours et à quel moment, comment peut-on l'acquérir et où, avec quels capteurs, selon quel point de vue, avec quelles modalités sensorielles.
Planification pour la navigation	Il s'agit dans ce type de planification de choisir et d'organiser un ensemble de primitives de localisation et de primitives de mouvement asservi sur des capteurs extéroceptifs visuel en vue d'atteindre un but ou d'explorer un environnement. Il s'agit par exemple de primitives de suivi d'une route, de contournement d'un obstacle, de poursuite d'une cible par asservissement. Cette forme de planification combine d'une certaine façon les deux précédentes.
Planification pour la manipulation	Comme le type précédent, l'objectif principalement dans ce type de planification est d'élaborer une stratégie avec des primitives de prise, de manipulation, de reconnaissance d'objets et d'assemblages en utilisant des retours sensoriels sur les contacts, les forces, les couples, la texture du touché (capteurs haptiques), ou des capteurs généraux (vision);
Planification pour la communication	Requise pour l'interaction avec l'homme et en cas de coopération ou de coordination multirobots. Il s'agit de déterminer et d'organiser les requêtes et les retours d'interaction pour l'activité en cours
Planification de tâches	La forme la plus générale et la plus abstraite. On cherche à déterminer et à organiser l'ensemble des activités du robot dans le temps et à leur attribuer des ressources, compte tenu des évolutions prévisibles dans l'environnement.

Tableau 2. 4: Les formes de planification

2.5.2. Modèle et spécifications

2.5.2.1. Modèle formel

Le cadre formel de référence auquel on fait généralement référence pour définir la sémantique des représentations utilisées en planification, est celui des graphes de transition d'états étiquetés [Ghallab, 2001]. C'est un quadruplet G = (S, A, E, δ) où :

- $S = \{s_1, s_2, ..., s_n\}$ est un ensemble fini et récursivement énumérable d'états ;
- $A = \{a_1, a_2, ..., a_n\}$ est un ensemble fini et récursivement énumérable d'actions ;
- $E = \{e_1, e_2, ..., e_n\}$ est un ensemble fini et récursivement énumérable d'événements ;
- $\delta: S \times A \times E \rightarrow 2^s$ est une fonction de transition entre états. Elle est définie par : si a est une action et δ(s, a) est non vide alors a est applicable à l'état s.

Ce système de transition d'états peut être représenté par un graphe direct dont les nœuds sont les états de S. Les arcs sont appelés transitions d'états.

Planifier dans ce cadre revient à trouver un chemin dans G entre un état initial et un but [Ghallab, 2001]. Ayant un problème spécifié par G, un état initial $s_0 \in S$, et un sous-ensemble d'états buts $s_g \in S$, on cherche une séquence d'actions (a_1, a_2, ..., a_n), telle que: $a_n(...a_2(a_1(s_0))..)$ soit un état de s_g .

2.5.2.2. Représentations et langages

Pour fonctionner, tout algorithme de planification a besoin d'une description de l'univers (représentation) sous une forme facilement compréhensible. Le choix d'une représentation adaptée aux problèmes à résoudre est un des éléments les plus importants pour la conception d'un système efficace. De nombreuses approches peuvent être employés parmi elles on trouve [Ghallab, 2001] :

- **La représentation dans la théorie des ensembles :** chaque état du monde est un ensemble de propositions et chaque action est une expression spécifiant les propositions qui doivent appartenir à l'état courant pour que l'action puisse être exécutée, ainsi que celles qui seront ajoutées et enlevées de l'état suite à l'exécution de l'action ;
- **La représentation classique :** contrairement à la représentation précédente, des prédicats du premier ordre et des connecteurs logiques sont utilisés à la place des propositions ;
- **La représentation par variables d'états :** chaque état est représenté par un tuple de n variables d'états valuées $\{x_1, x_2, ..., x_n\}$ et chaque action est représentée par une fonction partielle qui permet de passer d'un tuple à un autre tuple de variables d'états instanciées.

Plusieurs langages de description des domaines et des problèmes de planification sont apparus dans l'optique de mettre en place un langage de planification qui est à la fois suffisamment expressif pour décrire une grande variété de problèmes mais assez restrictif pour permettre à des algorithmes efficaces d'agir, on retrouve ainsi :

- STRIPS [Fikes et al. , 1971] le langage de représentation qui eut bien plus d'influence que sa méthode algorithmique: ce qu'on appelle le langage « classique ».
- ADL (action description langage) [Pednault ,1989] une extension significative en planification classique permet de relaxer certaines restrictions du langage STRIPS et d'encoder des problèmes plus réalistes.
- Enfin PDDL (Planning Domain Definition Langage) [McDermott, 1998] est une tentative pour standardiser les langages de description des domaines et des problèmes de planification. C'est un langage créé en 1998 par Drew McDermott pour la compétition IPC-98 dans le but de développer un seul langage de description des domaines de planification utilisé par tous les planificateurs de cette compétition. Il est ensuite devenu comme un standard utilisé par tous les planificateurs.

2.5.3. Principaux algorithmes et planificateurs

Il existe une pléthore d'algorithmes qui s'adresse au problème de planification**.** Selon [Ghallab et al., 2004] ces techniques peuvent être regroupées en classes et sous classes selon le tableau ci-dessous :

Classe	**Sous-classe**
1. Planification classique	• Planification dans un espace d'états • Planification dans un espace de plans
2. Planification non classique	• Planification fondée sur les graphes (Graphplan) • Les techniques de satisfaction des contraintes
3. Les stratégies heuristiques de contrôle	• Les heuristiques indépendantes du domaine • Règles de contrôle dans la planification • Réseaux hiérarchisés de tâches • Situation calculus • La logique dynamique
4. Planification avec temps et ressources (extension de Planification non classique)	• La planification temporelle • Planification avec des ressources
5. Planification sous l'incertitude	• Planification basée sur MDP (Markov Decision Processes) • Planification basée sur le modèle de checking • Incertitude avec les techniques non classiques
6. Autres approches de planification	• Planification basée des cas • Analyse du domaine • Fusion et réécriture des plans

Tableau 2. 5: Classes des algorithmes de planification

✦ Comme détaillé dans la section 2.2.2 du chapitre précedent, jusqu'à présent ce sont les techniques de la classes 1,3 et 5 qui sont les plus utilisées dans la composition des services Web, nous proposons au niveau de l'annexe A de s'approcher davantage de certaines techniques de ces classes afin de découvrir leur mécanisme de fonctionnement, c'est également l'occasion de détailler quelques concepts communs qui serons utilisés par l'approche de planification que nous adapterons .

2.6. L'Autonomic Computing: les systèmes autogérés

L'informatique autonomique où l'«Autonomic Computing » est apparue dans un contexte centré sur la problématique de l'administration de larges systèmes, elle se propose de rendre les systèmes informatiques capables de s'autogérer, de façon à minimiser le besoin d'intervention humaine [Horn, 2001]. Cette approche est motivée par le constat suivant : la complexité des systèmes informatiques a dépassé notre seuil de compétence, ils doivent donc être capables de se gérer de façon autonome ; Ce n'est pas l'homme qui doit s'adapter aux machines pour les utiliser mais elles doivent évoluer en fonction des enjeux.

C'est dans [Kephart et al. , 2003] où le concept d'informatique autonome a été remarquablement développé. Dans l'ouvrage, Kephart s'inspire du système nerveux humain qui permet par exemple à chacun de faire des efforts physiques sans avoir à se soucier du calcul du rythme cardiaque, de la pression sanguine ou de la quantité d'adrénaline, et de prendre des décisions nécessaire dans certain cas s'il le faut. Kephart affirme ainsi qu'un futur système autonome agirait de même, permettant de surpasser sa complexité et offrant aux administrateurs le temps pour se focaliser sur des problèmes de haut niveau. Il définit alors l'AC en mettant en avant l'apport de l'informatique autonomique dans les phases d'exploitation et de maintenance du cycle de vie des systèmes. « The essence of autonomic computing systems is self-management, the intent of which is to free system administrators from the details of system operation and maintenance and provide users with a machine that runs at peak performance 24/7 » [Kephart et al. , 2003].

2.6.1. Les propriétés auto-*

Dans son manifeste de 2001, IBM a énoncé plusieurs points essentiels qui, ensemble, caractérisent les systèmes autonomiques [Horn, 2001]. Ces critères sont usuellement regroupées sous le terme générique auto-* (Fig.2. 5) :

i. **Auto-configuration** (self-configuring) : tout système autonomique doit se reconfigurer continuellement afin de s'adapter aux variations de son contexte [Horn, 2001]. L'environnement d'exécution du système peut subir des modifications aussi diverses qu'imprévisibles. Il est donc nécessaire, en vue de continuer à satisfaire les buts de haut-niveau fixés par l'administrateur, que le système puisse changer sa propre configuration. En pratique, cela peut se manifester par l'ajout ou la suppression de composants, le déploiement de composants sur de nouvelles plateformes matérielles, etc. L'adaptation est la clé de voûte de l'évolution du système, et va donc lui permettre de s'organiser, de se réparer et de se protéger face aux changements de son milieu de vie.

ii. **Auto-optimisation** (self-optimizing): le système autonomique doit pouvoir utiliser au mieux les ressources dont il dispose [Horn, 2001]. Il doit donc en permanence considérer son propre état afin de déterminer comment il peut s'améliorer. Le système autonomique doit s'auto-évaluer en fonction de plusieurs critères de qualité, tels que la consommation mémoire, le temps de réponse, la consommation en énergie, et en fonction de son contexte d'exécution actuel : disponibilité des ressources, taux d'utilisation des ressources, point de congestion, etc. La décision d'optimiser tout ou partie du système revient donc à analyser ces différents critères, et à faire des compromis, principalement en fonction des buts de haut-niveau exprimés par l'administrateur. Par exemple, un système autonomique sur une plateforme mobile favorisera l'amélioration des performances lorsqu'il sera branché à une source de courant, mais privilégiera l'économie d'énergie lorsqu'il fonctionnera sur batterie.

iii. **Auto- guérison** (self-healing) : lorsqu'un dysfonctionnement survient au sein d'un système autonomique, ce dernier doit être en mesure de déceler le problème, de trouver une solution pour le résoudre, et de l'appliquer [Horn, 2001]. La robustesse du système autonomique doit lui permettre de faire face aux défaillances, en les détectant et en les corrigeant. Le dynamisme et l'hétérogénéité de l'environnement font qu'il est difficile de

prendre en compte toutes les situations possibles. Face à un état non-prévu ou incohérent occasionnant une défaillance, un système autonomique doit être en mesure d'empêcher cette dernière de mettre hors service la partie incriminée du système, ou de se propager à l'ensemble du système. Cette propriété caractérise donc la robustesse et la tolérance aux fautes du système autonomique.

iv. **Autoprotection** (self-protecting)**:** Le système autonomique doit être en mesure de se protéger contre les éventuelles menaces de l'environnement extérieur aussi bien que de son contexte interne [Horn, 2001]. De telles menaces peuvent être de différentes natures : causées par un environnement devenu hostile, par des utilisateurs maladroits ou malveillants, voir une partie du système devenant incontrôlable (bug, virus, porte dérobée). L'autoréparation a pour but de résoudre ces problèmes lorsqu'ils sont rencontrés, mais, comme le dit l'adage, « mieux vaut prévenir que guérir ». Un système autonomique doit donc anticiper les causes possibles de dysfonctionnement et doit mettre en place des mécanismes de défense appropriés. Lors de la découverte d'une faille de sécurité, le système pourra par exemple automatiquement télécharger et remplacer le logiciel impliqué.

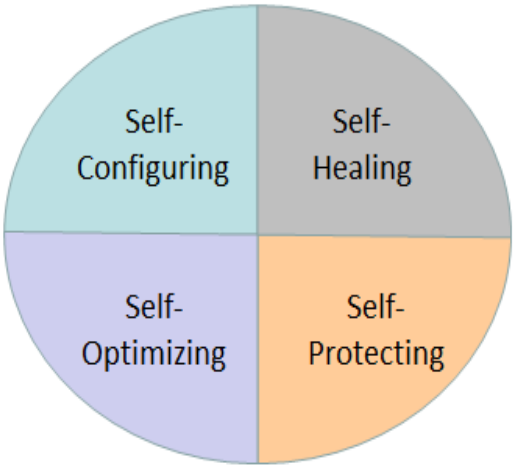


Fig.2. 5: Les propriétés auto-* de l'Autonomic Computing

Ces quatre propriétés fondamentales des systèmes autonomiques sont couramment appelées les propriétés CHOP (Configuring-Healing-Optimizing-Protecting). Ces propriétés ont été complétées par [Sterritt et al. , 2003] d'autres attributs comme l'auto-connaissance, l'auto-surveillance et l'auto-réglage. En fait, depuis l'origine du manifeste d'IBM en 2001[Horn, 2001], la liste des propriétés des systèmes autonomiques n'a cessé de croître en fonction des besoins des applications et des objectifs [Tianfield, 2003] [Sterritt, 2005]. Cette liste de propriétés, appelées propriétés auto-* étendues (extended self-* properties) n'est pas exhaustive : un nombre toujours croissant de caractéristiques auto-* sont mises en évidence. Les critères caractérisant les systèmes autonomiques font souvent débat, chacun souhaitant ajouter ou mettre en avant des propriétés parmi d'autres. Il faut cependant noter qu'il est possible d'exprimer ces propriétés en fonction d'une ou plusieurs des quatre propriétés auto-* fondamentales, ou inversement.

Dans la section suivante, nous allons décrire comment les propriétés de l'AC sont mises en pratique et avec quels impacts sur l'architecture des systèmes.

2.6.2. Structures des systèmes autonomiques : la boucle de contrôle autonomique

Dans cette section nous allons voir quelles sont les approches architecturales utilisées pour concevoir un système afin de lui donner le caractère autonomique. Les différentes études et implémentations des systèmes autonomiques et de leur élaboration ont mis en avant un modèle de référence pour la boucle de contrôle autonomique en informatique [Kephart et al., 2003] ,qui est souvent appelée la boucle MAPE-K (Monitor, Analyse, Plan, Execute, Knowledge). Ce modèle est de plus en plus utilisé pour détailler les aspects architecturaux des systèmes autonomiques. Il est illustré au niveau de la Fig.2. 6 et est similaire au modèle générique d'agents [Russell et al. , 2009], dans lequel un agent intelligent perçoit son environnement à travers des capteurs et utilise sa perception pour déterminer des actions à exécuter sur l'environnement.

Dans la boucle MAPE-K, les éléments gérés représentent des ressources logicielles ou matérielles auxquels un comportement autonomique est fourni en les couplant avec un gestionnaire autonomique. Les capteurs, aussi appelés senseurs ou sondes, collectent des informations, aussi appelées métriques, sur les éléments gérés. Les actionneurs prennent en charge les changements à effectuer sur les éléments gérés. Ces changements peuvent être vus à deux granularités différentes : le niveau gros grain par le changement de plusieurs éléments en même temps ou le niveau petit grain par le changement de paramètres internes à un élément. La distinction architecturale de ce gestionnaire autonomique contribue à une complexification du système. Au terme de la maturité de l'administration autonomique, cette distinction s'effacera pour devenir presque conceptuelle plus qu'architecturale.

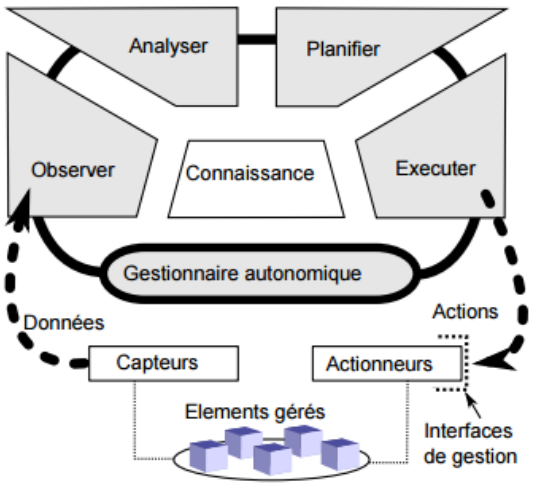


Fig.2. 6: Boucle de contrôle autonomique MAPE-K [Sharrock, 2010]

Comme illustré dans la Fig.2. 6 la boucle MAPE-K est composée des éléments suivants :

- **Le gestionnaire autonomique :** les informations collectées par les capteurs permettent au gestionnaire autonomique d'observer les éléments gérés et d'exécuter les changements au travers des actionneurs. Ce gestionnaire autonomique est un composant logiciel qui est idéalement configuré par des administrateurs humains en utilisant une description de haut niveau de l'expression de leur stratégie (les politiques). Le gestionnaire utilise ensuite les informations des capteurs et sa connaissance interne pour planifier et exécuter, en fonction des politiques décrites, toutes les actions de bas niveau nécessaires pour atteindre l'objectif de la stratégie.
- **L'observation ou « Monitoring » :** la tâche d'observation de la boucle MAPE-K apporte des mécanismes de capture de certaines propriétés de l'environnement. Pour cette tâche on utilise les verbes sonder ou par abus de langage « monitorer » (issu de l'anglais « Monitoring »). Ces propriétés sont à priori toutes importantes pour la gestion autonomique du système. Le gestionnaire autonomique a besoin d'informations appropriées pour reconnaître le fonctionnement anormal ou la sous-performance d'éléments du système et a besoin de savoir interpréter ces propriétés, d'où une éventuelle mise en forme des informations remontées par les sondes (ou capteurs). En général on parle d'observation active et passive ou « monitoring » passif et actif. Le « monitoring passif » n'a pas besoin de modifier le fonctionnement interne du système observé (par exemple, le rajout d'un code spécifique supplémentaire dans un logiciel). Il observe les interactions du système et ses états en étant le moins intrusif possible, c'est à dire en évitant au maximum de perturber le fonctionnement normal du système. L'observation ou « monitoring actif » fait appel à un certain niveau à des techniques d'ingénierie qui modifient ou rajoutent du code ou des instruments à l'implémentation des éléments du système. Par exemple, la capture des appels de fonctions pour une application ou la capture des appels de bas niveau pour le système d'exploitation. Le surplus de code pour le « monitoring » actif peut souvent être rajouté automatiquement par des outils spécifiques.
- **L'analyse :** La tâche d'analyse de la boucle de contrôle autonomique MAPE-K prend en compte les données d'observation fournies par les sondes pour les comparer suivant les politiques et les stratégies globales de gestion. Cette comparaison permet de déterminer une différence entre l'état d'exécution et d'opération effectif du système et celui attendu. Cette tâche fournit donc des indicateurs quantitatifs et qualitatifs de l'opération du système.
- **La planification :** La tâche de planification de la boucle de contrôle autonomique produit une série de changements à effectuer sur les éléments gérés. Ces changements dépendent de la comparaison effectuée par la tâche précédente d'analyse. Ils sont une fonction des différences de fonctionnement réel et attendu. Cette tâche planifie aussi de manière temporelle les changements à apporter au système. En effet, il peut y avoir des contraintes temporelles retardant les changements ou des priorités à prendre en compte. Parmi les contraintes temporelles notons celle induite par le temps d'exécution total de la boucle de contrôle autonomique.
- **L'exécution :** La tâche d'exécution se charge de respecter la planification des changements et d'exécuter en temps voulu les actions nécessaires sur les éléments gérés. Cette tâche fait le lien entre les calculs de la boucle de contrôle et le monde réel. Elle peut être en charge de

distribuer des actions sur plusieurs machines si un changement fait intervenir plusieurs éléments gérés.

- **La base de connaissances** : Reliant les différentes tâches de la boucle MAPE-K, la base de connaissances K peut provenir de différentes sources collectant l'information de manière automatique ou même manuelle. Il peut s'agir d'historiques de fonctionnement, d'observations du comportement du système, de données accumulées et stockées (par exemple les logs des sondes) ou même d'informations rajoutées par des experts humains pouvant aider le gestionnaire autonomique. Les quatre tâches peuvent venir chercher des informations ou rajouter des informations dans cette base.

- L'Autonomic Computing est un nouveau paradigme visant à rendre les systèmes informatiques capables de s'autogérer, l'autogestion dans le contexte de l'AC se projette sur quatre axes : l'auto-configuration, l'auto-optimisation, l'auto- guérison et l'autoprotection.
  Il est important de souligner à ce niveau que l'informatique autonomique ne peut pas se résumer à insuffler ces quatre propriétés aux systèmes. L'AC est une vision plus générale qui permet, en se basant sur une architecture typique (la boucle de contrôle autonomique), d'agencer les systèmes, aussi variés soient-ils, pour qu'ils puissent exposer les propriétés auto-*.
  Il serait alors intéressant d'étudier en détail cette architecture afin de l'adapter au contexte de la CSW en vue de générer un processus de composition autonomique.

2.7. La cinquième force

En considérant les aspects présentés ci-avant, nous croyons qu'une approche qui réunit les quatre axes de recherches pré-étudiés constitue une réponse efficace au besoin de la composition automatique et dynamique de services Web, en effet :

- **Force 1 :** Le Web sémantique est une vision d'un Web intelligent ou les ressources ont un sens compris, aussi bien par la machine que par l'humain, dans un tel contexte l'agent logiciel devient conscient de ce qu'il est, de ce qu'il peut faire et de ce que les autres peuvent faire et, n'aura donc plus besoin d'être guidé par l'agent humain. C'est pour cela que là où il y a besoin d'automatisation de processus liés aux ressources web, le Web sémantique peut faire preuve d'efficacité ; le domaine de la recherche d'information (Information retrieval) [Manning et al. ,2009] est peut-être celui le plus concerné par les progrès du Web sémantique surtout pour les volets précision et complétude [Gagnon ,2013]. A côté on trouve également le domaine de maintenance des bases documentaires et celui du commerce électronique. Enfin dans le domaine des services immatériels le Web sémantique fait espérer une véritable orchestration automatisée de processus très courants dans le monde de l'industrie et du commerce. C'est la filière du Services Web Sémantiques qui est ici concernée et c'est justement l'aspect qui nous intéresse le plus.
  La description sémantique des interfaces et des fonctionnalités d'un service Web lui permet d'être automatiquement découvert, composé et invoqué. Annoter sémantiquement les services gérés en utilisant l'approche de modélisation (OWL-S, WSMO, SA-WSDL) la plus adaptée est donc à envisager.

L'autre concept que nous empruntons au domaine du WS est celui d'ontologie, dans le cadre du processus de composition nous allons faire appel à un assemblage de sources d'origine très diversifié et donc très hétérogène dans leur description. On imagine alors qu'une intégration aisée se réalise via une couche d'abstraction sémantique sous formes d'ontologies décrivant les concepts liés au domaine, aux services et à l'utilisateur ainsi que les relations qui les relis, donnant ainsi une vue unifiée aux intervenant dans ce processus.

Ainsi nous espérons surtout trouver une solution à l'automatisation et à l'interopérabilité via l'usage des technologies promises du Web sémantique.

- **Force 2 :** D'un point de vue socio mimétique un système multi-agents est un regroupement d'agents capables de raisonner, d'agir et d'interagir. Deux concepts nous intéressent dans ce domaine dérivés de l'intelligence artificielle distribuée, d'abord la notion d'agent comme étant une entité individuelle dotée de plusieurs capacités avancées qui font d'elle une entité autonome, ensuite le problème social qui se résume en un seul mot : l'interaction. Il est à constater qu'au niveau architectural il y a une grande ressemblance entre un SMA, et l'environnement que nous étudions, à savoir un ensemble de services censés être suffisamment intelligents pour interagir afin de répondre à une demande complexe, c'est éventuellement cette ressemblance qui a motivée les travaux de composition des services web basé sur les SMA qu'on retrouve dans la littérature.

  Nous reconnaissons, nous aussi, le potentiel du SMA d'offrir les mécanismes nécessaires pour assurer une gestion décentralisée et distribuée de la composition dans un environnement aussi dynamique que celui des services Web et nous le retenons comme architecture de base de notre approche.

  Force 3 : La planification en intelligence artificielle est une approche de raisonnement qui permet à un agent de choisir ses actions en vue d'accomplir ses buts, ce n'est pas la seule approche algorithmique pour résoudre ce problème, mais c'est la seule qui propose une perception du problème, plus ou moins, identique à celle du problème étudié, à savoir : « Etant donnés: Un état initial du monde, la description d'un but et un ensemble d'actions susceptibles de changer le monde. Trouver: Une suite d'actions capable de changer l'état initial en un état satisfaisant le but ». Ce qui laisse penser que les algorithmes de planification peuvent constituer le mécanisme de raisonnement adéquat pour guider le comportement des agents à leur but -le service composé- . Sauf que la plupart des planificateurs existants (annexe A) se basent sur des algorithmes centralisés qui ne sont pas applicables dans notre contexte décentralisé. Ainsi la planification est retenue comme principe de base de raisonnement, mais aucun des algorithmes cités dans la section précédente ne sera utilisé, nous chercherons un algorithme de planification adapté à un contexte distribué, organisé sous forme d'un SMA et guidé par un objectif.

- **Force 4 :** L'ambition de ce travail est de proposer une approche dynamique et automatique de composition des services Web caractérisée par son aspect autonomique (an autonomic Web service composition approach), où on peut identifier les quatre propriétés auto fondamentales : (i) auto-configuration, (ii) auto-optimisation, (iii) auto-guérison et (iv) l'autoprotection. Une composition auto-

configurable serait de pouvoir automatiquement découvrir les participants et les sélectionner parmi les services disponibles. La composition auto-optimisée, quant à elle, sélectionne automatiquement les participants qui maximisent la qualité de service (QoS), alors qu'une composition qui se répare automatiquement détecte en temps réel que des critères/exigences ne sont plus vérifiés et réagit pour ajuster la situation. Finalement plus qu'une autoprotection, ce qui est intéressant d'étudier dans le contexte de CSW est plutôt l'auto-adaptabilité. La composition auto-adaptable continue à fonctionner malgré l'occurrence d'un changement dans le comportement de l'environnement et des services externes et ne nécessite pas l'intervention humaine pour adapter les services au changement.

Dans cette optique, les techniques de l'informatique autonomique semblent être l'outil privilégié pour atteindre cette ambition. En fait, l'informatique autonomique est essentiellement un concept architectural qui propose d'organiser les systèmes de façon à dégager un comportement autonomique « self » tout en minimisant le besoin d'intervention humaine. Puisque nous nous intéressons dans un premier temps au comportement du self-Healing, car la plus part des approches de compositions des services Web actuelles sont reprochées de ne pas être conçues pour résister dynamiquement aux erreurs, nous proposons alors d'adapter l'architecture proposée par l'AC en l'occurrence, le boucle MAPE-K pour assurer une sûreté de fonctionnement dynamique. Nous prenons également en charge le comportement de l'auto-optimisation et l'auto-configuration mais en se basant sur d'autres techniques (cf. chapitre 4 et 5).

Ainsi nous croyons que la conception d'une solution de composition autonomique des services Web sémantiques basée sur une architecture multi-agents en utilisant la planification comme modèle de raisonnement, est un compromis pertinent. L'architecture multi-agents est un pas vers une composition dynamique, décentralisée et scalable. L'utilisation des technologies du WS permet une composition automatique en se basant sur les descriptions formelles des ontologies. Les techniques de la PIA offrent un cadre de raisonnement adapté au problème de la composition. Et l'AC propose les propriétés à vérifier pour générer un comportement autonomique ainsi que l'architecture à adopter pour atteindre ce comportement.

Ce compromis génère une **cinquième force.** En effet, une caractéristique commune à laquelle contribuent et convergent les quatre axes, est **l'autonomie**. L'autonomie représente le caractère distinctif de l'approche et de la synergie escomptée de l'H4F. Cette synergie est illustrée dans la Fig.2. 7 qui décrit l'interconnexion entre les quatre domaines de base identifiés:

- un agent simule un SWS (1) et, les spécifications du SWS représentent la base de compétences de l'agent (2).
- les ontologies partagent les concepts et les données sémantiquement entre les différents agents (7).

- l'agent raisonne en se basant sur des algorithmes de planification, dans ce cas on parle de la planification multi-agents (3). Le problème de planification dans ce contexte devient donc « Etant donné une description de l'état initial, un ensemble de buts globaux, un ensemble d'agents , et pour chaque agent un ensemble de capacités et de buts privés, trouver un plan pour chaque agent qui réalise ses buts privés, tel que l'ensemble de ces plans sont coordonnables les uns aux autres et les buts globaux sont aussi bien atteints ».

- La propriété de l'autoréparation (l'auto-guérison) est déléguée à un des agents du SMA qui est considéré un agent autonomique (4), le mécanisme de l'autoréparation travaille sur les plans produits par les algorithmes de planification (5) pour les adapter (régler) (6).

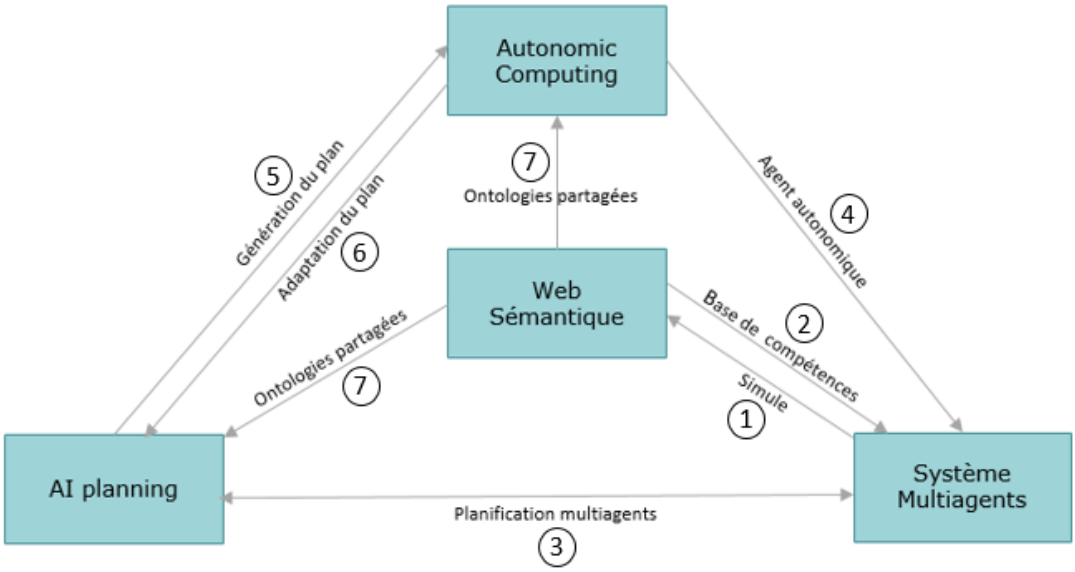


Fig.2. 7: Interactions entre les domaines de l'H4F [Adadi et al., 2014]

Cependant, l'utilisation de ces techniques issues de différents mondes pose de nouveaux problèmes et défis : comment transposer les modèles originaux d'une façon correcte et saine dans le monde de services Web et comment gérer leur diversité et leur interopérabilité. Par exemple, comment un agent qui ne comprend que des opérateurs en langage de modélisation de planification (PDDL, STRIPS..) puisse simuler un service dont les capacités sont décrites en langage sémantique (OWL-S, WSMO…). La réponse à ces questions constitue elle aussi l'une des contributions fondamentales de ce travail.

2.8. Conclusion

Convaincus qu'une démarche efficace de composition dynamique et automatique de services Web ne peut s'appuyer que sur un background pluridisciplinaire, nous avons exploré, dans ce chapitre, un existant riche venant du domaine de robotique, des technologies Web et d'architectures logicielles réparties, et ce dans une quête des sources méthodologiques susceptibles de nous aider à dresser un cadre théorique solide.

Ainsi le Web sémantique a été identifié comme la technologie adéquate pour assurer une composition automatique en se basant sur les descriptions formelles des ontologies. En plus, un système multi-agents dans lequel des agents raisonnent en utilisant des algorithmes de planification nous permettra de composer dans un environnement dynamique et décentralisé. Finalement les principes de l'Autonomic Computing semblent être adaptés pour générer le comportement autonomique souhaité.

Notre contribution, qui sera exposée dans les chapitres suivants, se concentre sur la construction d'une liaison covalente entre ces quatre domaines de base afin de proposer une approche de composition des services Web homogène et intégrée caractérisée par sa dynamique et son autonomie.

Deuxième partie

Contribution

Chapitre 3

Architecture proposée : iDyCoS

Comment peut-on percevoir sans concevoir ?
Emmanuel Kant

3.1. Introduction

L'aspect clé de la conception d'un système est son architecture. Une architecture permet d'exposer de manière compréhensible et synthétique la structure d'un système en termes de composants et d'interactions entre ces composants. Elle occupe ainsi une position centrale dans le processus de la construction de notre système de composition dynamique et automatique des services Web.

Guidé par une liste d'exigences architecturales, ce chapitre présente l'architecture sur laquelle se base notre système en adoptant une approche progressive qui, à partir d'une macro architecture, intègre les composants adéquats et définit les interactions nécessaires pour construire une architecture complète, valide et conforme.

3.2. Macro architecture

Notre contribution consiste à proposer une approche pour la composition dynamique et automatique de services Web basée sur l'hypothèse des quatre forces présentée dans le chapitre précèdent. Le système est supposé être une boite noire qui encapsule le processus de composition en prenant en entrée la requête de l'utilisateur et en lui retournant le service qui répond au mieux à sa demande d'une façon transparente. La description architecturale constitue un artefact central dans la conception d'un tel système.

En effet, la norme ANSI / IEEE 1471-2000 [IEEE, 2005] définit l'architecture d'un système comme étant « *le processus de description de la structure d'un système, telle qu'elle résulte de la donnée de ses composants et de leurs interfaces internes ainsi que des relations de ces composants avec leur environnement, dans l'ensemble de sa dimension temporelle (i.e. en allant de la conception à la gestion de l'évolution du système). Et ce dans le but de piloter efficacement sa mise en place* ». Ainsi, une description architecturale est principalement requise pour la spécification de la structure de notre système et la formalisation de l'approche proposée.

En partant d'un point de vue global (Fig.3. 1), l'architecture d'une plateforme de composition de services Web, doit comprendre principalement trois types de participants (acteurs): le fournisseur, l'utilisateur et le système de composition. L'utilisateur, qui peut être humain ou agent logiciel, exprime ses besoins via un intermédiaire (une interface graphique) (1), ces besoins incluent une requête exprimée en termes d'entrées/sorties et qui constitue le but pour le système de composition (2), ainsi que des éventuelles préférences (3). Le processus de composition se déclenche alors, il doit, théoriquement, commencer par la collecte des

services élémentaires auprès des fournisseurs, ensuite l'assemblement des services trouvés. Enfin, il doit retourner à l'utilisateur un résultat qui représente l'exécution du service composite formé (6).

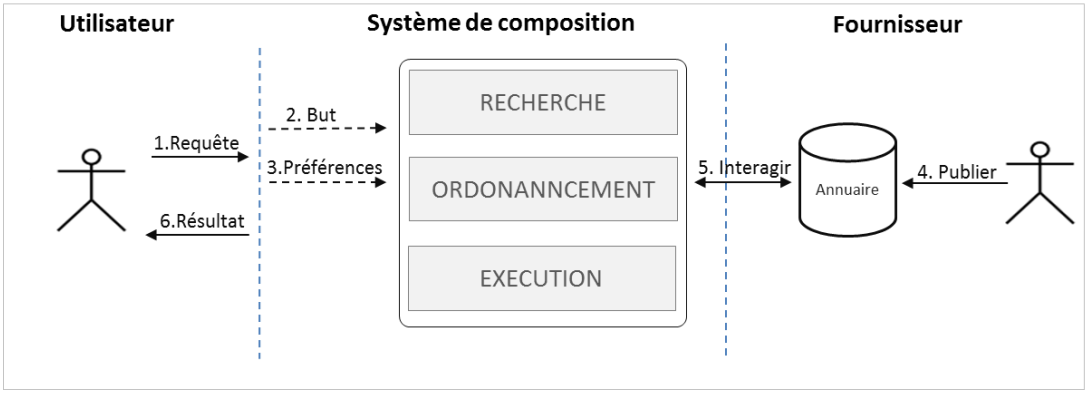


Fig.3. 1: Macro architecture du système de la CSW

L'architecture proposée ci-avant est loin d'être une architecture implémentable, c'est plutôt une tentative de décortication et d'analyse du problème que nous essayons de résoudre. En effet, nous avons trouvé qu'il est plus méthodique de commencer par une perception générale de la problématique sous forme d'une « macroarchitecture», qui s'intéresse à la définition du comportement global du système en faisant abstraction des détails d'implémentation et, d'essayer de l'affiner progressivement.

Ainsi l'analyse de cette abstraction de haut niveau du modèle de composition à concevoir, suscite quelques réflexions:

- il a été précisé dans la description précédente que le passage de l'environnement utilisateur à l'environnement système de composition est assuré par des simples interfaces graphiques. En prenant en considération que les langages de spécification utilisés par l'utilisateur ne sont pas compréhensibles par le système qui utilise ses propres langages de spécification internes, ce mode de communication devient alors insuffisant. L'architecture doit donc intégrer des composants assurant le rôle d'adaptateur ou de traducteur de requêtes.
- Par ailleurs, la collecte des services élémentaires est loin d'être une simple opération de recherche classique dans les annuaires des services Web disponibles. On ne cherche plus un seul service Web en se basant sur des critères syntaxiques, on est face à une requête complexe qui nécessite la combinaison de plusieurs services Web censés se reconnaitre entre eux. Des mécanismes de correspondance entre la requête et l'offre des services existants et entre les services disponibles eux-mêmes doivent alors être définis sur un plan sémantique.
- l'assemblement des services élémentaires, entraine évidement une gestion d'ordonnancement des opérations et d'interaction des services composants qui doit être prise en charge par l'architecture proposée.
- En cas de plusieurs alternatives d'assemblement, le système doit inclure les mécanismes d'évaluation nécessaires pour choisir la meilleure solution ou la solution la plus adaptée.

- Enfin l'exécution est supposée être dynamique et doit par conséquent gérer les problèmes d'incohérence et de disponibilité rencontrés en temps réel.

Ces axes de réflexion vont orienter la suite de ce travail, dans la mesure où ils constituent, en quelque sorte, une étude préliminaire des exigences : une description de ce que le système doit faire.
La prise en charge des exigences liées à ces axes aidera donc à façonner l'approche proposée et à alimenter la macro architecture. Mais avant ceci, il est nécessaire de compléter et de formaliser la liste des exigences architecturales qui servira, par la suite, de référence pour l'évaluation du système proposé.

3.3. Exigences architecturales

L'architecture présentée dans la section précédente est préliminaire, elle nous a aidé à introduire les principaux enjeux architecturaux auxquels notre plateforme de composition doit répondre. A base de cette première réflexion, nous présentons dans cette partie les exigences de l'architecture sous une forme formalisée. La définition des exigences qui sont divisées en deux types : fonctionnelles et non fonctionnelles est conforme au standard IEEE 1471 [IEEE, 2005].

3.3.1. Exigences fonctionnelles

L'objectif des exigences fonctionnelles est de spécifier ce que le système est supposé faire. Le tableau ci-après décrit les exigences fonctionnelles retenues pour notre système.

Référence	Exigence
EF1	Le système doit assurer la traduction de la requête exprimée en langage naturel en un langage sémantique compréhensible par le système
EF2	Le système doit assurer la recherche sémantique des services Web élémentaires potentiels (susceptibles de résoudre la requête une fois assemblés)
EF3	Le système doit définir l'ordre de l'exécution des services utilisés dans la composition
EF4	Le système doit proposer la meilleure solution qui prend en compte les besoins fonctionnels ainsi que les préférences précisées par l'utilisateur
EF5	L'exécution du service produit doit être autonome et dynamique
EF6	L'opération de la composition doit être effectuée de façon automatique et transparente vis-à-vis de l'utilisateur
EF7	Le système doit être générique et indépendant du domaine des services web (commerce, santé, biologie…)
EF8	Le système doit offrir des interfaces graphiques intuitives et adaptées pour faciliter l'interaction avec l'utilisateur d'une part et pour fluidifier l'opération de traduction d'autre part

Tableau 3. 1: Liste des exigences fonctionnelles

Les exigences fonctionnelles identifiées définissent les sous problèmes de la composition. En effet à contrario de la plupart des travaux de composition actuels qui réduisent le problème de composition à l'opération d'ordonnancement des services, nous considérons la

composition des services Web une problématique composée qui se traduit en quatre sous problèmes fondamentaux: un problème de découverte (exigence EF2), un problème d'ordonnancement (exigence EF3), un problème de sélection (exigence EF4) et un problème d'exécution (exigence EF5) des services Web. Chacun de ces quatre sous problèmes dictent un ensemble de défis que notre infrastructure de composition de services Web doit surmonter.

3.3.2. Exigences non fonctionnelles

En complément des exigences fonctionnelles, les exigences non fonctionnelles, appelées aussi exigences de qualité, définissent la réponse souhaitée du système vis-à-vis des cas d'utilisation, ce qui permet d'affecter les précisions nécessaires aux éléments de l'architecture fonctionnelle du système. Cette section présente les scénarios de qualité que nous jugeons importants pour une application qui implémente notre approche. La liste des exigences de qualité est conforme au modèle défini dans [Bass et al., 2003].

EQ1 : Demande d'un service complexe pour la première fois	
Source du Stimulus	Utilisateur
Stimulus	Un service est demandé pour la première fois
Environnement	Runtime
Artefact	Système de composition
Réponse	Le service composite doit être sauvegardé pour une utilisation ultérieure, le système doit apprendre des expériences précédentes et améliorer ses performances lorsque des requêtes similaires sont soumissionnées.

EQ2 : Demande d'un service simple	
Source du Stimulus	Utilisateur
Stimulus	L'utilisateur demande un service simple qui ne nécessite pas une composition
Environnement	Runtime
Artefact	Système de composition
Réponse	le système doit détecter qu'il s'agit d'une requête simple et n'effectuer que les étapes nécessaires pour résoudre la requête

EQ3 : Demande d'un service simple ou complexe	
Source du Stimulus	Utilisateur
Stimulus	L'utilisateur exprime un besoin sous forme d'une requête
Environnement	Runtime
Artefact	Système de composition
Réponse	Les étapes de la composition doivent produire un service composite qui respecte les caractéristiques intrinsèques d'un service Web

EQ4 : Préparation de la composition	
Source du Stimulus	Système de composition
Stimulus	Processus de la préparation de la composition est déclenché

Environnement	Runtime
Artefact	Système de composition – Annuaires des services Web
Réponse	En se basant sur les services Web disponibles (simples ou composés) le système doit chercher sémantiquement les services élémentaires potentiels

EQ5 : Exécution du plan de composition	
Source du Stimulus	Système de composition
Stimulus	Processus de l'exécution du service composite produit est déclenché
Environnement	Runtime
Artefact	Système de composition
Réponse	Le système doit assurer la détection et la correction des fautes d'exécution d'une façon proactive et il doit s'adapter aux aléas du temps réel

EQ6 : Aucune correspondance n'a été trouvée	
Source du Stimulus	Système de composition
Stimulus	A la sortie du processus de la composition aucune solution n'a été trouvée
Environnement	Runtime
Artefact	Système de composition
Réponse	Le système retourne un échec de la résolution de la requête actuelle et propose l'alternative fructueuse la plus proche de la requête demandée

Tableau 3. 2: Liste des exigences de qualité

Afin d'avoir une vision plus claire sur les choix architecturaux à faire, nous proposons, dans la suite de cette section, une reformulation des exigences architecturales qui reprend et étend le formalisme mathématique introduit dans le premier chapitre (section 1.3.1). En posant le problème ainsi, nous serons capables d'exprimer les exigences fonctionnelles et de qualité dans un langage rationnel qui nous permettra d'évaluer les exigences d'une façon fiable et objective puisqu'elles seront mesurables.

3.3.3. Formalisme mathématique des exigences de l'architecture cible

En reprenant le formalisme présenté dans le premier chapitre selon lequel la CSW peut être vu comme une fonction de composition F_c tel que :

$$F_c : I \times D \rightarrow O$$

Où I est l'ensemble des requêtes possibles de l'utilisateur, O l'ensemble des sorties qu'obtient l'utilisateur et D l'ensemble des ensembles des descriptions de la communauté des services Web. La prise en considération des exigences fonctionnelles nécessite la décomposition de F_c en sous-fonctions. Nous définissons ainsi la décomposition suivante :

$$F_t : I \times D \rightarrow D'$$
$$F_g : D' \rightarrow \delta(C)$$
$$F_s : \delta(C) \rightarrow C$$
$$F_e : C \rightarrow O$$

- Dans cette décomposition, F_t est la fonction de traduction simulant l'exigence EF1, cette fonction traduit la description des entrées i ∈I d'une description d∈D en une description d'∈D'. D' est l'ensemble des descriptions compréhensibles par le système (sémantiques).
- F_g la fonction traduisant les exigences EF2 et EF3, elle génère un ensemble de composition cs ∈ δ(C) à partir de la description d' de la requête et des services Web. δ(C) est l'ensemble d'ensembles de compositions.
- F_s est la fonction qui, à partir d'un ensemble de composition cs ∈ δ(C), conclut quant à la pertinence de cs de services (en se basant sur des attributs non-fonctionnels) et en choisit une : c , avec c∈ δ(C). F_s traduit l'exigence EF4.
- Enfin, F_e est la fonction exécutant la composition c∈C et fournissant la sortie o∈O.

Ainsi, la recherche d'une méthode de composition revient à la recherche d'une méthode : $F_tOF_gOF_sOF_e$ qui génère une composition exécutable à partir d'un ensemble de descriptions de services Web et d'une requête, cette méthode est illustrée dans la figure suivante.

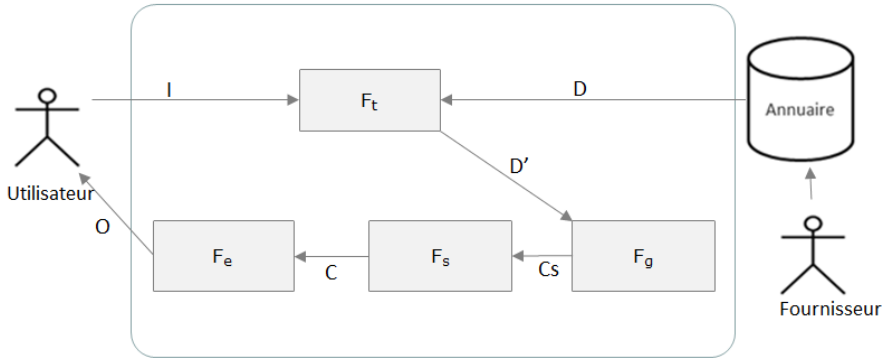


Fig.3. 2: Décomposition de la fonction F_c

En suivant la même logique mathématique, les scénarios de qualité définis auparavant peuvent être exprimés ainsi:

- Notons:
  - **S** l'ensemble des services Web;
  - $\mathbf{S_c}$ le sous ensemble des services Web composites ;
  - $\mathbf{S_s}$ le sous ensemble des services Web simples ;
- Soit d∈D et i∈I et o∈O tel que o= F_c(i,d)
- Supposant maintenant que o est l'exécution de la composition s', et d' la description d'un ensemble de service s_i tel que : i ∈{1,…, n}

Pour satisfaire les exigences de qualité, l'application $\mathbf{F_c}$ doit vérifier les critères suivants :

- **s'∈S** : la composition des services Web est un service Web, ceci fait de $\mathbf{F_c}$ une loi de composition interne qui offre le niveau d'homogénéité décrit par l'exigence EQ3. Ce critère veut aussi dire que la loi doit s'engager à répondre par un service quel que soit les variances de l'environnement en assurant le niveau nécessaire de

tolérance aux pannes et ce, en proposant des alternatives en cas d'échecs (exigence EQ5 et EQ6) ceci traduit l'efficacité du système.

- $s_i \in S$: le service élémentaire peut appartenir à l'ensemble des services simples comme il peut appartenir à l'ensemble des services composites, i.e un service composite une fois construit peut participer à d'autre cycle de composition comme service élémentaire tant que la loi F_c respecte l'exigence EQ3. En termes de qualité, ceci permet d'optimiser l'utilisation des ressources et donc d'offrir un meilleur niveau d'efficience (exigence EQ4).
- Si le nombre des services à composer est égal à 1 (un) et i= m ; alors la composition de s_m retourne s_m , lui-même, ceci veut dire que :
  - si $s_m \in S_s$, F_c doit comprendre qu'il ne s'agit pas de composition dans ce cas et elle doit adapter sa structure interne pour n'effectuer que les étapes nécessaires pour retrouver le service simple (exigence EQ2).
  - si $s_m \in S_c$, la loi F_c doit reconnaitre que s_m ,le service demandé, a été déjà construit lors d'un précédent cycle de composition et améliorer ses performances en apprenant des expériences de composition antérieures (exigence EQ 1).

Le tableau ci-après synthétise l'ensemble des exigences identifiées dans cette section. L'affinement de la macroarchitecture dépend principalement de la prise en charge de ces exigences, leur vérification ne sera, par contre, faite qu'à la fin de ce manuscrit (c.f chapitre 8) lorsque tous les aspects relatifs à l'approche proposée seront présentés et expliqués.

Type	Exigence	Référence
Fonctionnelle	Découverte	EF2
	Ordonnancement	EF3
	Sélection	EF4
	Exécution	EF5
	Autres	EF1, EF6, EF7, EF8
Qualité	Efficacité	EQ5, EQ6
	Fiabilité	EQ4
	Performance	EQ1
	Homogénéité	EQ3
	Adaptabilité	EQ2

Tableau 3. 3: Synthèse des exigences architecturales

La prise en compte des exigences est faite en suivant principalement les deux règles suivantes :

1. Les exigences fonctionnelles sont satisfaites par l'attribution des responsabilités appropriées tout au long de l'architecture;
2. Les exigences de qualité sont satisfaites d'abord, par l'adaptation des responsabilités attribuées dans l'étape précédente et ensuite par la définition appropriée des interactions entre les éléments qui meublent l'architecture. Des structures

supplémentaires peuvent, éventuellement, être intégrées si les deux mesures précédentes ne suffisent pas pour vérifier une des exigences.

Grâce au formalise mathématique illustré dans la Fig.3. 2, nous avons déjà une idée sur le style architectural à adopter, c'est celui qui organise (décompose) le processus de composition en sous phases où chaque phase prend en charge un sous-problème de composition (une sous fonction). La présentation de ce style architectural fait l'objet de la section suivante.

3.4. Style architectural : composition par phases à deux niveaux

L'idée pivot de l'architecture proposée pour résoudre la problématique de la composition est de construire une approche progressive de composition dynamique et automatique des services Web. En partant d'une requête déclarative simple exprimée par un utilisateur non-spécialiste, l'architecture proposée doit encapsuler la complexité des interactions entrainées par l'opération de composition et offrir un service composite qui satisfait la requête de l'utilisateur. Pour ce faire, nous proposons de concevoir le parcours de la résolution de la requête de composition sous forme d'un cycle de vie de quatre phases organisées en deux niveaux de composition : abstrait et concret. Ainsi nous pouvons gérer les différentes exigences chacune d'elles au niveau de la phase correspondante et avec la façon la plus optimale, par conséquent l'architecture gagne plus de modularité, moins de couplage, un bon niveau d'évolutivité (scalability), et plus important encore, le niveau d'autonomie souhaité.

3.4.1. Approche basée sur un cycle de vie

Décidément la problématique traitée se caractérise par la composition ; non seulement elle vise la production des services composés mais sa nature même est aussi composée d'autres sous-problématiques. La façon la plus judicieuse pour résoudre cette problématique est donc de décortiquer le processus de composition en phases et de déléguer à chaque phase la résolution d'un sous-problème. Les phases se déroulent dans un cadre d'autonomie et de continuité, c'est-à-dire que chaque phase évolue indépendamment des autres et peut utiliser des méthodes, des technologies et des algorithmes dédiés et spécifiques afin d'accomplir son rôle dans la composition. Mais ses sorties doivent impérativement former/alimenter les entrées des phases suivantes dans le cadre d'un processus continu.

Nous retrouvons ainsi les quatre phases suivantes :

a. Phase d'Analyse

Nous devons faire la distinction entre les langages de spécification internes et externes des services. Les langages externes sont utilisés par les utilisateurs pour exprimer leurs besoins, ils sont souvent des langages naturels qui manquent de précision et de complétude. Les spécifications internes doivent par contre être formelles et précises pour garantir la réussite du processus de génération de la solution. Le passage des spécifications utilisateurs aux spécifications système est géré dans la phase d'analyse qui constitue une phase préparatoire visant à identifier clairement l'objectif à atteindre dans le cycle de composition en cours dans un langage compréhensible par le système.

b. Phase de Planification

La planification est une phase essentielle. Elle détermine les services Web qui participeront à la composition et leur ordre d'exécution dans le plan de composition. Ceci consiste d'abord à chercher dans les registres de services des candidats susceptibles de participer à la résolution des besoins fonctionnels de la requête. Les aspects conformité et compatibilité des services sont traités lors de cette étape. Cette première sous- phase liste les candidats potentiels pour la réalisation de la composition abstraite des services; dans la deuxième étape des algorithmes spécifiques d'ordonnancement et de composabilité sont utilisés afin de déterminer l'ordre des tâches qui seront exécutées par les services Web découverts. Cette sous phase résulte un plan abstrait d'exécution des services composants.

c. Phase d'Evaluation

Dans cette phase le système interroge les registres des services web disponibles, pour sélectionner des instances de services Web appartenant au plan abstrait généré.

Une grande quantité des services Web concrets qui implémentent la même description abstraite peut être identifiée. Dans ce cas, les critères pris en compte ne sont plus suffisants pour permettre une sélection pertinente de services Web. Une méthode d'évaluation est utilisée donc à ce niveau pour choisir les instances de services qui une fois exécutées selon le plan de composition elles délivrent le meilleur service le plus proche des attentes de l'utilisateur. Le résultat de cette phase est un plan concret d'exécution.

d. Phase d'Exécution

La phase d'exécution réalise la composition concrète. Les services sont déployés sur la plateforme d'exécution, les liaisons avec les services sont établies et l'invocation est réalisée. Dans cette phase les incohérences et les imprévus relatifs au temps réel sont gérés par un retour en arrière vers la phase de l'évaluation ou carrément vers la phase de planification. Cette phase résulte à l'utilisateur une exécution d'un service Web composite en réponse à sa requête.

Ce découpage nous permet déjà de dresser le cadre guidant notre architecture (Fig.3. 3),

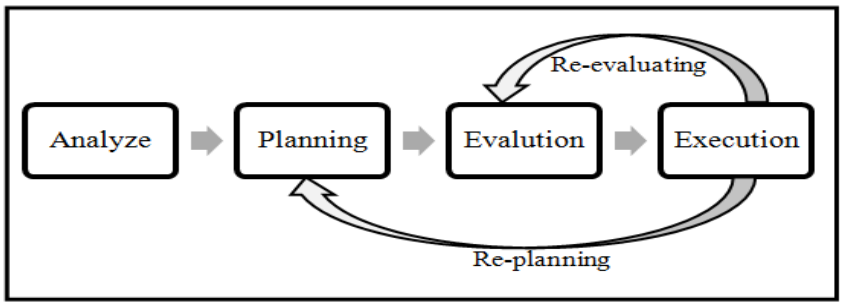


Fig.3. 3: Cycle de vie de la composition des services Web [Adadi et al., 2014]

En commençant par une définition abstraite et en la rendant progressivement concrète et exécutable le modèle à quatre phases, illustré ci-avant, permet ainsi de générer un plan de composition à deux niveaux de granularité:

- Niveau abstrait : formé par la phase d'analyse et de planification;
- Niveau concret : qui comprend les deux dernières phases d'évaluation et d'exécution.

L'approche à deux niveaux constitue elle aussi un pivot de l'architecture proposée, ci-après nous proposons une emphase sur le principe de l'approche et les gains escomptés de son adoption.

3.4.2. Approche de composition à deux niveaux

3.4.2.1. Principe de l'approche de composition à deux niveaux

Pour une résolution de bout en bout du problème de la composition de services, nous proposons le principe de la composition à deux niveaux. L'approche à niveaux est la solution architecturale qui nous permet d'atteindre les niveaux de modularité, de concision et de scalabilité souhaités. En implémentant cette approche le processus de composition procède d'abord à la génération d'un plan abstrait en se basant sur les types de services Web (composition abstraite), ce plan est ensuite concrétisé par un plan exécutable en sélectionnant les instances des services Web les plus appropriées (composition concrète). L'idée de base de notre approche de composition à deux niveaux est la séparation entre (i) les types des services Web, qui sont des regroupements de services Web similaires (en termes de paramètres fonctionnels) et (ii) les instances des services Web qui sont invocables.

La séparation de la représentation des types de services Web de la représentation de leurs instances permet principalement de:

- réutiliser les plans déjà construits pour répondre à des demandes similaires même si certaines préférences ont été modifiées ;
- travailler efficacement avec un grand nombre de services Web ;
- et plus important encore, cette approche permet **une optimisation de l'espace de recherche des services Web**. En effet, pour un système de composition qui doit scruter un espace gigantesque de services Web pour construire un plan, l'optimisation de l'espace de recherche est un enjeu important. Ce gain peut être souligné et démontré formellement en considérant :

(1) $S = \{s_1, ..., s\alpha\}$: liste de α types de services Web disponibles.

(2) $Is = \{is_1, ..., is_\beta\}$: liste de β instances de services publiées dans l'annuaire des SWS, notons que le mapping de S dans Is est surjective (de type « un à plusieurs »). Supposons que chaque type de service S_k possède M instanciations, donc $\beta = M \times \alpha$.

(3) P: un plan abstrait construit après la composition abstraite. Supposons que P comprend λ type de services Web, donc un plan abstrait peut être trouver en cherchant dans un espace $O(\alpha^\lambda)$.

(4) $W = \{W_1, ..., W_L\}$: une liste de L plans exécutables sélectionnés après la composition concrète. Puisque chaque type de services correspond à M instanciations, l'espace de recherche pour sélectionner un plan exécutable est $O(M^\lambda)$ et l'espace de recherche pour la sélection de L plan concrets est $O(L \times M^\lambda)$. ce qui fait que l'approche à niveaux peut produire un plan exécutable en cherchant dans un espace de $O(\alpha^\lambda)) + O(L \times M^\lambda)$.

Par contre une composition qui n'utilise pas la distinction type/instance, elle trouve un plan exécutable en cherchant dans un espace de $O(L \times \beta^\lambda))$. Sachant que cet espace est toujours

plus grand que celui présenté précédemment (démonstration ci-après), l'approche à niveaux conduit donc à une réduction de l'espace de recherche de $O(L \times \beta^{\lambda}))$ à $O(\alpha^{\lambda}))$ $+O(L \times M^{\lambda})$. Cette réduction est significative, par exemple pour $\alpha = 10$, $M = 5$, $L = 3$ et $\lambda = 3$, on a $\beta = 5x10 = 50$ et $L \times \beta^{\lambda} = 3 \times 50^3 =$ **375000** alors que $\alpha^{\lambda} + L \times M^{\lambda} = 10^3 + 3 \times 5^3 =$ **1375**

Démonstration:

Démontrons que $\boldsymbol{L \times \beta^{\lambda} \geq \alpha^{\lambda} + L \times M^{\lambda}}$

On a $\boldsymbol{L \times \beta^{\lambda} = L \times \alpha^{\lambda} \times M^{\lambda} = \alpha^{\lambda} \times (L \times M^{\lambda})}$ *, posons* $\boldsymbol{\alpha^{\lambda} = A}$ *et* $\boldsymbol{L \times M^{\lambda} = B}$ *et prouvons que* $\boldsymbol{A \times B \geq A + B}$

On a $\boldsymbol{A \geq 2}$ *ou* $\boldsymbol{B \geq 2}$ *(puisque le cas de A=1 ou B=1 ne correspond pas à une composition car il s'agit d'un seul service)*

Donc $\boldsymbol{1/A \leq 1/2}$ *et* $\boldsymbol{1/B \leq 1/2}$ *, cela implique que* $\boldsymbol{(1/A + 1/B) \leq 1}$*, donc* $\boldsymbol{(A+B/\ A \times B) \leq 1}$ *et donc* $\boldsymbol{A + B \leq A \times B}$

3.4.3. Implémentation de l'approche de composition par phases à deux niveaux

En s'appuyant sur le style architectural présenté dans cette section, un premier portrait global de l'architecture proposée peut être vu sous forme de deux modules logiques (Fig.3. 4):

(1) composeur abstrait : la phase d'analyse et la phase de planification peuvent être simulées par un module qui offre une composition dite fonctionnelle des types de services, et ce pour créer une fonctionnalité qui n'existe pas encore.

(2) Composeur concret: les phases d'évaluation et d'exécution, quant à elles, sont encapsulées dans un module logique dénommé « Composeur concret », ce module permet la sélection des instances des services élémentaires en se basant sur des paramètres non-fonctionnels qui formeront ensemble le nouveau service composite à exécuter.

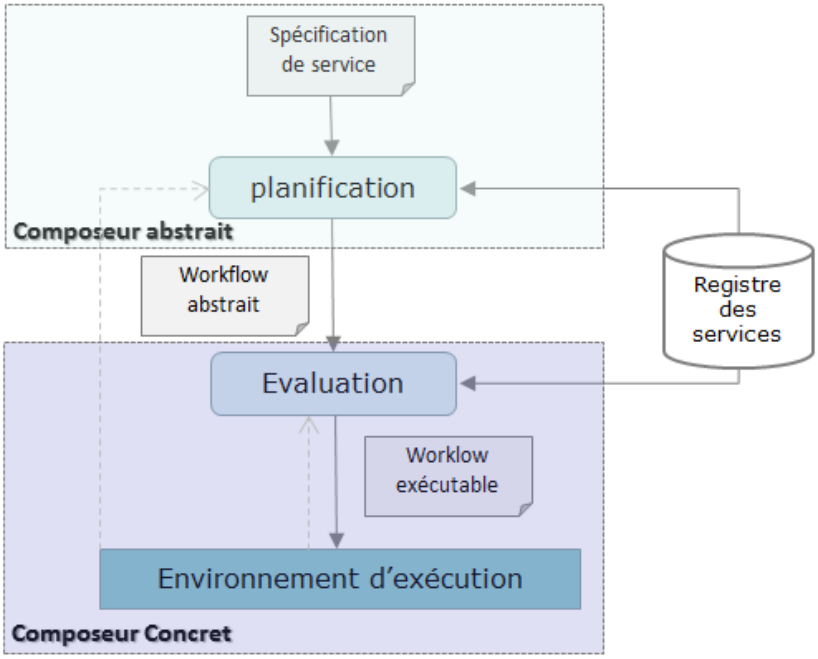


Fig.3. 4: Architecture par phase en deux niveaux pour la CSW [Adadi et al , 2016]

A partir de l'architecture préliminaire présentée auparavant, l'implémentation du style architectural décrit dans cette section permet de définir une version rehaussée de la macro architecture. Comme illustré dans Fig.3. 4, en plus des deux modules logiques, l'architecture contient également un annuaire de service contenant les informations sur les services disponibles et sur les services offerts par les fournisseurs. Lorsqu'un nouveau service est demandé, la phase d'analyse livre une spécification du service sous forme d'une requête sémantique au planificateur, ce dernier utilise des techniques de recherche et de planification pour créer une composition des types de services. L'objectif du module composé par ces deux phases est donc d'explorer qualitativement les services abstraits disponibles pour produire un workflow abstrait, i.e un plan (on suppose qu'un plan faisable existe) qui satisfait les besoins fonctionnels.

Afin de transformer le plan abstrait en un plan exécutable, des instances correspondantes aux types des services appartenant au plan doivent être choisies, le composeur concret utilise des techniques d'optimisation et d'évaluation pour sélectionner les meilleures instances des services Web afin de produire un workflow exécutable, dans ce cas l'exploration des services Web se fait à un niveau quantitatif en intégrant l'annuaire dans lequel les instances des services Web sont publiées. Le workflow ainsi généré doit être déployé dans une infrastructure d'exécution (l'état de l'art recommande d'utiliser un moteur de workflow comme par exemple WebSphere Process Choreographer [IBM, 2016] pour l'exécution du workflow). Le défi de cette phase est de rendre l'exécution auto-guérissante i.e. capable d'assurer une tolérance de pannes adaptée, évoluée et automatique.

3.5. Architecture proposée

3.5.1. Description de l'architecture

Après avoir présenté le résultat de la capture des exigences de l'architecture et décrit la caractéristique qui distingue notre approche à savoir la composition par phase à deux niveaux. Nous pouvons à présent dévoiler l'architecture complète de notre approche de composition. L'architecture nommée iDyCoS pour « intelligent and Dynamic Composition of Web Services » est le fruit des analyses précédemment présentées dans ce chapitre ainsi que les discussions relatives à l'hypothèse des quatre forces présentée dans le chapitre 2.

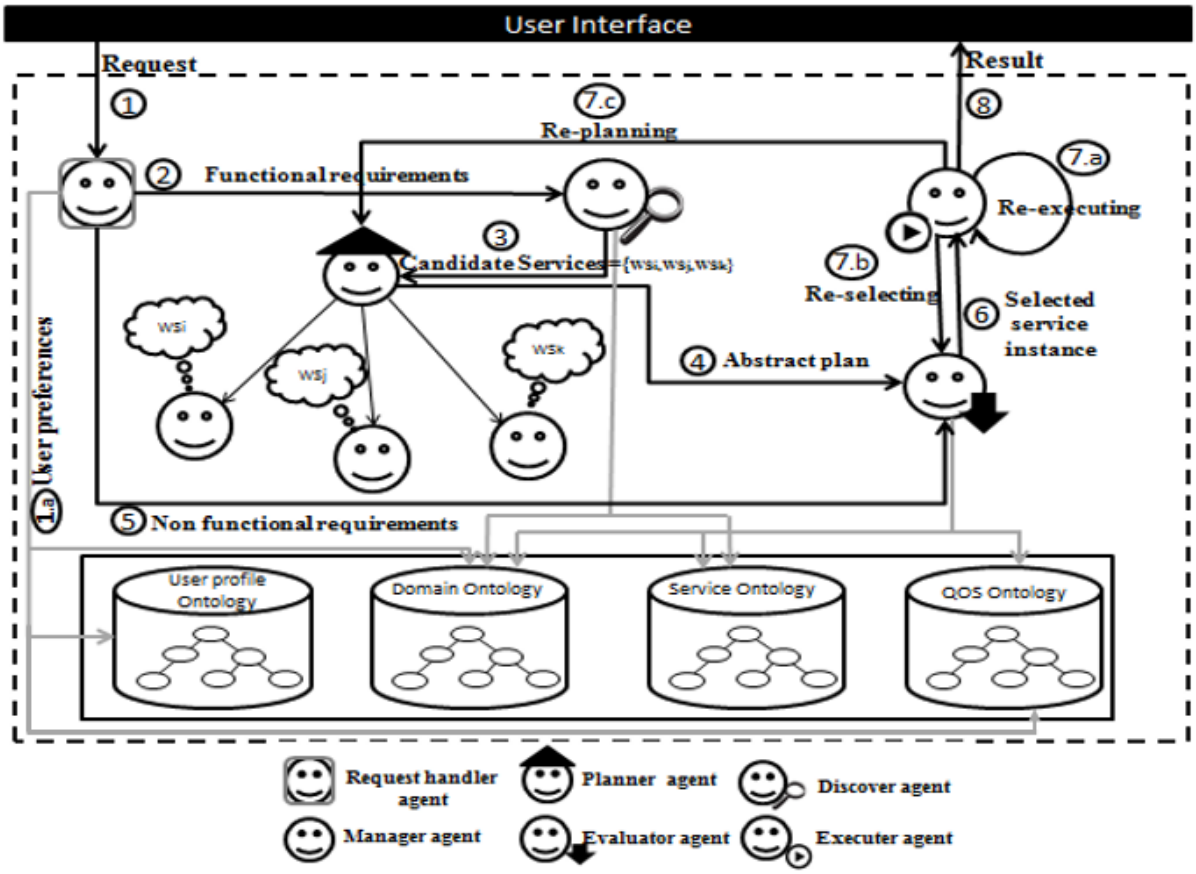


Fig.3. 5: Architecture iDyCos Web [Adadi et al., 2014]

Une lecture de la Fig.3. 5 nous permet d'avoir une idée générale sur la structure de l'architecture conceptuelle iDyCoS. En effet, comme illustré sur la figure ci-dessus, l'architecture proposée est composée de trois éléments principaux: un ensemble d'agents intelligents, quatre ontologies et une interface graphique de communication pour interagir avec l'utilisateur.

Ces composants sont distribués en deux niveaux : abstrait et concret à travers lesquels on retrouve les différentes phases de résolution de la requête de composition (cf section précédente). Le point de départ est une requête reçue à travers l'interface utilisateur, qui subit à travers le passage d'un agent à un autre une chaine de modifications, pour arriver finalement à l'état résultat communiqué à l'utilisateur par la même interface. Les interactions identifiées dans ce processus sont de type agent –agent, agent –ontologie et agent- interface de communication.

Les responsabilités affectées aux composants de l'architecture et leurs différentes interactions, déterminent le niveau de conformité par rapport aux exigences architecturales identifiées, ci-après le rôle de chaque composant sera mis en lumière avant de présenter un scénario typique de composition qui décrit les interactions entre les éléments de l'architecture proposée.

3.5.2. Les responsabilités des composants de l'architecture

3.5.2.1. Agents

Six types d'agents capables de découvrir, composer, exécuter et contrôler automatiquement les services Web ont été utilisés, à savoir l'agent « Request Handler », l'agent « Discover », l'agent « Planner », l'agent « Executer », l'agent « Evaluator » et l'agent « Manager ». Une description générale de ces agents et des tâches qu'elles leurs sont affectées est donnée dans le Tableau 3. 4. Plus de détails sur les mécanismes qui conduisent ces tâches seront données dans les prochains chapitres.

Type d'agent	Compétence
Request handler	▪ capable de décrire (traduire) les données d'entrées en requêtes sémantiques (cf chapitre 4); ▪ afin d'adapter la requêtes à ses besoins, l'utilisateur peut eventuellement activer l'option de la génération automatique des préfereneces, dans ce cas l'agent ajoute des préférences personalisées à la requête en utilisant l'ontologie du profil utilisateur et l'ontologie de domaine ; ▪ envoie la description sémantique des besoins fonctionnels à l'agent
Discover	▪ il a la capacité de vérifier l'annuaire des services pour voir s'il existe un service simple ou composite précedement crée qui peut satisfaire la requête; ▪ il utilise des algorithmes de matching (correspondance) spécifiques pour déterminer la liste des services abstraits, qui une fois composés peuvent offrir la fonctionnalité demandée (cf chapitre 4).
Planner	le modèle de planification (cf chapitre 4) produit un plan valide de composition basé sur la théorie de synthèse des plans, le rôle du planner dans ce modèle est de : ▪ contôler les interactions entre les agents managers et communiquer avec les autres types d'agents (Discoverer, Evaluator) ; ▪ il est aussi en charge d'initier et clôturer le processus de planification en envoyant le but à atteindre aux managers.
Manager	▪ simule un service abstrait découvert ; ▪ il est initié par la description sémantique du service qu'il simule, des mécanismes de traduction permettent alors de construire la base de connaisances et de compétances de l'agent à partir de la déscription sémantique qui lui est associée ; ▪ échange des propositions de plan avec les autres managers dans le respect des règles de dialogue afin de construire un plan solution (cf chapitre 4).
Evaluator	▪ cherche les services concrets implémantant les services abstraits qui appartiennent au plan de composition; ▪ estime (calcule) les paramatres non fonctionnels de toutes les alternatives du service composite en se basant sur des mécanismes de caluls des Qos et sur l'ontologie QoS (cf chapitre 5) ; ▪ utilise un alogorithme specifique pour classer les alternatives du service composite et choisit la meilleure.

Executer	▪ invoque les services concrets selectionnés dans le respect du plan de composition généré ; ▪ il a la capacité de détécter et réparer automatiquement les erreurs d'éxecution grace à son architecture basée sur une boucle autonomique (cf chapitre 5)

Tableau 3. 4: Responsabilités des agents

3.5.2.2. Ontologies

En plus des agents un autre élément clé de l'architecture proposée est les ontologies. Afin de mener à bien leurs tâches, les agents intelligents ont besoin d'accèder aux informations stockées dans quatre ontologies, ci-après la description du rôle de chaque ontologie.

Ontology	Description
Domain Ontology	Contient les concepts d'un domaine bien donné, leur hiérarchie et les relations entre eux
Service Ontology	Joue le rôle d'un annuaire UDDI avec des fonctionnalités étendues pour permettre de stoker les attributs sémantiques, à ce niveau, aucune restriction en termes d'utilisation d'un type donné de spécification sémantique n'est imposée (i.e., OWL-S, WSMO, SWSF, WSDL-S ou SAWSDL)
QoS Ontology	Spécifie les connaissances et les vocabulaires partagés relatifs aux propriétés de QoS et la relations qui les relient
User profile Ontology	Intègre les concepts et les propriétés utilisés pour modéliser les préférences de l'utilisateur

Tableau 3. 5: Responsabilités des ontologies

3.5.2.3. Interface Utilisateur

L'interface utilisateur joue un rôle important dans l'architecture puisqu'elle représente le point de communication avec l'utilisateur externe. L'interface conçue offre quatre opérations principales :

- L'initiation du processus de composition par l'utilisateur;
- Arrêt de la composition par l'utilisateur à n'importe quel point du processus ;
- Redémarrage du processus de composition à partir de son point d'arrêt ;
- Retour du résultat à l'utilisateur.

En plus d'être un point de communication, l'interface utilisateur joue un rôle pivot dans la phase d'analyse, qui sera détaillé dans le chapitre suivant (cf chapitre 4 section 4.3.1.1).

3.5.3. Interaction entre les composants de l'architecture

Comme décrit dans la Fig.3. 5, le processus de composition commence par une spécification abstraite et progressive vers une spécification concrète (exécutable), il est déclenché lorsqu'un utilisateur (humain ou logiciel) interagit avec l'interface d'utilisateur pour demander un service. Afin d'y répondre le système doit d'abord comprendre la requête, pour ce faire l'agent « Request handler » (1) utilise des règles de traitement et d'analyse pour traduire la requête exprimée en langage naturel en une requête sémantique en respectant l'ontologie de

domaine et celle de QoS. Suite à cette analyse préliminaire, le système obtient une représentation sémantique et formelle de la requête de l'utilisateur. Dans le cas où l'option de génération des préférences est activée, le Request Handler enrichit la requête sémantique par l'ajout des préférences récupérées automatiquement à partir de l'ontologie du profil de l'utilisateur (1.a). Ce travail de gestion de requête résulte finalement en une liste des besoins fonctionnels décrits sémantiquement envoyée à l'agent Discover (2), et une partie non-fonctionnelle formée par les préférences -manuelles (entrées par l'utilisateur) et/ou automatiques-, envoyée directement à l'agent Evaluator (5) (Fig.3. 6).

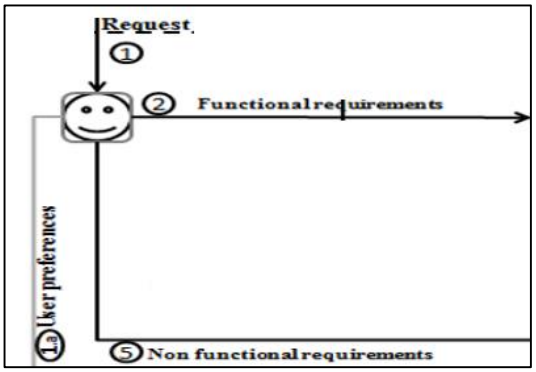


Fig.3. 6: Extrait 1 de l'Architecture iDyCos Web

Le Discover vérifie d'abord s'il existe un service (simple ou composite) au niveau de l'ontologie des services qui correspond exactement au service demandé, si un tel service existe, il est directement envoyé à l'Evaluator, de cette façon on évite un effort non nécessaire de planification. Sinon le Discover cherche des services candidats qui ont le potentiel de participer à la construction du nouveau service composite, la recherche s'appuie sur un algorithme de Matchmaking en utilisant l'annuaire des services sémantiques (ontologie des services) et l'ontologie de domaine. Les services découverts sont envoyés à l'agent Planner (3) (Fig.3. 7).

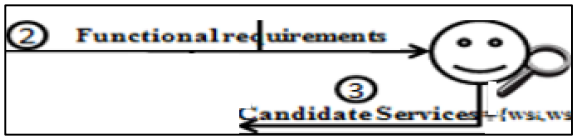


Fig.3. 7: Extrait 2 de l'Architecture iDyCos Web

Le Planner coordonne une liste d'agents Mangers qui simulent les services découverts, ces derniers collaborent en utilisant les techniques PIA pour co-construire un plan de composition abstrait, qui est ensuite envoyé à l'Evaluator (4). A ce moment le deuxième niveau de composition, où on gère des instances concrètes des services Web, commence (Fig.3. 8).

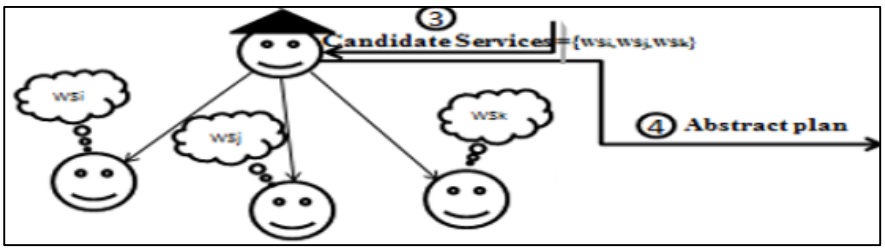


Fig.3. 8: Extrait 3 de l'Architecture iDyCos Web

Afin de sélectionner le meilleur service composite, l'Evaluator utilise la partie non-fonctionnelle de la requête sémantique envoyée par le Request Handler (5) et l'ontologie QoS (puisque toutes les alternatives du service composite répondent aux besoins fonctionnels), il classe les alternatives concrètes du service composite en utilisant des techniques de gestion de QoS, et le plan concret ayant la plus grande valeur des QoS est envoyé à l'agent « Exécuter » pour exécution (6) (Fig.3. 9).

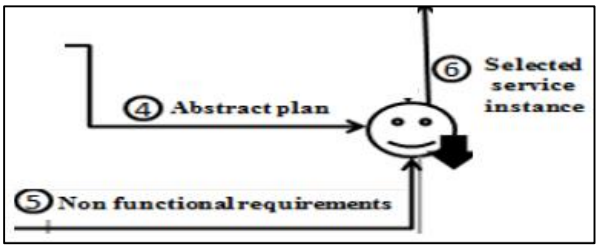


Fig.3. 9: Extrait 4 de l'Architecture iDyCoS Web

L'Executer est aussi responsable sur le stockage du service composite généré dans l'annuaire des services sémantiques pour une éventuelle réutilisation ultérieure. Durant l'exécution si une erreur est détectée, le mécanisme d'autoréparation guidant le comportement de l'Executer décide l'action à entreprendre pour rétablir la situation, cela peut conduire à une ré exécution du service composite (7.a), à une ré évaluation du plan de composition derrière (7.b) ou carrément à une replanification en partant des données de l'état de monde après l'apparition de l'erreur. Le résultat est passé à l'utilisateur à travers l'interface Utilisateur (8) (Fig.3. 10).

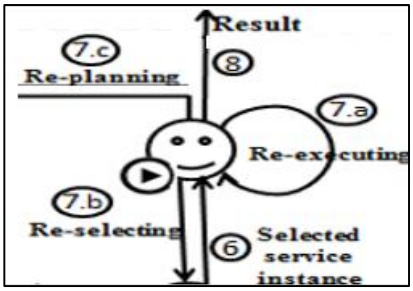


Fig.3. 10: Extrait 5 de l'Architecture iDyCos Web

Le diagramme (Fig.3. 11) formalise le processus décrit ci-dessus dans une notation BPM (Business Process Managemen) [Ko et al. , 2009].

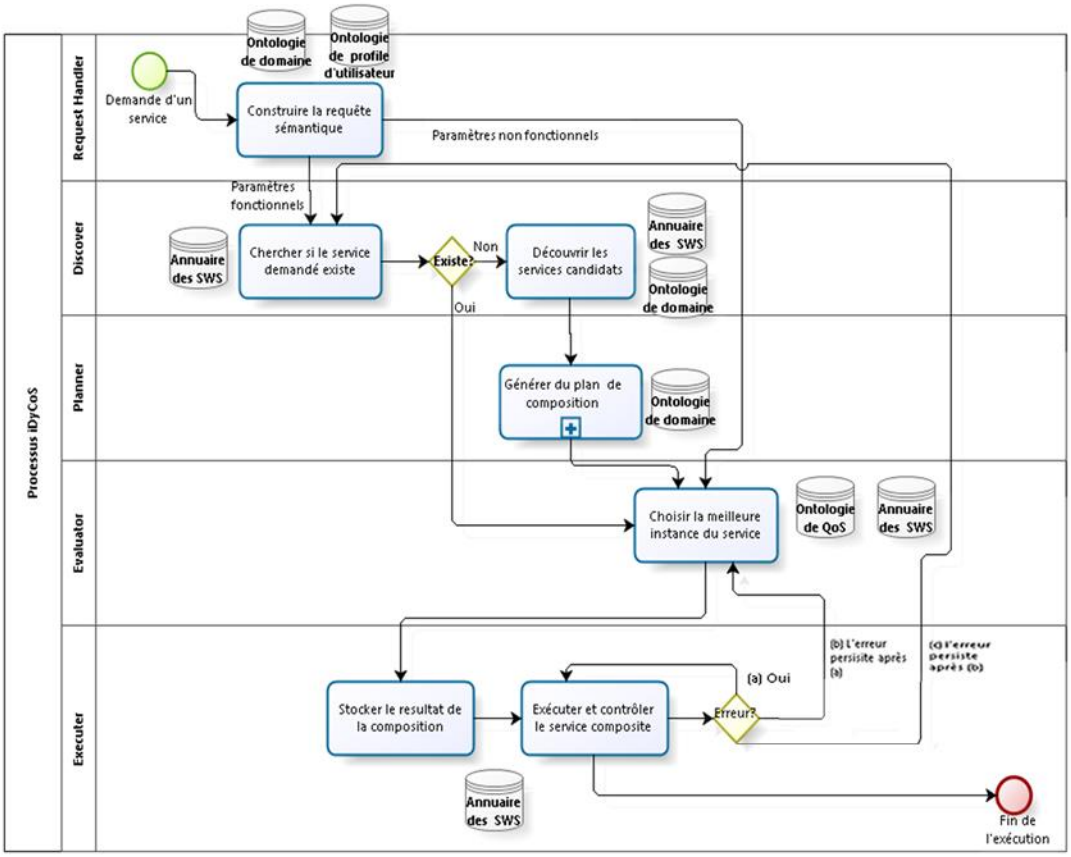


Fig.3. 11: Interaction entre les composants de l'architecture iDyCoS

3.6. Exemple introductif

Dans cette section nous proposons une illustration sur un exemple simplifié (didactique) du modèle de composition proposé.

Considérons le scénario où Mourad, qui vit à Montréal, décide de rendre visite à sa famille au Maroc. Sa famille habite à Fès.

Si Mourad choisit de confier l'organisation de son voyage à notre système. Il n'a besoin que de soumettre la requête suivante via l'interface du système : « **je veux aller à Fès le mardi 01 aout 2017 »,** pour que le processus de composition se déclenche :

a. en première étape, la demande est transformée en une requête sémantique, exprimée en termes d'entrées/sorties et des éventuelles préférences. La requête sémantique peut donc être exprimée ainsi :

   R_{voy}= **in** : {date : 01/08/2017, destination : Maroc-Fès}
   out : {confirmation}
   preferences : {moyen de transport aérien, moindre coût}

b. en se basant sur les éléments sémantiques de la requête le Discover cherche les services élémentaires abstraits qui ont le potentiel de participer à la résolution de cette requête, il trouve alors la liste des services suivante: S_{voy}= {S_1,S_2,S_3} tel que :
   S_1 : Une compagnie de transport international offrant un service de transport de Canada au Maroc ;

S_2 : une compagnie de transport local offrant un service de transport à l'intérieur du Maroc;

S_3 : un service de banque (représentant la banque de Mourad) ;

c. en prenant en entrée la liste S_{voy} et la requête sémantique $R_{voy,}$ le Planner produit un plan P_{voy} qui définit l'ordre de l'exécution des services abstraits découverts. C'est donc S_1 qui va intervenir en premier pour déplacer Mourad de Montréal à Casablanca, ensuite S_2 le prendra à Fès et le paiement du voyage va être assuré par S_3.
d. plusieurs services concrets implémentent la même interface fonctionnelle (par exemple pour S_1 on peut trouver Royal Air Maroc, Air Canada et Jet aiways). L'Evaluator va nous faire alors le choix des meilleurs services concrets disponibles en prenant en considération les préférences explicitées par l'utilisateur, les préférences générées et les critères de qualités de services (QoS).
e. enfin l'Executer va invoquer les services concrets sélectionnés dans l'ordre défini, puis il entre dans une boucle de contrôle de la bonne exécution du plan (c'est lui donc qui va vérifier que le voyage est toujours maintenu dans la date souhaitée sinon il bascule vers un service équivalent en toute transparence).

Le processus global de la planification du voyage de Mourad étant présenté, nous allons dans la suite de ce manuscrit détailler chacune des cinq étapes définies ci-avant. Nous ferons donc souvent appel à cet exemple afin d'illustrer les différentes techniques utilisées dans le modèle de composition iDyCoS.

3.7. Conclusion

Passerelle incontournable vers la mise en place de notre système, l'architecture proposée pour la composition dynamique et automatique de services Web a été décrite dans ce chapitre en s'appuyant sur une approche structuraliste qui, à partir d'une première ébauche naïve de l'architecture, cherche à faire émerger les principales problématiques afférentes au contexte étudié, puis en procédant à une analyse sobre visant à concevoir une solution efficace, ainsi :

- Dans une première étape des axes de réflexions critiquant la macro architecture ont provoqué l'élaboration d'une liste d'exigences fonctionnelles et de qualité;
- Ensuite, à même de se conformer aux exigences architecturales, un caractère distinctif de notre modèle a été introduit, il s'agit de la composition par phases à deux niveaux ;
- Enfin l'architecture iDyCoS a été dévoilée en explicitant ses composants et en illustrant les interactions entre ces composants par un exemple simplifié.

Au travers ce chapitre, nous avons fait référence à plusieurs techniques et mécanismes de composition nécessaires pour que chaque phase du processus de la composition puisse accomplir ses missions, le reste de cette partie se donne alors comme objectif l'étude de ces mécanismes en les organisant dans deux catégories : les techniques de la composition abstraite et les techniques de la composition concrète.

Chapitre 4 :

Techniques de composition abstraite

The purpose of abstraction is not to be vague,
but to create a new semantic level in which one can be absolutely precise.
Edsger Dijkstra

4.1. Introduction

L'approche de composition proposée se distingue par ses deux niveaux de composition: abstrait et concret. La composition concrète concrétise les résultats obtenus dans la composition abstraite, elle se base surtout sur des techniques d'optimisation et de tolérance aux pannes alors que la composition abstraite, qui vise à générer un plan abstrait définissant l'ordre de l'exécution des types de services, se base sur des techniques de correspondance sémantique et sur des algorithmes de planification.

Dans ce chapitre nous revenons, en détail, sur les techniques conçues pour accomplir la composition abstraite et ce, en explicitant leurs principes, en exposant les théories et les mécanismes y afférents, en évaluant leur performance et en présentant les points d'amélioration sous-jacents.

4.2. Stratégies et principes de modélisation

Les hypothèses et les principes communs à toutes les techniques de composition conçues sont présentés au niveau de cette section et ils seront valables pour le reste de ce travail. Ces hypothèses et principes concernent, en particulier, la méthode de modélisation des services abstraits et les besoins non fonctionnels ainsi que la communication inter agents.

4.2.1. Modélisation des services et de la requête

Afin de garantir une construction automatique de la fonctionnalité désirée, nous avons besoin d'un langage pour décrire les services disponibles dans un modèle formel interprétable par la machine empêchant les ambiguïtés et les problèmes d'interprétation.

Ce langage sera utilisé pour:

a. spécifier les besoins (la requête): la fonctionnalité demandée ainsi que les besoins non-fonctionnels doivent être exprimés dans un langage sémantique pour le traitement inter-système ;
b. représenter des services abstraits (type de service): au niveau de la composition abstraite, le processus de composition implique typiquement des algorithmes de raisonnement, une description de haut niveau centrée sur les compétences des services est donc suffisante à cette étape;
c. modéliser les paramètres non fonctionnels des services concrets : au niveau de la composition concrète, les services actifs sur internet doivent être identifiés afin de

pouvoir déployer le Workflow défini au cours du cycle de composition. Pour ce faire, les propriétés de qualité de services (QoS) doivent suivre le même formalisme que celui des paramètres non fonctionnels (PnF), pour que ces propriétés puissent servir de critères de sélection.

Dans le chapitre 2 (section 2.3.4.2), nous avons présenté les langages sémantiques les plus utilisés pour décrire les services Web sémantiques. Afin de choisir le langage le plus adapté nous mettons en confrontation, dans le Tableau 4. 1, les principales caractéristiques de ces langages.

Les critères de comparaison sont :

- Niveau d'abstraction: précise le niveau d'abstraction (conceptuel, technique) associé au langage.
- Niveau d'ouverture: représente le degré de standardisation permis par le langage.
- Description sémantique: précise la manière avec laquelle les services Web sont décrits sémantiquement.
- Eléments considérés dans l'annotation sémantique : indiquent les éléments inclus dans la description sémantique, ceci impacte directement la qualité de précision de la découverte et de la planification des services, les phases pilotes de la composition abstraite.
- Support des PnF : indique si l'approche supporte l'expression des paramètres non fonctionnels, l'un des critères de la précision de la composition concrète.
- Outillage : reflète le niveau de facilité et de maniabilité de l'utilisation de l'approche.

Critères	**OWL-S**	**WSMO**	**WSDL-S**	**SAWSDL**
Niveau d'abstraction	Conceptuel (basé sur une ontologie générique de services Web)	Conceptuel (basé sur les spécifications du Meta-Object Facility MOF [OMG, 2006])	Conceptuel (basé sur l'annotation sémantique de services Web WSDL)	Conceptuel (basé sur l'annotation sémantique de WSDL et BPEL)
Niveau d'ouverture (standardisation)	Très fort (basé sur OWL, WSDL)	Faible (manque d'intégration avec les standards existants)	Fort (basé sur OWL, WSDL, mais WSDL-S n'est pas un standard)	Fort (basé sur OWL, WSDL et BPEL)
Description Sémantique	Ontologie générique OWL-S	Ontologie WSMO	Annotation des fichiers WSDL	Annotation des fichiers WSDL et BPEL
Eléments considérés dans l'annotation sémantique	Input/Output, Préconditions/Effets	Input/Output, Préconditions/Effets	Input/Output, Préconditions/Effets	Input/Output
Support des PnF	Service Profile	Tous les éléments	Non supportés	Non supportés

Outillage	Matcher, éditeurs	Outil d'annotation, validateurs…	Editeurs	Editeurs

Tableau 4. 1: Comparaison des approches de description sémantique des services Web

La synthèse de la comparaison entre les approches de description sémantique est donnée ci-après:

WSDL-S et SAWSDL : ces deux langages représentent une extension de WSDL pour établir un support simple et générique qui consiste à ajouter des descriptions sémantiques pour les services Web, ils ne permettent pas la définition des propriétés non fonctionnelles ce qui limite la précision de la sélection des services concrets. En plus SAWSDL ne supporte pas la définition des Préconditions/Effets, ce qui restreint sérieusement les possibilités d'annotation du comportement fonctionnel des services Web (type de service). Pour une découverte et une planification précise les Préconditions/Effets sont essentiels [Steinmetz, 2008].

OWL-S et WSMO : représentent des approches révolutionnaires puisque tous les éléments relatifs aux services sémantiques sont reconsidérés, elles sont basées sur des fondements conceptuels pour la réalisation des services Web sémantiques. Les deux approches permettent une description sémantique précise basée sur des ontologies (Ontologie générique OWL-S et Ontologie WSMO) et permettent la modélisation des paramètres non fonctionnels. Cependant WSMO présente quelques limites par rapport à OWL-S, l'une des limites les plus importantes est le fait que cette approche ignore les standards existants, ce qui suppose que des mappings doivent être définis afin de garantir l'intégration avec les standards existants. En termes d'outillage, OWL-S tire parti de son ancienneté, ainsi il dispose de plusieurs outils allant de simple éditeur au composeur semi-automatique en passant par les outils de Matching et de validation. Alors que WSMO ne dispose que des outils pour l'édition, *WSML Editor*, aussi les outils associés à WSMO sont plus difficiles à développer et à utiliser car ils se basent sur WSML, un langage qui n'a pas été utilisé auparavant, tandis que OWL-S s'appuie sur RDF et XML.

Nous retenons donc OWL-S comme langage pour décrire sémantiquement les services Web; le modèle ServiceProfile d'OWL-S sera utilisé pour présenter les types des services Web en termes Input/Output/préconditions/Effets (IOPEs), la requête sémantique sera elle aussi décrite sur le même modèle. Tandis que des attributs de profil, qui peuvent contenir d'autres paramètres que les IOPEs, peuvent être utilisés pour décrire les PnF. Enfin nous utilisons le ServiceModel et le ServiceGrounding pour spécifier les instances des services Web.

4.2.2. Communication inter-agents

Le comportement intelligent et la progression du processus de composition dépendent en grande partie des interactions inter agents, de ce fait chaque agent sera doté d'un module de communication responsable sur le transport des messages avec les autres agents.

Dans notre travail, nous utilisons le langage FIPA-ACL [FIPA-ACL, 2016] comme langage de communication entre les différents agents. Celui-ci a pour rôle de structurer les messages construits par un agent donné. En le comparant à d'autres langages de communication comme KQML (Knowledge Query and Manipulation Language) [Finin et al., 1994], le langage FIPA-

ACL présente l'avantage d'une sémantique plus rigoureuse ainsi qu'un effort important de standardisation.

Par ailleurs, les agents communiquent sur le web via le protocole de transport standard SOAP.

4.2.3. Techniques de la composition abstraites

Etant l'objectif de la composition abstraite, la construction d'un plan abstrait nécessite différentes techniques. Dans le reste de ce chapitre les trois techniques qui pilotent ce premier niveau de composition seront détaillées, notamment :

- **L'analyse des spécifications :** le mécanisme par lequel une demande exprimée par un utilisateur est transformée en une requête compréhensible par le système.
- **Mécanisme de découverte**: le processus de pré-planification qui permet de sélectionner les types de services qui ont le potentiel de participer à la composition.
- **Modèle de planification :** une sous-architecture et un ensemble de mécanismes utilisant les techniques de planification (PIA) pour construire un plan de composition abstrait, i.e un plan définissant les services qui, une fois composés dans l'ordre défini, offrent le service demandé.

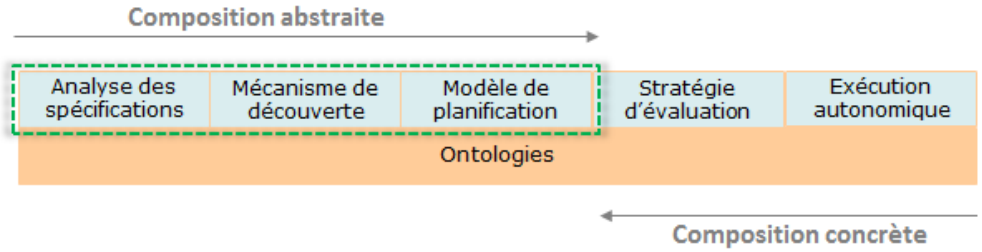


Fig.4. 1: Techniques de composition des services Web

4.3. Analyse des spécifications

A même de construire le plan du premier niveau de composition, le composeur abstrait a besoin d'une formulation sémantique de la requête utilisateur afin qu'il puisse la comprendre et la traiter, une technique de traduction des spécifications est donc de mise. Dans cette partie nous mettons la lumière sur le processus de traduction utilisé par la phase d'analyse qui transforme les spécifications du service demandé en une spécification sémantique comprise par le système.

4.3.1. Processus de la traduction des spécifications

La phase de l'analyse est pilotée par l'agent Request Handler Agent (RHA) dont la responsabilité est de construire sémantiquement la requête utilisateur. Pour ce faire, il suit un processus de trois étapes illustré dans la figure suivante:

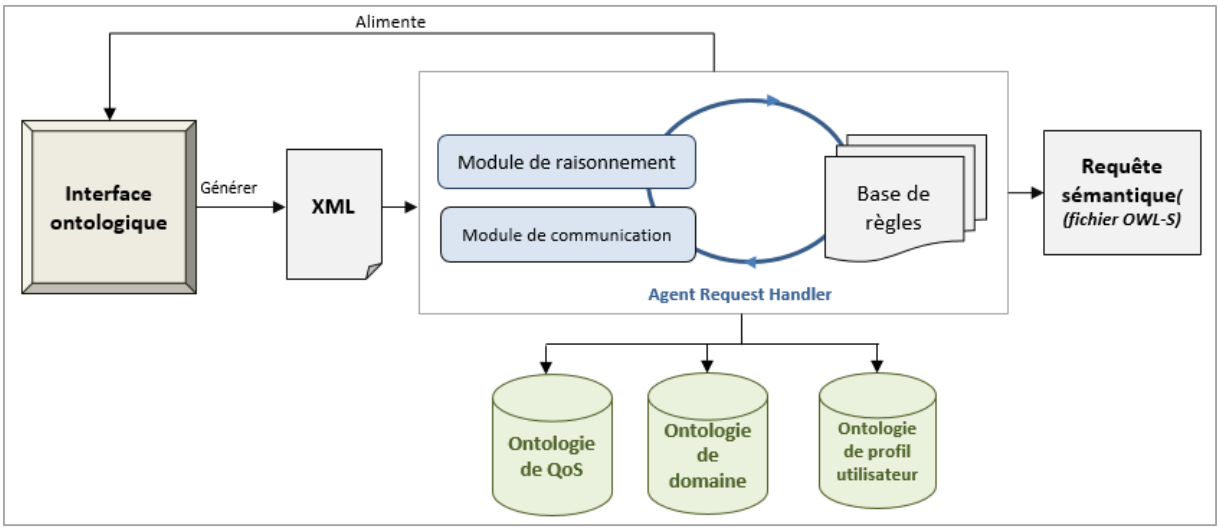


Fig.4. 2: Processus de traduction des spécifications

La **Fig.4. 2** décrit la structure de l'agent RHA composée de :

- Une base de règles qui inclut un ensemble des règles que le module de raisonnement utilise pour effectuer les transformations nécessaires pour obtenir le fichier OWL-S attendu.
- Un module de raisonnement qui constitue le cerveau de l'agent, il permet de raisonner en utilisant les différentes règles ainsi que les ontologies de domaine et de profil de l'utilisateur, afin de construire la requête sémantique.
- Un module de communication qui permet le transport des messages aux autres agents (cf. section précédente).

Cette structure permet à l'agent RHA de traduire la requête exprimée par l'utilisateur en une requête sémantique exprimée en termes d'IOPE en passant par trois principales étapes : (i) la génération du fichier XML, (ii) la traduction des préférences et des données métriques et (iii) le complément automatique de la requête sémantique. Ces étapes sont détaillées ci-après.

4.3.1.1. Génération du fichier XML

Notre travail n'aborde pas les problèmes relatifs au traitement automatique du langage naturel, à cet effet et afin d'exprimer son souhait, l'utilisateur choisit les termes à utiliser à partir d'une liste proposée par une interface de communication, ces termes sont alimentés à partir de l'ontologie de domaine et l'ontologie de QoS, c'est pour cette raison que l'interface intermédiaire est qualifiée d'ontologique (Ontology-Guided Interface). Cette interface se caractérise également par ses propositions dynamiques et incrémentales des termes sélectionnés et ce, en se basant sur la hiérarchie et les relations entre les concepts définies dans l'ontologie de domaine.

La description de la requête doit inclure les besoins fonctionnels et non fonctionnels, ainsi l'interface de communication doit être conçue de manière à accueillir la description des caractéristiques fonctionnelles de la requête (l'intitulé du service sa description, ses entrées et ses sorties) d'une part, et les préférences de l'utilisateur d'autre part. L'utilisateur n'est pas obligé de donner une valeur pour chacune de ses préférences mais il doit nécessairement avoir un moyen pour indiquer qu'une préférence donnée est plus importante (prioritaire) qu'une autre, pour ce faire l'interface propose un intervalle de poids de 1 à 5 pour que l'utilisateur puisse noter la priorité de chaque préférence renseignée, les préférences avec des poids élevés sont celles les plus importantes.

Une maquette de l'interface de communication ontologique conçue est illustrée dans la Fig.4. 3.

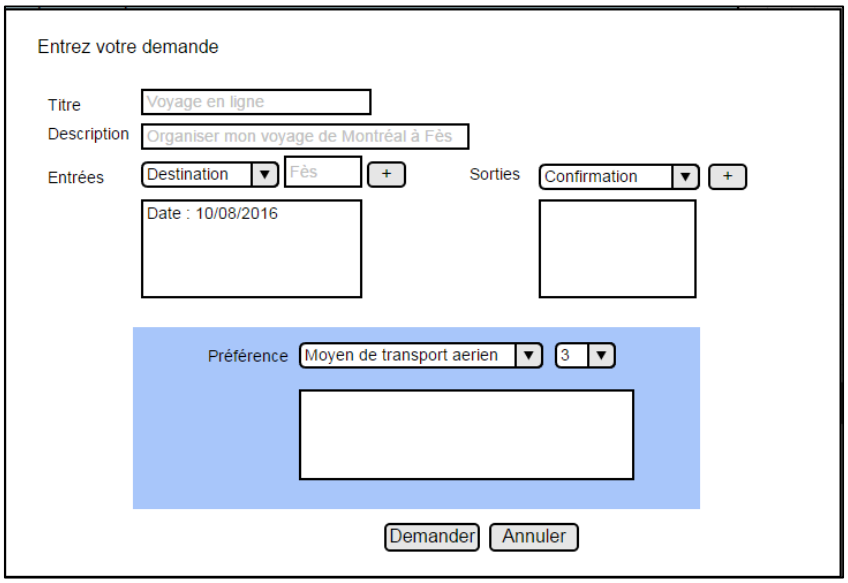


Fig.4. 3: Maquette de l'interface de communication ontologique

Une fois les besoins sont renseignés et la demande est soumise par l'utilisateur, un fichier XML contenant le détail des informations saisies à travers l'interface ontologique est généré.

4.3.1.2. Traduction des préférences et des données métriques

La structure du fichier XML généré contenant les éléments renseignés par l'utilisateur est définie ainsi :

```
<!-- Definition de la requête de l'utilisateurs-->
<request>
    <!-- Identifiant unique de la requête, automatiquement généré-->
    <ID> xxx </ID>
    <description> texte </description>
    <!-- Definition des  paramètres de la requête-->
    <Params>
          <Param>
               <type>in1</type>
               <value>v1</value>
          </Param>
          <Param>
               <type>out1</type>

          </Param>
    </Params>
   <!-- Definition des  préférences de l'utilisateurs-->
    <preferences>
          <preference>
             <type> Pr1</type>
             <!-- Une valeur de 1 à 5-->
             <weight> 2 </weight>
          </preferences>

   </preferences>
</request>
```

Fig.4. 4: Structure du fichier XML contenant la requête de l'utilisateur

L'agent explore alors ce fichier en utilisant les règles de traduction stockées dans la base des règles pour définir la requête sémantique. Nous énonçons ci-après les deux principales règles de traduction utilisées:

Règle1 : *Pour chaque balise <preference>, la sous balise <type> définit le type de la préférence : Pr_i et la sous balise <weight> définit le poids correspondant : w_i, ajouter alors à la liste PRF, qui stocke les préférences, l'enregistrement suivant : < Pr_i, w_i>.*

En appliquant cette règle, on obtient la liste des préférences et leurs poids, cette liste est ensuite encapsulée dans un message et envoyée directement à l'agent Evaluator si aucune préférence automatique n'est proposée.

Règle2 : Les balises <param> avec des sous balises <value> sont des entrés (input) de la requête, et les balises <param> sans sous balises <value> représentent les sorties (outputs) de la requête

Sachant que la structure de la requête sémantique (fonctionnelle) cible est définie ainsi :
R = {R.in, R.out, R.pre, R.effe}, où R.in, R.out, R.pre et R.effe sont respectivement la description sémantique des entrées, des sorties, des préconditions et des effets du service demandé. Cette règle permet alors de définir la partie métrique (inputs et outputs) de la requête sémantique.

4.3.1.3. Complément automatique de la requête sémantique

Maintenant que les préférences sont initiées et les inputs et outputs sont déterminés, l'agent RHA complète automatiquement la définition sémantique de la requête en se basant sur l'ontologie de domaine, en intégrant les préconditions nécessaires et les effets possibles associés aux concepts que représentent les inputs et outputs définis. L'ontologie du profil utilisateur est utilisée ensuite pour raffiner les préconditions et les effets identifiés, les

éléments stockés dans cette ontologie contribuent également à l'enrichissement automatique des préférences de l'utilisateur. Une emphase sur cette ontologie est donnée ci-après.

Le processus ainsi défini permet à l'agent reconstructeur de requêtes (RHA) de transformer le fichier XML représentant la requête utilisateur en une requête sémantique. Cette requête, qui représente le point de départ de tout un éventuel processus de composition, est encapsulée dans un message et envoyée au Discover tandis que la liste des préférences est envoyée directement à l'Evaluator.

4.3.2. L'Ontologie « profil utilisateur »

Un profil utilisateur est une représentation numérique des données uniques relatives à un utilisateur particulier. Modéliser un profil utilisateur nécessite la création d'une structure de donnée capable de stocker les attributs caractérisant un type d'utilisateur. En se basant sur leur capacité de modéliser les connaissances, les ontologies sont le meilleur candidat pour assurer efficacement le rôle de cette structure de données.

L'ontologie profil utilisateur peut alors être définie comme étant une structure qui permet de modéliser et de stocker sémantiquement les informations caractérisant un utilisateur. Ces données représentent les informations personnelles, les centres d'intérêts, les préférences d'un utilisateur ou d'un groupe d'utilisateurs dans un domaine bien défini. La génération et la mise à jour du profil utilisateur peut être effectuée en utilisant le feed-back de l'utilisateur, ceci peut être fait de deux façons : soit d'une façon statique en demandant à l'utilisateur de donner explicitement son avis sur la pertinence du services offerts (feed-back explicite), soit d'une façon automatique, avec cette dernière approche l'utilisateur n'est pas sollicité pour spécifier son retour, ce sont ses interactions avec le système qui sont capturées et analysées afin d'en extraire les intérêts et les préférences de l'utilisateur (feed-back implicite), cette opération peut être effectuée en utilisant des méthodes et des techniques du Web-usage mining (WUM) [Bosnjak et al., 2010].

Afin d'offrir à l'utilisateur des services personnalisés et contextualisés sans demander à l'utilisateur de construire son profil d'une manière explicite ou de définir ses préférences de façon exhaustive, le besoin de concevoir une ontologie pour le profil utilisateur a été identifié. La création de telle ontologie, planifiée dans une étape ultérieure de ce travail, dépend du domaine d'application du système de composition proposé d'une part, et nécessite des connaissances solides dans les domaines de l'Information Retreiving [Manning et al. ,2009] et de fouille de Web ainsi qu'un effort considérable de conception d'autre part. Elle demeure néanmoins une condition nécessaire pour renforcer la dimension « adaptation » du système de composition proposé et ce, pour améliorer la qualité des services offerts et pour augmenter la satisfaction des utilisateurs.

4.3.3. Exemple du voyage : analyse des spécifications

Pour conclure cette section nous présentons sémantiquement la requête de l'exemple du voyage qui représente le point de départ du processus de composition illustratif. Rappelons que l'utilisateur dans l'exemple veut se déplacer de Montréal à Fès à une date bien précise.

Nous supposons que l'utilisateur préfère voyager par un moyen de transport aérien avec un poids 4 et qu'il souhaite que le coût soit minimal avec un poids 1.

La requête générée relative au présent exemple peut être exprimée ainsi :

R_{voy}= {
R_{voy}. **in** : {date : 01/08/2017, destination : Maroc-Fès}
R_{voy} **.out** : {confirmation}
R_{voy}.**pre** = (être à Montréal, avoir crédit}
R_{voy}.**effe**= {être à Fès}
R_{voy}.**preferences** : {moyens de transport atériens : 4 , moindre coût :1}

4.4. Mécanisme de découverte

Théoriquement, la liste des services traités par le Planner pour construire le plan abstrait, doit inclure tous les types de services publiés dans l'annuaire des SWS. Néanmoins, ceci devient de plus en plus inapplicable avec l'augmentation du nombre des services publiés dans l'annuaire. Afin de surmonter ce problème, nous proposons de réduire le nombre de services passés au Planificateur. Pour ce faire l'agent Discover cherche des services candidats qui ont le potentiel de participer au plan de composition, cette opération, nommée la découverte, est guidée par un algorithme de Matchmaking qui se base sur des techniques sémantiques qui font de la découverte un mécanisme de filtrage dynamique et automatique.

Dans cette section nous détaillons le principe du mécanisme de découverte proposé, nous présentons l'algorithme de Matchmaking conçu et nous discutons sa performance en termes de complexité, de complétude et de précision afin de prouver son efficacité.

4.4.1. Principes et caractéristiques du mécanisme de la découverte

L'objectif ultime de la composition abstraite est de construire le plan abstrait. Il est important de noter, toutefois, que le modèle de planification utilisé pour construire ce plan, implique des mécanismes complexes : d'abord en termes d'utilisation de ressources, puisque l'approche de planification multi-agents nécessite de créer pour chaque service un agent qui le simule, ensuite en termes de complexité de calcul qui est généralement couteuse. C'est pour cette raison que nous avons jugé nécessaire d'avoir une phase préparatoire visant à optimiser et alléger le processus de construction de plan. Dans cette phase le Discover outillé d'un mécanisme de découverte prend en charge le filtrage des services publiés dans l'annuaire des services sémantiques pour exclure tout service non pertinent

En effet, donnant une requête sémantique, le Discover trouve les candidats potentiels pour résoudre la requête sans vraiment chercher la solution. Un candidat potentiel est un type de service qui contribue directement au but de la requête (au minimum un de ses effets et/ou sorties correspond au but) ou aux préconditions et/ou entrées d'un service type qui potentiellement contribue au but.

Le mécanisme de découverte est guidé par un algorithme dit de Matchmaking (correspondance) et se démarque par les quatre principales caractéristiques :

A. Nous envisageons d'effectuer l'opération de filtrage avant la planification :

Une autre alternative serait d'effectuer la découverte au besoin pendant la planification [Sirin et al., 2004]. Ceci risque de conduire à une faible performance de l'opération de planification en raison de nombre d'opérations supplémentaires requises pour explorer, à chaque fois lors de la construction de plan et l'annuaire des services sémantiques. Par ailleurs, et d'un point de vue bonnes pratiques, l'attribution de la tâche « chercher des services » et la tâche « raisonner sur leur ordre » au même agent présente une violation du principe d'Unique Responsabilité [Martin, 2003]. En effet, coupler ces deux responsabilités risque de porter atteinte à la cohésion et à la modularité de l'architecture proposée contrairement à ce qui est exigé (c.f chapitre 3). Notre vision s'attache, bien au contraire, de respecter le principe «diviser pour régner » [Martin, 2003], en proposant des techniques qui répondent aux différents sous problèmes de composition identifiés d'une manière indépendante. Ainsi la substitution ou l'optimisation d'une technique n'impactera pas les autres techniques. La façon la plus judicieuse d'appliquer cette vision à ce niveau est d'encapsuler la mission de filtrage des services dans une phase à part entière et de la placer juste avant la phase de génération de plan.

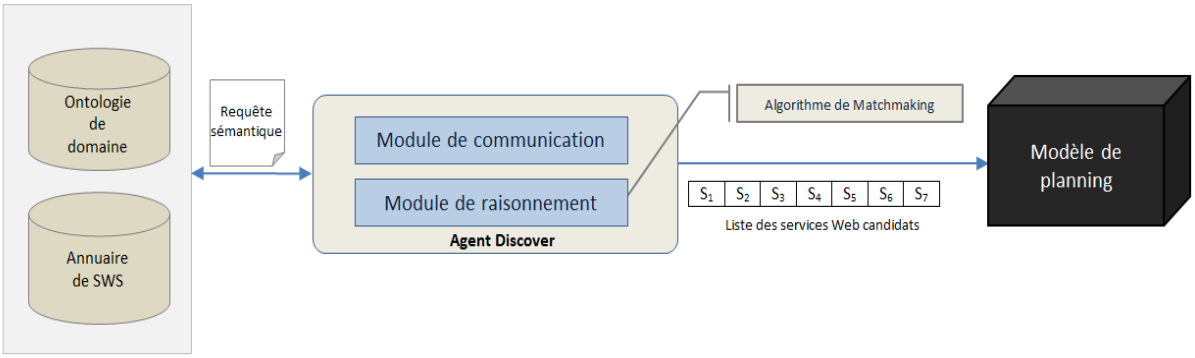


Fig.4. 5: Mécanisme de la découverte des services Web

Comme illustré au niveau de la Fig.4. 5, en considérant le modèle de planning une boite noire, l'agent Discover, l'acteur principal de cette phase, réagit indépendamment en se basant sur un algorithme de Matchmaking nécessitant en entrée une requête, un annuaire de services abstrait et produisant en sortie une liste des services candidats.

B. Le mécanisme proposé utilise une mesure sémantique du degré de correspondance pour évaluer la pertinence des services :

Le mécanisme de découverte proposé se veut être un mécanisme sémantique dans la mesure où il implémente une découverte approximative. En effet, il est très restrictif de ne prendre en charge que les services qui correspondent exactement aux besoins spécifiés. Il est à la charge de l'algorithme de «Matchmaking» d'effectuer les opérations de filtrage nécessaires pour retourner des services composables qui forment un résultat proche de la requête -Proche au sens sémantique-. Pour ce faire nous définissons ce qu'on appelle le degré de correspondance sémantique en utilisant la théorie de «subsumption reasoning» telle qu'introduite par [Paolucci et al., 2002]. Ceci dit, deux services sont proches sémantiquement s'il y'a une certaine correspondance sémantique qui les lie, une correspondance sémantique existe entre

deux services s'il y'a une correspondance sémantique entres leurs paramètres sémantiques (IOPE). Le degré de correspondance sémantique entre deux paramètres IOPE dépend de la correspondance sémantique entre les concepts associés à ces paramètres et, la correspondance sémantique entre deux concepts donnés est basée sur la relation liant ces deux concepts au niveau de leur ontologie de domaine.

Pour illustrer cette notion de correspondance sémantique, considérons, par exemple, un service Web de vente de véhicules (Selling vehicule Web Service) dont le paramètre de sortie est «Véhicule», et une requête de recherche dont le paramètre de sortie est spécifié comme « Voiture ». Bien qu'il n'y a pas de correspondance exacte entre le paramètre de sortie de la requête et celui du service Web, étant donné le fragment de l'ontologie de véhicule donnée ci-après (Fig.4. 6), le mécanisme de découverte peut déterminer un autre niveau de correspondance entre les deux paramètres puisque le concept « Vehicule » englobe le concept « Voiture ».

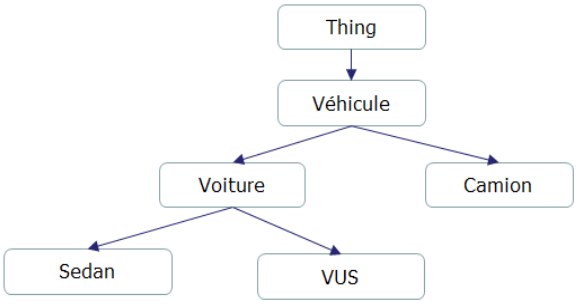


Fig.4. 6: Ontologie de véhicule

Nous distinguons quatre niveaux de correspondance sémantique entre deux concepts C_1 et C_2:

- **Exact**: Si C1 et C2 sont les mêmes classes ou si C1 est une sous classe immédiate de C2 dans l'ontologie du domaine ;
- **Plug in** : Si C2 englobe (subsume) C1 ;
- **Subsume** : Si C1 englobe (subsume) C2;
- **Disjoint** : S'il n'y a aucune relation entre C2 et C1.

Formellement, nous définissons la fonction degreeOfMatch telle que :

```
degreeOfMatch(outR,outA)
{
if outA=outR then return Exact
if outR subclassOf outA then return Exact
if outA subsumes outR then return PlugIn
if outR subsumes outA then return subsumes
otherwise fail
}
```

Fig.4. 7: Règles d'assignation du niveau de correspondance

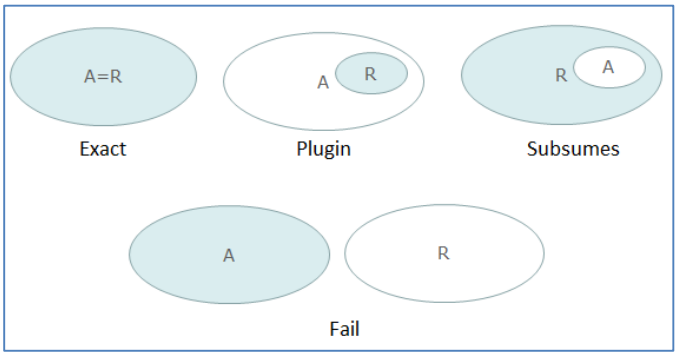


Fig.4. 8: Illustration des degrés de correspondance sémantique

Notons que de point de vue correspondance entre une offre C_1 et une demande C_2, les deux premiers niveaux de correspondance peuvent être considérés équivalents, puisque dans les deux cas l'offre satisfait la demande. Par contre, le troisième niveau de correspondance ne satisfait que partiellement la demande.

Si la correspondance entre les paramètres d'entrée/sortie (I/O) peut être résolue de la sorte, celle des préconditions/effets s'avère plus compliquée; les préconditions et les effets ne sont pas des concepts mais des conditions, leur principe de correspondance diffère de celui d'une signature d'I/O. Afin de pouvoir être filtrées, les préconditions/effets (P/E) sont d'abord décomposées en des listes d'atomes, les atomes de chaque liste doivent ensuite être identifiées (par exemple ‘‘Client {AvoirCarte} VisaCard", les atomes dans ce cas seront « Client » et « VisaCard »).Nous utilisons des clauses de Horn [Delgrande et al., 2013] pour modéliser les listes d'atomes. Chaque membre des listes des P/E est considéré comme un ternaire a(n) (atome, prédicat, atome). Le résultat de correspondance serait donc le résultat de correspondance sémantique de chaque élément du ternaire.

C. Les filtres utilisés se basent aussi bien sur les préconditions et les effets des services que sur les paramètres d'entrée/sortie :

Cette considération représente une caractéristique distinctive de l'approche de filtrage proposée. En effet, actuellement la plupart des algorithmes de Matchmaking [Emani, 2006] se limitent uniquement aux paramètres d'entrées et de sorties des services dans leur traitement, alors que la prise en compte des autres caractéristiques sémantiques des services peut améliorer considérablement la qualité et la flexibilité du filtrage. Selon notre point de vue, l'intégration des paramètres P/E (préconditions et effets) dans l'opération de correspondance des services offre principalement trois avantages :

(i) D'abord, la correspondance utilisant uniquement les I/O considère seulement les classes des entrées/sorties du service, toutefois il peut bien y avoir des services possédant des I/O qui appartiennent aux mêmes classes mais qui jouent des rôles tout à fait différents. Ce type de correspondance simple manque donc de précision et d'exactitude.

(ii) En plus, il n'est pas toujours optimal de raisonner sur l'ensemble des paramètres I/O d'un service, les services qui satisfont la même requête non pas nécessairement les mêmes paramètres d'entrée/sortie. Par exemple un service d'achat de livres peut demander en entrée le nom de livre et l'auteur, un autre peut demander en plus l'ID du client, si on raisonne avec la correspondance simple des I/O, ces deux services ne seront pas sur le même niveau de similarité en les comparant avec une requête spécifiant

seulement le nom du livre et de l'auteur, alors qu'au point de vue satisfaction du besoin réel de l'utilisateur, qui est d'« acheter un livre », les deux services sont valides.

(iii) Finalement, l'utilisation des préconditions et des effets permet de raisonner sur la sémantique de ce type de paramètre. En pratique, il n'est pas nécessaire de trouver un service qui produit exactement l'effet demandé. Il est suffisant de découvrir un service dont l'effet implique l'effet spécifié. De la même manière on peut seulement se contenter de trouver un service dont les préconditions sont impliquées par les préconditions spécifiées. Seule la correspondance des P/E peut prendre en considération ce fait. Par exemple, si le client demande un service pour faire une radiographie de son doigt, c'est tout à fait acceptable de lui proposer un service de radiographie de la main ou du membre supérieur, mais puisque le doigt n'est pas une sous classe de la main ni du membre supérieur, en considérant la correspondance simple ces deux services ne seront pas filtrés. Plus d'exactitude et plus de souplesse sont donc apportées par un filtrage prenant en compte les P/E.

D. Le principe de Matchmaking proposé rompre avec la logique des algorithmes conventionnels de correspondance :

Les algorithmes de Matchmaking sont considérés comme l'outil privilégié pour faire des filtrages, il en existe une multitude, mais ils obéissent tous au même principe, celui introduit par l'algorithme original proposé par [Srinivasan et al., 2004]. Dans sa version initiale ce dernier utilise uniquement une correspondance simple des I/O, mais il est tout à fait possible, en suivant sa logique, de l'étendre par l'introduction de la correspondance P/E. Il s'énonce ainsi : "*An Advertisement and a Query match if their Outputs and Inputs both, match*".

Formellement, un service Web Advt correspond à une requête si les IOPE du service correspondent à leurs homologues de la requête (algorithme ci-après) [Srinivasan et al., 2004].

$$\forall c \in Advt_{in}, \exists d \in Query_{in}, \\ s.t.\ match(c,d) \neq Fail$$

$$\forall c \in Query_{out}, \exists d \in Advt_{out}, \\ s.t.\ match(c,d) \neq Fail$$

$$\forall c \in Advt_{precondition}, \exists d \in Query_{precondition}, \\ s.t.\ match(c,d) \neq Fail$$

$$\forall c \in Query_{effect}, \exists d \in Advt_{effect}, \\ s.t.\ match(c,d) \neq Fail$$

Fig.4. 9: Algorithme classique de correspondance [Srinivasan et al., 2004]

Puisque le but de cette phase de découverte est de déterminer une chaine de services potentiellement composables et non pas une correspondance directe avec la requête, cet algorithme est inapplicable dans notre cas. Nous proposons alors un nouveau algorithme de Matchmaking qui, à contrario de l'approche présentée ci-avant, fait correspondre les éléments **d'entrée** sémantiques (entrées et préconditions) d'un service abstrait avec les

éléments de **sortie** sémantiques (sorties et effets) d'un autre service. Le principe de correspondance présenté ci-avant est, toutefois, utilisé au premier et au dernier niveau de filtrage pour boucler le chainage des services découverts avec la requête. L'intégralité de l'algorithme conçu sera présentée et détaillée dans la section suivante.

4.4.2. L'algorithme de Matchmaking proposé

- A un niveau d'abstraction élevé, le problème à résoudre par l'algorithme de Matchmaking peut être décrit ainsi :
  - Donnons une requête sémantique : R = {R.in, R.out, R.pre, R.effe}, où R.in, R.out, R.pre et R.effe sont respectivement les entrées, les sorties, les préconditions et les effets de la requête.
  - Et une liste S = {s_1, ..., $s\alpha$} des services abstraits disponibles dans l'annuaire des SWS.
  - Sachant que chaque service $s_i \in S$ est un quadruplet : s_i = {s_i.in, s_i.out, s_i.pre, s_i.effe}, où s_i.in, s_i.out, s_i.pre et s_i.effe sont respectivement les entrées, les sorties, les préconditions et les effets du type de service.

  ➔ Sélectionner (filtrer) un nombre *n* de types de services à partir de S, tel que l'espace de recherche est réduit mais le meilleur plan peut encore être construit.

- Afin d'atteindre cet objectif et en se basant sur la notion de correspondance sémantique décrite précédemment, nous définissons une surcharge de l'opérateur « = », pour simuler mathématiquement les filtres de la fonction de correspondance degreeOfmatch(X, Y), comme suit :

$$C_1 = C_2 \begin{cases} \text{si degreeOfMatch}(C_1, C_2) = \textbf{Exact} \\ \text{Si degreeOfMatch}(C_1, C_2) = \textbf{PlugIn} \\ \text{Sinon } C_1 \neq C_2 \end{cases}$$

C_1 et C_2 sont deux concepts appartenant à l'ontologie de domaine.

Les opérateurs « $\subset$», « $\cap$ », sont par conséquent définis en considérant le nouveau opérateur «=», par exemple pour que s_x.in : la liste des entrées d'un service s_x, soit incluse dans s_y.in : la liste des entrées d'un autre service s_y, il faut que pour toute entrée de s_x , il existe une entrée de s_y.in qui lui est égale « = » dans le sens défini ci-avant, nous avons donc l'équivalence suivante :

$$s_x.\text{in} \subset s_y.\text{in} \Leftrightarrow \forall\ s_x.\text{in}_i \in s_x.\text{in}, \exists\ s_y.\text{in}_j \in s_y.\text{in}\ \ \textit{telque}\ \ s_x.\text{in}_i = s_y.\text{in}_j$$

L'algorithme proposé (Fig.4. 10) se base sur l'idée basique qui stipule que quel que soit le modèle de planification utilisé, le plan de composition peut être représenté sous forme d'un graphe, par conséquent, afin d'appartenir à ce graphe un service abstrait a besoin d'une part, d'extraire ses paramètres d'entrées et ses préconditions des nœuds précédents et d'autre part, ses paramètres de sorties et ses effets doivent vérifier à leur tour les entrées et les préconditions des nœuds successeurs. Pour ce faire l'algorithme construit deux communautés de services abstraits, tels que les paramètres d'entrée et les préconditions des services

appartenant à la première communauté peuvent être tirés (directement ou indirectement) des entrées et des préconditions de la requête sémantique (2) et, les sorties et les effets des services appartenant à la deuxième communauté sont vérifiés par les sorties et les effets (directement ou indirectement) de la requête sémantique (3). Evidemment, l'intersection des deux communautés appartient au graphe de planification, puisque les services qui se trouvent à l'intersection des deux communautés possèdent la propriété suivante : leurs IOPE correspondent sémantiquement aux IOPE de la requête (4).

Finalement il est important de noter qu'en plus du filtrage des services candidats, l'algorithme permet également de donner une réponse efficace à une question décisive: **Devons-nous passer par une phase de planification?** Et ce, en vérifiant d'abord et avant tout calcul de correspondance, si un service existant (composite ou simple) est capable de satisfaire la requête passée en argument, si c'est le cas l'Evalutor est directement appelé sans passer par la planification (1). Dans le même sens l'algorithme vérifie également si le Pool des services candidats est vide (5). Dans le premier cas (1) la planification n'est pas nécessaire et dans le deuxième cas (5) la planification n'est pas possible.

Algorithme 1 : Algorithme de Machmaking

Algo *MatchmakingAlgorithm(R, S)*
(1) Si $\exists s_x \in S$ *telque :*
$s_x.in \subset R.in$ *et*
$s_x.pre \subset R.pre$ *et*
$R.out \subset s_x.out =$ *et*
$R.effe \subset s_x.effe$
Alors appler_Evaluator (s_x);*// planification n'est pas nécessaire*

Sinon

(2) Construire ***PreviousTree*** *tel que:*
– *R est le noeud racine*
– $\forall s_i \in S, s_i \in$ *PreviousTree si et seulement si:*
$s_i.pre \subset R.pre$ *ou* $s_i.in \subset R.in$ *alors child(R) =* s_i
Ou $\exists s_j \in$ *PreviousTree où :*
$s_i.pre \cap s_j.effe \neq \emptyset$ *ou* $s_i.in \cap s_j.out \neq \emptyset$ *alors child* (s_j) = s_i

(3) Construire ***NextTree*** *tel que:*
– *R est le noeud racine*
– $\forall s_i' \in S, s_i' \in$ *NextTree si et seulement si:*
$s_i'.effe \cap R.effe \neq \emptyset$ *ou* $s_i.out \subset R.out$ *alors child(R)=* s_i'
ou $\exists s_j' \in$ *NextTree où:*
$s_i'.effe \cap s_j'.pre \neq \emptyset$ *ou* $s_i'.out \cap s_j'.in \neq \emptyset$ *alors child*(s_j')=s_i'

(4) Construire ***CandidatePool*** *tel que:*
$\forall s_i \in S, s_i \in$ *CandidatePool si :*
– $s_i \in$ *(PreviousTree* $\cap$ *NextTree)*
– *ou* $s_i \in$ *PreviousTree et* $\exists s_{ii} \in$ *CandidatePool telque child(si)=* s_{ii}
– *ou* $s_i \in$ *NextTree et* $\exists s_{ii} \in$ *CandidatePool telque child*(s_i)= s_{ii}
(5) ***si CandidatePool*** *est vide // la planification n'est pas possible*
retourner *Echec*

sinon *appler_Planner (R,* ***CandidatePool****);*

Fin de si
Fin de si
Fin algo

Fig.4. 10: Algorithme de Matchmaking proposé

4.4.3. Evaluation du mécanisme de la découverte

D'abord, il est important de noter que cet algorithme est une **optimisation** en soi-même, il améliore les calculs engendrés par le modèle de planification en lui proposant en entrée un nombre optimisé de services candidats à gérer, et par conséquent il améliore l'efficacité du processus global de la composition en maintenant son exactitude et son exhaustivité. En effet, l'optimisation à ce stade de pré-planification vise la réduction d'espace de recherche, ceci est effectué d'abord, en adoptant une approche de composition à double niveaux, où la composition abstraite raisonne seulement et uniquement en termes de types de services et ne laisse que les services possédants des IOPEs différents être traités (cf chapitre 3). L'algorithme de Matchmaking assure, quant à lui, un deuxième niveau d'optimisation de l'espace de recherche de planification en construisant à partir des services abstraits un Pool de services qui ont le potentiel de participer au plan abstrait, ce Pool constitue le nouveau espace de planification de l'agent planificateur.

En termes de **complexité**, en cas normal, à chaque itération, l'algorithme doit parcourir l'annuaire des types de services n'appartenant pas à l'arbre previousTree (ou à l'arbre NextTree) à la recherche d'un nouveau élément à ajouter à ces deux arbres. Ce processus s'arrête si aucun élément n'a été rajouté lors de la dernière itération ou si l'espace à explorer devient vide. Donc au pire des cas où tous les services de l'annuaire sont des candidats et que à chaque itération on ajoute exactement un seul service au Pool (ce qui est peu probable), l'annuaire de *n* services doit être parcouru n fois ce qui donne une complexité au pire des cas d'ordre de $o(n^2)$. Cette complexité peut être optimisée par une meilleure utilisation des structures données pour l'exploration de l'annuaire.

En plus de la complexité, ce qui est également intéressant de discuter dans le cadre de cette évaluation d'algorithme, c'est la capacité de ce dernier d'améliorer sa performance en évitant les faux positifs (i.e., les types services retenus lors du filtrage alors qu'ils ne sont pas pertinents) et les faux négatifs (i.e., les types services qui sont pertinents mais qui ne sont pas inclus dans la liste des services filtrés). Les résultats des faux positifs et des faux négatifs sont utilisés respectivement pour mesurer/estimer le niveau de **précision et** le niveau **de complétude** (completeness) du mécanisme de découverte.

En fait, l'existence des **faux positifs** ne présente aucun risque sur la réussite du processus de composition, cela veut juste dire que des services, qui ne participeront pas dans le processus de planification, ont été sélectionnés. Les faux positifs ne nuisent donc en rien l'exactitude (correctness) de l'algorithme. Toutefois leur présence signifie que l'espace d'exploration de planification n'est pas efficacement réduit, leur élimination demande par contre des efforts de calculs importants qui risquent, en plus de l'augmentation de la complexité, de faire dévier l'objectif visé d'une optimisation de l'espace de planification à une vrai planification en recherchant des solutions et non pas des candidats. Nous nous contentons alors de ce niveau d'optimisation de l'espace de planification qui nous semble un parfait équilibre entre un bon niveau de filtrage et une bonne complexité des calculs.

- A l'inverse des faux positifs, les **faux négatifs**, menacent la réussite de la phase de planification ; cette dernière peut échouer si un service potentiel a échappé lors du filtrage et qu'il n'a pas intégré le pool des candidats. Ce risque ne se présente pas avec le mécanisme de découverte conçu. En effet le mécanisme de filtrage utilisé par l'algorithme de Matchmaking accepte les services candidats qui satisfont directement ou indirectement, complétement ou partiellement la requête on raisonne en termes de sémantique. Cette façon de filtrage ne laisse pas les faux négatifs s'échapper au pool des candidats (explication ci-après). La complétude (completness) de l'algorithme est donc garantie.

Expliquons formellement comment les faux négatifs n'échappent pas au mécanisme de filtrage proposé :

- Soit $s_f \in S$ et telque $s_{f=}\{ s_f.in, s_f.out, s_f.pre, s_f.effe\}$, où $s_f.in$, $s_f.out$, $s_f.pre$ et $s_f.effe$ sont respectivement ses entrées, ses sorties, ses préconditions et ses effets.

- Supposons que S_f est un service candidat, donc il a le potentiel de contribuer à la résolution de la requête sémantique, notamment à la résolution complète ou partielle de ses effets ou ses sorties ceci implique que :
  (1) $s_f.effe \cap R.effe \neq \emptyset$ ou $s_f.out \cap R.out \neq \emptyset$

- S_f peut aussi contribuer indirectement à la résolution de la requête sémantique:
  (2) $s_f.effe \cap s_j.Pre \neq \emptyset$ ou $s_f.out \cap s_j.in \neq \emptyset$ ou s_j est un autre service candidat

- Et puisqu'il est un service candidat, s_f lui aussi comme s_j a besoin que ses entrées et ses préconditions soient vérifiées par des contributeurs indirects:
  (3) $s_i.effe \cap s_f.Pre \neq \emptyset$ ou $s_i.out \cap s_f.in \neq \emptyset$ où s_i est un service candidat qui contribue indirectement à la résolution de la requêter

- Notons finalement qu'il est aussi possible que les entrées/préconditions de s_f correspondent complétement ou partiellement aux entrées/préconditions de la requête :
  (4). $s_f.Pre \cap R.Pre \neq \emptyset$ ou $s_f.in \cap R.out \neq \emptyset$

➔ (1) et (2) impliquent que $s_f \in$NextTree et (3) et (4) impliquent que $s_f \in$PreviousTree donc $s_f \in$(PreviousTree ∩NextTree) ce qui implique que $s_f \in$ CandidatePool.

Comme mentionné précédemment, les faux positifs et les faux négatifs sont utilisés pour mesurer la précision et la complétude de l'algorithme.

En effet, dans le domaine de la reconnaissance de formes [Segond, 2006] et l'information retrieval [Manning et al. ,2009], la **précision** est définie comme étant le nombre des offres pertinents rapporté au nombre total des services pour une requête donnée, donc plus l'algorithme devient flexible dans le filtrage, plus le nombre des faux positifs augmente et plus la précision diminue. Par conséquent, afin d'obtenir une grande précision l'algorithme doit être plus rigide dans la correspondance sémantique.

D'autre part, le **rappel** (Recall) est défini par le nombre des offres pertinentes retrouvées au regard du nombre des offres pertinentes publiées dans l'annuaire. Donc plus l'algorithme devient flexible dans le filtrage, plus le nombre des faux négatifs diminue et plus le rappel augmente. Par conséquent, afin d'obtenir un rappel élevé l'algorithme doit être plus flexible dans la correspondance sémantique.

Il y a donc un compromis entre le nombre des faux positifs et celui des faux négatifs retournés par l'algorithme de Matchmaking. La flexibilité de l'algorithme est proportionnelle à l'augmentation des faux positifs et à la diminution des faux négatifs. Il est donc nécessaire de régulariser la flexibilité de la correspondance afin d'avoir un équilibre entre une bonne précision et un bon rappel.

Finalement, il est également nécessaire de noter que cet algorithme ne prend pas en compte le caractère conditionnel des effets, i.e qu'ils peuvent ne pas être connus à la phase de la découverte. En perspective nous comptons faire évoluer l'algorithme pour qu'il supporte cette contrainte et pour qu'il optimise la scalabilité en réduisant la complexité de calcul.

4.4.4. Exemple du voyage : Mécanisme de découverte

Revenons à notre exemple illustratif. Après avoir défini la requête sémantique (le but) dans la phase d'analyse, nous allons maintenant appliquer l'algorithme de Matchmaking pour identifier les services candidats.

Rappelons d'abord les données d'entrée de l'algorithme :

« L'algorithme de Matchmaking prend en entrée une requête, un annuaire de services abstraits et produisant en sortie une liste des services candidats ».

Nous avons :

R_{voy}= ⌈ R_{voy}. **in** : {date : 01/08/2017, destination : Maroc-Fès}
⌈ R_{voy} **.out** : {confirmation}
⌈ R_{voy}.**pre** = (être à Montréal, avoir crédit)
⌊ R_{voy}.**effe**= { être à Fès)

S est l'annuaire des SWS, qui est peuplé par la description sémantique des services inscrits dans notre système. Chaque service $s_i \in S$ est donc défini ainsi s_i ={ s_i.in, s_i.out, s_i.pre, s_i.effe}, où s_i.in, s_i.out, s_i.pre et s_i.effe sont respectivement les entrées, les sorties, les préconditions et les effets du type de service. Pour la suite nous allons utiliser la notation graphique suivante pour simplifier la représentation de ces attributs sémantiques :

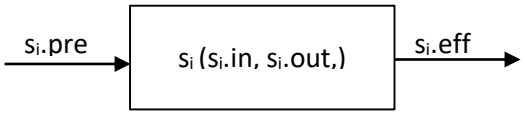


Fig.4. 11: Représentation graphique d'un service

Supposons qu'il n'existe pas de service dans S équivalent à R:

1- Nous allons d'abord construire la liste **PreviousTree** :
Nous cherchons des services avec une précondition équivalente à : R.pre= {Etre à Montréal, avoir crédit} et leurs successeurs (nœuds successeurs). L'exploration de S donne par exemple les services suivants :

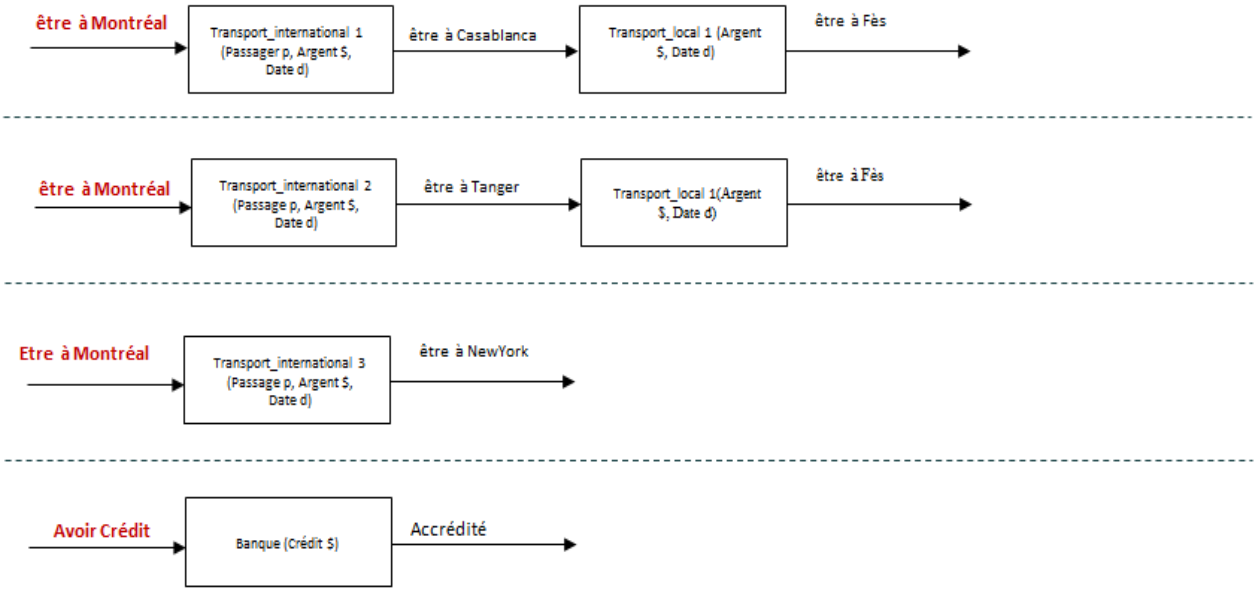


Fig.4. 12: Composition de l'arbre PreviousTree (exemple de voyage)

2- Ensuite, nous allons construire la liste **NextTree**:
Nous cherchons des services avec des effets équivalents à : R.effe= {Etre à Fès} et leurs antécédents (nœuds précédents). L'exploration de l'annuaire des SWS donne par exemple les services suivants :

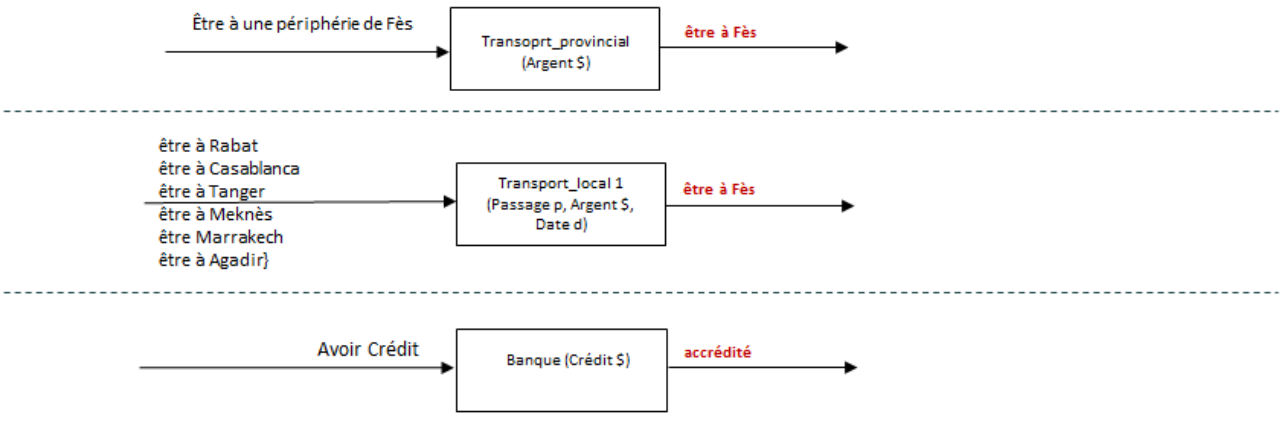


Fig.4. 13: composition de l'arbre NextTree (exemple de voyage)

3- Maintenant, il nous reste qu'à construire le **CandidatePool** formé par l'intersection des deux arbres et les services attachés. Comme illustré dans la figure ci-après le pool de services découverts qui sera passé au Planner contient les services suivants : {Transport_internationel1, Transport_internationel2, Transport_local1, Banque}

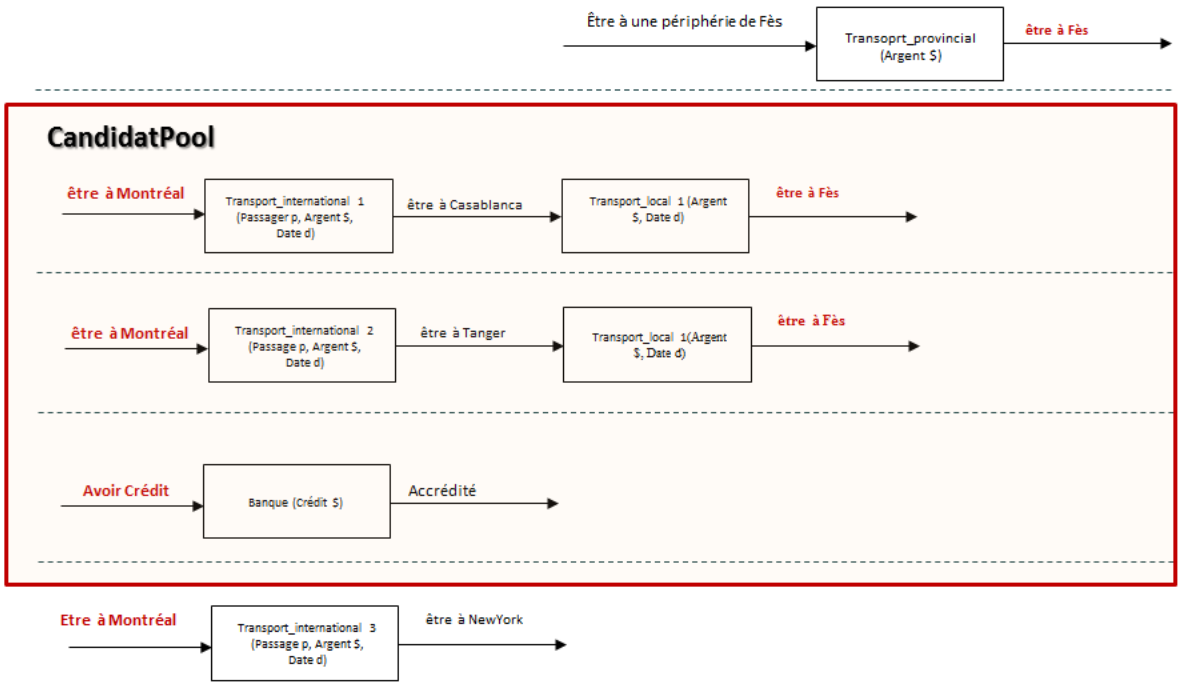


Fig.4. 14: Formulation du CandidatePool (exemple de voyage)

Notons que :

- **Transport_international :** est un service de transport qui relie deux emplacements appartenant aux différents pays : l'avion, le TGV et le bateau sont des exemples de services de cette catégorie de transport, mais tant qu'ils ont la même description sémantique en termes d'IOPE- à ce niveau d'abstraction- ils sont considérés comme un seul service. La variation des P/E possibles donne naissance à plusieurs sous-catégories comme par exemple : Transport_international1 qui relie Montréal à Casablanca et Transport_international2 qui relie Montréal à Tanger (précisément) et Transport_international3 qui ne peut relier Montréal qu'à New-York (la liste est évidement non exhaustive, il s'agit d'un exemple illustratif).
- **Transport_local :** qui est un service de transport qui relie deux emplacements appartenant au même pays : le train, les autocars et l'avion interne sont des exemples de cette catégorie.
- On entend par **Transport_provincial,** un service de transport qui relie une ville marocaine à ses périphériques (i.g. Fès et Sefrou ou Moulay yâcoub…): les grands taxis et les bus sont des exemples de cette catégorie.

Il est important de rappeler que l'exemple illustratif que nous utilisons dans ce manuscrit a été simplifié et adapté pour atteindre sa finalité explicative. Aussi, le caractère réaliste et exhaustif n'est pas pris parfois en compte en priorité dans l'exemple.

4.5. Modèle de planification

Permettre à plusieurs agents de planifier et de coordonner leurs activités de manière distribuée afin de produire un plan abstrait, tel est l'objectif de la composition abstraite. Afin de ce faire, plusieurs travaux préparatifs ont été menés jusqu'à présent; d'abord la demande de l'utilisateur a subi un processus d'analyse pour la reconstruire dans un format sémantique compréhensible par le système. Ensuite les services abstraits candidats qui ont le potentiel de résoudre la requête ont été élus. Ceci nous amène à la planification, étape la plus créative du processus abstrait, qui identifie les services qui contribuerons effectivement à la composition et détermine l'ordre de leur participation. Dans ce manuscrit nous défendons l'idée forte que la planification distribuée (référée aussi par planification multi-agents) définit une forme de rationalité collective susceptible de résoudre efficacement le sous-problème de construction des plans de composition. Dans ce sens, un planificateur de type distribué sous hypothèses appelé synthèse dialectique des plans, a été étudié afin de décider sur la possibilité de l'utiliser comme le mécanisme guidant le modèle de planification.

4.5.1. Fondement théorique du modèle de planification : synthèse dialectique des plans

4.5.1.1. Origine scientifique de la synthèse dialectique des plans

L'idée de base puise son inspiration dans l'épistémologie et plus particulièrement dans la théorie de réfutabilité de Karl Popper [Popper, 1963], selon laquelle les théories scientifiques ont un caractère hypothétique ou conjecturale. "Nous ne savons pas, nous pouvons seulement conjecturer." Souligne Karl Popper le fondateur de cette théorie. L'attitude scientifique est donc une attitude critique qui ne cherche pas des vérifications mais tout au contraire des tests qui peuvent réfuter la théorie mais non l'établir définitivement. Ce qui signifie que les nouvelles théories ne sont que des approximations meilleures que celles qui les ont précédées. Par exemple, la théorie de la relativité d'Einstein contient celle de Newton en tant qu'approximation, cette dernière englobant à son tour celles de Kepler et de Galilée. Et rien ne permet de penser que la théorie de la relativité ne sera pas un jour remise en question par une théorie au pouvoir explicatif plus grand, qui l'inclura comme une simple approximation. Le progrès scientifique résulte donc d'un processus itératif de propositions-réfutations qui nous permet d'approcher toujours plus d'une vérité objective.

Pas très loin, en philosophie, la dialectique définit un mode de raisonnement qui procède, pour démontrer une proposition, par la mise en parallèle d'une thèse et de son antithèse, et qui tente de dépasser la contradiction qui en résulte au niveau d'une synthèse finale. Cette forme de raisonnement trouve son expression dans le réputé « plan dialectique » dont la structure est « thèse-antithèse-synthèse » : je pose (thèse), j'oppose (antithèse) et je compose (synthèse) ou dépasse l'opposition [Hegel, 1991]. La dialectique s'enracine dans la pratique du dialogue entre deux interlocuteurs ayant des idées différentes et cherchant à se convaincre mutuellement.

En planification distribuée, le mimétisme qui se base sur l'analogie entre le processus hypothétique de la production des connaissances scientifiques (par réfutation) et la construction conjointe de plans par un groupe d'agents possédant des connaissances et compétences différentes, en utilisant le style dialectique pour organiser le dialogue permettant à ce groupe d'interlocuteurs (agents) de démontrer une proposition, est connu sous le nom de la synthèse dialectique des plans [Pellier,2005].

4.5.1.2. Principes de planification de la synthèse dialectique des plans

La synthèse dialectique des plans est un modèle de planification sous hypothèses, qui produit un plan en se basant sur un processus collectif de délibération permettant à un groupe d'agents de raisonner sur un but commun à atteindre. Le principe de l'approche s'appuie sur la capacité d'un agent d'élaborer un plan qui est, en quelques sortes, en contradiction avec ses connaissances, autrement dit, afin d'atteindre un objectif l'agent produit un plan qui peut être exécuté si certaines conditions sont vérifiées. Ainsi le processus ne s'arrête pas si certaines conditions ne sont pas marquées « vraies » dans sa base de connaissance, mais propose plutôt une conjecture, un plan sous hypothèses, qui devient un nouvel objectif pour les autres agents. A titre d'illustration supposons qu'une porte soit fermée, si l'agent n'est pas capable de l'ouvrir, même s'il peut réaliser l'ensemble des actions nécessaires pour remplir son but derrière celle-ci, il ne proposera aucun plan. En effet, la planification classique requière que l'ensemble des propriétés du monde nécessaires à l'exécution d'un plan soient satisfaites avant sa réalisation (les préconditions dans notre contexte). En revanche, si l'agent peut faire l'hypothèse que la porte est ouverte, il est possible qu'un autre agent soit capable de l'aider en proposant un plan pour l'ouvrir. Un agent peut ainsi, même s'il ne possède pas les compétences nécessaires, proposer une conjecture pour résoudre une partie du but global. Nous voyons bien ici l'intérêt de considérer la notion de conjecture plutôt que celle de plan. Bien entendu, les hypothèses formulées par les agents doivent être vérifiées pour qu'une conjecture soit considérée comme un plan solution.

L'approche proposée par Pellier [Pellier, 2005] est basée sur des cycles de propositions et contre-propositions des agents, appelés cycles de conjecture – réfutation. Dans le cadre d'un dialogue dialectique, les interactions entre agents prennent la forme d'un débat. Chaque agent possède un tableau dans lequel il enregistre les propositions des autres agents (Fig.4. 15), ce tableau définit l'état du dialogue mais également l'espace de recherche co-construit par les agents. Tour à tour les agents vont donc prendre la parole pour raffiner, réfuter ou réparer le plan courant en se basant respectivement sur leurs bases de connaissances et de compétences. Si le plan réfuté est réparé, il est plus robuste mais reste soumis à de nouvelles tentatives de réfutation. S'il est réfuté et ne peut être réparé, les agents l'abandonnent et poursuivent leur recherche sur un autre plan. Si un plan n'est pas réfuté, il est considéré comme acceptable et constitue la solution du système au but proposé.

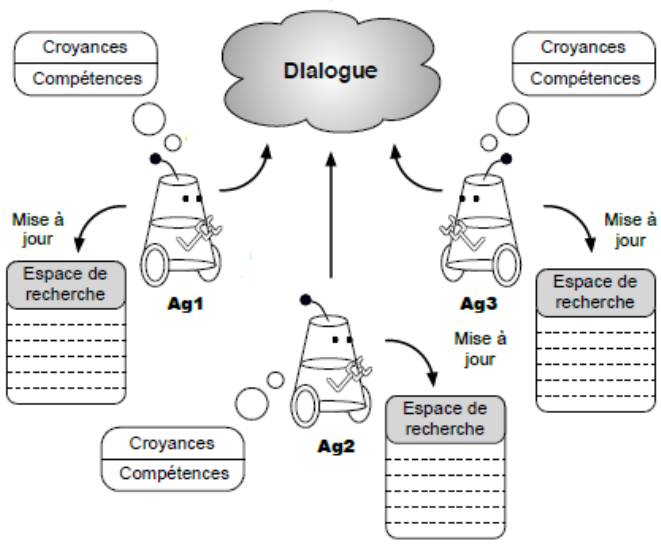


Fig.4. 15: Synthèse dialectique des plans [Pellier, 2005]

4.5.2. Synthèse dialectique des plans pour résoudre la CSW : potentiels et défis

Au contraire des méthodes largement consommées dans la composition des services Web (c.f chapitre 1: section 1.3.2), l'adoption de la synthèse dialectique des plans, conçue principalement pour une utilisation orientée robotique [Pellier, 2005], comme outil de composition des services Web, dénote une volonté manifeste de dévier des sentiers battus et d'explorer une nouvelle piste qui semble receler un potentiel considérable. En effet :

- la nature intrinsèquement distribuée des services Web et leurs autonomies expliquent en grande partie l'intérêt de l'utilisation de ce modèle de planification complétement automatique et entièrement distribué, aussi bien au niveau du contrôle (chaque agent possède ses propres compétences) qu'au niveau des données (chaque agent possède ses propres connaissances).
- La distribution n'est néanmoins pas un critère distinctif, car plusieurs approches de planification distribuée existent dans la littérature [Pistore et al., 2005] [EL Falou et al., 2010] [Tong et al., 2011]. En effet, le qualificatif « distribué » peut se comprendre à plusieurs niveaux, ce qui permet de classifier les approches de planification distribuée en trois catégories [Weiss, 2004] :
  - **Planification centralisée pour les plans distribués** : dans ce cas les plans qui doivent être exécutés dans un mode distribué peuvent être formulés de manière centralisée.
  - **Planification distribuée pour les plans centralisés** : la formulation d'un plan complexe peut exiger la collaboration de plusieurs spécialistes de la planification où chacun contribue par une partie à ce plan, jusqu'à ce qu'un plan global soit créé.
  - **Planification distribuée pour les plans distribués** : dans ce dernier type de la planification, le processus de synthèse et le processus d'exécution d'un plan multi-agents sont tous les deux distribués. A contrario du type précédent avec des plans centralisés, les agents ne risquent pas d'être en conflit les uns avec

les autres lors de l'exécution des plans partiels, au contraire ils doivent s'entraider les uns les autres pour réaliser leurs plans. On retrouve alors le modèle **orienté objectif** et le modèle **orienté agent**, dans ce dernier modèle la planification n'est pas guidée par un objectif commun (global) à atteindre au contraire du premier [Weiss, 2004]

- ➔ D'après ce bref descriptif, il apparait clair que la classe la plus adaptée pour notre cas est celle à laquelle appartient la méthode de planification proposée, notamment la planification distribuée pour les plans distribués orientée objectif. Le niveau de distribution de la méthode et le rôle de l'objectif commun pour orienter et façonner l'opération de planification s'ajoute alors à l'argument précèdent pour renforcer notre choix.

- Par ailleurs, classiquement, la plupart des méthodes de planification multi-agents découplent la phase de **planification** (chaque agent planifie sans tenir compte des autres agents) de la **résolution des conflits,** alors que la méthode de planification adoptée s'appuie, justement, sur ces conflits (débats) pour construire itérativement un plan solution.
- Une autre caractéristique essentielle de la méthode de planification proposée, tiens au fait que les agents participant à la planification raisonnent conjointement en s'appuyant sur leurs bases de compétences respectives pour atteindre un but commun en intégrant leurs croyances partielles sur le monde. Ils sont donc capables de raisonner même avec une vision partielle du monde et des données incomplètes. Ceci s'adapte parfaitement avec le caractère dynamique et ouvert de la gestion des services Web où nous cherchons à minimiser les connaissances partagées et à maintenir l'aspect privé des compétences de chaque service.
- Le dernier point fort de l'approche proposée réside dans son caractère hypothétique, offrant une grande flexibilité au processus de planification. En effet, ce dernier n'est pas bloqué en rencontrant des contradictions, au contraire les incohérences sont considérées des hypothèses et le but est résolu en résolvant récursivement ces hypothèses.

L'adoption de l'approche dialectique pour la synthèse de plans dans un contexte multi-agents comme modèle de planification guidant la phase de construction du plan, est donc principalement motivée par le fait qu'elle s'agit d'une planification distribuée orientée objectif, s'appuyant sur des interactions entre des agents capables de raisonner même avec une vision partielle du monde et des données incomplètes ou en contradiction avec leurs bases de connaissances, pour construire itérativement un plan solution.

Néanmoins cette approche, n'est pas directement applicable à notre problématique. Un travail d'adaptation a été donc mené en se basant sur des réflexions résultantes d'une étude approfondie de la méthode, afin de transposer les algorithmes dialectiques sur les services Web. Les principaux axes qui ont orienté ce travail de transposition sont:

- Peut-on considérer la possibilité de ne jamais converger vers une solution le revers de la médaille de la souplesse du processus de planification? Si oui quels sont les

mécanismes nécessaires à développer pour lutter contre ce risque et garantir que le débat atteint toujours une synthèse modélisant le plan solution ?

- Sachant que les agents raisonnent en termes d'opérateurs de planification et les services sont décrits en sémantique. Doit-on donc prévoir un autre processus d'analyse (traduction) pour traduire les descriptions services en langage de planification et vice versa ?
- Quelle relation (correspondance) peut exister entre agent et service ?
- Comment organiser le dialogue au niveau intra-agent pour atteindre progressivement le but?
- Comment s'organise le comportement au niveau inter-agent pour aboutir au raisonnement hypothétique désiré ?
- Du point de vue algorithmique, quelles structures de données à utiliser pour optimiser les calculs générés par le stockage, la mise à jour et l'exploration des plans partiels en cours (les conjectures) ?

La réflexion autour des axes ci-dessus a donné naissance à la troisième technique de composition conçue : **Modèle de planification.**

4.5.3. Modèle de planification proposé

4.5.3.1. Définitions préliminaires

Avant toute formalisation du modèle de planification, un éventail de définitions préliminaires est nécessaire d'être exposé afin d'unifier la compréhension des notions principales.

Au cœur de la vision dialectique se trouve, le **problème de planification** multi-agents qui est composé d'un **but** et un ensemble **d'agents** qui doivent coopérer afin d'atteindre ce but. Chaque agent détient une base de connaissances et une autre de compétences. Cette dernière est formée d'un ensemble **d'opérateurs**, des actions qu'un agent est capable d'exécuter d'une façon autonome dans un environnement donné. Un **plan solution** est destiné à résoudre un problème de planification, alors qu'un plan partiel ou une **conjecture** résout un problème sous certaines **conditions**. Ci-après nous présentons une définition formelle des concepts pivots, notamment : un opérateur de planification, agent, problème de planification, conjecture, hypothèse et plan solution.

Définition 1 (Opérateur de planification)

Un opérateur de planification est un triplet o =<name(o), precond(o), effect(o)> avec :

- name(o), le nom de l'opérateur, $n(x_1, \ldots, x_k)$ ou n est un symbôle et $x_1, \ldots, x_k$ définissent les paramètres de l'operateur;
- precond(o) et effect(o), les pré conditions et les effets de o, respectivement définissant des littéraux qui doivent être vrais dans l'état où l'opérateur est appliqué et les littéraux qui doivent être ajoutés $(effect(o)^+)$ ou supprimés $(effect(o)^-)$, pour calculer l'opération de transition.

➔ Une instance d'un opérateur est appelée : Action.

Définition 2 (Méthode)
Une méthode est un triplet m = (name(m), precond(m), reduction(m))

- name(m) est une expression de la forme n(x_1, ..., x_k) où n représente le nom de la méthode et x_1, ..., x_k ses paramètres
- precond(m) représente les préconditions (i.e. un ensemble de prédicats) devant être vérifiées dans l'état courant pour que m soit appliquée.
- reduction(m) définit la séquence d'opérateurs ou de méthodes pour réaliser m.

Définition 3 (Agent)
Un agent est un triplet ag = <name(ag), operators(ag), beliefs(ag)>,où :

- name(ag), le nom de l'agent;
- operators(ag), un ensemble d'opérateurs, i.e., les compétences de l'agent;
- beliefs(ag), ensemble de littéraux, i.e., les croyances initiales de l'agent.

Définition 4 (Problème de planification)
Un problème P pour un domaine D est un triplet P= (AG, s_0, g) où :

- AG définit un ensemble de noms d'agents;
- s_0, l'état initial qui appartient à S;
- g, le but, définit un ensemble cohérent de prédicats instanciés i.e., les propriétés du monde qui doivent être atteintes.

Nous faisons l'hypothèse restrictive que l'union des croyances des agents d'un problème de planification est cohérente, i.e., pour deux agents α et β, si une proposition p ∈ belief(α) alors ¬p / ∈ belief(β). Cependant, aucune hypothèse n'est faite sur le possible partage de croyances entre les agents en termes de faits ou d'opérateurs.

Définition 5 (Conjecture)
Une conjecture est un quadruplet χ = (A, <, I, C) tel que:

- A={a_0,…,a_n} est un ensemble d'actions;
- < est un ensemble de contraintes d'ordre sur les actions A de la forme $a_i<a_j$, i.e., a_i précède a_j;
- I est un ensemble de contraintes d'instanciation portant sur les variables des actions A de la forme x=y, x≠y ou x=cst tel que cst ∈ D_x et D_x est le domaine de x;
- C est un ensemble de liens causaux de la forme $a_i \xrightarrow{p} a_j$ telles que a_i et a_j sont deux actions de A, la contrainte d'ordre ai<aj existe dans <, la propriété p est un effet de a_i et une précondition de a_j et les contraintes d'instanciation qui lient les variables de a_i et de a_j portant sur la propriété p sont contenues dans I.

Une conjecture est une notion pivot dans le modèle de planification proposé, elle se présente comme un ensemble d'actions, i.e., d'opérateurs instanciés, qui sont liés par des contraintes d'ordre et des liens causaux. Les contraintes d'ordre permettent de définir l'ordre d'exécution des actions et les liens causaux indiquent les effets produits par une action qui sont nécessaires à l'exécution d'une autre action.
Fig.4. 16 présente la représentation graphique d'une conjecture adoptée dans ce travail. Un rectangle représente une action. Les préconditions sont situées au-dessus du rectangle et les

effets en dessous. Les flèches représentent les contraintes d'ordre, les flèches pointillées représentent les liens causaux. Les hypothèses (définition 6) sont représentées par des astérisques (*). Cette représentation est basée sur [Pellier, 2005].

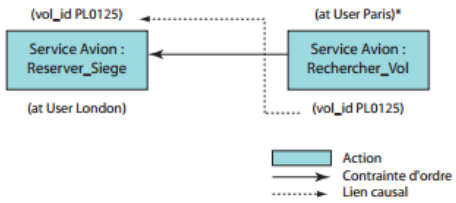


Fig.4. 16: Représentation graphique d'une conjecture [Pellier, 2005]

Définition 6 (Hypothèse)

Soit une conjecture χ = (A, <, I, C), une Hypothèse formulée par χ est définie comme une précondition p d'une action $a_j \in A$ tel que pour toute action $a_i \subset A$; le lien causal $a_i \xrightarrow{p} a_j \notin C$
Une hypothèse est donc une précondition nécessaire au déclenchement d'une action dont la particularité est de n'être soutenue par aucun lien causal.

Définition 7 (Solution plan)
Un conjecture χ = (A, <, B, C) est un plan solution pour un problème de planification P=<s_0, g, AG>, si toutes les hypothèses de χ sont vérifiées.

4.5.3.2. Sous architecture de planification

Notre objectif est de concevoir des algorithmes de planification distribuée pouvant contribuer à la création d'un service composite vu comme un plan définissant des relations de précédence et de causalité entre des services élémentaires. Nous considérons que chaque service est un agent autonome capable de planifier. Lorsque la phase précédente (découverte) ne réussit pas à répondre à la requête d'un utilisateur (un nouveau service composite doit être produit), cette requête devient le but d'un processus distribué de planification dont le plan solution, s'il existe, est une représentation du service composite constitué des agents ayant pris part à sa construction.

A partir de chaque service abstrait découvert, nous définissons alors un agent Manager dont le but est de simuler le comportement du service. Il s'agit d'une entité autonome capable de raisonner et de communiquer avec les autres agents dans le but de co-construire un plan de composition des services Web.

La sous-architecture proposée s'appuie également sur un agent Planificateur (le Planner), qui assure une double mission : contrôler le statut de l'ensemble des interactions des agents Manager d'un côté et, d'interagir avec les autres types d'agents (Discover et l'Evaluator) faisant partie de l'architecture iDyCoS d'autre coté. Ceci ne doit pas être compris comme un contrôle centralisé qui entrave et contredit la vision décentralisée défendue par cette thèse. L'objectif est plutôt de centraliser la vision globale du sous-système de planification en une entité pour faciliter les tâches de communication, favoriser la modularité du système et aussi améliorer sa scalabilité (cf. section 4.5.4).

Un agent Manager est initialisé avec la description sémantique du service qui le simule et avec une base de connaissances. La description OWL-S permet de définir la base de compétences de l'agent tandis que la base de connaissances (issue de l'ontologie de domaine par exemple) apporte les connaissances qui seront nécessaires à l'agent pour raisonner. Quant à le Planner, il est initialisé par la description sémantique de la requête (le but à réaliser) (Fig.4. 17).

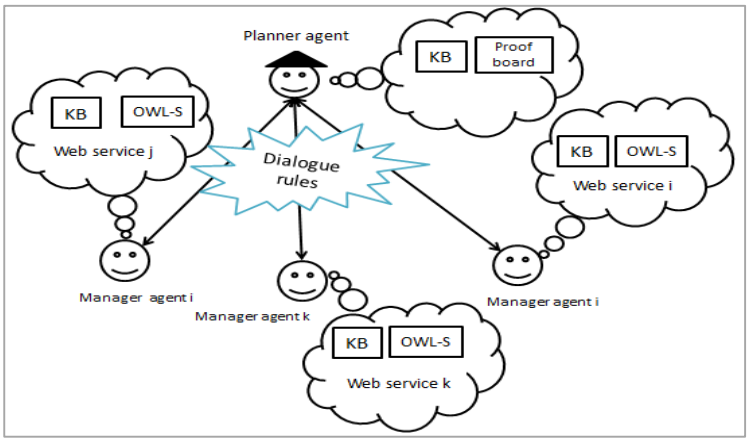


Fig.4. 17: Sous-architecture de planification [Adadi et al., 2014]

Raisonner, interagir et contrôler sont donc les principales fonctions des agents qui participent à la planification, une structure adaptée a été ainsi conçue pour chaque type d'agent afin qu'ils accomplissent leurs missions.

Fig.4. 19 illustre la structure d'un agent Manager, elle respecte la constitution de la structure générique d'un agent que nous avons suivi jusqu'à présent, à savoir un module de raisonnement et un module de communication. Ces deux modules sont liés par une base de compétences et une base de connaissances, le Planner (Fig.4.18) contient un composant supplémentaire, le tableau de démonstration.

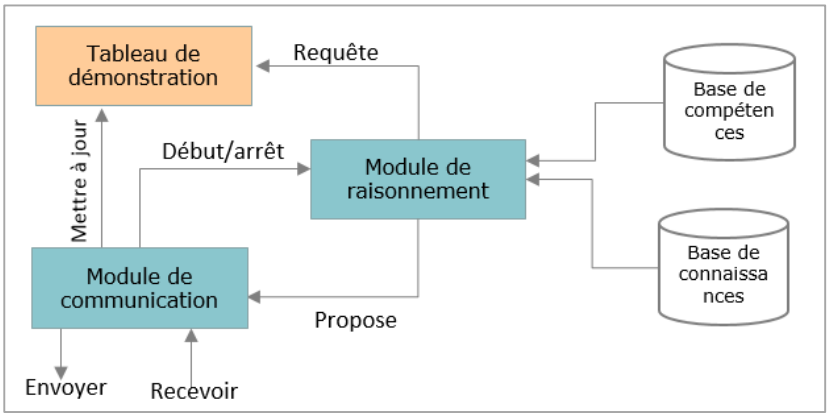


Fig.4. 18: Structure de l'agent Planificateur

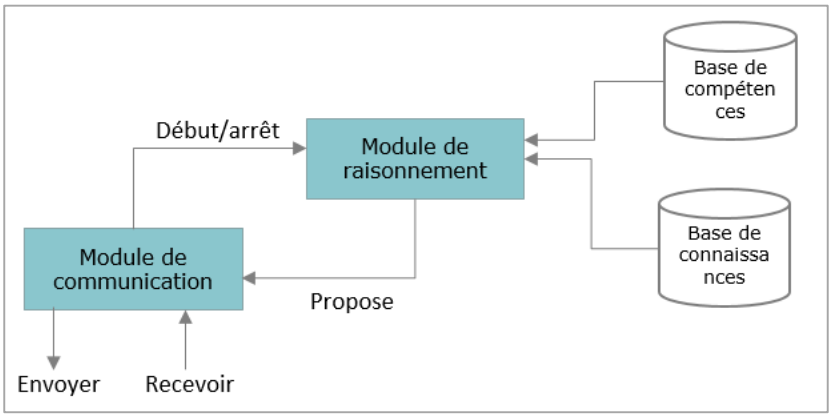


Fig.4. 19: Structure de l'agent Manager

Le tableau ci-après donne une description de chaque module/Composant par type d'agent :

Module/Composant	Manager	Planner
Module de Raisonnement	Le module de raisonnement est le module central de l'agent. Il définit le comportement de l'agent en dirigeant ses interactions. Il s'appuie sur la boucle de raisonnement lui permettant de traiter les plans partiels et sur le gestionnaire de dialogue pour soumettre les propositions aux autres agents. Ce module est guidé par une boucle de raisonnement pour l'agent Manger	Ce module permet au Planner d'assurer son rôle de coordonnateur en mettant à sa disposition les mécanismes nécessaires pour le suivi de la construction du plan: - lance le plan initial, - reçoit les raffinements des Managers et les mises à jour du tableau de conjecture, - informe qu'un plan solution se trouve dans le tableau, - demande des confirmations aux managers qu'ils ne peuvent plus raffiner un plan partiel, - arrête le processus.
Module de Communication	Le module de communication (gestionnaire de dialogue) gère la réception et l'interprétation des messages provenant des autres agents, Le dialogue est construit autour d'un ensemble de primitives (**Tableau 4. 4**) qui, lors de l'envoi ou de la réception d'un message, indique le comportement que doit adopter l'agent, i.e., spécifie le changement d'état dans l'automate du module de raisonnement.	
Base de connaissances	La base de connaissances, alimentée principalement à partir des ontologies de l'approche iDyCoS, apporte les connaissances qui seront nécessaires à l'agent pour raisonner	
Base de compétences	Il s'agit de la base qui stocke les compétences d'un agent. Pour le Manager, c'est la description OWL-S du service que l'agent simule qui permet de définir cette base et ce, en se basant sur un mécanisme de traduction sémantique/planification (c.f section 4.5.3.3).	
Tableau de démonstration		la partie publique qui stocke les contributions des agents, cette structure de données est dotée d'outils de programmation nécessaires pour gérer (recherche, exploration et la mise à jour) des plans partiels.

Tableau 4. 2: Description des composants de la structure de l'agent planificateur et des agents Mangers

4.5.3.3. Mécanismes de planification

La concrétisation de la conceptualisation proposée nécessite d'expliciter certains mécanismes, notamment :

- le mécanisme de création de la base de compétences de l'agent à partir de la description sémantique OWL-S du service Web associé et vice versa i.e. le mécanisme de génération de la description sémantique du service composite à partir du plan construit ;
- les mécanismes de raisonnements utilisés par un agent pour planifier localement ;
- les mécanismes et les primitives qui permettent aux agents de coordonner leurs contributions de façon à garantir la convergence vers un plan solution.

Dans cette partie nous allons donc essayer d'entrevoir les mécanismes nécessaires à la mise en œuvre de l'approche de planification proposée au travers la définition de l'algorithme de traduction, les règles de dialogue, la structure algorithmique du tableau des conjectures, la boucle de raisonnement et les mécanismes de raffinement.

A. Algorithmes de traduction de la sémantique vers la planification

Le traitement du sous-problème de construction de plan de composition sous forme d'un problème de planification nécessite l'existence d'une correspondance fidèle et fiable entre les deux domaines sémantique et de planification et ce, afin de garantir la crédibilité des résultats fournis par le modèle de planification.

L'utilisation du langage OWL-S, rend cette correspondance quasi-directe, d'ailleurs l'une des raisons qui ont motivées le choix de ce langage pour décrire la sémantique des services Web (cf. section 4.2.1), est sa syntaxe qui est proche de celle des langages de planification « Plan-like language », nous proposons ainsi de transformer les données sémantiques en des données de planification en se basant sur le tableau d'équivalence ci-dessous.

Domaine de planification	Domaine sémantique
But	Requête sémantique
Opérateur	Profil du service (Service Profile)
Méthode	Service web composé, processus composé (composite process)
Base de connaissances, état initial	Extrait de l'ontologie de domaine

Tableau 4. 3: Correspondance Sémantique/Planification

La traduction des notions entre les deux domaines est basée sur un algorithme de traduction [Guitton , 2006]. Cet algorithme prend en entrée une description OWL-S et retourne en sorite une description PDDL [McDermott, 1998] équivalente. L'application de cet algorithme nécessite d'imposer un certain nombre d'hypothèses et de restrictions :

- les structures de contrôle qui régissent l'enchaînement des sous-processus dans un processus composite sont variées (Sequence, Concurrent, Split, Split+Join, Unordred, Choice, If-Then-Else, Repeat-Until, Repeat-While), mais seule la structure de contrôle « Sequence » est prise en compte, puisque les autres peuvent être exprimées sous la forme de cette structure basique;

- OWL-S permet de tenir compte de l'indéterminisme dû à l'exécution d'un service, c'est-à-dire qu'il laisse la possibilité de définir le résultat d'un service Web en fonction de son comportement (succès, erreur ou encore absence de réponse) via l'utilisation des classes ConditionalEffect et ConditionalOutput qui permet de définir la condition sous laquelle un résultat est produit. Comme la composition aura lieu avant l'exécution du service, nous ne considérons qu'un seul résultat possible, celui qui correspond au succès de l'exécution du service;
- Nous émettons l'hypothèse que le premier processus de la description sémantique d'un service Web (la requête sémantique) est un processus composite dont les préconditions contiennent le but que ce service permet d'atteindre.

Principe de l'algorithme : L'algorithme de traduction réutilisé (cf. Algorithme 2) est un algorithme récursif. Son principe est de transférer le premier processus (correspond à la requête initiale) en un opérateur si ce dernier est un processus atomique, sinon il traduit récursivement les processus intervenant dans les structures de contrôle de ce processus composite, puis crée la méthode correspondante. Il permet d'ajouter les préconditions et les effets du processus atomique à l'opérateur correspondant. Cet algorithme règle aussi les problèmes de gestion des préconditions, des effets et d'ajout de buts à atteindre.

Selon cet algorithme nous avons deux cas de figure : cas d'un service Web atomique et celui d'un service Web composite.

- **Traduction des processus atomiques :** Si le processus à traduire est un processus atomique, alors la correspondance est intuitive. Un opérateur de planification est créé avec, comme identifiant, l'identifiant de ce processus, c'est-à-dire le nom de l'opération du service correspondant. Les préconditions et les effets du processus atomique sont ajoutés à l'opérateur correspondant.
- **Traduction des processus composites :** Un service composite ne correspond pas à une opération d'un service Web, mais permet de définir l'ordre et les conditions d'exécution des différentes opérations du service. Ainsi le cœur du processus composite est la structure de contrôle qui définit cet enchaînement. La première étape de la traduction d'un processus composite consiste donc à traduire la structure de contrôle. Puis la méthode correspondante est créée. Les préconditions permettant son déclenchement sont ajoutées ainsi que la décomposition, i.e., la séquence d'opérateurs correspondant aux processus invoqués dans la structure de contrôle.

L'algorithme de traitement des structures de contrôle est récursif car les éléments d'une séquence peuvent être des processus mais également d'autres structures de contrôle. Pour chaque composant (représenté par la classe perform), l'algorithme de traduction de processus est invoqué. Celui-ci maintient une liste des processus qui interviennent dans la structure de contrôle en liant les variables de ces processus avec le processus appelant, mais également en liant les variables entre elles dans le cas où l'un des processus consomme une donnée produite par l'autre.

- **Gestion des préconditions :** Les préconditions d'un processus ont une correspondance directe avec les préconditions d'un opérateur ou d'une méthode du fait que l'hypothèse formulée sur l'écriture des préconditions et des effets dans la

description OWL-S. Les préconditions peuvent être hypothétiques, i.e., leur vérification n'est pas imposée pour déclencher l'opérateur ou la méthode.

- **Gestion des effets :** Tout comme les préconditions, les effets d'un processus atomique sont traduits directement. Mais ils doivent être distingués en effets positifs et en effets négatifs au moment de la création de l'opérateur. Nous choisissons de représenter les effets négatifs, dans la description OWL-S par des prédicats commençant par « not ».

Notons qu'à la fin de la phase de planification, le même principe et les mêmes correspondances sont utilisés pour effectuer la traduction inverse, i.e de la planification à la sémantique, et ce afin de fournir en sortie de la phase de construction du plan un fichier OWL-S décrivant le service composite construit, en encapsulant ainsi le mécanisme qui a permis d'obtenir ce fichier.

Algorithme 2 : Algorithme de traduction de la sémantique vers la planification

```
Algo Traduction_Processus(Processus p)
        Si p est atomique Alors
        Créer nouvel opérateur o
        o.nom=p.nom
        o.paramètre=p.inputs
        o.préconditions=p.préconditions + prédicats_typage(p.inputs)
        o.effets=p.effets + prédicats_typage(p.outputs)
        Sinon //p est composite
        composition= Traduction_structure_contôle(p.structure.contôle)
                Pour chaque processus p2 de composition
                Traduction_processus (p2)
                Fin de pour
        Créer nouvelle méthode m
        m.nom=p.nom
        m.paramètre=p.inputs
        m.préconditions=p.préconditions + prédicats_typage(p.inputs)
        m.décomposition=décomposition_processus(composition)
        Fin de si
Fin algo
```

Fig.4. 20: Algorithme de traduction de la sémantique vers la planification

B. Règles de dialogue

La communication inter-agents consiste en l'envoi d'actes de communication ou messages entre agents. C'est par l'intermédiaire de ces actes que les agents vont pouvoir débuter ou encore suspendre l'élaboration d'un plan-solution. Un agent envoie un message dans deux cas de figure :

1. S'il a reçu un message nécessitant d'être traité, il construit alors un message pour répondre à celui qu'il a reçu et l'envoie à l'émetteur de ce dernier ;

2. S'il exécute une action dont les effets consistent en l'envoi d'un ou plusieurs messages, alors il construit ces messages et les envoie au(x) destinataire(s) spécifié(s).

La structure d'un message est définie par un langage de communication pour les agents (ACL) et repose sur la théorie des actes de langage [Austin, 1962]. Puisque nous utilisons la plateforme JADE [Bellifemine et al., 2001] pour implémenter un prototype du système, le

concept d'acte de communication est simulé par les performatives de FIPA [FIPA-ACL, 2016], l'ensemble des actes de communication utilisés, ainsi que leurs significations sont présentées dans le tableau ci-après.

Acte de dialogue	**Rôle**
CFP	L'action d'appel à propositions pour raffiner une conjecture donnée
PROPOSE	Soumission d'une proposition pour raffiner une conjecture donnée
FAILURE	L'action de dire à un autre agent qu'une tentative de raffinement d'une conjecture donnée a échoué
QUERY_IF	L'action de demander aux Managers s'ils peuvent encore pousser davantage le raffinement du plan en cours.
CONFIRM	Le Manager informe le Planner qu'il ne peut plus raffiner le plan en cours
DISCONFIRM	Le Manager informe le Planner qu'il peut encore raffiner le plan en cours
INFORM	L'action d'informer qu'un plan solution a été trouvé dans le tableau de conjectures (tableau de démonstration)
CANCEL	L'action avec laquelle le Planner donne ses ordres aux Managers pour arrêter la planification

Tableau 4. 4: Actes de dialogue

C. Tableau de démonstration

l'agent Planner possède un tableau [Englemore , 1988] dans lequel il enregistre les propositions des autres agents. Le tableau de démonstration (Proof Board) sert de support au raisonnement mais également au dialogue. D'un point de vue algorithmique, le tableau peut être vu comme un graphe ET/OU orienté dont les nœuds ET représentent les conjectures et les nœuds OU représentent les hypothèses contenues dans le nœud « conjecture » père (Fig.4. 21). Une conjecture solution est une conjecture terminale, c'est-à-dire qui ne formule plus d'hypothèses. Ainsi, pour qu'un nœud « conjecture » soit résolu, tous ses nœuds « hypothèses » doivent être résolus. En revanche, il suffit qu'un des nœuds « conjecture » soit résolu pour qu'un nœud « hypothèse » le soit également.

Le tableau de démonstration est initialisé avec un plan partiel π_0 défini par :

- deux actions a_0, a_∞ telles que les préconditions de a_∞ représentent le but G soumis à l'ensemble des agents et les effets de a_0 l'état initial des croyances de l'agent ;
- une relation d'ordre ($a_0 \prec a_\infty$) ;
- les contraintes d'instanciation relatives à la description des préconditions et des effets de a_0, a_∞ ;
- un ensemble de liens causaux vide.

π_0 n'est pas un plan solution car le but n'est soutenu par aucune relation causale (les hypothèses ne sont pas vérifiées).

Les agents vont donc raffiner ce plan par l'ajout d'autres services (opérateur dans notre terminologie) dont les effets sont produits pour réaliser le but. Ces services peuvent eux-mêmes avoir des préconditions non soutenues causalement, ce qui induit de nouveaux raffinements ou provoque des réfutations avec d'autres services qui nécessitent des

réparations. Nos algorithmes garantissent que ce processus converge vers un plan solution lorsqu'il existe.

Le tableau de démonstration définit donc l'état des interactions entre agents mais également l'espace de recherche co-construit par les agents.

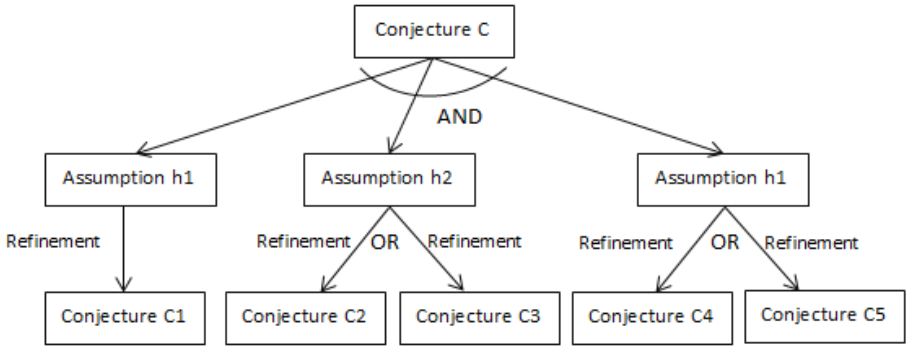


Fig.4. 21: Structure du tableau de démonstration [Adadi et al., 2015b]

D. Boucle de raisonnement

La définition des règles qui régissent le dialogue et la structure du tableau de démonstration est nécessaire mais ne spécifie pas les mécanismes qui guident l'interaction. Pour répondre à cette question, nous proposons d'introduire la boucle de raisonnement qui permet à un agent Manger de planifier localement.

Comme illustré dans Fig.4. 22 (cf. algorithme 3), à chaque itération un Manager choisit une conjecture à partir du tableau de démonstration (1) et cherche à lui appliquer un raffinement (2), faisant ainsi progresser la recherche du plan solution. Les cycles de raisonnement continuent jusqu'à trouver un plan solution dans le tableau de démonstration (3). Si tous les agents ne peuvent plus raffiner aucune conjecture, le processus de planification échoue.

Algorithme 3 : Boucle de raisonnement

```
Algo boucle de raisonnement
Tant que raisonne = vrai faire
plans =une conjecture non résolue du tableau de démonstration//  (1)
        Si plans est vide alors
        soumettre  echec de résolution de plans ;
        raisonne =faux ;
        Sinon
        Sélectionner  une conjecture π ∈ plans ;
        threats =Assumptions(π) // Assumptions(π)  est l'ensemble des hypothèses formées par π
                Si threats est vide alors
                soumettre  unr réussite de résolution de  π
                raisonne = faux ;// (3)
                sinon
                Sélectionner  une hpothèse φ ∈ threats;
                refinements =Refine(φ, π) ;// en ajoutant les opérateurs de raffinement (liens causaux ou sous-
                plans) (2)
                        Si refinements est vide  alors
                        soumettre  echec de résolution de φ
                        sinon
                        Sélectionner un raffinement ρ ∈ raffinements
                        soumettre ρ comme un  raffinement de φ ;
                           Fin de si
                   Fin de si
        Fin de si
Fin tant que

Fin algo
```

Fig.4. 22: Boucle de raisonnement

E. Mécanismes de raffinement

L'opérateur de raffinement utilisé dans la boucle de raisonnement définie précédemment est la technique principale servant à démontrer qu'une hypothèse peut être vérifiée. Nous distinguons deux mécanismes de raffinement : les raffinements par ajout d'un lien causal et les raffinements par ajout d'un sous-plan partiel.

- **Les raffinements par ajout d'un lien causal**

Le premier mécanisme de raffinement consiste à ajouter un lien causal. Cette technique de raffinement est la plus simple. S'il existe déjà au sein du plan partiel π une action a_i qui a pour effet p alors il suffit d'ajouter au plan partiel π un lien causal $a_i \xrightarrow{P} a_j$ indiquant dorénavant que l'action a_i démontre l'hypothèse p formulée par l'action aj. Un exemple de ce type de raffinement est donné dans Fig.4. 23. Le raffinement solution est un triplet constitué d'un lien causal, d'une contrainte d'ordre ($a_i \prec a_j$), ainsi que des contraintes d'instanciation σ nécessaires à l'unification de p avec les effets de ai.

L'intérêt de ce type de raffinement est de minimiser l'ajout de nouvelles actions au plan partiel et ainsi de réduire le nombre d'hypothèses introduites.

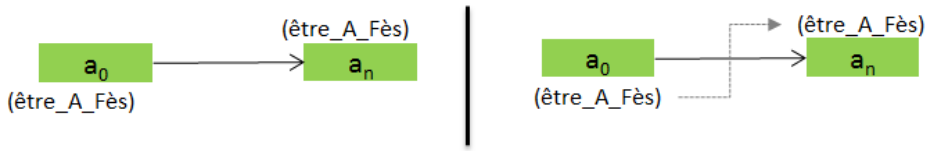


Fig.4. 23: Raffinement par ajout d'un lien causal

- **Les raffinements par ajout d'un sous-plan partiel**

Le second mécanisme de raffinement consiste à ajouter un sous-plan partiel pour démontrer la validité d'une hypothèse. Rappelons qu'une hypothèse p est considérée comme un sous-but à atteindre pour les autres agents. Le raffinement solution contient le sous-plan partiel produit π_0, un lien causal, $a_i \xrightarrow{P} a_j$ permettant de spécifier quelle action a_i de π_0 démontre l'hypothèse p, et une contrainte d'ordre ($a_i \prec a_j$). Fig.4. 24 montre un exemple de raffinement par ajout d'un sous-plan partiel pouvant lui-même contenir des hypothèses.

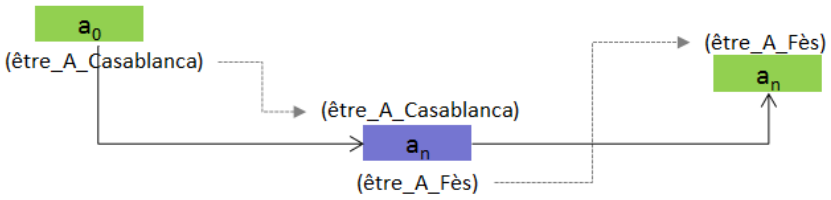


Fig.4. 24: Raffinement par ajout d'une sous-conjecture

4.5.3.4. Dynamique du modèle de planification

Après avoir décrit la structure du sous-système de planification et les mécanismes qui le guident, essayons maintenant d'étudier la dynamique générée par ce sous-système à travers un automate de contextualisation d'un agent Manger.

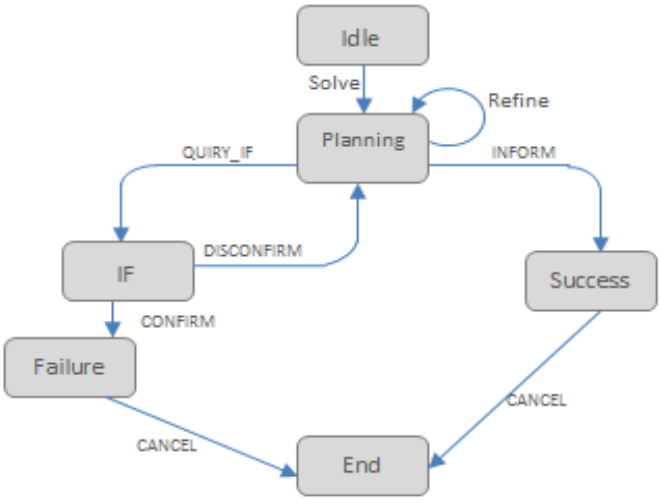


Fig.4. 25: Automate de contextualisation d'un agent Manger

Initialement, les agents sont en état d'inactivité (IDLE) dans l'attente d'un but à résoudre. A la réception du premier appel à proposition, le dialogue est ouvert et les agents commencent à échanger des raffinements, faisant ainsi alimenter le tableau de démonstration dans le respect

des actes de dialogue convenus. Une fois le dialogue est initié, les agents atteignent l'état « Planning » où ils continuent à exécuter leur boucle de raisonnement jusqu'à qu'un ordre d'arrêt sur échec ou sur succès soit reçu. Dans le premier cas les agents entrent dans un état IF, le Planner s'assure alors que le plan en cours ne peut plus être raffiné en envoyant une demande de confirmation à tous les Mangers, si les agents confirment qu'ils ne sont plus capables de faire avancer la synthèse dialectique, i.e., de raffiner les hypothèses restantes, le dialogue est fermé sur l'échec et l'activité des agents se termine. Sinon les agents reviennent à leur état de « Planning » pour poursuivre le raisonnement. Dans le deuxième cas où la série des raffinements proposés a réussi de vérifier toutes les hypothèses du tableau de démonstration, les agents entrent directement dans l'état « Success » et le Panner conclue le dialogue en annonçant l'existence d'un plan solution.

L'automate de contextualisation garantit que la terminaison du processus de synthèse de plans ne peut avoir lieu que si l'ensemble des agents sont dans l'incapacité de faire progresser la co-construction d'un plan ou alors qu'un plan-solution est présent dans le tableau. Dans le cas réel il est en plus nécessaire, pour assurer la terminaison de l'interaction, de se donner un critère bornant l'effort de planification, comme par exemple, une borne sur la durée, le nombre d'hypothèses existantes ou la longueur d'un plan.

4.5.4. Modèle de planification : analyse et points d'amélioration

Etant convaincus que la planification distribuée représente le cadre théorique adéquat pour la phase de construction du plan, la conception d'un modèle de planification consistait donc à transposer une méthodologie de planification multi-agents, jugée « à potentiel », dans le contexte de la composition de services Web. Cette projection ne saurait être démontrée possible et faisable sans l'introduction d'un ensemble de mécanismes fiables et efficaces :

- D'abord le passage de la sémantique à la planification et vice versa, se base sur l'idée fondamentale d'associer un agent à un service, ensuite la conception d'un algorithme de traduction renforce le pont entre les deux domaines.
- La notion de conjecture ainsi que le fondement théorique derrière (lien causaux, contraintes d'ordre, hypothèse, plan solution) sont essentiellement introduits pour concrétiser l'aspect hypothétique de la méthodologie.
- Les règles de dialogue et le tableau de démonstration représentent, quant à eux, des outils de communication essentiels pour maintenir un socio mimétisme fidèle à la dialectique et à l'argumentation, essence et source d'inspiration de la méthodologie de planification adoptée.
- Finalement, les algorithmes de planification basés essentiellement sur l'opérateur de raffinement constituent le cerveau et le moteur du modèle de planification proposé.

Par ailleurs, l'évaluation du modèle de planification proposé fait ressortir quelques pistes d'amélioration qu'il est essentiel d'expliciter avant de revenir à l'exemple de voyage pour illustrer le modèle de planification:

- D'abord, il est à noter que le modèle de planification proposé se base sur des algorithmes de planification opérant sur les préconditions et les effets. Cette méthode ne tient donc pas compte des I/O. Aussi pour des soucis de déterminisme on suppose que les résultats d'une action sont parfaitement connus i.e., on ne prend pas en compte le caractère conditionnel des effets. L'exploration de la faisabilité d'une planification métrique (incluant les I/O) ainsi que la prise en charge de l'aspect conditionnel des effets sont des variables d'optimisation sur lesquelles on compte jouer dans une deuxième étape une fois l'approche de base est prouvée et validée.
- Un point important à améliorer dans l'approche est le passage à l'échelle et le temps de réponse en fonction du nombre des services composés. La complexité intrinsèque du processus de planification distribuée sur laquelle repose notre approche de composition automatique se limite pour le moment à une classe de problèmes nécessitant une trentaine de services. Toutefois, il est possible d'envisager une généralisation de l'approche dans laquelle l'entité minimale ne serait plus un agent mais un groupe d'agents regroupés en structure organisationnelle formée d'un ensemble de Managers et un Planner qui seront capables de composer des services de plus haut niveau d'abstraction.
- Il est également important de noter que le processus proposé ne vise pas à être optimal, nous effectuons une planification que l'on qualifie de planification en profondeur d'abord (visant une convergence plus rapide d'abord). Cela signifie que les agents travaillent coopérativement afin de pouvoir extraire au plus tôt un plan solution, peu importe s'il est le meilleur ou pas. Ceci ne constitue pas une limitation de l'approche, puisque à ce niveau d'abstraction les critères d'évaluation sont plutôt qualitatifs et exigent d'abord et avant tout la satisfaction des besoins fonctionnels, l'optimisation sera, par contre, une question fondamentale dans la prochaine phase : phase d'évaluation.

4.5.5. Exemple de voyage : Modèle de planification

Toujours dans une quête d'illustrer les techniques de composition conçues et utilisées, nous continuons avec le processus de composition relatif à la requête de voyage.

Supposons pour simplifier que le Discover a proposé trois services pour la planification et que chaque service est représenté par un agent :
Agent TI: offrant un service de transport international entre pays.
Agent TL: capable d'assurer un transport local à l'intérieur du Maroc.
Agent banque : représentant la banque de Mourad, qui est en charge de payer les différentes réservations que Mourad sera amené à faire.
Imaginons maintenant le dialogue qui pourrait se dérouler entre les trois agents pour aider Mourad à se rendre au Maroc pour visiter sa famille:
Agent Planner : « Mourad est à Montréal et il doit se rendre à Fès. Pouvez-vous l'aider ? »
Agent TI: « Je ne peux malheureusement pas l'aider, je ne sais pas comment aller à Fès. »
Agent TL : « En ce qui me concerne, je peux l'emmener à Fès à condition que tu sois capable de le rendre à Casablanca et que tu me paies la somme de 160 MAD. »

Agent banque : « Je peux payer la somme de 160 MAD, le compte de Mourad est créditeur. »
Agent TI: « je peux assurer le transport de Montréal à Casablanca. En revanche, il faut que Mourad puisse s'acquitter de la somme de 1000 $. »
Agent banque : « Parfait, je crois que nous tenons la solution au problème, je peux également payer les 1000 $ demandés. »
Le plan solution est donc : « *Se rendre à Casablanca avec l'aide de l'agent TI puis de Casablanca à Fès grâce à l'agent TL, l'agent banque quant à lui payera les charges de voyage*»
En se basant sur le modèle présenté dans cette section, formellement le processus de planification se déroulera ainsi :

- **Initialisation:** le Planner envoie la première conjecture qui représente le but à tous les Mangers (Fig.4. 26).
- **Premier raffinement***:* seulement l'agent TL peut raffiner la conjecture initiale, mais sa contribution forme des nouvelles hypothèses qui doivent être résolues (Fig.4. 27).
- **Deuxième raffinement:** les Mangers doivent maintenant raffiner deux hypotheses : Avoir_solde et être_A_Casablanca, la première hypothèse peut être résolue par l'agent banque, pendant que l'agent LI peut résoudre la deuxième, mais ce dernier a besoin de 1000$ (Fig.4. 28).
- **Plan solution :** Encore une fois l'agent banque intervient pour résoudre la dernière hypothèse. En faisant ainsi toutes les hypothèses seront résolues et nous obtenons le plan solution représenté dans Fig.4. 29.

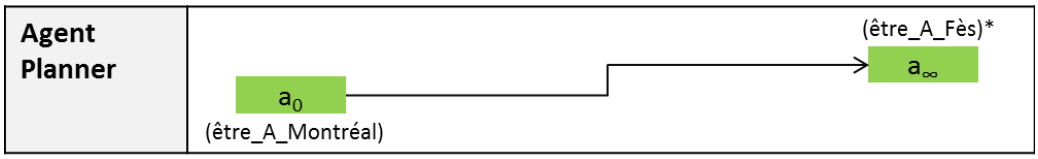


Fig.4. 26: Initialisation du raffinement (exemple de voyage)

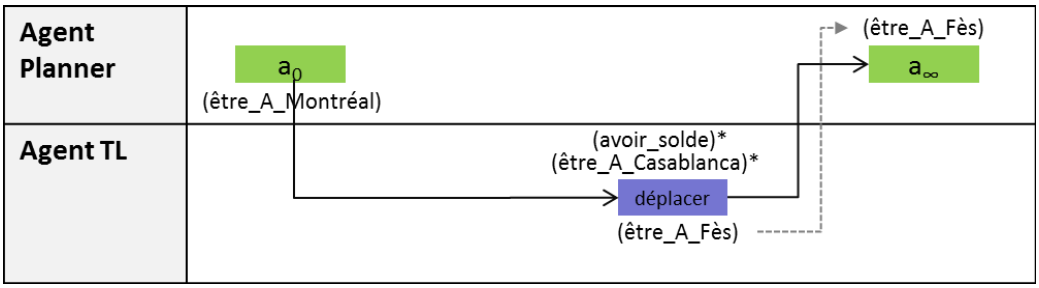


Fig.4. 27: Premier raffinement (exemple de voyage)

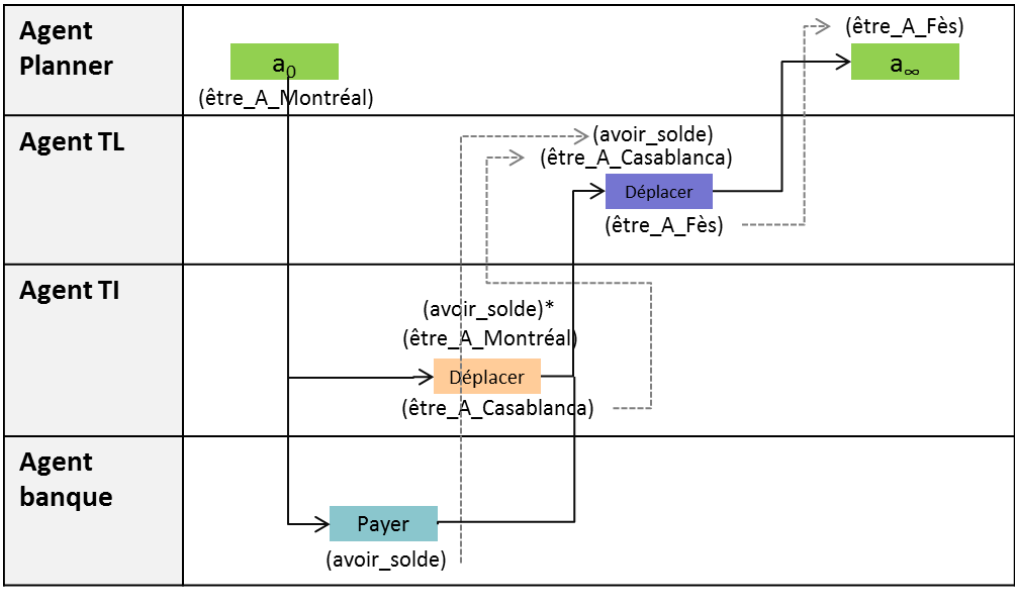


Fig.4. 28: Deuxième raffinement (exemple de voyage)

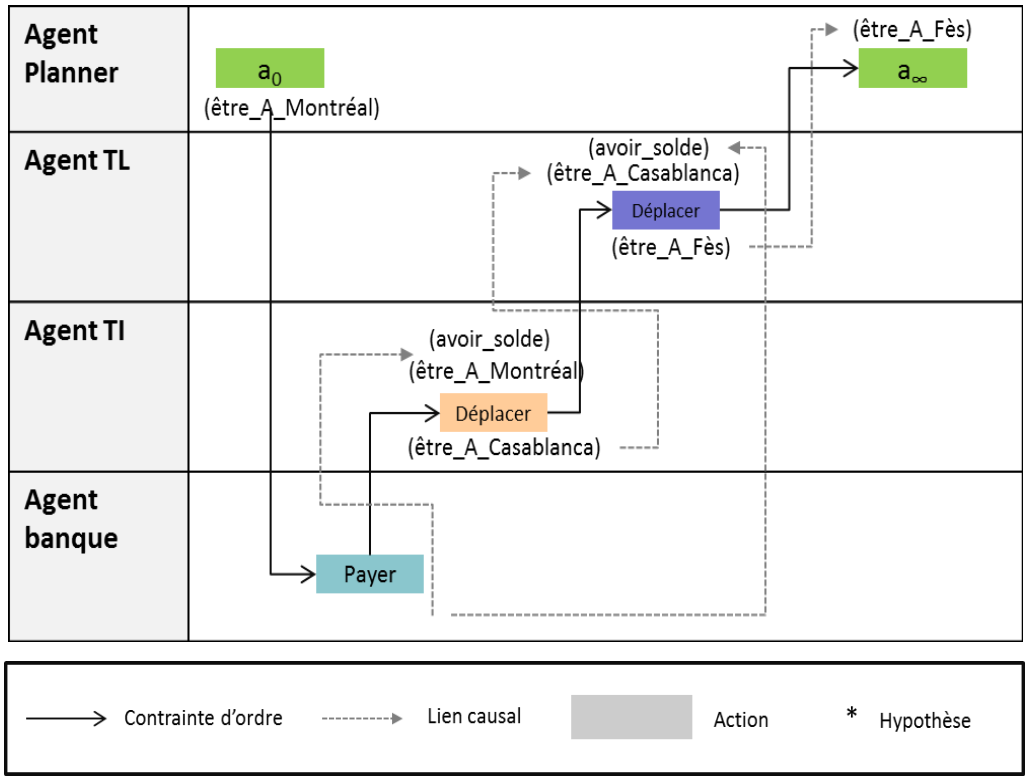


Fig.4. 29: Plan solution (exemple de voyage)

4.6. Conclusion

Au cœur des techniques de composition abstraite exposées dans ce chapitre se trouve le modèle de planification. Jugée comme le mécanisme adéquat pour déterminer les services Web qui participeront à la composition et leur ordre d'exécution dans le plan de composition, une méthode de planification distribuée dénommée la synthèse dialectique des plans a été étudiée et transposée dans le monde des services Web.

Pour soutenir le modèle de planification un mécanisme de découverte a été conçu. Son originalité tient dans l'optimisation apportée par un pré-calcul des correspondances sémantiques relatives aux services abstraits compatibles avec le but global à atteindre.

La requête traitée dans le mécanisme de la découverte et dans le modèle de planification est une requête revue par l'agent RHA, ce dernier est doté d'une capacité d'analyse sémantique permettant de traduire les spécifications informelles des utilisateurs en une requête sémantique.

Dans le chapitre suivant nous continuons l'exposition des techniques de composition, en mettant en lumière, cette fois, la stratégie de sélection utilisée et le processus d'exécution autonomique établi, ces deux techniques construisent et fondent ce que nous avons convenu d'appeler la composition concrète.

Chapitre 5

Techniques de composition concrète

We are all hungry and thirsty for concrete images
Salvador Dali

5.1. Introduction

Le service composite abstrait en combinaison avec les préférences précisées par l'utilisateur, conduisent le processus de la composition concrète. Cette dernière se compose de la phase de l'évaluation dans laquelle l'Evaluator choisit une instance parmi un ensemble de services composites concrets répondant tous aux mêmes exigences fonctionnelles mais dont l'offre en termes de qualité de services diffère d'une instance de service à une autre. La phase d'exécution prend lieu à la fin de la composition concrète, qui coïncide avec la fin du processus de composition global. Cette dernière phase prend en charge l'exécution du Worklow concret défini par l'instance du service composite élue en prenant en considération l'aspect autonomique dans la gestion des comportements anormaux qui peuvent survenir en temps réel lors de l'exécution.

Dans la suite de ce chapitre nous étudions en détail les techniques de composition conçues pour que les phases de la composition concrète se déroulent conformément à l'architecture iDyCoS, nous discutons alors:

- **La stratégie d'évaluation :** le mécanisme sur lequel l'Evaluator s'appuie pour sélectionner l'instance adéquate pour répondre à la requête en cours.
- **L'exécution autonomique** : la technique qui donne à la phase de l'exécution la qualité d'auto-guérissante et à l'agent Executer la qualité d'agent autonomique.

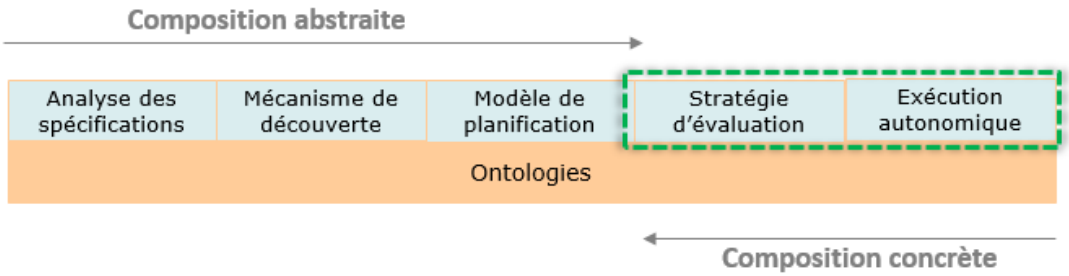


Fig.5. 1: Techniques de composition des services Web

5.2. Stratégie d'évaluation

5.2.1. Problème de sélection

A cette étape du processus de composition où une définition dite qualitative du service composite décrivant ses capacités fonctionnelles est déjà établie, grâce aux techniques du premier niveau de composition. Nous sommes à présent appelés, en vue de compléter le

processus, à instancier cette description en invoquant les instances des services élémentaires publiées dans le registre des services. Notons que l'opération de recherche et d'invocation des services concrets est une opération compliquée qui implique de nombreuses contraintes liées à l'orchestration des flux de données, la correspondance des types de données et la correspondance des protocoles d'invocation. Mais elle ne nécessite, par contre, aucune méthode ou technique spécifique. Dans le cadre de la présente technique, cette opération sera donc considérée comme déjà effectuée. Idéalement, cette opération de recherche et d'invocation des services concrets ne devrait ressortir qu'une seule instance pour chaque service abstrait élémentaire. Néanmoins, avec la prolifération des services Web, il faut prévoir l'existence de plusieurs instances de services Web offrant les mêmes fonctionnalités (même service abstrait). Cette relation de type « un à plusieurs » s'accentue davantage pour les services composites (e.g. si un service élémentaire possède m instances et qu'un service composite possède n services élémentaires, on peut alors former $n \times m$ instances du service composite en question).
Etant donné ceci, une question fondamentale s'impose : Comment choisir l'instance adaptée à la requête de l'utilisateur quand plusieurs services répondant aux fonctionnalités demandées, sont disponibles ? Cette question décrit le problème plus général de sélection de services Web.

En littérature, la sélection des services consiste à choisir, parmi les services Web découverts, ceux qui répondent au mieux aux exigences de l'utilisateur sur la base des besoins fonctionnels et/ou non fonctionnels. Les besoins non fonctionnels (ou extra-fonctionnels) sont définis comme un ensemble de caractéristiques totalement indépendantes des besoins fonctionnels d'un service mais qui décrivent, justement, les modalités de réalisation de ces besoins fonctionnels (ou propriétés fonctionnelles), ils s'expriment généralement à travers la qualité de service (QoS). Cette dernière rassemble les propriétés d'une instance d'un service qui lui confèrent l'aptitude à satisfaire des besoins déclarés ou implicites [ISO, 2005]. Ces besoins peuvent être liés à des paramètres tels que l'accessibilité, la disponibilité, le temps de réponse, le coût, la fiabilité, etc. Le Tableau 5. 1 présente une liste extensive des propriétés non-fonctionnelles décrites dans la littérature [O'sullivan et al., 2002] [Rosenberg, 2009] [Kumar et al., 2009] [Comuzzi et al., 2009]. Les propriétés de QoS constituent ainsi un critère décisif dans le choix d'un service approprié parmi des offres de plus en plus compétitives et aux fonctionnalités souvent similaires.

Propriétés non fonctionnelles d'un services Web				
Disponibilité	Classification	Introspection	fiabilité des messages	WsRF (Web Service Relevancy Function)
Accessibilité	Contrôle	Lactance	Temps de réponse	Usabilité
Précision	Conformité	Chances de succès (likelihood of success)	Réputation	Débit
Piste d'audit	Cryptage des données	Non-Répudiation	Robustesse	
Authentification	Documentation	Notification	Temps aller-	

			retour (Round Trip time)
Autorisation	Temps d'exécution	Pénalité	Montée à l'échelle (Scalability)
Bonnes pratiques	Intégrité	Prix	Sécurité
Capacité	Interopérabilité	Temps de rafraîchissement	Successibilité

Tableau 5. 1: Exemple de propriétés non fonctionnelles (QoS) d'un service Web

En s'accordant avec cette vision, une pléiade de techniques de sélection de services ont été proposées dans la littérature; à titre illustratif nous citons Optimization Algorithm pour la sélection des services [Baldon et al., 2007], Integer linear programming [Talantikite et al., 2008], Broker-based architecture [Dong-Hoon et al., 2009], Negotiation model pour la sélection des services [Swarnamugi et al., 2010] et bien d'autres.

Cependant, le contexte de la composition remet à la table des questions fondamentales qui font que ces techniques ne sont pas directement applicables à notre approche de composition :

- Comment automatiser et résoudre le conflit sémantique qui peut exister entre ce qu'offre le service en termes de QoS et ce qu'attend l'utilisateur (préférences de l'utilisateur) ?
- Comment peut-on estimer les propriétés de QoS du service composite à partir de celles des services élémentaires le composant ?
- Une fois les QoS des alternatives (instances des services composites) sont calculées, quel mécanisme de classement à utiliser pour juger quant à la pertinence d'une alternative par rapport aux autres ?

Ces questions ont dirigé notre réflexion vers la conception d'une stratégie d'évaluation efficace et adaptée basée sur les propriétés de QoS qui prend en charge les considérations ci-dessus.

5.2.2. Principe de la stratégie d'évaluation

Avant toute réflexion sur la stratégie à entreprendre, il est intéressant de noter d'abord qu'on se retrouve toujours dans la même logique de correspondance (Matching). En effet, lors de la composition abstraite nous disposions, d'un côté, d'une requête décrivant les fonctionnalités attendues par l'utilisateur et d'autre coté d'un ensemble de types de services offrant chacun un certain nombre de fonctionnalités. Et il fallait trouver un mécanisme qu'à partir des services abstraits élémentaires, devrait construire un type de service composite qui satisfait les fonctionnalités demandées.

Maintenant nous disposons d'une part d'un niveau de qualité attendu par l'utilisateur, formé par les préférences explicitées par l'utilisateur et les propriétés de QoS générées par le système et, d'autre part d'une offre en termes de services concrets assurant tous les fonctionnalités demandées avec un certain niveau de qualité. Et il faut trouver, là encore, le

mécanisme adéquat pour choisir l'alternative du service composite qui comprend les instances de services répondant au mieux au niveau de qualité attendu par l'utilisateur.

En capitalisant alors sur ce qui a été élaboré jusqu'ici, nous savons d'ores et déjà que la première chose à faire dans le cadre de cette stratégie d'évaluation, est d'établir d'abord une base d'entente sémantique (une ontologie) entre la cible et ce qu'on dispose en termes de qualité de service, tout en se basant sur une description OWL-S des services Web enrichie avec les propriétés QoS (comme convenu au début du chapitre 4 : section 4.2.1).

Ensuite, il y aurait lieu de préciser le mécanisme de calcul susceptible de synthétiser la composition des attributs de QoS qui correspond à la valeur estimée des propriétés de QoS du service composite en se basant sur la valeur de ces propriétés pour chaque service élémentaire.

Finalement, l'évaluation doit être formalisée dans le cadre d'un algorithme prenant en entrée l'ensemble des attributs de QoS relatives aux compositions possibles et retournant l'alternative qui maximise ces attributs du point de vue utilisateur.

La stratégie d'évaluation proposée tourne alors autour de trois aspects clés : (i) l'ontologie de QoS, (ii) le calcul de la composition des attributs de QoS et (iii) l'algorithme de sélection. Dans ce qui suit, nous présentons notre contribution au niveau de chacun de ces aspects.

5.2.3. L'ontologie de QoS

Comme introduit dans le chapitre 3, l'ontologie de QoS définit les propriétés de QoS, leurs relations et instaure un savoir partagé et commun entre les utilisateurs et les fournisseurs concernant les propriétés de QoS.
L'utilité de la définition d'une ontologie de QoS applicable à la caractérisation non-fonctionnelle des services, a été étudiée à de nombreuses reprises. En effet, son utilisation constitue un moyen efficace pour introduire la « sémantique » dans **la couche non-fonctionnelle des services Web** et la hisser au rang de **QoS sémantique**. De fait, par la définition d'un modèle commun des connaissances du système, une ontologie de ce type permet:

- aux fournisseurs de services de publier la QoS offerte par leurs services selon une sémantique partagée par toutes les parties prenantes ;
- aux consommateurs de spécifier de manière non-ambiguë leurs besoins en termes de qualité de services (préférences);
- au système de composition de faciliter le calcul automatique de conformité entre offres et demandes non-fonctionnelles des services.

Les travaux suivants, parmi les plus connus, définissent des ontologies appliquées dans un contexte non-fonctionnel:

Ontologie	**Description**
WS-QoS	Les auteurs de [Tian et al., 2003] définissent le canevas WS-QoS de support des contraintes de QoS dans les SOA ; ce canevas inclut une

	ontologie de QoS afin de réaliser la sélection dynamique de services Web en se fondant sur les besoins en termes de services ou de réseau. Cependant, les auteurs de cette ontologie ont choisi de l'écrire dans un pseudo-langage fondé sur une structure XML, perdant par la même occasion les avantages liés à l'utilisation d'OWL (comme une sémantique bien claire et les possibilités accrues de raisonnement logique).
DAML-QoS	L'ontologie DAML-QoS [Zhou et al., 2004] a été écrite en OWL et présente certains avantages, notamment ses liens avec OWL-S utilisés pour décrire fonctionnellement les services. Initialement développée avec DAML+OIL puis OWL 1.0, un de ses principaux problèmes était la mauvaise utilisation des contraintes de cardinalité OWL pour exprimer les bornes sur les propriétés de QoS. Mais ce problème a pu être levé grâce à la meilleure expressivité des nouvelles versions d'OWL et au support de la quantification sur les types des données. Afin de permettre la définition et la mesure de la QoS dans un système, l'ontologie DAML-QoS est organisée selon un principe de couches. Elle définit trois niveaux (ou étages) distincts dans l'ontologie : le niveau de profil de la QoS, le niveau de définition des propriétés de QoS et le niveau des métriques.
QoSOnt	QoSOnt a été introduite dans [Dobson et al., 2005], il s'agit d'une ontologie assez proche de DAML-QoS, qui se concentre sur la définition de métriques et sur le calcul de correspondance entre les besoins de QoS. L'ontologie QoSOnt a été définie de manière suffisamment générique pour permettre son extension dans les cas particuliers, elle reste ainsi indépendante de toute vision particulière de la QoS. De plus elle est organisée sous forme de plusieurs couches afin de favoriser son extensibilité et sa réutilisabilité. Les couches les plus hautes sont fondées sur les couches les plus basses, on peut descendre ainsi à partir des couches de domaine (par exemple network et service) jusqu'aux concepts de base de QoS. La couche la plus haute, Usage Domains, permet de lier les attributs de QoS définis dans la couche inférieure aux différents domaines d'application qui y sont définis. Cependant, comme pour WS-QoS, les auteurs ont défini leur ontologie dans un pseudo-langage fondé sur une structure XML

Tableau 5. 2: Analyse des ontologies de QoS existantes

De ces différents travaux, il ressort que de nombreuses ontologies ont été déjà définies pour l'expression des notions de Qualité de Service. Les concepts qu'elles proposent appartiennent à des niveaux d'abstractions les plus divers, cependant, de par la nature même des ontologies, il est toujours possible de dériver ses propres concepts « métiers » des classes déjà existantes. En conséquence de quoi, en cas de nouveau besoin en termes d'ontologie, il apparaît peu pertinent de développer ses propres modèles en partant de zéro, mais bien au contraire de se fonder, sinon de s'inspirer de l'existant.

Le modèle d'ontologie de QoS proposé s'inspire donc de l'existant (ontologies présentées ci-avant), il est organisé en trois niveaux:

Fig.5. 2: Niveaux de l'ontologie de QoS

(1) Le premier niveau: décrit le profil de QoS en termes de classes, d'attributs et de relations entre eux. Comme illustré dans la Fig.5. 2, selon notre point de vue chaque instance de service correspond à un profil de QoS qui définit les éléments suivants :

- **Profil_QoS** : le concept qui englobe à travers la relation « avoirAttribut » l'ensemble de paramètres de QoS d'une instance de service.
- **Attribut_QoS** : un attribut possède un nom (nom_attribut) et peut être soit une propriété commune (qui reflète un paramètre technique d'un service comme la disponibilité par exemple) ou une propriété de domaine (qui décrit une propriété métier relative au domaine du service comme par exemple la qualité des sièges dans un service de transport ferroviaire)
- **Métrique** : définit la mesure d'une propriété, une métrique peut être numérique (une valeur exact ou un intervalle) ou non numérique (texte, booléenne, grade...)
- **Valeur** : indique la valeur de la métrique de l'attribut
- **Unité** : indique l'unité de l'attribut
- **Tendance** : définit l'influence de la variation de l'attribut sur la qualité de service, elle peut avoir deux valeurs : croissante, dans ce cas la qualité de service augmente avec l'augmentation de la valeur de l'attribut (e.g la disponibilité, la fiabilité …), ou décroissante, dans ce cas la qualité de service baisse avec l'augmentation de la valeur de l'attribut (e.g., temps de réponse, coût….).
- **Agrégat** : un élément important dans le présent contexte de composition puisqu'il définit la formule de calcul de la valeur de l'attribut de QoS en cas de service composite.
- **Poids :** reflète l'importance et la priorité de l'attribut dans le profil de QoS à travers un poids.

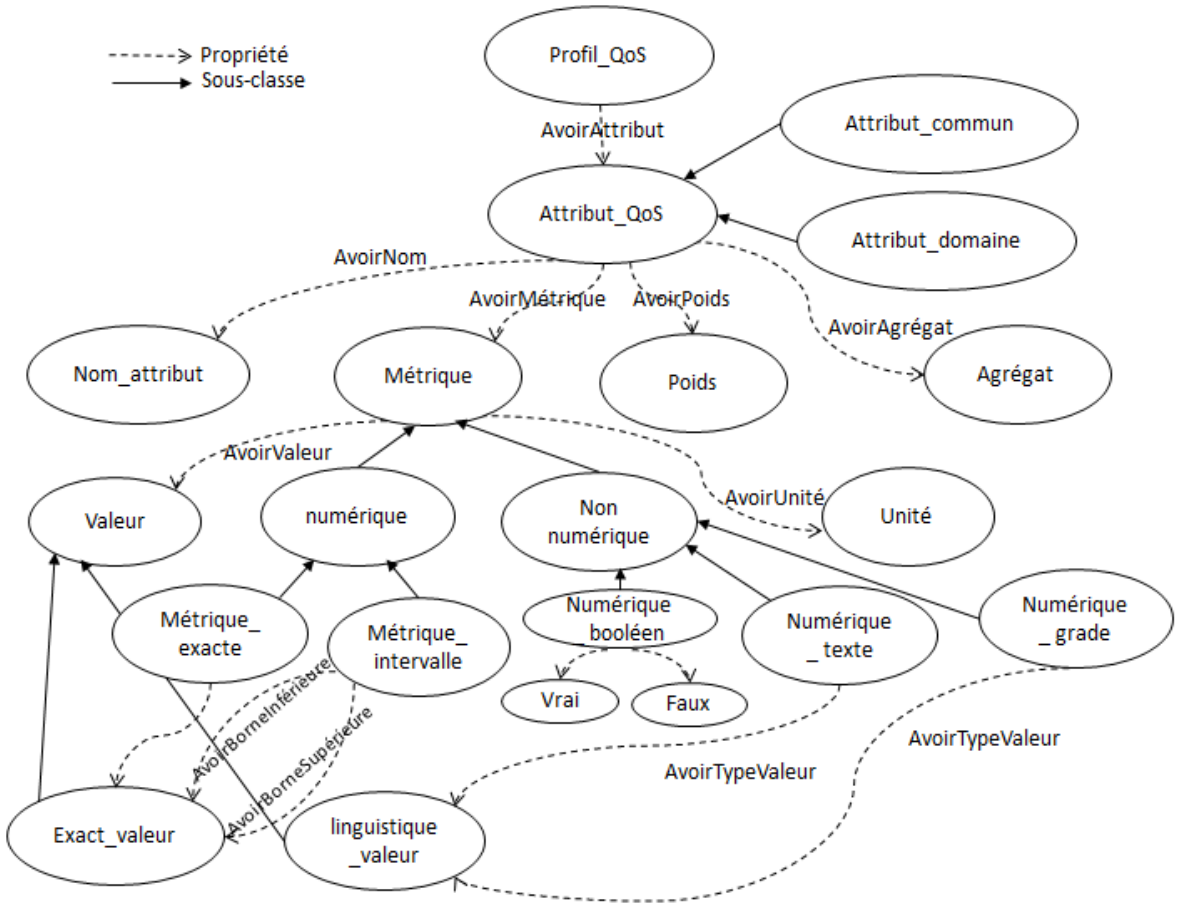


Fig.5. 3: Profil de QoS

(2) Le deuxième niveau: contient des propriétés dites techniques qui sont valables quel que soit le domaine du service étudié. Nous considérons à ce niveau le modèle décrit dans la référence [Yu et al., 2011], qu'on peut qualifier d'holistique et général puisqu'il englobe les principaux éléments de QoS communs : Accessibilité, Exactitude, Disponibilité, Capacité, Economie, Gestion des exceptions, Intégrité, Interopérabilité, Performance, fiabilité, Réputation, Robustesse, Scalabilité (monté en charge), Sécurité et Stabilité (Fig.5. 4).

Fig.5. 4: QoS techniques [Yu et al., 2011]

(3) Le troisième niveau : A contrario des deux autres niveaux qui sont stables et génériques, la structure et la constitution de ce niveau change en fonction du domaine étudié. Il s'agit dans ce niveau de recenser pour chaque domaine étudié les propriétés de QoS de services propres à ce domaine (e.g la propriété « qualité de son » ou « qualité du streaming » dans le domaine des médias), de définir les relations qui réunissent ces propriétés et de les organiser sous forme de profils de QoS selon le premier niveau.

5.2.3.1. Modèle de qualité de services (offre et cible)

En se basant sur la sémantique définie dans l'ontologie de QoS et l'extension d'OWL-S qui prend en charge la description des QoS, nous pouvons, à présent, introduire une définition complète et granulaire de l'offre d'un service Web Sémantique (SWS) comme suit:

Définition 8

$SWS_{OWL\text{-}SQ}$ = {N, D, IOPE, pQoS, O} où N est le nom du service, D est sa description, IOPE sont ses propriétés fonctionnelles, pQoS est le profil QoS du service (ses attributs leurs types et leurs valeurs) et O est l'ontologie de domaine.

D'autre part, comme mentionné antérieurement (c.f chapitre 4- section 4.3 : Analyse des spécifications), nous utilisons l'ontologie de QoS pour alimenter l'interface de communication ontologique afin de capturer les préférences de l'utilisateur sous forme d'une liste de couple {q,w} où q est la référence d'un attribut de QoS et w est le poids correspondant normalisé avec un facteur 1/5 i.e wi $\in$ [0,1]. A cette liste d'autres critères de qualité de services internes (générés automatiquement) sont ajoutés pour former la cible en termes de QoS (ou besoins non fonctionnels). Nous définissons alors « QoS », la liste des critères de qualité de service à fournir, ainsi :

Définition 9

$QoS = QoS_i \cup QoS_u$ où:

- $\rightarrow$ QoS_i : désigne la liste des critères de QoS internes
- $\rightarrow$ QoS_u: désigne la liste les préférences spécifiées par l'utilisateur

La contribution principale de la stratégie d'évaluation serait donc de définir le mécanisme de correspondance entre : (i) le modèle de l'offre en termes d'instance de services $SWS_{OWL\text{-}SQ}$ et (ii) le modèle de cible en termes de qualité de service QoS.
Pour ce faire, une précondition est nécessaire, il s'agit de déterminer ou estimer la valeur d'un attribut pour un service composite donné à partir des valeurs des attributs des services élémentaires, pour cette finalité un focus sur la propriété « agrégat » prédéfinie dans l'ontologie de QoS est donné dans la suite.

5.2.4. Agrégation des attributs de QoS dans la composition

« Ce qui est mesurable est gérable », afin de rendre les services composites gérables à travers leurs couches non-fonctionnelles à l'instar des services atomiques, nous avons alors besoin de calculer la résultante de QoS en se basant sur la valeur des attributs de QoS des services élémentaires. Il existe deux méthodes pour ce faire :

- Dans le cadre d'une approche qui ne distingue pas entre le niveau abstrait et le niveau concret de la composition, la sélection peut se faire pendant la composition. i.e., au fur et à mesure de la construction du plan, on fait appel au meilleur service élémentaire possible (en raisonnant sur l'ensemble de ses attributs de QoS). Contrairement à ce que cela laisse penser, cette stratégie conduit à une meilleure qualité des services élémentaires mais n'implique pas nécessairement une meilleure qualité de composition; puisque le fait de joindre deux meilleurs services ne donne pas nécessairement le meilleur résultat, cette démarche est donc exclue.
- L'autre méthode est basée sur la composition de chaque attribut de QoS pour l'ensemble des services élémentaires. Cependant cette méthode ne consiste pas en une simple addition ou multiplication des valeurs des attributs de QoS, chaque attribut doit

être traité d'une manière appropriée selon sa nature et sa signification. Notre démarche de calcul des attributs de QoS du service composite, expliquée ci-après, adhère à cette méthode.

Dans le cadre de ce travail nous utilisons une version adaptée de la formule d'agrégation initialement proposée par [Cardoso et al, 2004]. En effet, la formule de calcul de la résultante d'un attribut de QoS que nous proposons dépend de la fonction de conjonction utilisée dans la définition du processus de composition (la balise <process:CompositeProcess> dans la description OWL-S) d'une part, et de la classification des attributs d'autre part.

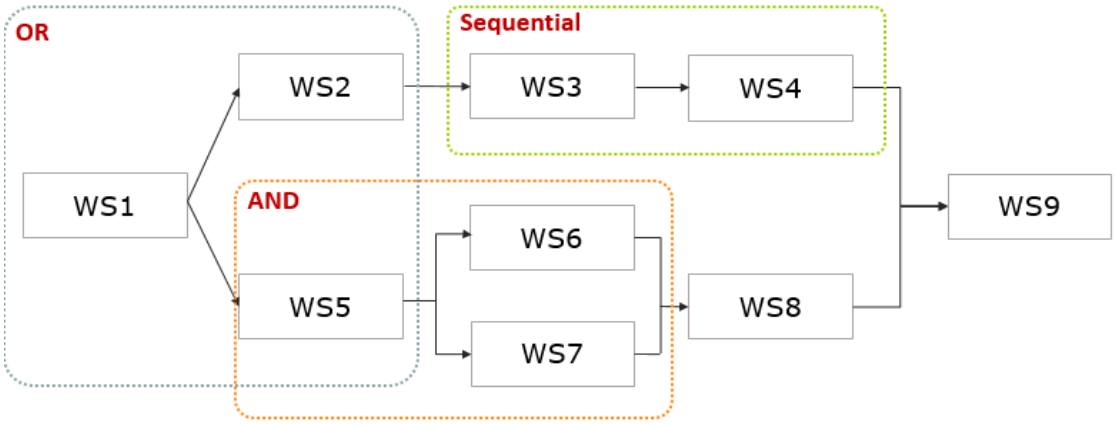


Fig.5. 5: Liaisons de composition dans un service composite

Fig.5. 5, présente un graphe de composition standard contenant les trois principaux types de liaisons possibles entre deux services élémentaires:

- Une liaison de type séquence : relie deux services où, le deuxième attend les résultats de l'invocation du premier, pour être invoqué.
- Une liaison « AND » : signifie que suite à l'appel d'un service donné, deux services (au moins) vont être déclenchés en parallèle.
- Alors qu'une liaison « OR » : signifie qu'un service parmi les services possibles va être invoqué.

D'un autre côté, afin de déterminer comment les attributs de QoS sont regroupés dans une composition nous avons également besoin d'utiliser une classification des attributs selon leurs réaction dans la composition, nous utilisons, pour ce faire, les classes ci-dessous:

Classe	**Description**	**Exemple**
Additive	les valeurs de ces attributs sont ajoutées aux valeurs des attributs du service suivant en se basant sur le graphe de composition	durée, utilisation de l'énergie
Multiplicative	les valeurs de ces attributs sont multipliées par les valeurs des attributs du service suivant en se basant sur le graphe de composition	sécurité, accessibilité, connexion
Extrême	des attributs qui dépendent de la valeur maximale ou le minimal de l'ensemble de services de la	débit

	composition	
Autres	des attributs qui ont une façon particulière de réagir dans la composition, en général ils fusionnent les autres réactions	coût

Tableau 5. 3: Les classes des attributs de QoS

En se basant sur la fonction de conjonction et la classification des attributs nous proposons les formules d'agrégation des valeurs des attributs de QoS présentés dans le Tableau 5. 4:

	Unité	Séquentiel	AND	OR	Exemple
Additive	Ad	$\sum_{i=1}^{n} ad_i$	$\max(ad_i)$	$\max(ad_i)$	durée
Multiplicative	mu	$\prod_{i=1}^{n} mu_i$	$\prod_{i=1}^{n} mu_i$	$\min(mu_i)$	disponibilité, fiabilité
Extrême	Ex	$\min(ex_i)$	$\min(ex_i)$	$\min(ex_i)$	latence
Autres	ot	$\sum_{i=1}^{n} ot_i$	$\sum_{i=1}^{n} ot_i$	$\max(ot_i)$	coût

Tableau 5. 4: Formules d'agrégation des attributs de QoS dans une composition

En utilisant les formules du tableau ci-dessus, nous pouvons calculer la valeur de la résultante de tous les attributs de QoS d'une instance d'un service composite en se basant sur le graphe de composition (ex. Fig.5. 5) et sur la classification des attributs de QoS (Tableau 5. 3).

5.2.5. Algorithme de sélection

Le dernier aspect de la stratégie de l'évaluation est l'algorithme de sélection qui détermine quelle instance de l'ensemble des services composites S^{ins} sera sélectionnée en se basant sur les spécifications de QoS, l'algorithme proposé se déroule en trois phases:

A. Phase 1: Construction de la matrice de sélection

L'algorithme construit dans un premier temps la matrice Q_{nm}, où *n* représente le nombre des alternatives des services Web composites concrets, et *m* représente le nombre de propriété de QoS à évaluer. $q_{i,j}$ désigne alors la valeur de l'attribut j pour le service composite S_i^{ins}, cette valeur est calculée par la formule d'agrégation décrite dans la partie précédente.

$$Q = \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1m} \\ q_{21} & q_{22} & \cdots & q_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ q_{n1} & q_{n2} & \cdots & q_{nm} \end{bmatrix}$$

B. Phase 2: Normalisation de la matrice

Pour qu'ils puissent être calculés et comparés les uns avec les autres, les différents attributs de QoS doivent suivre le traitement unifié d'une méthode adimensionnelle. Ainsi, nous normalisons les attributs dont la valeur est de classe numérique (et ceux qui peuvent être convertis à cette classe de valeur) par l'une des équations (1) ou (2) selon la propriété «tendance » de cet attribut (croissante ou décroissante).

$$v_{i,j} = \begin{cases} \dfrac{q_{i,j} - q_j^{min}}{q_j^{max} - q_j^{min}} & \text{Si } q_j^{max} - q_j^{min} \neq 0 \\ 1 & \text{Si } q_j^{max} - q_j^{min} = 0 \end{cases} \quad (1)$$

$$v_{i,j} = \begin{cases} \dfrac{q_j^{max} - q_{i,j}}{q_j^{max} - q_j^{min}} & \text{Si } q_j^{max} - q_j^{min} \neq 0 \\ 1 & \text{Si } q_j^{max} - q_j^{min} = 0 \end{cases} \quad (2)$$

Où : $v_{i,j}$ est la valeur normalisée du $j^{ième}$ attribut du $i^{ème}$ service

$q_j^{max} = \max (q_{i,j})\ 1 \leq i \leq n$, q_j^{max} est la valeur maximale de $j^{ième}$ colonne de matrice de qualité Q

$q_j^{min} = \min (q_{i,j})\ 1 \leq i \leq n$, q_j^{min} est la valeur minimale de $j^{ième}$ colonne de matrice de qualité Q

→ l'équation (1) est appliquée lorsqu'on tend à maximiser la valeur de $q_{i,j}$, $v_{i,j}$ prend alors la valeur $(q_{i,j} - q_j^{min})/ (q_j^{max} - q_j^{min})$, en cas optimal où $q_{i,j}= q_j^{max}$, $v_{i,j}= 1$, $v_{i,j}$ prend aussi la valeur 1 si $q_j^{max} - q_j^{min}=0$.

→ l'équation (2) est appliquée lorsqu'on tend à minimiser la valeur de $q_{i,j}$, $v_{i,j}$ prend alors la valeur $(q_j^{max} -q_{i,j})/ (q_j^{max} - q_j^{min})$, en cas optimal où $q_{i,j}= q_j^{min}$, $v_{i,j}= 1$, $v_{i,j}$ prend aussi la valeur 1 si $q_j^{max} =q_j^{min}$

Nous obtenons ainsi une matrice de sélection normalisée Q' tel que :

$$Q' | nq_{i,j} | = \textit{norme}\ (Q)$$

N.B : Avant de normaliser la matrice Q, un tableau $NR = \{nr_1, nr_2, \ldots\ldots nr_m\}$ est à établir en se basant sur l'ontologie QoS. Rappelons que le premier niveau de cette ontologie stocke des informations sur le profil de chaque propriété de QoS y compris sa tendance.
nr_j possède une valeur binaire (0 ou 1) où 0 indique que $q_{i,j}$ doit être normalisé en utilisant l'équation(1) alors que la valeur 1 indique que c'est l'équation (2) qui doit être utilisée pour normaliser $q_{i,j}$.

C. Phase 3: Introduction des poids

A ce stade, l'algorithme multiplie le vecteur de poids W, formé par les valeurs de poids introduites par l'utilisateur pour chaque critère de qualité, par la matrice de qualité normalisée en utilisant la formule 3. La somme pondérée ainsi calculée, définit le niveau de

correspondance entre ce qu'offre les alternatives de service en termes de qualité et la qualité attendue par l'utilisateur.

$$QoS\ (S^{ins}_i) = sum\_weight(Q', W) = \sum_{j=1}^{m} (v_{ij} \times w_j) \qquad (3)$$

Ou w_j indique le poinds de $j^{ième}$ attribut de QoS. On a $w_j \in [0,1]$ et $\Sigma w_j = 1$

D. Phase 4: Tri des instances de services composites

La dernière phase retourne l'instance de service recommandée pour l'exécution en utilisant la formule (4) et (5) qui trie les instances du service composite selon leur valeur de qualité de service QoS (S^{ins}_i) en utilisant la fonction Rank, puis retourne l'instance qui correspond à la valeur optimale de qualité de services avec la fonction select :

$$S_{rank} = rank\ (\ QoS\ (S^{ins}_i)) \qquad (4)$$

$$S_{optimal}\,(t) = select(S_{rank}) \qquad (5)$$

La synthèse de cet algorithme de sélection peut être exprimée ainsi :

select (rank (sum_weight (norme (QoS),W)))

L'idée clé de cet algorithme est de trouver l'instance de service la plus proche des spécifications de QoS, pour ce faire il trie les alternatives de services composites concrets en se basant sur leurs poids de point de vue utilisateur et retourne la meilleure alternative possible. Le mécanisme de fonctionnement de l'algorithme est repris dans les étapes ci-après.

Etape-1: Construire la matrice de sélection Q.

Etape-2: Normaliser la matrice Q en utilisant l'équation-1 et l'équation-2.

Etape-3: Evaluer le degré de correspondance (matching) entre l'offre et la demande en termes de QoS

Etape-4: Sélectionner la meilleure instance de service

Fig.5. 6: Algorithme de sélection

Les trois aspects ainsi définis, notamment l'ontologie de QoS, le calcul de la composition des attributs de QoS et l'algorithme de sélection, forment la technique de composition proposée pour conduire le comportement de l'agent Evaluator de l'architecture iDyCoS, afin de sélectionner les instances de services à exécuter.

« Mesurez ce qui peut être mesuré et rendez mesurable ce qui ne peut pas l'être », l'idée de base de la stratégie s'identifie parfaitement avec cette fameuse citation de Galilée [Galilée, 1982]. En effet, avec la prolifération des services Web nous étions dans l'obligation de chercher un autre critère de sélection pour choisir l'instance de service adaptée, la QoS semble le meilleur candidat pour tenir compte de cet état de fait, ainsi afin de rendre les services composites gérables à travers leurs couches non-fonctionnelles à l'instar des services atomiques, il fallait concevoir des mécanismes de mesures permettant de (i) calculer sur une base sémantique la résultante de QoS pour chaque critère de qualité pour un service composite donné puis de (ii) mesurer le degré de correspondance entre la qualité de service des instances disponibles et la qualité de service spécifiée par l'utilisateur, enfin (iii) de

proposer l'instance de service qui semble réaliser le compromis entre une qualité maximale et une qualité conforme.

Pour mettre en application la théorie présentée dans le cadre de cette stratégie d'évaluation, on reprend notre exemple illustratif de voyage.

5.2.6. Exemple de voyage

Afin de vérifier l'applicabilité de la stratégie d'évaluation proposée, nous utilisons, encore une fois, notre exemple illustratif.

Considérons, à titre indicatif, les critères de qualité suivants:

- **Disponibilité**: signifie l'aptitude d'un service d'être prêt, dans un contexte spécial et temporel, à produire les fonctionnalités demandées.
- **Coût**: représente la contrepartie que le demandeur doit payer (assurer) afin d'obtenir le service demandé.
- **Temps de réponse:** mesure le temps total entre l'envoi de la requête et la réception de la réponse.
- **Réputation**: représente une évaluation du service par le client en se basant sur son niveau de satisfaction suite à une expérience d'utilisation du service.

a. Liste de spécifications de QoS

A l'aide de ces critères nous définissons ainsi la liste de QoS:

QoS	QoS_1		QoS_2		QoS_3		QoS_4	
	Coût	0.4	Temps de réponse	0.2	Disponibilité	0.3	Réputation	0.1

b. Calcul de la valeur des attributs de QoS :

Rappelons que le plan de composition retenu dans la phase précédente est composé de trois services: un service de transport international, un service de transport local et un service de banque.

Supposons maintenant que pour le service élémentaire *transport international* l'Evaluator a pu trouver trois services concrets qui l'implémentent notons-les: s_{ti1}, s_{ti2}, s_{ti3} . Une seule instance du service banque : s_b a pu être invoquée et un seul service de transport local a été trouvé: s_{oncf}

Les alternatives du service composite concret qu'on peut former à la base des instances élémentaires trouvées sont:

$S^{ins}= \{S_1^{ins}, S_2^{ins}, S_3^{ins}\}$ tel que:

$S_1^{ins} = \{ s_{ti1}, s_b, s_{oncf}\}$, $S_2^{ins} =\{ s_{ti2}, s_b, s_{oncf} \}$, $S_3^{ins} =\{ s_{ti3}, s_b, s_{oncf} \}$

Nous donnons les valeurs des attributs de qualité de services pour chaque service élémentaire comme suit:

	Coût	**Temps de réponse**	**Disponibilité**	**Réputation**
s_{ti1}	500	10	0.8	0.6
s_{ti2}	400	10	0.7	0.5
s_{ti3}	600	10	0.7	0.8
S_b	10	5	0.9	0.7
S_{oncef}	10	20	0.5	0.3

Nous calculons alors les valeurs QoS des services composites en se basant sur les formules d'agrégation (Tableau 5. 4) (en considérant le graphe de composition Fig.5. 5: c'est la liaison « séquentielle » qui est utilisée pour la fonction de conjonction)

	Coût	**Temps de réponse**	**Disponibilité**	**Réputation**
Formule d'agrégation	Addition	Addition	Multiplication	Addition
S_1^{ins}	520	35	0.36	1.6
S_2^{ins}	420	35	0.315	1.5
S_3^{ins}	620	35	0.315	1.8
*Valeur maximale**	1000	55	1	3
*Valeur minimale**	300	5	0.1	0.3

**Indique les valeurs maximales/minimales du critère de qualité à prendre en considération lors de la normalisation*

c. Sélection de la meilleure instance

La matrice *Q* est donc définie ainsi

$$\boldsymbol{Q}= \begin{vmatrix} 520 & 35 & 0.36 & 1.6 \\ 420 & 35 & 0.315 & 1.5 \\ 620 & 35 & 0.315 & 1.8 \end{vmatrix}$$

Puisque nous tendons à minimiser *le coût* et le *temps de réponse* nous utilisons l'équation (2) pour normaliser les valeurs de ces deux attributs, alors que pour les valeurs de la *disponibilité* et la *réputation* nous utilisons l'équation (1) de maximisation.

Nous obtenons alors la matrice normalisée *Q' :*

$$\boldsymbol{Q'}= \begin{vmatrix} 0.68 & 0.4 & 0.03 & 0.48 \\ 0.83 & 0.4 & 0.02 & 0.44 \\ 0.54 & 0.4 & 0.02 & 0.55 \end{vmatrix}$$

Maintenant, afin de pondérer les valeurs des attributs nous multiplions la matrice normalisée par le vecteur du poids dont les valeurs ont été définies au début W= {0.4, 0.2, 0.3, 0.1}, en utilisant la formule (3) nous obtenons alors le vecteur suivant :

$QoS(S_1^{ins})$	0,409
$QoS(S_2^{ins})$	0,462
$QoS(S_3^{ins})$	0,357

Notons que chaque ligne de ce vecteur représente la valeur de QoS globale estimée pour une instance de service composite.

Finalement en utilisant la formule (4) c'est le servie S_2^{ins} qui est élu et passera en exécution, puisqu'il représente l'instance avec la valeur maximale de QoS.

5.3. Exécution autonomique

L'autonomique est un axe angulaire dans l'approche proposée. En effet, cette notion trouve une projection dans tous les mécanismes relatifs à chacune des phases de la composition proposées ainsi, la découverte des services candidats, la génération des plans de composition et l'optimisation des participants maximisant la qualité de service, s'effectuent par le système d'une façon tout à fait autonome et dynamique sans avoir besoin d'aucune intervention humaine. Pour compléter ce comportement autonomique, la phase d'exécution doit à son tour assurer des mécanismes en symbiose avec cette philosophie commune.

La phase d'exécution est la dernière phase dans le processus de composition, c'est la phase de concrétisation et de génération des résultats percevables par l'utilisateur final. Son rôle principal est le déploiement du plan exécutable généré par la phase précédente. Intuitivement, cette phase doit assurer une mission de veille, idéalement ceci se traduit en un **contrôle** de la bonne exécution du service composite, en une **détection** en temps réel que des critères/exigences ne sont plus vérifiés, enfin en un **ajustement** approprié et transparent de la situation en cas de violation des critères/exigences convenus.

En informatique autonomique ce genre de comportements est connu sous le nom du self-healing (auto- guérison). A notre connaissance, rares sont les travaux qui ont essayé de perfectionner les mécanismes de composition au point de prendre en charge la tolérance aux pannes. Encore moins rares ceux qui ont tenté d'adapter les techniques de l'Autonomic Computing au contexte de la composition pour atteindre cet objectif. Pourtant cet aspect de tolérance aux pannes est générateur de valeur ajoutée académique et surtout industrielle.

Ainsi, le mécanisme proposé dans cette section est le résultat de l'exploration des techniques émergeant dans l'informatique autonomique et ce, afin de concevoir un agent exécuteur autonomique, dont le fonctionnement génère un comportement de self-healing affectant l'ensemble du processus de composition.

5.3.1. Architecture de l'agent Executer

Le travail sur la phase de l'exécution de l'architecture iDyCoS a comme objectif la conception d'une architecture qui permet à l'agent exécuteur d'assurer en plus de sa fonction principale d'exécution, une fonction de monitoring (de contrôle). La référence théorique de ce travail de conception est le paradigme de l'Autonomic Computing, plus particulièrement l'architecture modèle de la boucle MAPE qui représente une structure typique pour implémenter un comportement autonomique.

Comme exposé au niveau du chapitre 2, en général, un manager autonomique, assure l'autogestion des éléments gérés qui peuvent être une ressource ou un ensemble de ressources qui ont la particularité d'être observables et contrôlables. Pour accomplir sa mission, un

manager autonomique doit implémenter une ou plusieurs boucles intelligentes de contrôle, connu sous le nom du MAPE loop. Une structure qui implémente cette boucle permet de :

1. Surveiller son propre comportement afin de détecter les défaillances de prestation de services (collecter, regrouper et filtrer les évènements des éléments gérés);
2. Analyser ces défaillances en vue de diagnostiquer les erreurs qui les ont causés (comparer ces artefacts contre une base de données des symptômes, les éléments analysés peuvent alors donner une indication sur la classe d'anomalie supposée et sur les solutions possibles);
3. Planifier des stratégies de réparation appropriées (choisir une des solutions du problème) ;
4. Exécuter ces plans pour restaurer le comportement normal du système (effectuer les actions de la solution qui continueront à être exécutées par le manager autonomique pour contrôler les éléments gérés).

Le principe décrit ci-avant est parfaitement applicable dans notre cas. En effet, on peut rendre l'agent exécuteur un manager autonomique qui autogère l'exécution, un processus observable et contrôlable à travers l'implémentation de la boucle intelligente. Sauf que, étant donné le processus de composition des services Web proposé, l'application de la boucle MAPE conventionnelle présentera quelques soucis d'optimisation. Par exemple quand un service n'arrive plus à produire le résultat attendu, l'analyse va être faite et le système va essayer de trouver un autre plan. Mais la re-planification peut parfois ne pas être nécessaire si, par exemple, le service a été retardé en raison de la lenteur du réseau, dans ce cas une simple réexécution du service peut être suffisante.

Ainsi l'architecture cible de l'agent exécuteur doit implémenter une version adaptée de la boucle MAPE, dans ce qui suit nous exposons notre proposition à cet égard, en s'appuyant sur l'examen et l'évaluation de plusieurs approches des systèmes auto guérison [Friese et al., 2005] [Gurguis et al., 2005] [Madhusudan et al., 2006] [Mikic-Rakic et al., 2002] [Naccache et al., 2007] [Frei et al., 2013] [Schneider et al., 2014].
L'architecture proposée encapsule le mécanisme de self-Healing en deux principaux composants :

Le premier composant est le module de l'exécution, son objectif est:

- l'exécution du service composite en s'appuyant sur l'infrastructure d'exécution préparée pour ce besoin.

Le deuxième composant réunit les modules responsables:

- du contrôle et de capture des anomalies d'exécution (Module Monitoring);
- et de l'analyse des erreurs et la détermination des solutions (Module Réparation).

Les éléments incarnant le rôle de ces composants ainsi que leurs interactions sont illustrés dans la figure ci-après qui met en relief l'architecture interne de l'agent Executer. Cette architecture implémente une boucle MAPE revue composée par : (i) un module d'exécution, (ii) un module de communication, (iii) un module de Monitoring et (iv) un module de Réparation en plus (vi) d'un annuaire des anomalies.

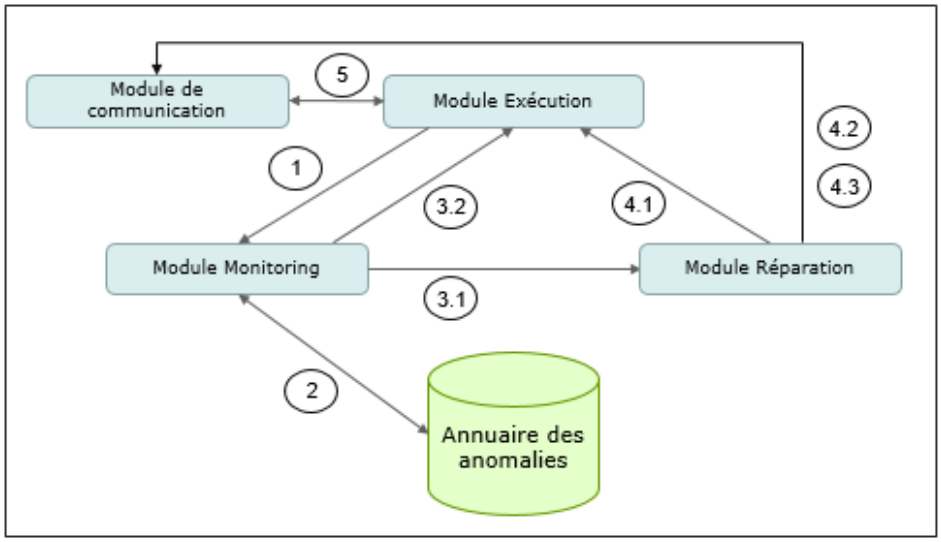


Fig.5. 7: Architecture de l'agent Exécuter

- **Le module Communication** : Module commun entre les différents types d'agents, toute communication entre l'agent Executer et les autres agents passe par ce module.
- **Le module Exécution :** Responsable sur l'exécution du service composite. Malgré son rôle central, ce module ne sera pas détaillé dans le cadre de ce travail car il n'est sujet d'aucune technique de composition particulière. L'infrastructure nécessaire pour cette opération consiste principalement en un moteur d'exécution, le moteur WS-BPEL prévu à être utilisé dans le cadre de ce projet est ActiveBPEL [ActiveBPEL, 2016], c'est un projet open source écrit en java. En fait, les deux moteurs open sources les plus populaires sont Apache ODE [ApacheODE, 2016] et ActiveBPE. En raisonnant performance, Apache ODE est meilleur, mais ActiveBPEL possède un excellent support, son moteur est appuyé par une communauté très active.

- **Le module Monitoring :** Pour être en mesure d'assurer un comportement de self-healing, il est nécessaire de détecter d'abord si un comportement anormal est survenu et ce le plutôt possible et en parallèle avec l'exécution du service composite. Le module Monitoring assure cette fonction, il utilise un contrôle basé évènement (event-based monitoring) [Mühl et al., 2006], ce type de contrôle est utilisé lorsqu'il est nécessaire de vérifier les critères non-fonctionnels d'un service. Il s'agit de l'approche la plus transparente et la moins interventionniste, puisqu'en parallèle de l'exécution, le module Monitoring écoute les événements lancés par le moteur BPEL. Aucun impact sur la performance n'est donc attendu. Par contre, il est nécessaire de mentionner que cette approche dépend du type du moteur d'exécution utilisé. Si nous utilisons ActiveBPEL par exemple, il est possible d'intercepter une série d'événements sous le standard d'activités BPEL (i.e. invoke, receive, etc.) ainsi que leur changement d'état. Chaque activité peut alors être, par exemple, dans l'état «prête à être exécuter», dans l'état «en exécution», dans l'état «terminée» ou dans l'état «erreur», en écoutant ces évènements il est possible de raisonner sur quelques critères de QoS.

- **Le module Réparation :** Permet au système de continuer l'exécution même après la découverte d'un comportement erroné. Dans le présent travail nous prévoyons trois stratégies de réparation (Fig.5. 8):
  - ***Réexécution:*** cette stratégie consiste, tout simplement, à re–exécuter le service qui a provoqué le problème.
  - ***Resélection****:* selon cette stratégie, pour résoudre le problème survenu, il faut essayer de re- sélectionner un autre service capable d'offrir les mêmes fonctionnalités du service défectueux avec un niveau acceptable de QoS.
  - ***Replanification*****:** il s'agit, au niveau de cette stratégie, de faire appel au Planner pour générer un nouveau plan qui, une fois exécuté, offre les mêmes fonctionnalités du service défectueux avec un niveau acceptable de QoS.

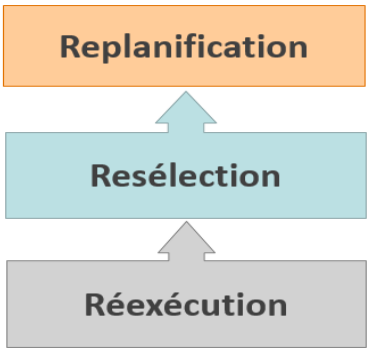


Fig.5. 8: Les trois stratégies de réparation

- **Annuaire des anomalies :** Afin de structurer l'aspect tolérance aux pannes, les « pannes » possibles sont organisées dans des catégories de contexte. La notion de contexte fait référence à ce niveau aux facteurs de l'environnement de l'exécution qui peuvent impacter cette dernière (et causer des erreurs). Dans notre cas nous nous concentrons sur la catégorisation des aspects en relation avec les QoS du service. A titre illustratif nous nous pouvons considérer dans une première implémentation trois catégories d'anomalies :
  - non-réponse: le service invoqué ne répond pas ;
  - sécurité: le niveau de sécurité attendu ne peut être achevé ;
  - performance: le service demandé ne s'exécute pas avec le niveau d'efficacité attendu.

5.3.2. Cycle d'exécution autonomique

En se référant aux interactions schématisées dans Fig.5. 7, le cycle d'exécution se déroule ainsi : après l'élection du meilleur plan concret (service composite concret), il est passé au module Exécution (5). Pendant l'exécution du plan, cette dernière est contrôlée en continue et en temps réel par le module Monitoring qui, regroupe, filtre et compare les évènements interceptés du module Exécution (1). Le Module Monitoring se base sur la catégorisation proposée par l'annuaire des anomalies pour identifier l'état de violation des contraintes/critères de contexte (2). Si à n'importe quel stade durant l'exécution, le module Monitoring détecte une violation des exigences, il notifie le module Réparation (3.1), à ce moment, une des stratégies de réparation prend place.

Il existe trois façons pour remédier aux erreurs de l'exécution :

(a) La première action à prendre est la réexécution du plan actuel, dans la plupart des cas les erreurs causées par le retard ou la non réponse d'un service Web dû à la lenteur du réseau, peuvent être résolues après quelques réexécution du plan concret (4.1).

(b) Dans le cas où la réexécution échoue, le module Réparation vérifie les changements du contexte de l'utilisateur à partir de la dernière évaluation faite. Si un changement a eu lieu, l'Evaluator est notifié, ce dernier est appelé pour générer un nouveau classement des services concrets et envoyer le premier service pour l'exécution. Si aucun changement n'a eu lieu, l'Evaluator choisit le candidat suivant dans la liste de classement des services composites concrets qui ne contient pas le(s) instance(s) défectueuse(s) et l'envoi pour exécution (4.2).

(c) Si le problème persiste ou si la violation des critères est au niveau du type de service (toutes les instances disponibles d'un ou plusieurs services composants), le module Réparation demande à l'agent Planner de concevoir un plan partiellement ou complétement nouveau, à partir de l'état actuel pour intégrer les dernières informations disponibles (4.2).

Parce que l'Executer ne peut pas exécuter les actions de réparation pendent l'exécution, une fois une violation des critères/exigences est détectée, le processus d'exécution est suspendu, il devient en mode « réparation » (3.2). Une fois les actions de réparation sont accomplies, le processus revient en mode « exécution ». Le module Exécution continue alors l'exécution et, parallèlement, le module Monitoring reprend le contrôle de l'exécution du nouveau plan (5).

5.3.3. Exécution autonomique: analyse et points d'amélioration

Les trois modules d'exécution, de monitoring et de réparation forment l'épine dorsale de la technique d'exécution autonomique proposée. En effet, l'interaction entre ces trois modules permet de générer un comportement d'auto-guérison à la portée de tout le processus de composition.

L'objectif de la présente technique de composition n'est pas de proposer les mécanismes nécessaires pour exécuter les services composites, mais plutôt de mettre en place un mécanisme de protection de cette exécution quels que soient les mécanismes d'exécution utilisés, pour concevoir un tel mécanisme, la réflexion a été menée au niveau de trois dimensions complémentaires :

- **la méthode de Monitoring** : il s'agit d'une méthode basée évènements, cette méthode est efficace lorsqu'il s'agit de contrôler la violation des exigences de qualités de services (ce qui est le cas), mais comme expliqué auparavant, cette méthode dépend de l'implémentation interne du module Exécution (moteur d'exécution). La recherche doit être menée dans le sens d'hybrider la méthode basée évènements avec d'autres méthodes de Monitoring [Moser et al., 2010] afin de diminuer la dépendance entre le module Exécution et le module Monitoring.
- **la méthode d'analyse** : l'analyse s'appuie essentiellement sur une base de symptômes qui catégorise les anomalies/erreurs de contextes les plus connus. L'amélioration à ce

niveau concerne l'aspect dynamique, i.e., analyser les comportements anormaux non reconnus par l'annuaire des anomalies et alimenter dynamiquement ce dernier.

- **Les stratégies de réparation** : il s'agit à ce niveau d'une affaire de décision, *en cas de violation des exigences est ce que la re-planification (et éventuellement la redécouverte) est nécessaire ou pas ?*
  Pour des raisons de performance, la solution proposée à cet égard vise à éviter le maximum possible la re-planification, pour ce faire elle organise les actions de réparation dans une hiérarchie: Réessayer- Substituer – Restructurer. L'action d'un niveau supérieure ne peut être entreprise qu'après l'échec de l'action du niveau inférieur. Là encore nous constatons les bienfaits d'une vison de composition à deux niveaux qui donne la possibilité d'effectuer des actions de réparation sans toucher à la structure (abstraite) du service composite produit.
  D'autres améliorations dans le respect de cette orientation peuvent être explorées. Il s'agit notamment de pousser l'investigation dans le sens de rendre le processus de décision des actions de réparation plus efficace et plus fiable, notamment en utilisant une analyse basée sur des techniques de Machine Learning [Todica et al., 2012].

5.4. Conclusion

Ce chapitre met en exergue les techniques de la composition concrète, ces techniques se caractérisent par leur manipulation des instances de services et par la production d'un résultat finalisé, percevable par l'utilisateur. En effet, la stratégie d'évaluation examine chaque instance possible du service composite pour choisir la plus adaptée en se basant sur les propriétés de QoS. L'architecture de l'agent Exécuter permet, quant à elle, de fiabiliser la phase de l'exécution et de générer un comportement de self-healing à la portée de l'ensemble du processus de composition. Ces deux techniques complètent les trois techniques exposées dans le chapitre précédent, ensemble elles façonnent et dynamisent le processus de composition.

Un nouveau tournant va être pris dans les chapitres suivants dont l'enjeu sera de prouver la viabilité et la pertinence de l'approche proposée dans des conditions réelles et de mettre en évidence son utilité pour résoudre des problématiques complexes.

Troisième partie

Application et évaluation

Chapitre 6

Domaine d'application : Gouvernement Electronique

Chaque problème que j'ai résolu est devenu une règle,
qui a servi ensuite à résoudre les autres problèmes.
René Descartes

6.1. Introduction

Un modèle, qu'il soit simple ou compliqué, n'a d'utilité que s'il peut être appliqué à un système réel et s'il permet de faciliter la compréhension d'une problématique donnée. Ce chapitre s'inscrit donc dans une démarche d'application, de test et d'évaluation de l'approche de composition des services Web proposée.

Jusqu'à présent notre approche s'est présentée comme une Framework conceptuel de composition dynamique de services Web qui sert à automatiser l'ensemble des étapes du processus de composition de la découverte jusqu'à l'exécution en passant par la planification et l'évaluation. Elle propose en outre des techniques de composition qui résolvent efficacement les problèmes liés à l'hétérogénéité sémantique, l'intégration, la décentralisation et de la dynamique. Même s'il parait que l'e-business constitue le domaine d'application le plus évident d'un tel Framework, nous estimons que notre proposition peut efficacement être testée et évaluée dans un domaine qui adhère à la logique de Service mais qui requiert un espace d'échange d'information plus large, plus dynamique et plus hétérogène.

Une vue holistique sur les secteurs de l'économie numérique [OCDE, 2015] montre qu'il existe principalement cinq importantes applications : e-commerce, e-gouvernement, l'éducation à distance (e-learning), la télémédecine, le divertissement électronique, qui désignent l'utilisation des technologies de l'information et de la communication pour améliorer la qualité des services liés respectivement à des transactions commerciales, à des procédures administratives, à des formations (programmes éducatifs) à des soins sanitaires et au divertissement.

Dans cette collection, l'e-gouvernement présente un potentiel intéressant. En effet, l'e-gouvernement se distingue par opposition aux scénarios des autres domaines « E-» par le défi d'atteinte de l'interopérabilité et de l'intégration auxquelles s'intéresse particulièrement notre approche de composition de services Web. Nous estimons alors que notre approche pourrait être efficacement testée et évaluée sur le domaine d'e-gouvernent.

Dans la suite de ce chapitre nous examinons minutieusement le domaine d'e-gouvernement. Nous proposons ensuite une projection de notre architecture de composition des services Web

sur ce domaine tout en soulignant les retombées bénéfiques de l'adoption de notre approche pour faire face aux principaux défis techniques de ce domaine.

6.2. E-gouvernement : Etude du domaine

6.2.1. L'émergence du gouvernement électronique

Le gouvernement électronique (ou encore e-gouvernement) est un concept qui est apparu dans l'administration publique vers la fin des années 1980. La formule la plus classique consiste à définir ce concept comme le fruit de la rencontre entre le numérique et le gouvernement. Il s'agit, selon l'Organisation des Nations unies (ONU) [UN, 2008], de «l'utilisation des technologies de l'information et des communications (TIC) pour améliorer les activités des organisations du secteur public et de leurs agents. De tels efforts peuvent être dirigés vers la prestation de services aux citoyens (Front Office), ou à la modernisation des pratiques du travail et à la réalisation et l'amélioration de l'efficacité opérationnelle des bureaux (back-office) ».

L'e-gouvernement est considéré alors comme l'application de moyens électroniques pour transformer les services publics en les rendant plus accessibles et plus efficaces [InfoDev, 2002]. Selon la commission européenne [European Commission, 2005], cette transformation doit se conjuguer avec des changements au niveau organisationnel pour que l'e-gouvernement puisse atteindre son objectif d'optimisation des schémas des relations dont le gouvernement est partie prenante [IPCS, 2003].

L'apparition de l'e-gouvernement est corrélée à l'émergence de quatre grandes tendances interdépendantes sur le marché international au cours de ces vingt dernières années :

1. **L'innovation** : l'ère numérique est associée à la généralisation des vagues successives d'innovations axées sur la technologie de l'information et de la communication. Des technologies comme internet, le commerce électronique et le mobile ont apporté avec eux, une connectivité ubiquiste, un accès en temps réel, un échange d'information moins couteux, et un grand volume de données et d'information. La généralisation de l'utilisation de ces technologies par les individus, les organisations et notamment les gouvernements est devenue une réalité dont l'usage doit être orienté vers l'augmentation de la valeur, pour les organisations publiques c'est la valeur publique qui est concernée.
2. **La société d'information:** la transition de l'économie industrielle vers l'économie d'information que le monde est en train de vivre, rend l'information une nécessité stratégique pour les organisations [Eckerson, 2002]. La société d'information est un terme qui désigne alors une société pour laquelle la création, la distribution et la manipulation des informations sont devenues les activités économiques et culturelles les plus significatives [Whatis, 2016]. Actuellement, l'un des défis majeurs des gouvernements est justement d'assurer la transformation en société d'information, qui passe nécessairement par l'instauration d'une démarche e-gouvernement.

3. **La mondialisation** : les changements de conditions de concurrence ont forcé les organisations d'adopter progressivement des stratégies de mondialisation [Lambert et al., 2000], l'e-gouvernement s'inscrit, justement dans le cadre du processus accéléré de mondialisation, qui concerne aussi bien le secteur public que le secteur privé.
4. **La démocratie**: la reconnaissance de l'importance de l'interaction entre les gouvernements et leurs citoyens dans le processus de la prise de décision est l'un des aspects fondamentaux de la modernisation, la commission européenne considère, même, ce type de participation élargie, un principe vital de la démocratie [European Commission, 2005]. Dans ce sens l'e-gouvernement est un outil susceptible de rénover l'interaction gouvernement-citoyen.

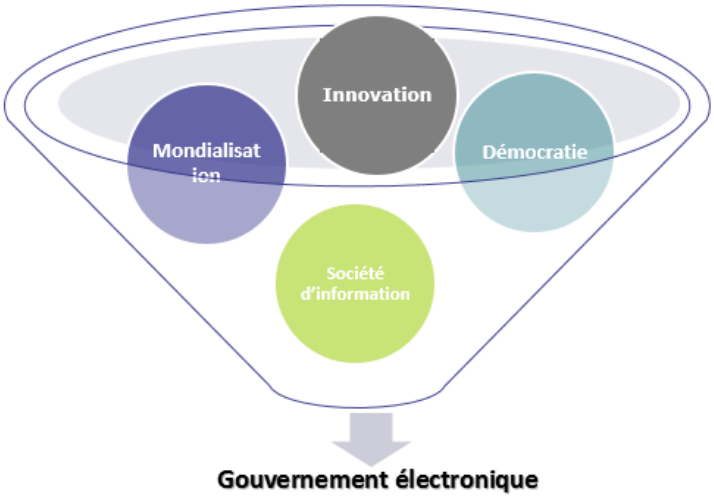


Fig.6. 1: Contexte de l'émergence de l'e-gouvernement

6.2.2. Les domaines de l'e-gouvernance

L'e-gouvernement implique l'automatisation ou l'informatisation des procédures existantes non-informatisées, ce qui conduit à de nouveaux styles de leadership, de nouvelles façons de débattre et de décider des stratégies, d'interagir avec les sociétés commerciales, d'écouter les citoyens et les communautés, d'organiser et de livrer l'information, donc cette informatisation conduit essentiellement à de nouvelles façons de gouverner. Par conséquent, le processus qui consiste à utiliser, améliorer, inventer et gérer les outils d'e-gouvernement pour gouverner est appelé **l'e-gouvernance** [Heeks, 2001].

L'e-gouvernance peut être vue alors comme un concept plus large qui définit et évalue les impacts que les technologies ont sur la pratique et l'administration des gouvernements et sur les relations entre les fonctionnaires et la société en général. L'e-gouvernance englobe aussi une série de mesures nécessaires à mettre en place par les organismes publics afin d'assurer la mise en œuvre réussie des services d'e-gouvernement pour le grand public.

Le modèle de l'e-gouvernance de référence [Heeks, 2001] décompose ce concept en trois domaines qui recouvrent les projets de l'e-gouvernement.

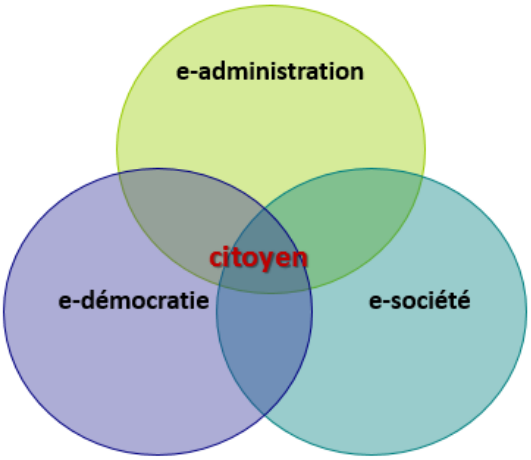


Fig.6. 2: Domaines de la gouvernance en ligne

a) l'e-administration (administration en ligne, administration électronique ou encore cyber administration) qui est en fait l'application d'e-gouvernement dans sa relation avec les citoyens et les entreprises en tant qu'administrés. C'est le domaine de la prestation électronique de service qui consiste à offrir aux administrés - citoyens et entreprises - la possibilité de procéder en ligne à leurs transactions avec l'administration publique.
b) l'e-démocratie est l'axe d'e-gouvernement qui développe la relation avec le citoyen en tant qu'acteur politique. C'est le domaine du e-voting (ou vote électronique), mais aussi des forums de discussion pour permettre aux citoyens d'échanger et d'enrichir les débats politiques.
c) l'e-société est l'axe de développement des technologies de l'information et de la communication dans la société.

A priori, ces trois domaines ne sont pas mutuellement exclusifs, leurs frontières n'étant pas clairement établies. Cette définition permet de distinguer les projets qui touchent principalement les processus administratifs (e-administration), de ceux qui touchent plutôt la participation politique (e-démocratie) et de ceux qui touchent le développement sociétal dans son ensemble (e-société).

6.2.3. Les schémas d'interactions de l'e-gouvernement

L'organisation des services électroniques du gouvernement dans trois domaines génère plusieurs typologies d'interactions [Urs, 2012] que l'e-gouvernement tend à rendre plus conviviales, plus transparentes et moins cheres. Le Tableau 6.1 présente la matrice des interactions d'e-gouvernement qui font intervenir les partenaires principaux du gouvernement notamment le citoyen, l'entreprise et le gouvernement lui-même.

	Citoyen	**Gouvernement**	Business
Citoyen	C2C	C2G	C2B
Gouvernement	**G2C**	**G2G**	**G2B**
Business	B2C	B2G	B2B

Tableau 6. 1: la matrice "X2Y" des interactions de l'e-gouvernement

i. Government to citizen (G2C)

Par « *Government to Citizen* » [Garson et al., 2008], il faut entendre l'ensemble du dispositif communicationnel entre un gouvernement et ses citoyens. Il s'agit, généralement de donner aux citoyens la possibilité de communiquer de manière permanente et directe avec leur gouvernement, de se renseigner, de s'informer, de demander et de recevoir des documents, de réaliser des téléprocédures, de déclarer et de payer leurs impôts ou taxes.

Le sens inverse de ce type de relation (C2G) inclus les interactions dont le citoyen devient un collaborateur et un acteur pour la prise de décision (i.e., les réclamations en ligne, le vote en ligne).

ii. Government to business (G2B)

Il est évident que, dans toutes les étapes de leur évolution, les entreprises soient soumises à un nombre élevé de démarches, de procédures ou de formalités administratives. Le *Government to Business* [Garson et al., 2008] simplifie la relation de l'entreprise avec l'administration et constitue un gain de temps et un facteur non négligeable de renforcement de la productivité.

iii. Government to government (G2G)

Les TIC permettent une meilleure communication intergouvernementale. Avec la mise en place d'intranets gouvernementaux, l'e-gouvernance devient un moyen efficace pour inciter les agents publics des différents départements et organismes de l'État à mieux communiquer, à s'échanger des informations, à partager leurs expériences et leur savoir-faire à travers les espaces de travail au sein des portails collaboratifs. Le schéma d'interaction « *Government to government* »[Garson et al., 2008] correspond à ce niveau d'intégration intergouvernementale.

6.2.4. Modèle d'évolution de l'e-gouvernement

La mise en place d'un gouvernement électronique est un processus à long terme qui implique l'implémentation de projets couvrant les domaines d'e-service, e-démocratie et e-administration et l'instauration des relations fiables et durables avec l'ensemble des parties prenantes.

Selon [Layne et al., 2001] et [Chen, 2002] cette mise en place de l'e-gouvernement suit un modèle d'évolution à quatre phases.

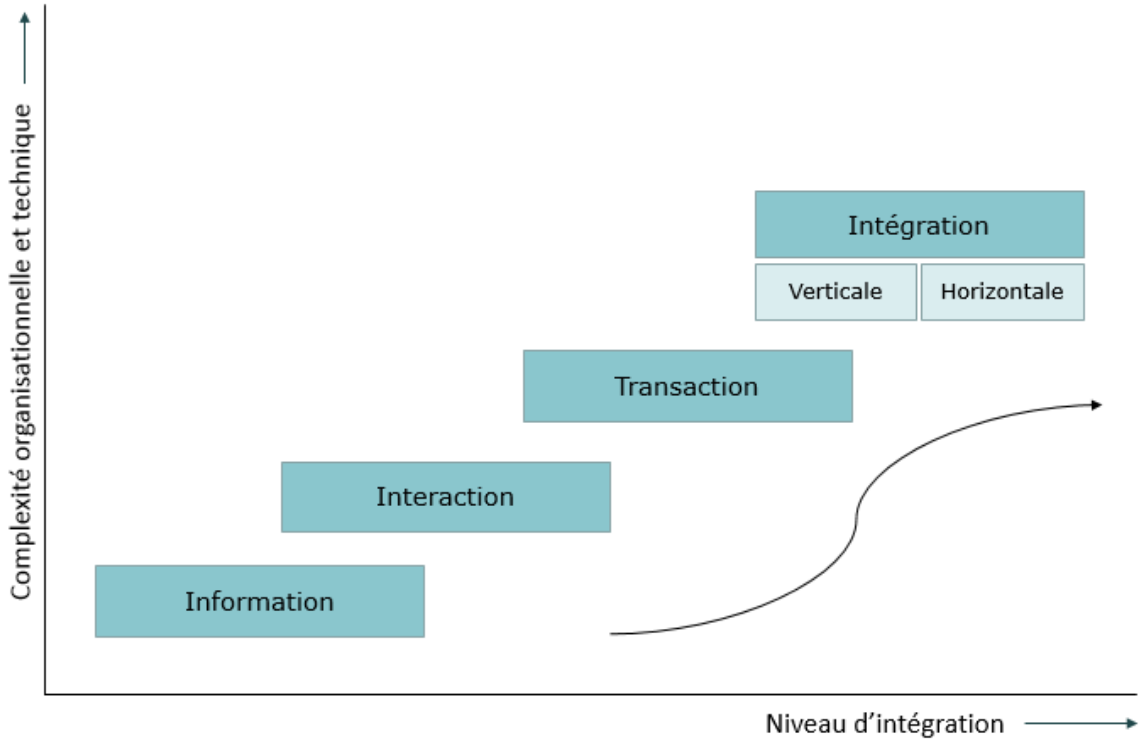


Fig.6. 3: Cadre conceptuel de l'évolution de l'administration électronique [Layne et al., 2001]

Comme illustré dans la Fig.6.3, le cadre conceptuel de l'évolution de l'administration électronique est décomposé en quatre grandes phases d'évolution:

1) **Phase d'information** (présence en ligne):
   Pour la plupart des organisations, la phase d'information est la première étape de développement du gouvernement électronique. Le but de cette phase est de disposer d'une présence en ligne sur Internet afin d'exploiter ce nouveau canal de communication. Le site Web s'ajoute à l'ensemble des autres canaux d'information et de diffusion d'informations officielles notamment les dépliants, les centres d'appel téléphonique, les centres de service et les fax. Cette présence peut être décrite comme un catalogue d'information qui comporte des informations d'ordre général sur l'organisation mais aucune communication personnalisée ni bidirectionnelle avec les citoyens n'existe au niveau de cette phase.
2) **Phase d'interaction :**
   À la phase d'interaction, l'organisation propose un lien de communication plus personnalisé avec les citoyens en implantant une prestation électronique de service qui utilise la messagerie électronique, un moteur de recherche, des forums publics, des listes d'envoi et le téléchargement de formulaires. C'est le début d'une communication électronique bidirectionnelle avec le citoyen et d'une gestion systématique du contenu et des documents. On peut dire que cette phase jette les bases de la mise en place d'un guichet de services « virtuels » d'information plus personnalisée.

3) Phase de transaction :

La phase de transaction est un prolongement de la phase d'interaction, elle y ajoute des télé- procédures qui permettent de soumettre des documents électroniques et le paiement en ligne afin de réaliser une transaction complète. On assiste au début du « self-service » qui automatise fortement les aspects information, interactions et traitements de certaines transactions. On intègre au potentiel de communication celui du traitement de l'information. On peut ainsi gérer électroniquement le cycle complet d'une transaction depuis la collecte des données en passant par les traitements jusqu'à son archivage. On assiste alors à une simplification et à une automatisation des procédures administratives entre le gouvernement et ses partenaires.

4) Phase d'intégration:

Pour définir la phase d'intégration, on présente deux types de projets d'intégration de prestation électronique de service. Le premier touche l'intégration des services électroniques intra-organisationnels (intégration verticale) tandis que le deuxième touche l'intégration des services électroniques inter-organisationnels (intégration horizontale).

I. Intégration verticale : le service intra organisationnel intègre les services offrant des fonctions similaires ou complémentaires qui sont décentralisées à plusieurs niveaux (national, régional, local).

II. Intégration horizontale : il s'agit d'une intégration des services inter-organisationnels qui vise une intégration totale de prestation électronique de service de diverses organisations sur un même portail. Le citoyen peut alors bénéficier d'un point d'entrée unique pour faire une demande de service qui exige la collaboration de plusieurs organisations.

Ces deux types d'intégration ne sont pas mutuellement exclusifs, il est possible qu'une organisation implante un projet qui touche à la fois l'intégration interne de ses processus et l'intégration des processus inter-organisationnels avec ses partenaires.

On ressent à cette étape le besoin d'une étude plus approfondie des quatre phases d'évolution d'un point de vue technologique afin de localiser la ou les phases où il serait opportun de contribuer. La revue littéraire des enquêtes traitant l'évolution du gouvernement électroniques [St-Amant, 2004] [Commonwealth of Australia, 2003] et l'étude des différents modèles d'évolution de l'e-gouvernement [Layne et al., 2001] [Chen, 2002] [Deloitte Research, 2000] [Seiffert, 2002] ont permis de dresser l'état de synthèse illustré dans le Tableau 6. 2.

Caractéristiques	Phase d'information	Phase d'interaction	Phase de transaction	Phase d'intégration
Fonctionnalités	Consultation de pages html	▪ Echanges électroniques et communication personnalisée avec le citoyen ▪ Gestion des contenus	▪ Procédures administratives en ligne (formulaires) ▪ Traitement à distance de transactions	Traitement en ligne de demandes personnalisées
Communication	Communication unidirectionnelle	▪ Communication bidirectionnelle	▪ Echange interactif d'information	Accès à un dossier électronique personnalisé
Technologie	Technologie Web de base	▪ Moteur de recherche ▪ Forum public ▪ Téléchargement de formulaires	▪ Echanges électroniques de données sécurisées ▪ Authentification en ligne ▪ Signature électronique	Technologie évoluée d'intégration
Systèmes d'informations	Systèmes d'information en silos isolés	▪ Systèmes d'information en silos isolés	▪ Coopération avec les Systèmes d'information déjà implantés	Intégration complète des systèmes d'information

Tableau 6. 2: Caractéristiques techniques des phases d'évolution de l'administration électronique

Pour la plupart des gouvernements, la phase d'intégration est la prochaine phase prévisible de l'administration électronique. En effet, comme présenté dans le Tableau 6.2, les trois premières phases sont des phases de transformations organisationnelles mineures des processus actuels de service à l'aide des TIC. Ces transformations ne permettent pas une intégration complète de la prestation électronique de service. On utilise progressivement les technologies interactives multimédia (courriel, forum, messagerie électronique), de gestion de documents électroniques (PDF, moteur de recherche) pour les relier aux systèmes d'information actuels afin de réaliser électroniquement certaines transactions standardisées. Parallèlement, des mécanismes de support se mettent en place pour assurer la sécurité et la confidentialité des renseignements personnels (authentification, cryptage).

La phase quatre est par contre une phase de transformations majeure des processus de prestation électronique de service et des structures organisationnelles afin de permettre d'offrir des services électroniques complets, étendus et fortement intégrés intra et inter-organisationnels via un portail de service « one stop » qui intègre fortement le Front-Office et le back-office d'une organisation ainsi que les portails d'autres organisations, d'autres paliers de gouvernements et/ou de certains partenaires privés.

Le principe de portail de service « one stop » qui nécessite l'intégration d'un ensemble hétérogène et distribué semble formuler l'équation adéquate pour le déclenchement du processus de composition tel que nous l'avons conçu.

Convaincus que le domaine de l'e-gouvernement est le meilleur «testbed » pour notre approche de composition de service Web, l'étude de ce domaine, nous a permet de repérer les problématiques potentielles sous-jacentes à l'émergence et au développement du gouvernement électronique auxquelles nous pouvons contribuer. Désormais, motivés donc par la volonté d'offrir un cadre technologique pour une intégration complète des services d'e-gouvernement, nous poursuivons notre étude dans ce sens. Mais avant de s'attarder sur des questions purement techniques, nous recommandons vivement le lecteur de consulter l'annexe B dans lequel nous proposons de contextualiser notre contribution en proposant un état des lieux sur les travaux et les initiatives réalisées dans ce domaine à l'échelle national et international. En effet, l'examen de l'état actuel des stratégies de gouvernement électronique en posant un regard particulier sur l'évolution du domaine au Maroc, confirme l'intérêt de traitement de la problématique de l'intégration des services électroniques dans un contexte national favorisé par une orientation stratégique visant la concrétisation de la société d'information.

La suite de ce chapitre tente donc de répondre à la question «Comment l'approche iDyCoS pourrait-elle contribuer à mettre en place une vision one stop shop ?».

6.3. L'approche iDyCoS pour l'implémentation d'un portail «one stop shop»

6.3.1. L'approche centrée citoyen

L'un des défis les plus importants auquel les services e-gouvernement offerts par les administrations publiques (AP) se heurtent, est la satisfaction des principaux clients du gouvernement : les citoyens et les entreprises.

Le problème réside dans le fait que l'organisation actuelle des instances administratives repose sur une division du travail entre plusieurs domaines ou compétences. En conséquent, les procédures et les services administratifs sont adaptés et distribués au niveau de plusieurs institutions publiques. Alors que, les demandes des clients (citoyens et entreprises) nécessitent généralement la mobilisation de plus d'une seule compétence ou d'un seul établissement publique. Par exemple si un citoyen déménage, il aura besoin de changer son adresse, de mettre à jour les documents civils, de faire des nouvelles installations d'électricité et de téléphone, d'informer l'employé et l'école etc… Selon la logique d'affaire actuelle, pour résoudre le problème dans son intégralité, le citoyen doit suivre plusieurs processus (des procédures administratives ainsi que des processus business) en présentant à des établissements publiques et privés des informations et des documents qui sont parfois les mêmes, ainsi la redondance des informations, et les nombres des institutions sollicités retardent et compliquent le processus de résolution de problème. Mais ce qui interpelle le plus dans cette situation, c'est le fait que dans ce genre de scénarios, c'est le client qui doit, lui seul, découvrir quels processus à entreprendre, dans quel ordre et chez quel établissements il doit se rendre. Ceci résulte généralement en des dossiers incomplets qui retardent le début effectif du traitement des affaires administratives, ce qui conduit à des processus compliqués, gourmand en temps, coûteux, et dont le client n'est pas satisfait.

Afin de palier à ces limitations, les programmes d'e-gouvernement doivent, à l'instar de l'approche e-business, faire la transition vers une approche centrée client. Aujourd'hui les citoyens exigent des services publics comparables à ceux offerts par d'autres secteurs d'activités, telles que les banques et les assurances. Ils demandent des services intégrés, personnalisés, rapides et flexibles. En plaçant le client au centre de la préoccupation de l'administration, le principe le plus important propre à l'e-gouvernement devient alors sans doute la prestation de service orientée client. Ce principe, emprunté au secteur privé, a donné toute sa mesure avec le World Wide Web. Selon ce principe, les services publics doivent être conçus dans le but de satisfaire les besoins des citoyens ou d'aider ces derniers à remplir leurs obligations civiques. L'organisation formelle de l'administration revêt une importance secondaire et, à la place de cela, différentes parties de l'administration, ou même des niveaux d'administration, sont rapprochés par leur relation commune avec des communautés identifiables dans la population au sens large.

On assiste alors à un changement de paradigme, d'un paradigme centré sur le gouvernement, à un paradigme centré sur les citoyens, en mettant davantage l'attention sur le contexte (ex. : social, organisationnel, et les facteurs institutionnels) dans lequel le gouvernement électronique se développe et sur les résultats pour les clients. Les changements au niveau modèle d'affaires sont décrits dans la figure ci-après [Québec, 2014] :

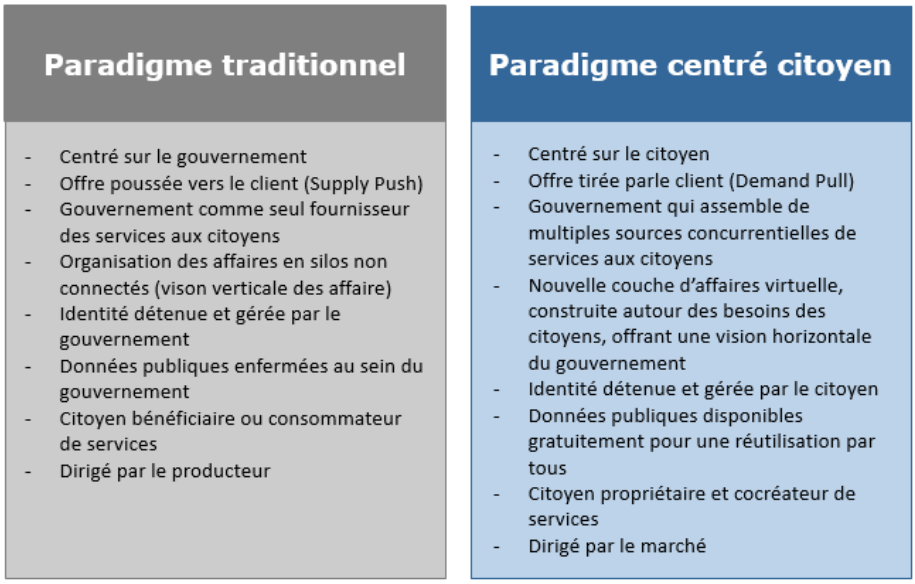


Fig.6. 4: Changement de paradigmes dans la livraison de service public [Québec, 2014]

Comme illustré dans la Fig.6.4, dans le paradigme traditionnel, l'accent est mis sur l'efficacité interne, la rationalité et les entités fonctionnelles, la hiérarchisation du pouvoir et la gestion basée sur les règles. Quant au paradigme cible, il est centré sur la valeur ajoutée, la satisfaction des besoins du client, la flexibilité dans la livraison des services, la gestion en réseaux avec des partenaires internes et externes. Ce nouveau paradigme met l'emphase sur l'innovation et la gouvernance de sorte que l'organisation puisse continuer à se réinventer.

Sur le plan technique, l'atteinte du paradigme centré client nécessite une forte coopération et intégration inter et intra organisationnelle qui permet l'échange automatique des informations

de façon transversale entre les différentes administrations, autrement dit ceci nécessite le passage d'une architecture unitaire en silos à un seul système gouvernementale intégré qui encapsule la complexité des procédures administratives et simplifie la relation avec les citoyens. Ce concept est généralement appelé le guichet virtuel de services gouvernementaux (ou encore one stop shop)

6.3.2. Guichet virtuel de services gouvernementaux (one stop shop)

Comme mentionné auparavant, la phase d'intégration est le stade ultime de développement de l'administration électronique. Cette étape qui constitue un changement de paradigme dans la livraison du service public qui est axée alors citoyen, développe l'idée d'un gouvernement complètement intégré de manière analogue à la notion de l'entreprise étendue [CGI ,2013] dans le secteur privé.

L'approche de services intégrés consiste donc à servir directement la clientèle, dès le premier contact, en une seule intervention et dans le mode qu'elle préfère en développant des partenariats entre les organisations participantes. La concrétisation de cette approche passe par une intégration totale de la prestation électronique de service de diverses organisations sur un même portail. Le citoyen peut alors bénéficier d'un point d'entrée unique pour faire une demande de service qui exige la collaboration de plusieurs organisations. Ce type de portail de service est appelé « one stop shop».

Dans la littérature anglo-saxonne, l'interface entre un état électronique et ses utilisateurs (ou clients) est le plus souvent appelé « One-Stop-Shop » (parfois aussi « Single Window »), ce qui est généralement traduit par «guichet virtuel». Dans une vision guichet unique, les administrés ne devront plus se rendre dans les différents départements et guichets d'une administration afin d'obtenir des services. C'est dans le même ordre d'idée que la plupart des gens fréquentent les supermarchés au lieu de se rendre séparément chez le boucher, le laitier ou le boulanger pour acheter de la nourriture [Lenk, 1998]. La création de valeur pour le citoyen est considérable. Le citoyen pourra bénéficier d'un service intégral et l'output se fera directement en ligne même pour des procédures complexes impliquant de nombreuses entités (plusieurs organisations, partenaires privés). Le citoyen pourra bénéficier de nouvelles fonctionnalités pour de longues procédures qui exigent du temps et des déplacements fréquents dans de nombreuses organisations Fig.6. 5. Notons au passage que l'idée d'un point d'entrée unique n'est pas nouvelle, ce qui est relativement nouveau est l'idée d'un point d'entrée virtuel, créé en utilisant les nouvelles technologies.

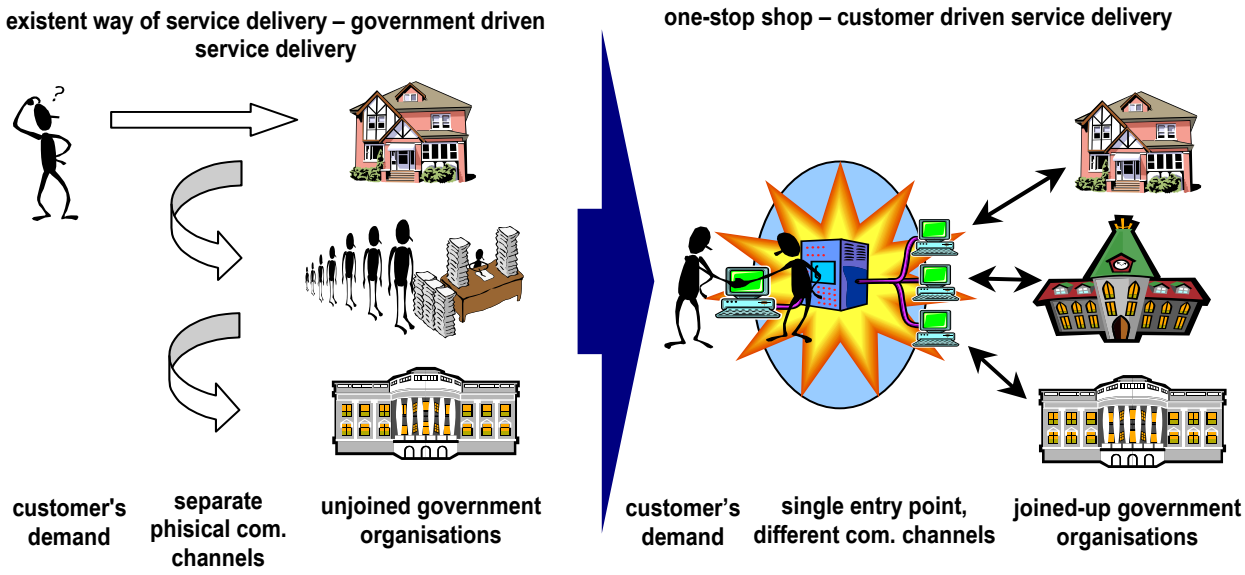


Fig.6. 5: Comparaisons entre l'approche traditionnelle et l'approche and one stop de la livraison des services gouvernementaux [Vintar et al., 2002]

Trois distinctions supplémentaires du guichet virtuel sont proposées dans [Bent et al, 1999]:

- Le « First-Stop » est un point d'information qui guide le client vers les services appropriés, en fonction de ses besoins.
- Le « Convenience Store » offre différents services transactionnels au sein d'un même espace. Il regroupe des services décentralisés et certains services plus complexes ne peuvent y être menés à bien.
- Le « True One-Stop » intègre la plupart des services nécessaires pour satisfaire les besoins de groupes d'utilisateurs spécifiques ou d'événements spécifiques.

C'est ce dernier type de guichet virtuel qui nous intéresse. Nous estimons toutefois qu'il ne s'agit pas simplement d'un regroupement des procédures existantes et d'une transposition en version électronique. L'implémentation d'un « true » guichet nécessite un grand effort de réflexion au niveau de l'approche conceptuelle qui pourrait assurer le niveau d'interopérabilité et d'intégration requises.

L'étude des travaux de recherches dans ce sens [Angelis, 2009], [CEFACT-ONU, 2005] montre que même s'il existe de nombreuses formules possibles pour créer un guichet (physique ou virtuel), deux grands modèles se dégagent [CEFACT-ONU, 2005]:

a) Une autorité unique qui reçoit des informations, soit sur papier soit par voie électronique, les communique à toutes les autorités publiques concernées et coordonne les contrôles afin d'empêcher tout blocage intempestif dans la chaîne logistique (Fig.6. 6).
b) Un système automatisé unique de collecte et de diffusion des informations (du domaine public) qui combine la collecte, l'utilisation et la diffusion (ainsi que le stockage) par voie électronique des données administratives. il s'agit, généralement, d'un système qui permet aux citoyens (entreprises) de ne soumettre qu'une seule fois des données normalisées que le système traite et redistribue aux organismes ayant à connaître de la transaction. Plusieurs possibilités existent:

i) Système intégré: Les données sont traitées par le biais du système (Fig.6. 7);
ii) Système connecté (décentralisé): Les données sont envoyées à l'administration qui les traitera (Fig.6. 8);
iii) Une combinaison de i) et ii).

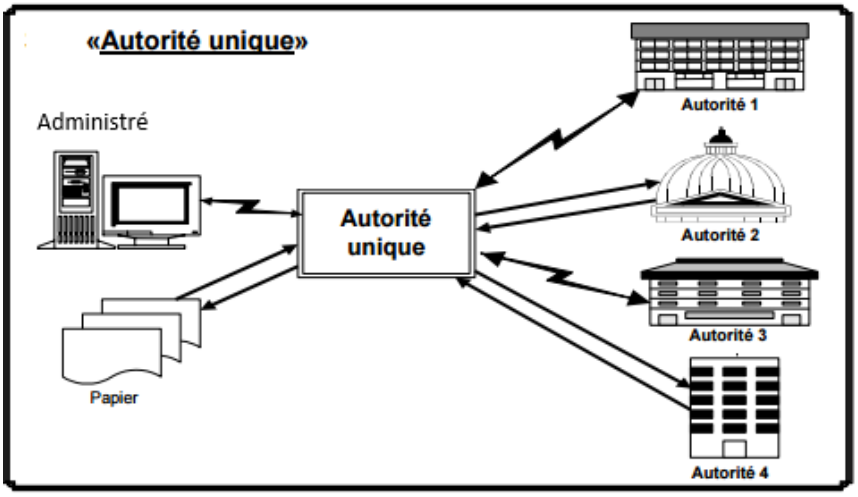


Fig.6. 6: Le modèle autorité unique du guichet unique [CEFACT-ONU, 2005]

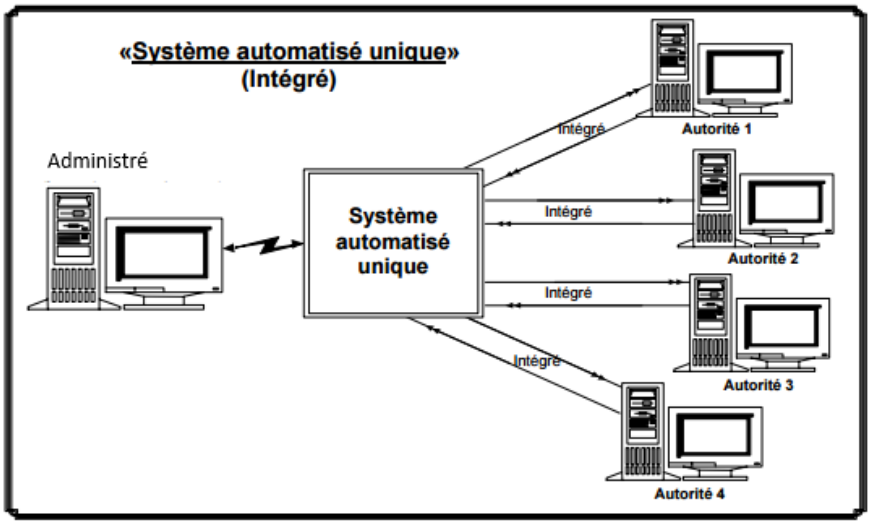


Fig.6. 7: Le modèle système automatisé intégré unique du guichet unique [CEFACT-ONU, 2005]

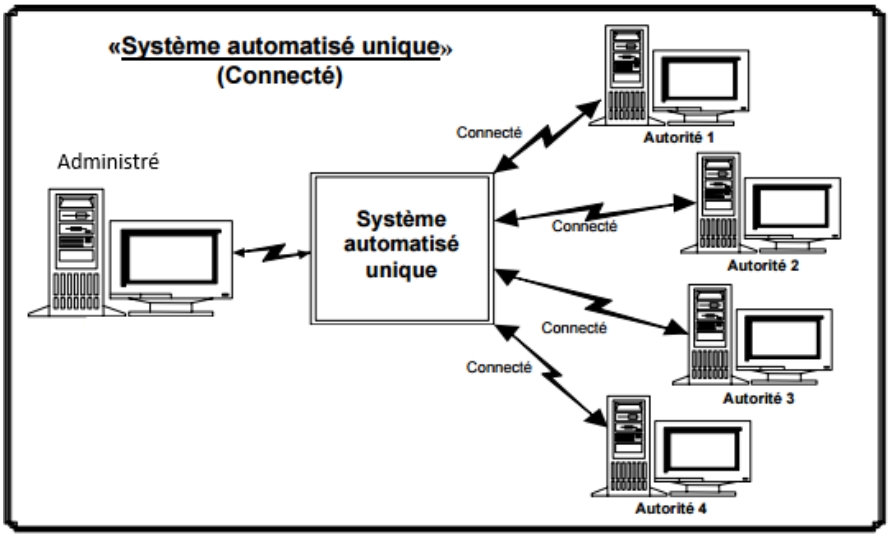


Fig.6. 8: Le modèle système automatisé connecté unique du guichet unique [CEFACT-ONU, 2005]

En se basant sur un modèle de système automatisé unique intégré (Fig.6. 7), nous proposons l'approche iDyCoS comme une approche conceptuelle et un cadre technologique adapté pour la mise en place d'un guichet virtuel des services électroniques d'administration. La section suivante éclaircie davantage notre proposition en projetant notre architecture dans le contexte d'e-gouvernement.

6.3.3. Projection de l'Architecture iDyCoS sur le domaine d'e-gouvernement

Le présent exercice, qui consiste à projeter l'architecture iDyCoS sur le domaine de l'administration publique afin de mettre en place un cadre technologique de la vision de guichet virtuel, prend tout son sens dans le constat suivant :

D'une part, étant une plateforme de composition des services Web, notre approche tâche de répondre à la demande de l'utilisateur d'une façon transparente en encapsulant la complexité de la production du service demandé, c'est donc un point d'entrée unique dont le fonctionnement est orienté usager. Notre approche vérifie bien alors les critères d'une approche conceptuelle d'un portail one stop shop.

D'autre part, la vision de guichet virtuel, de par sa nature et sa vocation, nécessite un niveau d'interopérabilité et d'intégration élevé, ce qui forme justement l'une des forces de notre approche. Un portail gouvernemental de type « True One-Stop » semble donc être le cas d'application le plus approprié pour mettre en relief les atouts de notre approche.

Dès lors, l'utilisation de l'approche iDyCoS pour l'implantation d'un guichet virtuel semble être opportune.

L'architecture iDyCoS est reprise dans la Fig.6. 10, en respectant le schéma de relations induit par la phase d'intégration, à savoir : une interaction C2G et G2C marquant respectivement le début et la fin du processus administratif et une interaction G2G illustrant le mécanisme

encapsulant le déroulement des différentes procédures administratives d'une façon transparente par rapport au citoyen (Fig.6. 9).

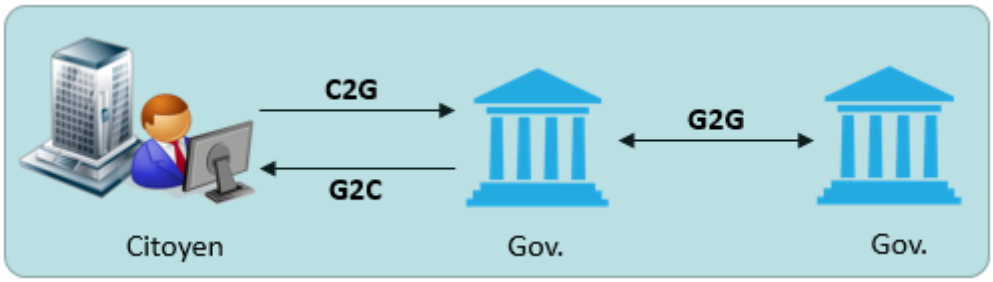


Fig.6. 9: Schéma d'interactions X2Y cible

Pour implémenter le schéma de relations ci-avant nous introduisons le concept de couches au niveau de l'architecture iDyCoS, comme décrit au niveau de la Fig.6. 13, la plateforme est présentée dans une architecture en quatre couches [Adadi et al., 2015]:

La couche « **Legacy system** » : contient les applications existantes disponibles dans chacune des administrations publiques (AP) impliquées dans le système.

La couche **« Ontologie » :** l'ontologie est le concept clé utilisé pour résoudre le problème d'hétérogénéité fortement présent dans un contexte e-gouvernement. Nous reprenons donc les quatre ontologies de l'architecture iDyCoS dans une couche dédiée en dessus de la couche « Legacy system ». Il est à noter, qu'en pratique, la projection de l'approche sur le domaine de l'e-gouvernement nécessite la conception et l'implémentation d'une ontologie du périmètre gouvernemental couvert ainsi que l'intégration du niveau métier adapté dans l'ontologie de QoS. Une proposition dans ce sens est présentée dans le chapitre suivant.

La couche **« Composition » :** est le noyau du modèle basé sur l'approche iDyCoS que nous proposons dans ce chapitre. Elle permet d'offrir les fonctionnalités nécessaires pour l'arrière-guichet virtuel, comme la découverte, la composition et l'exécution des services administratifs en se basant sur les techniques présentées précédemment dans cette thèse.

La couche « **Communication** »: il s'agit d'un portail gouvernemental de type Gouvernement à Citoyen (G2C), c'est l'interface de communication de l'architecture iDyCoS qui joue le rôle de ce point d'entrée unique où le citoyen peut demander et recevoir des services e-gouvernement personnalisés. Rappelons qu'il s'agit d'une interface sémantique qui vise à rendre l'interaction entre le gouvernement et ses citoyens plus intelligente, plus conviviale et plus transparente.

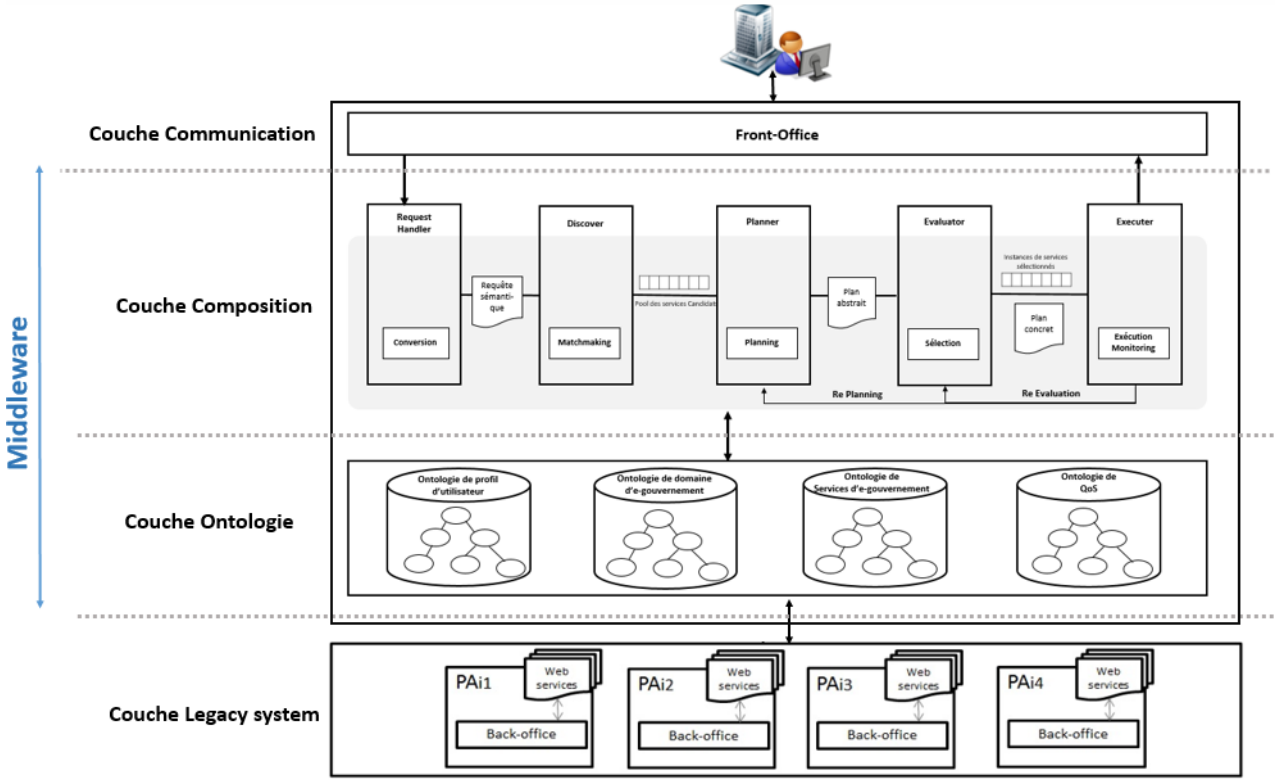


Fig.6. 10: L'architecture iDyCoS projetée sur le domaine d'e-gouvernement

L'architecture en couches du système établi un pont entre le besoin du client de l'administration (citoyen et entreprise) et l'offre des AP, représentés respectivement par le front office et la couche Legacy system. Ce pont est confronté à deux défis majeurs : **l'interopérabilité et l'intégration.**

Dans un contexte e-gouvernement, (a) l'interopérabilité signifie la capacité des organisations gouvernementales à partager des informations et d'intégrer les processus métiers par l'utilisation des standards et des normes de travail [Scholl et al., 2007]. Alors que (b) l'intégration est la formation d'une plus grande unité d'entités gouvernementales, temporaires ou permanentes, dans le but de la fusion des processus et / ou le partage d'informations [UK, 2005].

Si les différentes ontologies utilisées constituent le nerf de la guerre pour faire face au premier défi, il convient de noter que les deux couches Communication et Composition assure un double niveau d'intégration :

(i) Une intégration front-office (l'administration visible par l'usager): assurée par une interface de communication sémantique qui implémente le paradigme centré utilisateur pre-étudié et s'impose comme l'unique point d'accès aux services d'administration en intégrant et en fusionnant tout autre canal d'accès à ces services.

(ii) Une intégration back-office (fonctionnement interne de l'administration) : l'architecture orientée services (SOA) conjuguée aux différentes techniques de composition proposées, permettent une intégration verticale et horizontale entre les applications d'administration électronique existantes (couche Legacy systems). Ces applications sont capables de coopérer, malgré leur hétérogénéité, grâce au cadre conceptuel proposé (architecture en

couches) qui offre les mécanismes nécessaires pour coordonner les processus d'e-gouvernement afin d'offrir des services intégrés, standardisés et transparents.

C'est ainsi que le middleware, la partie formée par les couches Ontologie et Composition, est essentiellement perçue comme une instance de back office chargée d'orienter le citoyen à partir de son problème au(x) service (s) approprié(s), en encapsulant la complexité des établissements publics et en assurant une intégration rationnelle des processus de l'administration via une interopérabilité entre les systèmes informatiques déployés.

Techniquement, en se basant sur une architecture en couches composée essentiellement d'agents intelligents et d'ontologies et, à partir d'une demande informelle, et éventuellement incomplète, les agents intelligents explorent l'annuaire des services d'e-gouvernement où les services des AP sont publiés sémantiquement, pour trouver la bonne combinaison des services qui satisfait les exigences du citoyen, l'utilisation du modèle de planification proposé rend ce processus automatique et dynamique. L'exécution autonomique assure l'exécution du service ainsi formé dans un enivrement hétérogène et dynamique.

En conséquence de quoi, le système fournit des services à valeur ajoutée plus commodes et plus adaptés aux besoins et aux attentes du citoyen, il permet également d'accroître l'efficacité en termes de coûts et de la productivité en réutilisant les services d'AP tout en gardant leurs processus internes et les systèmes existants intacts.

6.4. Conclusion

Le cas d'utilisation présenté dans ce chapitre porte sur une facette particulière du gouvernement électronique: le guichet virtuel, un point d'accès unique mettant en ligne des services gouvernementaux intégrés. Cette vision permet de transformer le fonctionnement des administrations publiques et leur ouverture envers les usagers de leurs services.

De par ses caractéristiques intrinsèques, nous avons vu que ce cas d'application constitue un support idéal à la présentation des contributions scientifiques identifiées dans ce manuscrit, à savoir : Une composition complète et intégrée (cf. chapitre 3), sémantique et intelligente (cf. chapitre 4), et dynamique et agile (cf. chapitre 5) des services Web.

Une étude de cas liée à ce domaine sera détaillée dans le chapitre suivant dont l'objectif est de mettre en place une preuve de concept du modèle de composition proposé.

Chapitre 7 :

Etude de cas et implémentation

A theory is something nobody believes, except the person who made it.
An experiment is something everybody believes, except the person who made it.
Albert Einstein

7.1. Introduction

Dans le cadre d'une preuve de concept, nous avons étudié dans un premier temps le potentiel de l'approche proposée à contribuer à donner des réponses efficaces à des problématiques réelles, le domaine étudié étant l'e-gouvernement et la problématique traitée étant l'intégration des services d'administration publique dans un portail one stop shop.

Dans le présent chapitre nous poursuivons la preuve du concept mais cette fois sur un plan technique. Nous proposons un prototype à l'échelle pilote pour la démonstration des concepts clés constituant notre approche, l'aspect modulaire de l'architecture iDyCoS permet une construction incrémentale et progressive de ce prototype, ainsi nous présentons dans le cadre de ce travail les résultats des premiers livrables de ce prototype. Il s'agit d'un PoC (Proof of concept) de (a) l'ontologie de domaine et de (b) la troisième technique de composition en l'occurrence le modèle de planification.

La preuve de concept ainsi construite aidera à instrumenter l'évaluation de l'approche discutée à la fin de ce manuscrit.

7.2. Présentation du prototype

Nous proposons dans le cadre du présent scénario d'implémentation de mettre en place un moteur de composition reposant sur le modèle de planification présenté antérieurement. Le moteur gère des services décrits sémantiquement en manipulant les concepts décrits dans une preuve de concept de l'ontologie du domaine étudié.

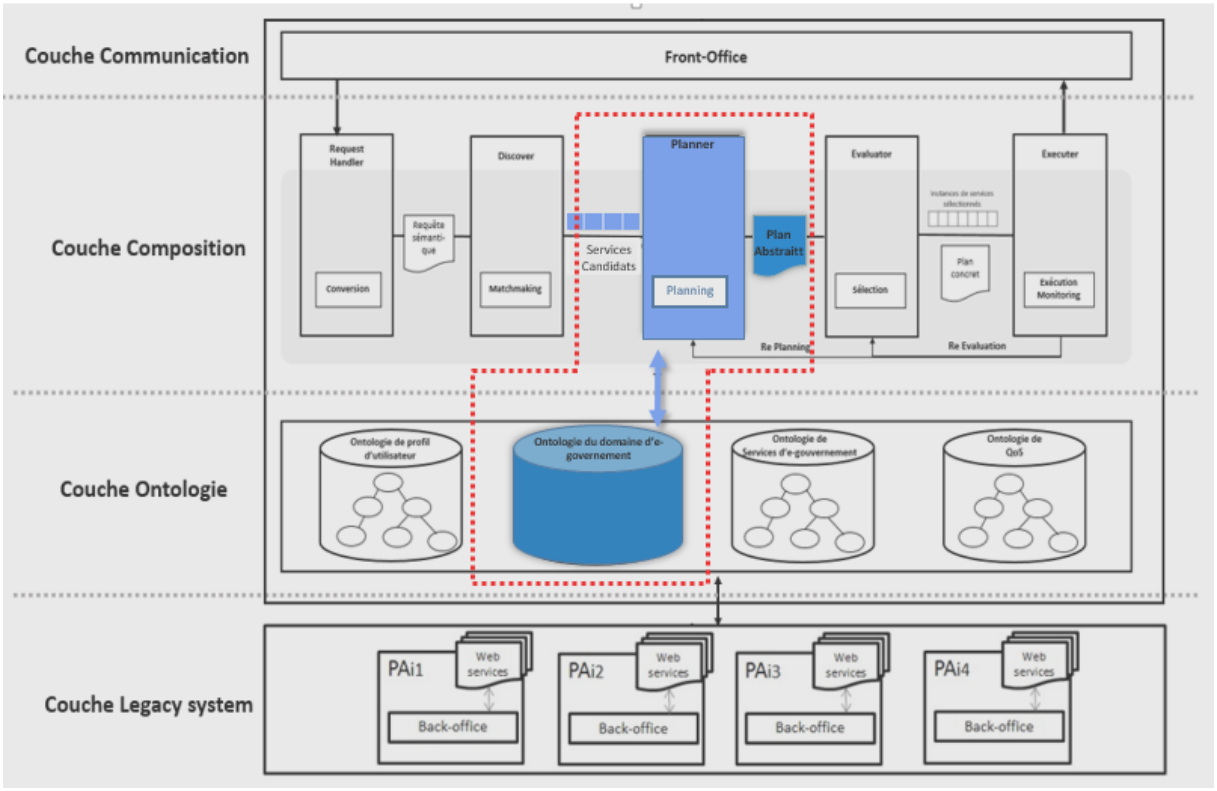


Fig.7. 1: Périmètre de la preuve de concept PoC

Comme illustré dans la Fig.7. 1, le présent PoC s'adresse au sous problème de planification, la dynamique que nous souhaitons mettre en place à cette étape est celle permettant la construction du plan abstrait de composition en synthétisant les interactions entre les différents agents managers simulant les services Web sémantiques découverts et ce en se basant sur l'ontologie de domaine. Le choix de ce périmètre est motivé par l'importance et la signification des résultats du mécanisme simulé. En effet, étant donné que l'approche iDyCoS privilégie une composition à deux niveaux, dont le fonctionnement du deuxième niveau (composition concrète) dépend des résultats produits au premier niveau (composition abstraite), le test des techniques de composition abstraite est à considérer en premier temps. Parmi les trois techniques de la composition abstraite, celle relative au modèle de planification peut être implémenté en « Standalone ». Le but des deux premières techniques étant préparatif (la technique de spécification prépare la requête sémantique en exerçant une fonction de traduction, et la technique de découverte prépare les services candidats en effectuant une fonction de filtrage), leurs résultats ne peuvent avoir de signification que dans le cadre de la troisième technique : Modèle de planification qui, elle par contre, produit un résultat significatif : un plan abstrait de composition.

Nous visons ainsi à travers ce prototype de mettre en pratique les différents aspects conçus dans le cadre de la phase de planification et d'identifier les contraintes techniques liées à l'implémentation pour un éventuel rehaussement de l'architecture proposée, les enjeux du PoC sont alors:

- ➔ Prouver la faisabilité du modèle de planification;
- ➔ Etudier et mettre en pratique les outils et les méthodes de conception et d'implémentation des ontologies;
- ➔ Gérer et manipuler les services Web sémantiques ;
- ➔ Mettre en œuvre les mécanismes assurant la cohérence et l'interopérabilité entre les domaines de H4F, dans ce cas il s'agit d'implémenter les adaptateurs nécessaires pour inter opérer le SMA/SWS (et l'ontologie)/et l'algorithme de planification.

Notons que ce cadre de travail est basé sur les deux hypothèses simplificatrices suivantes :

1. Nous rappelons l'hypothèse formulée au début du chapitre 4 qui suppose que les services intervenant dans le processus de composition sont décrits en langage OWL-S, qui constitue l'initiative la plus réfléchie et raisonnable pour la mise en œuvre des services sémantiques.
2. Nous nous positionnons dans la phase de composition abstraite, exactement dans la deuxième sous étape de la phase de « planification » à savoir la construction du plan, et par conséquent nous supposons que les services Web sémantiques impliqués dans le cycle de composition en cours sont déjà découverts.

Etant donné le périmètre, les objectifs et les hypothèses décrits ci-avant, la démarche employée au cours de la réalisation du PoC s'articule autour de quatre étapes principales:

a) Explorer une étude de cas détaillée et illustrative du domaine étudié dans le chapitre 6 (e-gouvernement), cette étude de cas est nécessaire pour l'évaluation et la vérification des différents composants développés en s'appuyant sur un cas réel.
b) Création d'une ontologie de domaine, qu'on nommera *WebGov*, en respectant les objectifs de ce type d'ontologies tels qu'ils sont décrits dans le chapitre 3.
c) Elaboration de la description sémantique des services découverts pour le cas étudié et ce en utilisant le modèle sémantique OWL-S détaillé dans le chapitre 2 ;
d) Implémentation d'un moteur de planification, auquel on attribuera le nom *SWSComposer*, en se basant sur la théorie du modèle de planification introduite dans le chapitre 4.

- ➔ En amont et tout au long de cette démarche, différentes technologies vont être étudiées en vue d'adopter celles les plus adéquates à notre besoin.

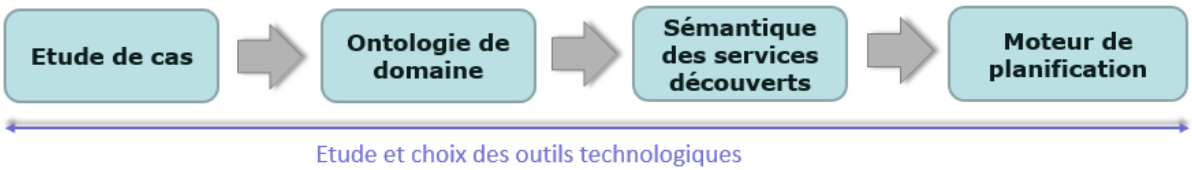


Fig.7. 2: Démarche de la réalisation du PoC

7.3. Etude de cas : Renouvellement de la carte nationale d'identité

Par l'intermédiaire d'un exemple démonstratif (planification d'un voyage) que nous avons fait évoluer au fur et à mesure de l'introduction des différentes notions abordées dans les chapitres précédents, nous avons pu illustrer les idées défendues dans ce manuscrit. En suivant la même démarche, nous explorons dans ce chapitre un exemple plus conséquent, une étude de cas appartenant au domaine d'e-gouvernement. Cette étude servira, dans un premier temps, à positionner les problèmes qui surgissent assez fréquemment dans le domaine de l'administration publique. Il sera repris tout au long de ce chapitre afin de vérifier et discuter les résultats des mécanismes mis en place dans le cadre du PoC. L'exemple choisi est celui relatif au renouvellement de la Carte Nationale d'Identité.

7.3.1. Description du problème

La demande d'obtention ou de renouvèlement des papiers administratifs est une opération fréquente chez les citoyens marocains. Malheureusement ce type d'opération prend beaucoup de temps et exige un effort considérable. Pour souligner ceci, nous décrivons un cas simplifié mais réaliste développé dans le cadre de notre prototype, qui porte sur un scénario typique où un citoyen demande le renouvellement de la Carte Nationale d'Identité (CNI).

La carte nationale d'identité est un document officiel qui permet à tout citoyen de justifier son identité et sa nationalité marocaine. Elle est obligatoire à l'âge de 18 ans et d'une durée de validité de 10 ans [cnie, 2016].
Cette pièce d'identité doit être changée/renouvelée dans les cas suivants :

- La modification du prénom, du nom ou de la date de naissance;
- La rectification du lieu de naissance ou du numéro de l'acte de l'état civile ;
- Le changement de domicile;
- La perte, le vol, ou la détérioration de la CNI;
- L'expiration de la durée de validité.

Dans ce dernier cas, une nouvelle carte est délivrée à l'intéressé après restitution de la carte nationale d'identité en sa possession et production d'un ensemble d'autres pièces qui sollicitent à leur tour d'autres services administratifs [service-public, 2016].

Considérons donc le scénario suivant illustrant la procédure de renouvellement de la CNI dans sa version traditionnelle (le descriptif se base sur les données récoltées lors de notre enquête sur terrain dans les différentes administrations impliquées dans ce processus, ainsi que sur les renseignements postés sur [service-public, 2016]) :

« Une enseignante habitant Fès âgée de 36 ans nommée Mariam, a besoin de renouveler sa CNI suite à son expiration. Pour ce faire, Mariam doit d'abord se rendre à l'arrondissement de police le plus proche pour avoir un certificat de résidence (CR), ceci nécessite que Mariam ait au préalable sont attestation de travail (AT) du lycée où elle travail.
Puisque Mariam est née à Tahanaout, une ville à côté de Marrakech, elle a besoin de voyager 550km pour avoir trois extraits d'acte de naissance (AN) du bureau d'état civil de la commune de Tahanaout.
Un extrait de l'AN doit être déposé avec l'AT en plus d'une preuve de résidence (facture d'eau ou d'électricité) à l'arrondissement de police pour avoir le CR.
Quant aux deux extraits de AN restants, ils doivent être joints au CR délivré en plus d'autres papiers (photographies, les empreintes biométriques, les frais ...) pour former le dossier requis pour le renouvellement de la CNI à soumettre à la préfecture de police de Fès. »

Ce processus est schématisé sur la figure ci-après :

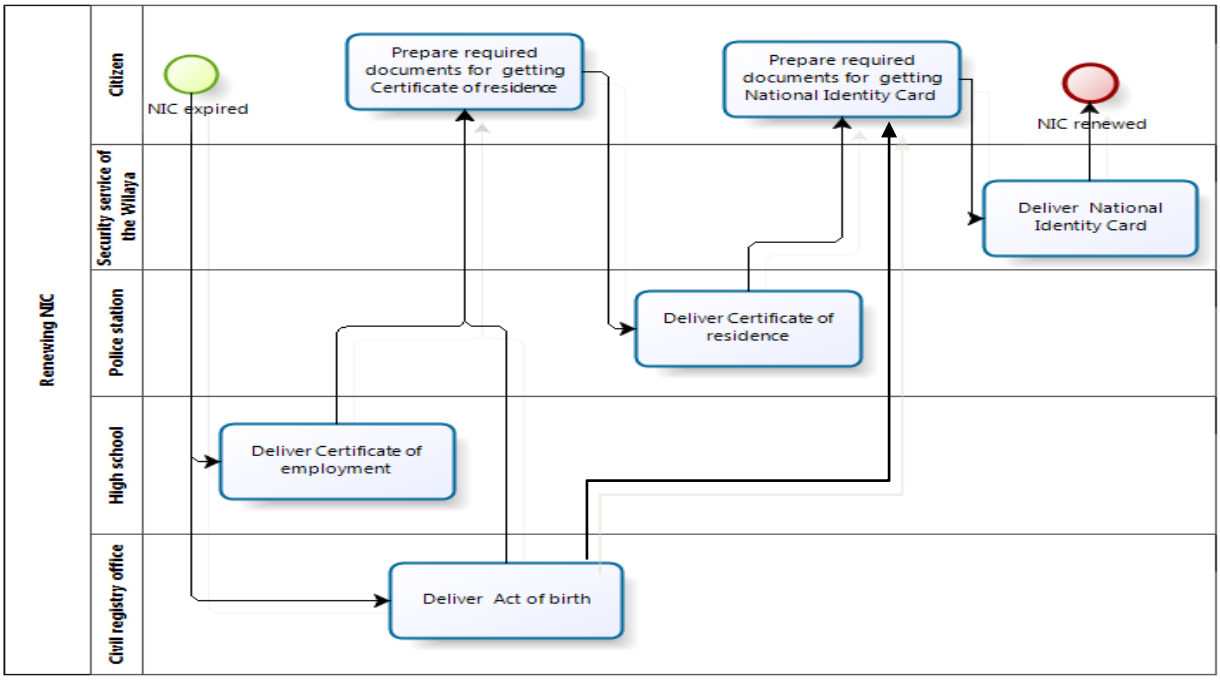


Fig.7. 3: Processus de renouvellement de la CNI [*Adadi et al., 2015b*]

7.3.2. Discussion du problème

Comme le montre la Fig.7. 3, le renouvellement de la CNI est une procédure longue et ardue dans le sens que le citoyen doit se rendre à plusieurs établissement situés au sein et à l'extérieur de sa localité.

En fait, la complexité de la procédure est accentuée par le manque d'échange de données entre les différents intervenants dans le processus en question.

La contribution fondamentale du système proposé serait donc d'offrir une livraison des services sans hiatus aux utilisateurs, en assurant une véritable coopération inter-organisationnelle permettant d'améliorer le partage d'information, tout en réduisant la nécessité que les utilisateurs fournissent des informations redondantes déjà détenues par l'administration.

Selon le modèle proposé, Mariam va pouvoir obtenir sa nouvelle CNI sans avoir à se présenter en personne au niveau des quatre administrations publiques impliquées dans ladite procédure, mieux encore elle n'a même pas besoin de connaitre les étapes de traitement administratif inter-organisationnel de la demande. Tout ce qu'elle a à faire, c'est de demander le service « Renouveler ma CNI » une fois connecté au système, et ce dernier va prendre en charge la livraison du service souhaité en invoquant les sous services nécessaires dans le respect du plan construit. Le processus de renouvellement de la CNI cible est schématisé dans la Fig.7. 4

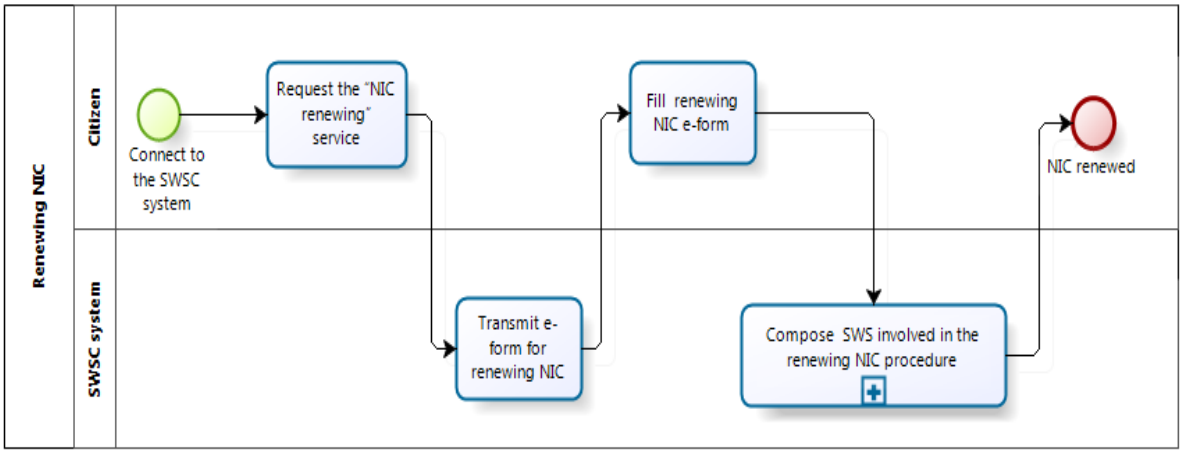


Fig.7. 4: Processus du renouvellement de CNI en utilisant le modèle proposé [Adadi et al., 2015b]

La contribution du système dans le processus cible est encapsulée dans le sous processus « *Compose SWS involved in the renewing NIC procedure* » illustré dans la figure précédente qui englobe, en effet, la mécanique de composition du service demandé.

Cette mécanique est dévoilée ci-après pour le cas étudié en mettant en exergue les éléments produits dans chacune des phases de composition :

Phase	**Produit**	**Valeurs pour le cas de « Renouvellement de CNI »**
Analyse	Requête sémantique	Entrées : Citoyen_Info; AN, CR, AT Sorties**:** CNI Précondition**:** Etre Eligible Effets: Avoir CNI
Découverte	Liste des services abstraits candidats pour planification	- Service constructeur de la CNI - Service de livraison du certificat de résidence - Service de livraison de l'acte de naissance - Service de livraison d'attestation de travail
Planification	Ordre de l'appel des services abstraits	1.1 Service de livraison attestation de travail 1.2 Service de livraison de l'acte de naissance 2. Service de livraison du certificat de résidence 3. Service de livraison de l'acte de naissance

		4. Service constructeur de la CNI
Evaluation	Liste des services élémentaires concrets	- Service constructeur de CNI Wilaya de sécurité centrale Fès - Service de livraison de certificats de résidence de l'arrondissement de police Azzouhour Fès. - Service de livraison d'actes de naissance de la commune de Tahanaout - Service de livraison d'attestations de travail de la direction provinciale de l'éducation nationale Fès
Exécution	Exécution du Workflow produit	

Tableau 7. 1: Eléments produits pour chaque phase du processus de composition pour le cas étudié

Il est à noter que ce scénario de composition est volontairement simplifié tout en restant inspiré du cas de la vie réelle. L'objectif est, en effet, d'illustrer le mécanisme de composition de notre modèle à travers un exemple simple mais réaliste. Dans ce qui va suivre nous allons présenter notre prototype appliqué à cet exemple, référencé ultérieurement par EC_CNI, en suivant la démarche convenue.

7.4. Conception et mise en place de l'ontologie de domaine

Nom de livrable	WebGov	
Version	1.0	
Description	Ontologie de domaine de l'administration publique définissant les concepts du domaine étudié et leurs relations	
Objectif	Avoir un consensus entre les différents intervenants concernant les connaissances du domaine étudié	
Inclusion		**Exclusion**
- Collecte de données ; - Organisation des données non structurées ; - Formalisation du modèle conçu.		- Réutilisation d'une ontologie existante
Méthodes	METHONTOLOGY	
Outils	Langage OWL, Protégé	

Fig.7. 5: Fiche descriptive du livrable.1

En se référant à la définition introduite dans le chapitre 2, une ontologie de domaine décrit le vocabulaire ayant trait à un domaine donné (ex. : l'enseignement, la médecine, la géologie…). Par l'utilisation de l'ontologie de domaine, nous visons à intégrer une couche de connaissances issues de sources hétérogènes. L'ontologie en sens générale et celle du domaine en particulier, forment ainsi l'instrument incontournable pour résoudre le problème de l'interopérabilité du domaine étudié –l'e-gouvernement-. Comme l'ontologie de domaine est censée modéliser un consensus entre les agents sur les concepts qui sous-tendent les données partagées, sa mise en place s'avère essentielle pour que le prototype puisse atteindre ses objectifs.

La conception d'ontologies est une tâche complexe qui nécessite la mise en place de procédés élaborés afin d'extraire la connaissance d'un domaine, manipulable par les machines et interprétable par les êtres humains. Face à une telle mission, deux approches sont envisageables : concevoir l'ontologie « from scratch » ou adapté une ontologie existante.

Même s'il est généralement admis que le formalisme d'une nouvelle ontologie par la réutilisation d'ontologies existantes est « plus avantageux » en termes de temps, de coût et d'effort, ce n'est pas toujours la bonne route à prendre. En effet, la plupart des ontologies existantes [oegov, 2016], les open sources en particulier, sont difficiles à manipuler et à customiser. L'avantage de la réingénierie d'une ontologie devient ainsi incertain, puisque l'investissement nécessaire pour comprendre les ontologies existantes pourrait être encore plus important que de construire une ontologie à partir de zéro.

Etant donné ceci, puisque nous travaillons sur une preuve de concept, nous avons choisi de concevoir et de développer une ontologie d'e-gouvernement partielle avec un niveau de granularité adapté à notre étude de cas.

Le travail présenté dans cette section concerne donc le développement d'une ontologie du domaine e-gouvernement qu'on nommera WebGov. La première partie de cette section s'interroge sur la manière d'outiller la construction de notre ontologie. Alors que la deuxième partie est consacrée au travail de construction de l'ontologie WebGov.

7.4.1. Méthode et outils de développement de l'ontologie

Une ontologie est toujours liée à une méthodologie de construction, à un outil de construction et à un langage de représentation d'ontologie.

7.4.1.1. Méthode de construction

Depuis le début des années 1990 et jusqu'à présent, l'ingénierie ontologique a été et demeure un champs de recherche actif [Gomez-Perez et al., 1999] , [Staab et al., 2001]. Cependant, il n'existe pas encore de consensus en matière de normes de construction. Et par conséquent, il n'existe pas encore de méthodes universellement reconnues pour la construction d'ontologies [Uschold, 1996]. Ceci relève beaucoup plus du savoir-faire que de l'ingénierie. En dépit de cela, de nombreux critères et principes permettant de guider la construction d'ontologies ont été proposés.
En effet, à partir de l'étude des différentes méthodologies de développement d'ontologie [Wache et al., 2001], deux principaux cadres méthodologiques sont identifiés. D'abord ceux basés sur l'expérience, comme la méthode proposée par [Grüninger et al., 1995] suite au projet TOVE ou encore celle exposée par [Uschold et al.,1995] fondée sur le modèle entreprise. Le deuxième groupe de méthodologies propose des modèles de prototype évolutif, comme METHONTOLOGY [Fernández et al., 1997] qui propose un ensemble d'activités pour développer des ontologies basées sur leur cycle de vie et sur le raffinement du prototype, ou la méthode 101 Method [Noy et al., 2001] qui propose une approche itérative pour construire une ontologie.

Parmi ces méthodes, seule METHONTOLOGY est reconnue être la méthode la plus mature et la plus complète [Wache et al., 2001], c'est cette méthode que nous utilisons dans le cadre de ce prototype.
Notons qu'à ce niveau, nous présentons une lecture des principales méthodes utilisées principalement pour construire une ontologie à partir de zéro, les méthodes de construction des ontologies par la réutilisation d'autres ontologies ne font pas partie de la portée de cette discussion.
METHONTOLOGY, la méthode sélectionnée pour construire l'ontologie WebGov, a été proposée par Fernandz en 1997 au sein d'un laboratoire d'intelligence artificielle de l'université polytechnique à Madrid, cette méthode permet de construire des ontologies au niveau des connaissances à l'aide d'un modèle conceptuel intermédiaire, sans avoir besoin d'une connaissance a priori de concepts. La construction de l'ontologie est réalisée selon un cycle constitué de sept étapes: (1) la spécification (but et utilisateurs visés de l'ontologie), (2) l'acquisition des connaissances (récolte des donnés), (3) la conceptualisation (structuration du domaine au niveau des connaissances), (4) l'intégration (intégration des sous ontologies) (5) l'implémentation (expression du modèle formel à l'aide d'un langage d'implémentation), (6) l'évaluation (validation et vérification) et (7) la documentation. Notons que les sept étapes de construction respectent le standard IEEE relatif au cycle de vie de conception et mise en place d'un système informatique [IEEE, 1996].

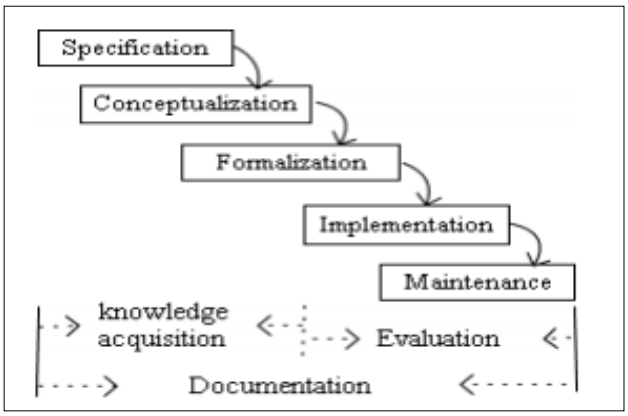


Fig.7. 6: IEEE 1074-2006 Cycle de vie de développement d'un système informatique [IEEE, 1996]

En plus de la méthodologie de construction, un langage de développement ontologique et un outil de développement doivent également être décidés.

7.4.1.2. Langage de spécification

Nous adoptons OWL [W3C, 2003] présenté dans le chapitre 2 comme langage de spécification ontologique. Ce langage caractérisé par sa sémantique formelle, a été choisi pour sa qualité de calcul de cohérence dans le contrôle et la classification qui est un critère crucial pour le développement correct et cohérent d'un modèle ontologique de domaines complexes.

7.4.1.3. Environnement de développement

Malgré l'existence d'une plateforme logicielle, ODE [Blázquez et al., 1998], développée spécialement pour guider l'utilisation de METHONTOLOGY. Nous proposons de travailler avec protégé [Protégé, 2016] qui supporte lui aussi METHONTOLOGY et OWL mais avec plus d'intuitivité et plus d'ouverture technologique.

Protégé est un environnement graphique de développement d'ontologie le plus utilisé pour construire les ontologies avec multiples langages. C'est un logiciel libre d'utilisation qui supporte le modèle de frames contenant des classes, des slots (attributs) et des facettes (valeurs des propriétés et contraintes) ainsi que des instances des classes et des propriétés pour permettre le contrôle et la visualisation d'ontologies. Il regroupe une communauté d'utilisateurs assez importante et constitue une référence pour beaucoup d'autres outils.

7.4.2. Construction de l'ontologie de domaine

Construire une ontologie du domaine e-gouvernement requiert un effort humain substantiel et une collaboration étroite entre les spécialistes maîtrisant les langages de description d'ontologie et les experts du domaine. En s'appuyant sur la boite à outils des technologies présentées ci-avant, l'ontologie WebGov a été construite conformément à la méthode METHONTOLOGY. Cette ontologie couvre, à terme, le domaine d'e-gouvernement, mais dans une première étape, expérimentale, il a été décidé de se focaliser à un corpus de connaissances portant uniquement sur l'administration publique et ses services. En outre, il est vite apparu nécessaire pour exploiter ce corpus de recourir à des interviews d'experts du domaine, afin, d'une part, de garantir une restitution fidèle du modèle réel et, d'autre part, de tenir compte de la façon dont les experts conçoivent et manipulent les connaissances du domaine dans leurs activités. Les entretiens des experts ont été dirigées par la modélisation car elles sont intervenues alors que la construction de l'ontologie avait déjà débutée à partir des données et des documents disponibles. Elles ont essentiellement servi à valider et clarifier la structuration des connaissances métier. Dans ce sens les entretiens avec les experts ont conduit à optimiser la structuration proposée pour l'ontologie et à introduire des nouveaux concepts permettant de factoriser des concepts issus des termes candidats extraits des données recensées.

De ce fait, la validation de l'ontologie a été effectuée à trois niveaux : la validation des connaissances dites métiers constituant le niveau informel de l'ontologie, a été basée sur l'avis des experts du domaine. La validation des aspects logiques et structuraux de l'ontologie comme les problèmes d'incohérence logique, d'incomplétude ou de redondance, a été traitée automatiquement par les outils de développement utilisés. Quant à la validation du contenu de l'ontologie au niveau formel, c'est-à-dire son adéquation avec la partie du monde réel qu'il représente, elle a été décidée par les concepteurs.

Nous présentons dans ce qui suit les étapes de la réalisation de WebGov qui s'inspirent et respectent les phases de la méthodologie METHONTOLOGY. Nous mettons particulièrement en exergue les trois stations cruciales qui marquent le passage de l'informel vers le formel.

7.4.2.1. Spécification

Le but de l'étape de spécification est d'identifier l'objectif primaire de l'ontologie, pour quelle finalité elle sera construite et quelle est la portée de la connaissance couverte. La Fig.7.7 montre un extrait du document de spécification de l'ontologie WebGov.

Ontologie WebGov
Domaine
Administration publique
But
La finalité de la construction de l'ontologie WebGoV est d'établir un modèle de connaissance consensuel autour des services administratifs électroniques
Niveau de formalité
Formel
Portée
L'ontologie doit se concentrer sur le monde de l'administration publique. le niveau de granularité est directement lié aux termes identifiés
Source de connaissances
a) Entretiens avec les experts du domaine ; b) Informations administratives et services de référence ; c) Les sites Web suivants: http://www.service-public.ma/ http://www.egov.ma/ www.maroc.ma/fr/applications-mobiles/idarati www.maroc.ma http://www.cnie.ma/

Fig.7. 7: Document de spécifications de l'ontologie

7.4.2.2. Conceptualisation

La phase de conceptualisation permet d'articuler le niveau du discours et le niveau ontologique et de penser le passage de l'un à l'autre. L'objectif de cette phase est donc d'organiser et de structurer les connaissances acquises en utilisant une représentation externe indépendante du langage et de l'environnement d'implémentation. Il s'agit plus précisément, de convertir la connaissance acquise d'un point de perception informel du domaine vers une spécification semi-formelle, en utilisant une représentation intermédiaire qui peut être comprise par les développeurs de l'ontologie ainsi que par les experts du domaine.

Ainsi afin d'établir cette modélisation semi-formelle les tâches ci-après ont été effectuées dans l'ordre suivant :

1. Création d'un **glossaire des termes** de domaine pour déterminer les termes faisant partie du domaine, le glossaire est enrichi par une description et des synonymes pour chaque terme (Tableau 7. 2);
2. Identifier **les relations binaires entre** les différentes classes de l'ontologie (Tableau 7. 3);
3. Construire **un arbre de classification** des concepts pour spécifier les sous-classes et les classes supports (Fig.7. 8).

Terme	Synonymes (acronymes)	Description	Rôle
Evènement de vie	Situation de vie, évènement d'affaire,	Une expérience de vie qui implique un service(s) administratif (s)	Concept
Règlement	Loi	Un ensemble de règles qui organisent une procédure administrative	Concept
Citoyen	Person, Enterprise	Une personne qui appartient légalement à un pays et qui possède des droits et des responsabilités envers ce pays	Concept
Administration publique	Etablissement publique, institution publique, agence publique, instance publique	Un organisme public qui délivre un service public aux citoyens	Concept
Carte nationale d'identité	CNI	Un document administratif qui sert de preuve d'identité	Concept
Passe par	Fait face à	Passer par un événement de la vie	Relation
Gouverné par	Structuré par, organisé par	Organiser et contrôler une procédure administrative	Relation
Délivré par	Produit par, fournit par	Produire un objet administratif	Relation

Tableau 7. 2: Un fragment du glossaire de termes élaboré

Nom de la relation	Délivrer	Sert à	Produire	Passe par
Concept source	Administration publique	Service administratif	Service administratif	Citoyen
Cardinalité source	(0,n)	(0,n)	(0,n)	(0,n)
Concept cible	Service administratif	Evènement de vie	Document administratif	Evènement de vie
Cardinalité cible	(0,n)	(0,n)	(0,n)	(0,n)
Relation inverse	Délivré par	Besoin d'un	Lié à	Arrive à

Tableau 7. 3: Un extrait du tableau de relations binaires de l'ontologie WebGov

Situation professionnelle
- **Actif**
  - **Salarié**
    - **Fonctionnaire**
    - **Employé**
  - **Travailleur Independant**
- **Eleve_etudiant**
- **Retraité**
- **Sans**

Fig.7. 8: Un exemple d'un arbre de classification des concepts de l'ontologie WebGov

7.4.2.3. Formalisation et implémentation

Finalement, en se basant sur les résultats de la phase de conceptualisation, l'ontologie a été formalisée avec l'outil Protégé en transformant le modèle conceptuel en un modèle implémenté.

La structure ontologique créée, se caractérise par sa nature top-down, dans le sens où l'assimilation du domaine est axée sur la compréhension globale de la structure des services administratifs en se basant sur des concepts clés qui entrainent le développement de concepts supports. Plus spécifiquement, le concept « évènement de vie » est utilisé comme un concept structurant. Par évènement de vie on entend une situation particulière dans la vie d'un citoyen, comme le mariage, l'accouchement, la construction de maisons, etc. Pour aider les citoyens à gérer ce type d'évènements, les « administrations publiques », un autre concept structurant, offrent des « services ». L'exécution des services administratifs suit un processus gouverné par des lois -un « règlement»-, qui produit et/ou requiert un ensemble de « documents administratifs ». Les concepts structurants soulignés ci-avant forment les éléments de la structure de haut niveau de l'ontologie WebGov illustrée dans la Fig.7. 9. La traduction en langage de spécification ontologique de cette structure est générée automatiquement sous forme de fichier OWL, un aperçu de ce fichier, qui sera utilisé par les autres composants du PoC, est donné dans la Fig.7. 10.

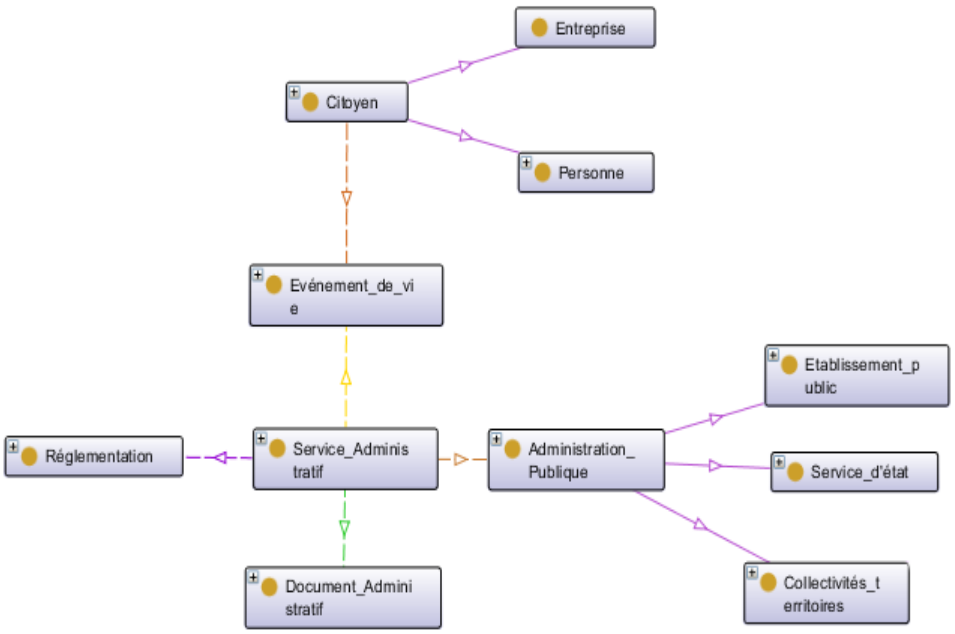


Fig.7. 9: Un aperçu de haut niveau de la structure de l'ontologie de domaine WebGov

```
<owl:ObjectProperty rdf:about="http://localhost/gov.owl#Passe_par">
    <rdfs:domain rdf:resource="http://localhost/gov.owl#Citoyen"/>
    <rdfs:range rdf:resource="http://localhost/gov.owl#Evénement_de_vie"/>
</owl:ObjectProperty>

<!-- http://localhost/gov.owl#Produit -->

<owl:ObjectProperty rdf:about="http://localhost/gov.owl#Produit">
    <rdfs:range rdf:resource="http://localhost/gov.owl#Document_Administratif"/>
    <rdfs:domain rdf:resource="http://localhost/gov.owl#Service_Administratif"/>
</owl:ObjectProperty>

<!-- http://localhost/gov.owl#Evénement_de_vie -->

<owl:Class rdf:about="http://localhost/gov.owl#Evénement_de_vie"/>

<!-- http://localhost/gov.owl#Personne -->

<owl:Class rdf:about="http://localhost/gov.owl#Personne">
    <rdfs:subClassOf rdf:resource="http://localhost/gov.owl#Citoyen"/>
</owl:Class>

<!-- http://localhost/gov.owl#Service_Administratif -->

<owl:Class rdf:about="http://localhost/gov.owl#Service_Administratif"/>


<owl:ObjectProperty rdf:about="http://localhost/gov.owl#Passe_par">
    <rdfs:domain rdf:resource="http://localhost/gov.owl#Citoyen"/>
    <rdfs:range rdf:resource="http://localhost/gov.owl#Evénement_de_vie"/>
</owl:ObjectProperty>

<!-- http://localhost/gov.owl#Produit -->

<owl:ObjectProperty rdf:about="http://localhost/gov.owl#Produit">
    <rdfs:range rdf:resource="http://localhost/gov.owl#Document_Administratif"/>
    <rdfs:domain rdf:resource="http://localhost/gov.owl#Service_Administratif"/>
</owl:ObjectProperty>

<!-- http://localhost/gov.owl#Evénement_de_vie -->

<owl:Class rdf:about="http://localhost/gov.owl#Evénement_de_vie"/>

<!-- http://localhost/gov.owl#Personne -->

<owl:Class rdf:about="http://localhost/gov.owl#Personne">
    <rdfs:subClassOf rdf:resource="http://localhost/gov.owl#Citoyen"/>
</owl:Class>

<!-- http://localhost/gov.owl#Service_Administratif -->

<owl:Class rdf:about="http://localhost/gov.owl#Service_Administratif"/>
```

Fig.7. 10: Un extrait de la description OWL de l'ontologie de domaine WebGov

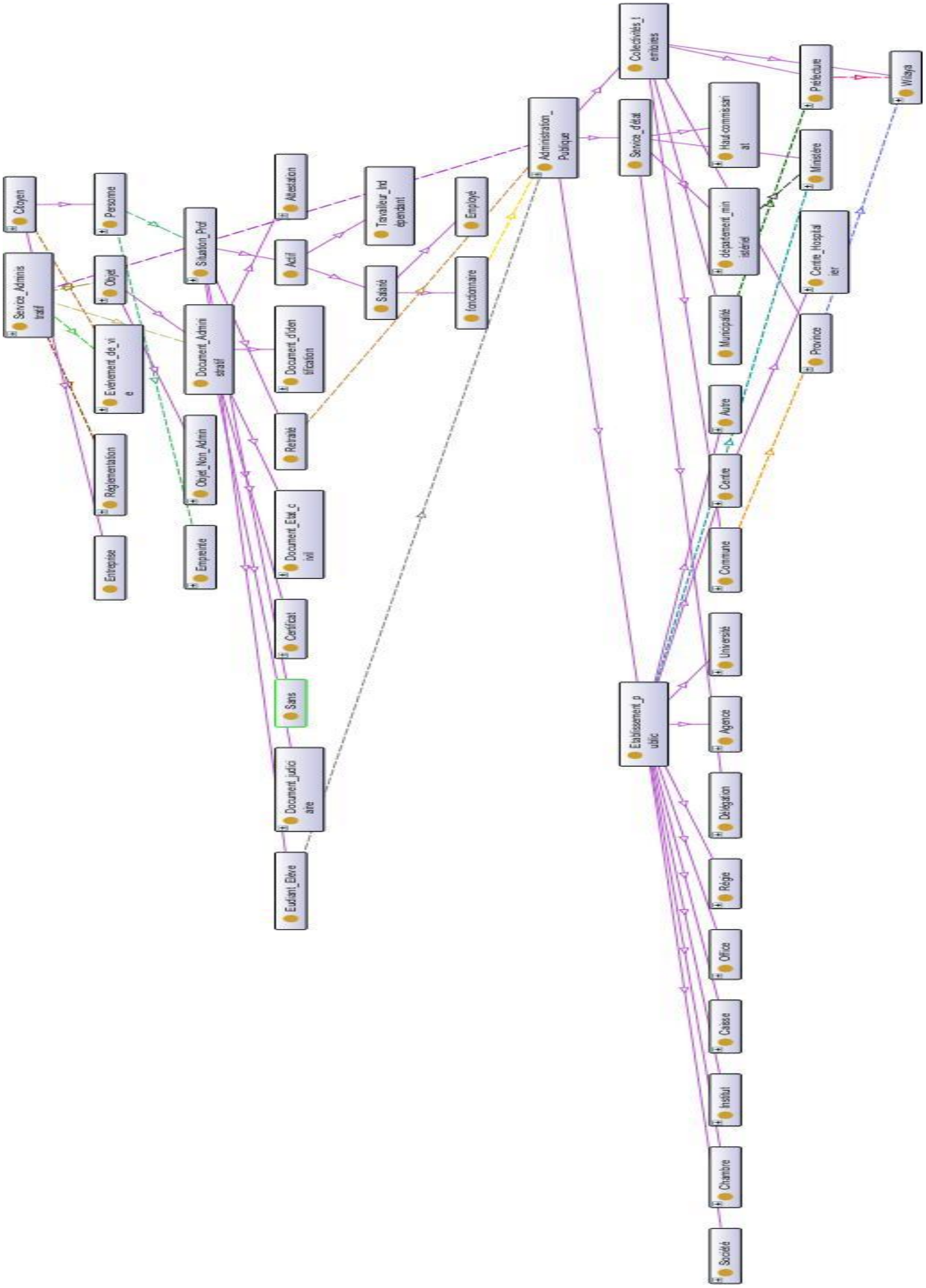


Fig.7. 11:Un aperçu détaillé de la structure de l'ontologie de domaine WebGov

Un aperçu plus détaillé de la structure de l'ontologie de domaine proposée est donné dans la Fig.7. 11.

L'ontologie est constituée de 52 concepts, 18 attributs spécifiques, 17 relations (hors les relations de spécialisation) et 140 instances. Une ontologie beaucoup plus détaillée serait nécessaire pour l'utilisation du système avec les paramètres du monde réel. En fait, la finalité de cette expérimentation est de démontrer la faisabilité technique de l'approche. Nous jugeons alors que la taille actuelle est suffisante et adaptée au besoin du prototype.

Par ailleurs, comme illustré dans la Fig.7. 12, l'ontologie est organisée dans un ensemble de sous ontologies, qui tournent autour des concepts structurants et mettent en évidence d'autres concepts dits supports, la figure ci-après montre un exemple de la sous ontologie « citoyen » :

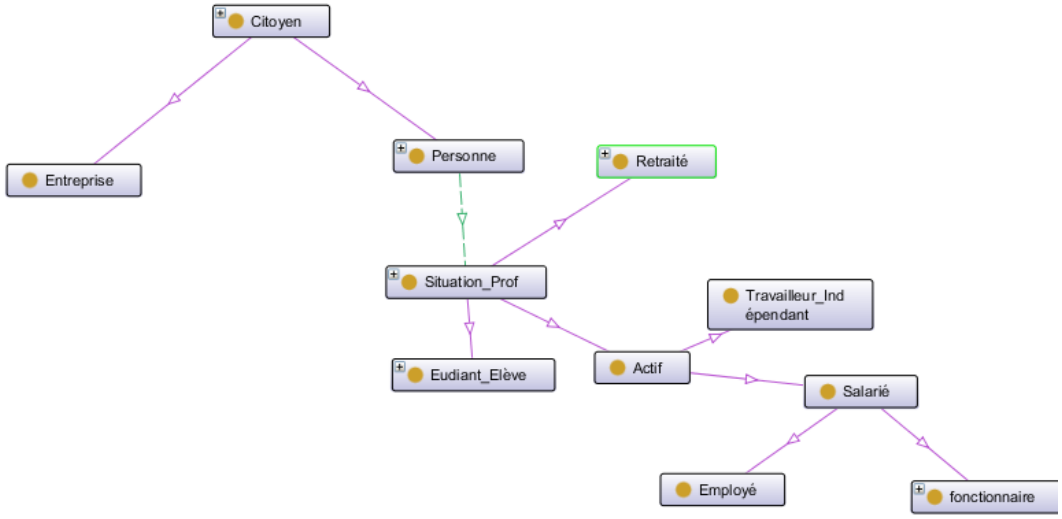


Fig.7. 12: La sous-ontologie « citoyen »

L'ensemble des individus (instances des classes) recueillis ont été aussi définis dans cette étapes de formalisation, la figure ci-après représente un extrait des individus liés à la sous ontologie « structure administrative ».

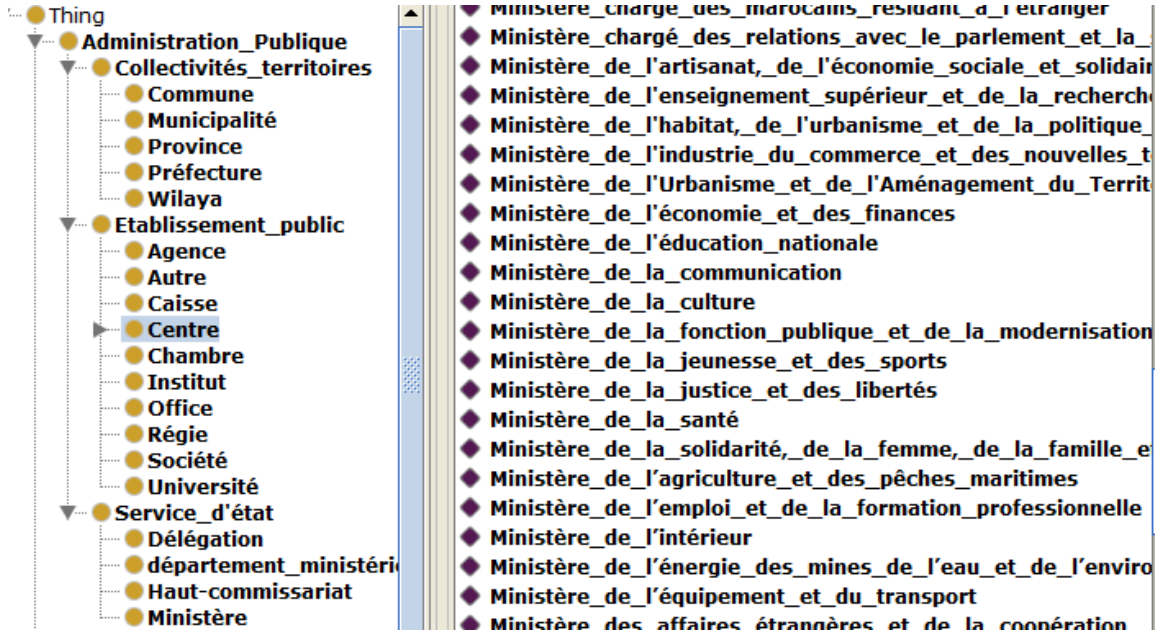


Fig.7. 13: Les individus de la sous ontologie « structure administrative »

7.5. Création de la description sémantique des services découverts

Nom de livrable	SWSList	
Version	1.0	
Description	Ce lot du prototype se constitue d'une liste de services Web sémantiques	
Objectif	Créer les différentes descriptions OWL-S des services utilisés dans le cas d'étude EC_CNI	
Inclusion		**Exclusion**
- Livraison des fichiers ci-dessous : - ConstruireCNI.owls - Livraison_AttestationDeTravail.owls - Livreaison_ActDeNaissance.owls - Livraison_CertificatDeResidance.owls - Utilisation de l'ontologie WebGov		- La description OWL-S de la requête sémantique
Langages	OWL-S, SWRL	
Outils	Java, Eclipse, OWL-S IDE	

Fig.7. 14: Fiche descriptive de livrable.2

En respectons le processus métier relatif à l'étude de cas exposée et en prenant en considération les hypothèses évoquées précédemment, cette étape d'implémentation du PoC consiste à créer les différentes descriptions OWL-S des services supposés être découverts. Il s'agit de décrire chaque service comme un processus atomique qui, à son tour, doit être décrit en termes de son flux de données (inputs et outputs) ainsi qu'en termes de ses contraintes d'utilisation (préconditions et effets). Ces descriptions vont être fournies en entrée du moteur de planification en vue de générer le plan de composition abstrait.

7.5.1. Description conceptuelle des services Web sémantiques

La conception de haut niveau du processus de composition exposée dans le Tableau 7. 4, propose une liste de cinq services abstraits issus de la phase de la découverte qui seront impliqués dans le cycle de composition étudié, le tableau ci-après propose un zoom sur les aspects sémantiques de ces cinq services pour guider la traduction en description OWL-S.

Service	Description	Signature		Préconditions	Effets
		Entrées	Sorties		
ConstruireCNI	Service de construction de la CNI	- Citoyen - Acte de naissance - Certificat de résidence - Liste d'objets non administratifs (empreintes, photos, frais)	- Carte nationale d'identité	- Avoir Acte de naissance - Etre Eligible - Avoir certificat de résidence	- Avoir CNI
Livraison_AttestationDeTravail	Service qui livre l'attestation de travail selon la situation professionnel du citoyen	- Situation Professionnelle du citoyen	- Attestation de travail	Situation Professionnelle conforme	Avoir Attestation de travail
Livreaison_ActDeNaissance	Service délivrant l'acte de naissance	- Citoyen - Objet non administratif : frais (droits)	- Acte de naissance		Avoir Acte de naissance
Livraison_CertificatDeResidance	Service offrant le certificat de résidence	- Citoyen - Acte de Naissance - Certificat de résidence - Liste Objets non administratifs (facture d'eau ou d'électricité, droits)	- Certificat de résidence	- Avoir Acte de naissance - Attestation de travail	Avoir Certificat de résidence

Tableau 7. 4: Description conceptuelle des SWS

7.5.2. Langages et outils de création des services Web sémantiques

Il s'agit principalement dans cette étape de manipuler des services Web sémantiques décrits en langage OWL-S. Pour ce besoin, nous avons utilisé l'éditeur OWL-S IDE [Srinivasan, 2006] qui nous a assistés pour effectuer rapidement et efficacement cette tâche. Concrètement, ce plug-in Eclipse [Eclipse, 2016] a été utilisé pour générer le squelette de la description sémantique des services à partir des classes java. La description OWL-S du modèle de processus du service généré, décrit le service en termes de son flux de données seulement, mais pas en termes de ses contraintes d'utilisation (préconditions et effets). Pour compléter la description, nous avons exploité l'éditeur des modèles de processus fournis par le même éditeur.

Notons, dans le même sens, que l'ontologie OWL-S [Martin et al. , 2004] n'offre pas d'attributs pour décrire les conditions, qui représentent dans ce cas les préconditions des services sémantiques. Pour pallier cette limite nous nous sommes servis du langage de règles

SWRL [Horrocks et al., 2004] (Semantic Web Rule Language), un langage de règles basé sur OWL DL et intégrable au langage OWL-S pour renforcer l'expressivité sémantique en termes de contraintes d'utilisation (c.f annexe C).

7.5.3. Mécanisme de création des services Web sémantiques

En utilisant les outils précédemment présentés, le travail effectué dans cette étape consiste à :

(a) Créer la classe Java implémentant le service web ;
(b) Générer la description OWL-S du service à l'aide du support Java2OWL-S fourni par l'éditeur OWLS-IDE;
(c) Utiliser l'éditeur des modèles de processus (process models) pour compléter la description OWL-S du modèle de processus du service généré.

Pour illustrer le mécanisme, nous présentons ci-après un exemple commenté de la création de la description sémantique du service de construction de la CNI.

(a) Implémenter la classe java

```
public interface ConstruireCNI {
        public  CNI construireCNE( Citoyen citoyen_info, ActDeNaissance
an_du_citoyen ,CerificatDeResidance cr_du_citoyen , Objet_non_admin
droitCNI, Objet_Non_Admin photo, Objet_Non_Admin empreinte);
  }
```

(b) Générer la description sémantique du service correspondant: fichier owl-s

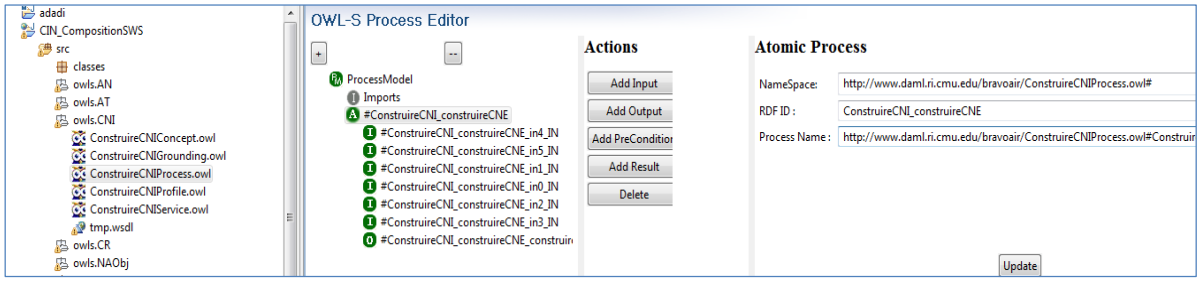


Fig.7. 15: Vue graphique du fichier OWL-S généré

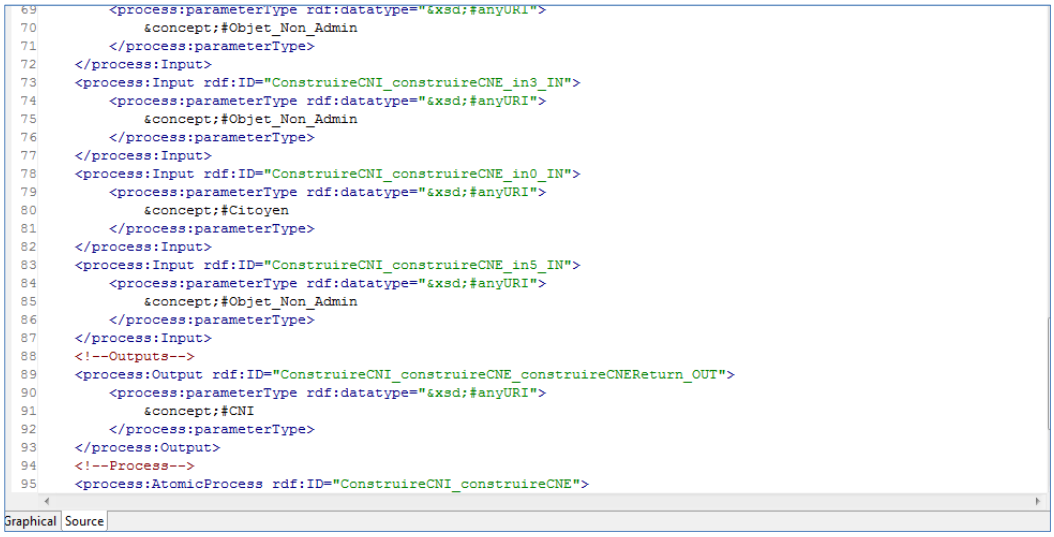


Fig.7. 16: Vue XML du fichier OWL-S généré

(c) Complémenter l'annotation sémantique de la description générée

```
<!ENTITY this "http://localhost:8080/WebGov.owl">
```

Import de l'ontologie webGov pour l'utilisation de ses concepts

```
<rdf:RDF
          .
          .
          .
          .
  xmlns="&this;#"
  xmlns:swrl="&swrl;#"
  xmlns:expr="&expr;#"
  xml:base="&this;#">
```

Espace de nom SWRL

Définition du type de données des paramètres en se basant sur l'ontologie WebGov (&this)

```
    <!--Inputs-->
       <process:Input rdf:ID="ConstruireCNI_CitoyenInfo_IN">
          <process:parameterType
    rdf:datatype="&xsd;#anyURI">&this;#Citoyen
          </process:parameterType>
        </process:Input>

       <process:Input rdf:ID="ConstruireCNI_ANduCitoyen_IN">
          <process:parameterType
          rdf:datatype="&xsd;#anyURI">&this;#ActDeNaissance
          </process:parameterType>
       </process:Input>
             .
             .
             .
             .
     <!--Outputs-->
         <process:Output rdf:ID="ConstruireCNI_CNIduCitoyen_OUT">
           <process:parameterType rdf:datatype="&xsd;#anyURI">&this;#CNI
           </process:parameterType>
        </process:Output>

             .
             .
             .
             .
```

```
<!--Precondition-->
<expr:SWRL-Condition rdf:ID="avoirActDeNaissance">
           <expr:expressionLanguage rdf:resource="&expr;#SWRL"/>
           <expr:expressionObject>
                 <swrl:Imp>
                       <swrl:body>
                       <swrl:AtomList>
           <rdf:first>
             <swrl:IndividualPropertyAtom>
               <swrl:propertyPredicate
rdf:resource="&this;#avoirActDeNaissance"/>
```

Conditions exprimées en langage SWRL

```
                <swrl:argument1
rdf:resource="#ConstruireCNI_CitoyenInfo_IN"/>
                    <swrl:argument2
rdf:resource="#ConstruireCNI_ANduCitoyen_IN"/>
             </swrl:IndividualPropertyAtom>
            </rdf:first>
             <rdf:rest rdf:resource="&rdf;#nil"/>
           </swrl:AtomList>
                        </swrl:body>
                   </swrl:Imp>
             </expr:expressionObject>
       </expr:SWRL-Condition>
                .
                .
                .
                .
```

Description des flux de données (entrées, sorties, préconditions, effets)

```
    <!--Process-->
    <process:AtomicProcess rdf:ID=" GetCNI ">
        <process:hasInput rdf:resource="#ConstruireCNI_CitoyenInfo_IN"/>
        <process:hasInput rdf:resource="#ConstruireCNI_ANduCitoyen_IN"/>
        <process:hasInput rdf:resource="#ConstruireCNI_CRduCitoyen_IN"/>
        <process:hasInput rdf:resource="#ConstruireCNI_DroitCNI_IN"/>
        <process:hasInput rdf:resource="#ConstruireCNI_Photo_IN"/>
        <process:hasInput rdf:resource="#ConstruireCNI_Empreinte_IN"/>
        <process:hasOutput rdf:resource="#ConstruireCNI_CNIduCitoyen_OUT"/>
        <process:hasPrecondition rdf:resource="# avoirActDeNaissance "/>
<process:hasPrecondition rdf:resource="# avoirCerificatDeResidance "/>
        <process:hasEffect rdf:resource="# avoirCNI "/>
    </process:AtomicProcess>

</rdf:RDF>
```

7.6. Mise en place du moteur de planification

Nom de livrable	SWS Composer	
Version	1.0	
Description	Un moteur de planification offrant en sortie le plan de composition d'un ensemble de services sémantiques abstraits fournis en entrée.	
Objectif	Mettre en place la sous architecture de planification en se basant sur la technique de composition « modèle de planification »	
Inclusion		**Exclusion**
- Mettre en place le SMA - Implémenter les algorithmes de planification - Coder les outils de traduction planification/sémantique - Application du moteur sur l'étude de cas EC_CNI		- RAS
Langages/plateformes	FIPA-ACL, JADE	
Outils	Eclipse	

Fig.7. 17: Fiche descriptive de livrable.3

Cette étape s'attache à concrétiser l'une des techniques de composition la plus importante proposée dans ce travail, le modèle de planification. Ce mécanisme qui guide la sous architecture de planification est considéré, en effet, le cerveau de l'approche IdyCoS du moment où il encapsule les algorithmes nécessaires pour que la communauté d'agents Managers coordonnée par un Planner puisse produire le plan de composition abstrait, c'est pour cela que nous attribuons le qualificatif « moteur de planification » à la composante qui implémente cette technique. Le résultat de cette étape complète l'édifice de ce prototype en exploitant les composantes précédemment livrées dans une structure idoine afin d'incarner la vision que nous entretenons pour ce PoC.

Nous détaillons ci-après les étapes de construction du moteur de planification baptisé *SWS Composer*. Nous déroulons ensuite les résultats obtenus.

7.6.1. Structure du moteur de planification

En se basant sur la théorie présentée dans les chapitres 3 et 4, nous proposons l'implémentation du moteur selon la structure illustrée dans la Fig.7. 18:

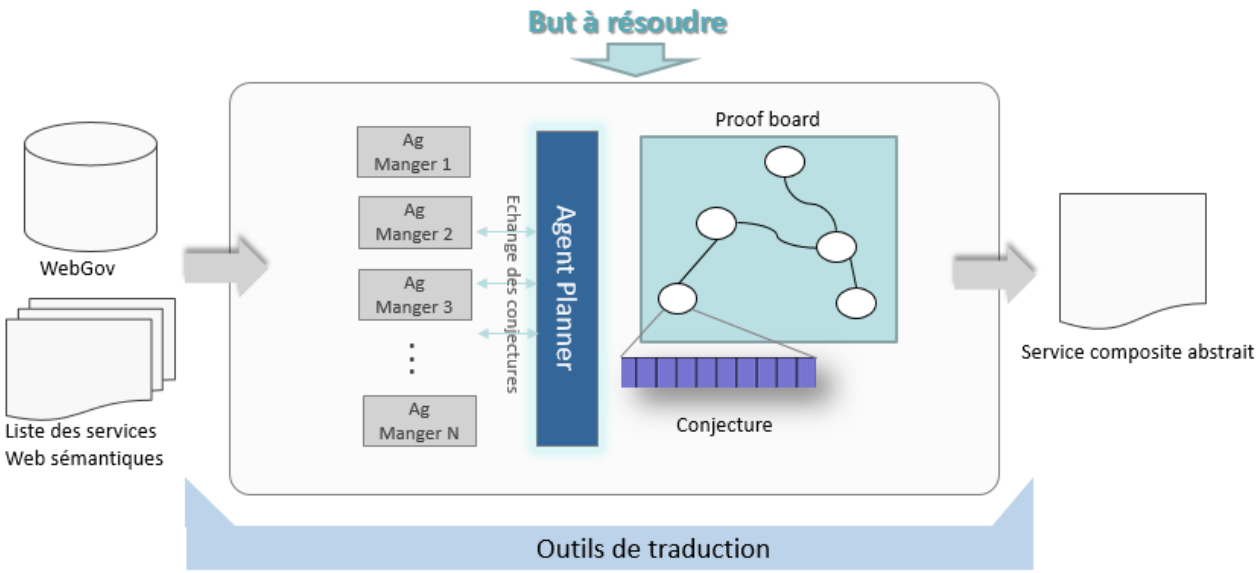


Fig.7. 18: Structure du moteur de planification

Le moteur implémente le mécanisme de planification, il produit un fichier OWL-S décrivant le service composite abstrait en utilisant l'ontologie de domaine précédemment décrite et en prenant en entrée les fichiers OWL-S formalisant les services découverts, le but à résoudre (la requête sémantique) déclenche le mécanisme de planification interne du moteur. Pour accomplir ses missions de planification, l'anatomie du moteur est structurée comme suit :

1. **Broof board (tableau de démonstration):** sous forme d'un graphe, cette structure de données héberge les conjectures et leurs relations ;
2. **Conjecture** : l'unité de données dans ce modèle, c'est aussi le contenu des messages qu'échangent les agents pour faire avancer la planification ;
3. **Planner** : l'agent superviseur, qui assure une mission de coordination ;

4. **Manager** : l'agent qui simule le comportement d'un service abstrait donné.
5. **Outils de traduction** : des algorithmes facilitateurs permettant le passage sémantique-planification et vice versa.

Ces cinq éléments constituent avec leurs interactions le moteur de planification, nous détaillerons la structure interne de chacun de ces éléments juste après la présentation des technologies sur lesquelles le développement de ces éléments s'est basé.

7.6.2. Outils de développement utilisés

Les contraintes en termes d'outils de réalisation du prototype étant relativement faibles; seul le support du langage OWL-S pour la sémantique et du langage FIPA-ACL pour la communication inter-agent est requis (c.f chapitre 4). Nos critères de choix se sont donc, avant tout, portés sur la qualité et la simplicité des outils technologiques. Dans la continuité du livrable précèdent, les développements ont été effectués dans un environnement Java en exploitant les composants offerts par les API java (structure de données, flux entré sortie, JDOM pour le pressing XML) et en se basant sur JADE, comme plateforme de construction et de gestion des interactions agents.

7.6.2.1. Plateforme multi agents JADE

Une plateforme multi-agents est un ensemble d'outils nécessaire à la construction et à la mise en service d'agents au sein d'un environnement spécifique. Ces outils peuvent servir également à l'analyse et au test des SMA ainsi créés. Parmi les plateformes fournies comme logiciels libres, il y a quelques plateformes plus connues pour avoir été utilisées dans le développement de plusieurs applications : JADE, Zeus, MadKit, AgentBuilder, Jack, JAFMAS, AgentTool, DECAF, RMIT…etc. Mais la plus connue c'est JADE.

La plateforme JADE (Java Agent DEvelopment framework) [Bellifemine et al., 2001] est un environnement de développement de SMA conforme aux normes de la FIPA [Huget, 2016] très diffusé. Il fournit, à l'instar des autres plateformes multi agents, une infrastructure gérant la communication, la gestion du cycle de vie des agents, ainsi qu'un ensemble d'agents « utilitaires » assurant le bon fonctionnement d'un SMA [Rimassa et al., 1999] (c.f Annexe C).

Dans une infrastructure un agent JADE est une sous classe de la classe Agent ; la programmation de l'agent se fait donc directement en Java en s'appuyant sur des librairies dédiées. De manière plus abstraite, un agent Jade sera composé de croyances et de capacités (attributs d'objet) et d'un ensemble de comportements (i.e. classes héritant de la classe Behaviour). C'est l'ordonnanceur de comportement qui définira l'ordre des comportements de l'agent et donc la séquence d'actions qu'il entreprendra. Les agents communiquent via des messages ACL.

Le choix de la plateforme Jade pour cette étape de première validation de l'approche proposée est motivé par les arguments ci-dessous:

- Une Plateforme qui permet le développement de systèmes multi-agents et d'applications conformes aux normes FIPA.

- Elle est implémentée en JAVA et fournit des classes génériques pour faciliter la création des systèmes multi-agents.
- Elle facilite la création des systèmes distribués basés sur les systèmes multi-agents et gère la communication entre les différents SMA répartis dans le web via le protocole IIOP « Internet InterOrb Protocol ».
- Elle offre une interface graphique utilisateur intuitive et maniable pour gérer et contrôler la société d'agents du système.
- Elle est open source avec une communauté très active derrière.

Les notions présentées dans cette introduction sur JADE sont détaillées au niveau de l'annexe C pour une meilleure explication des différents concepts et composants JADE utilisés pour le développement du SMA de SWS Composer.

7.6.2.2. Langage de communication entre agent FIPA-ACL

Comme convenu dans la première section du chapitre 4, nous utilisons dans le cadre de ce travail le langage FIPA-ACL [FIPA-ACL, 2016] comme langage de communication entre les différents agents (ACL).

FIPA- ACL a été défini pour répondre principalement à une critique assez vigoureuse contre KQML [Finin et al., 1994]; ses performatifs sont trop nombreux, donc redondants, et de plus ils ne couvrent pas en entier le champ des performatifs envisageables.

L'ACL de FIPA ressemble à KQML, mais il ne comporte que 22 performatifs. De plus, il est défini de façon beaucoup moins ambigüe : la sémantique des performatifs est décrite dans un langage spécialisé, le langage SL (Semantic Language) [SL, 2016]. Notons que ce langage n'implique aucun engagement de base vis-à-vis d'un langage pour le contenu. Plusieurs langages peuvent être utilisés pour coder le contenu, comme : SQL, PROLOG, JAVA.

Nos agents JADE utilisent des messages décrits en FIPA-ACL. Ces messages sont composés en général de :

- L'émetteur du message : un champ rempli automatiquement lors de l'envoi d'un message ;
- L'ensemble des récepteurs du message : un message peut être envoyé à plusieurs agents simultanément ;
- L'acte de communication : qui représente le but de l'envoi du message en cours (informer l'agent récepteur, appel d'offre, réponse à une requête,…) ;
- Le contenu du message ;
- Un ensemble de champs facultatifs, comme la langue utilisée, l'ontologie, le timeout, l'adresse de réponse.

Les aspects avancés de ce langage sont donnés en annexe (annexe C).

Après avoir survolé les outils et les langages utilisés, nous présentons le détail d'implémentation de chacun des cinq éléments clés de notre moteur de planification.

7.6.3. Implémentation du moteur de planification

7.6.3.1. Proof Board

Description	C'est l'entité qui héberge les conjectures échangées, elle simule la notion théorique décrite dans le chapitre 4 (section 4.3.3)
Nature	Graphe
Structure	La structure de cette entité implémente principalement : • Un mécanisme d'alimentation du graphe des conjectures qui fait avancer la planification. • Un mécanisme d'exploration de graphe à la recherche du plan solution.

Tableau 7. 5: Description de l'entité Proof table

La signature des méthodes formant la structure de cette entité est donnée ci-après avec des commentaires :

```
public class Graphe implements  Serializable{
//le graphe (proof board) est formé par des nœuds caractérisés par : une donnée (hypothèse ou raffinement) et un père qui est à son tour un noeud

   protected Vector<Noeud> nodes ;

          //…

         //Ajouter au graphe le raffinement/hypothèse « n » ayant comme père le noeud « p »
   public void addNode(node n, node p) {
          }

          //Déterminer les hypothèses formées par un nœud père « n »
   public Vector<Noeud> sonNodes(Noeud n) {
          }

          //Déterminer les conjectures non encore résolues
   public Vector<Noeud> leavesNodes() {
          }

          //Tester si toutes les hypothèses sont résolues
   public boolean ContainsleavesNodes( Noeud n) {
          }

        //Cette méthode retourne une valeur vrai si la taille du vecteur retourné par la    méthode leavesNodes() est nulle, c.à.d : toutes les hypothèses sont résolues
   public boolean isResolved() {
          }
```

7.6.3.2. Conjecture

Description	Plan partiel dont l'implémentation respecte les deux notions théoriques définies dans le chapitre 4 (définition 5)
Nature	Classe
Structure	Cette entité représente l'unité de donnée échangée entre les agents dans les messages ACL. elle simule un raffinement, c'est à dire la contribution d'un agent pour vérifier les hypothèses non résolues se trouvant au niveau du graphe

Tableau 7. 6: Description de l'entité Conjecture

La classe Conjecture se présente donc sous forme d'un ensemble de nœuds, elle est identifiée par un ID unique.

```
Public class Conjecture implements  Serializable {

private Vector<Noeud> conj;

private int id;
…
}
```

7.6.3.3. Agent Planner

Description	Agent superviseur
Nature	Agent JADE
Structure	La structure de cet agent permet de: ▪ Initier le débat ; ▪ Animer le débat via l'alimentation du proof table et la sollicitation des agents managers ; ▪ Clôturer le débat (succès ou échec)

Tableau 7. 7: Description de l'entité agent Planner

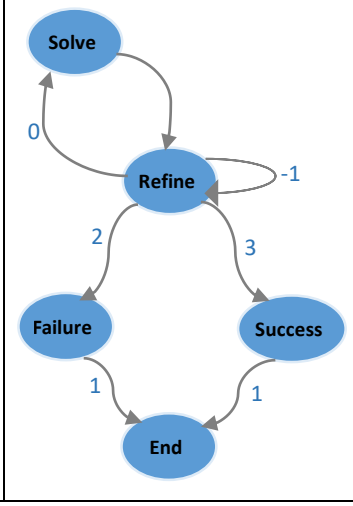
La structure de cet agent est donnée en code commenté ci-après:

```
// L'agent planificateur dispose de :
 private List<AID> managerAgents; // File des managers abonnés
 private Conjecture firstConjecture; // La première conjecture qui simule
                                     le but à résoudre
 private Graphe ProofBoard; //stocke l'état de le débat dans le proof
board
```

Cette entité implémente le comportement ci-dessous :

```
FSMBehaviour planner_beh= new FSMBehaviour();
planner_beh.registerFirstState(new solve(), "solve");
planner_beh.registerState(new refine(), "refine");
planner_beh.registerState(new failure(), "failure");
planner_beh.registerState(new success(), "success");
planner_beh.registerLastState(new end(), "end");
planner_beh.registerDefaultTransition("solve",
"refine");
planner_beh.registerTransition("refine", "solve",0);
planner_beh.registerTransition("refine", "refine",-1);
planner_beh.registerTransition("refine", "failure",2);
planner_beh.registerTransition("refine", "success",3);

planner_beh.registerTransition("success", "end",1);
planner_beh.registerTransition("failure", "end",1);
```



Nous présentons ci-après, à titre illustratif, un extrait de la boucle de raisonnement de l'agent Planner codée:

```
private class solve extends OneShotBehaviour{
@Override
public void action() {
/*************** Appel à proposition **************/
stop = 0;
ACLMessage cfp = new ACLMessage(ACLMessage.CFP);
for (int i = 0; i < managerAgents.size(); ++i) {
cfp.addReceiver(managerAgents.get(i));
}
try {
cfp.setContentObject(goal);
                    }
              …
myAgent.send(cfp);

                   }
  }

/*************************************************************************/
private class refine extends OneShotBehaviour{
          @Override
public void action() {
             /*************** En attente d'une réponse**************/

ACLMessage messageRecu =blockingReceive();
            /*************** Une contribution est reçue**************/

if (messageRecu.getPerformative() == ACLMessage.PROPOSE) {

try {
goal=(Conjecture)messageRecu.getContentObject();
for (  int i = 0; i < goal.getConj().size(); i++) {
if(goal.getConj().size()>0){
```

```
  /*************** Alimentation du Proof table **************/
for (  int i = 0; i < goal.getConj().size(); i++) {
ProofBoard.addNode(goal.getConj().elementAt(i).getData(),goal.getConj().ele
mentAt(i).getPredecessor().getData());
                    }

…..

…
   /*************** Test de reussite **************/
              if(ProofBoard.isResolved() ){
                  stop=3;//goto success
              }

                 }
   /*************** Test d'echec **************/
//Si les manager ont notifié leur echec de raffinement de la conjecture en
cours

        else if (messageRecu.getPerformative() == ACLMessage.FAILURE ){

Conjecture failureGoal = new Conjecture();
try{  failureGoal= (Conjecture)messageRecu.getContentObject();}
              …
….
failur.add( failureGoal);


int failure=0;// Pour tester que tous les manager ont répondu par
« failure »
for (int i = 0; i < failur.size(); i++) {
if( failur.get(i).getId()==failureGoal.getId())
                           {
              failure++;

                           }    }

              stop=-1;  //Aller à refine  (par défaut)
        /*************** ECCHEC **************/
              if( failure>=managerAgents.size()){//tous les manager ont
échoué à raffiner

              stop=2;  //Aller à FAILURE
              }
        }
          }
```

7.6.3.4. Agent Manager

Description	Agent simulant le manger comme décrit dans le chapitre 4 (section 4.3.2)
Nature	Agent JADE
Structure	La structure de cet agent l'aide à raffiner les conjectures reçues via les messages ACL en se basant sur les opérateurs dont il dispose;

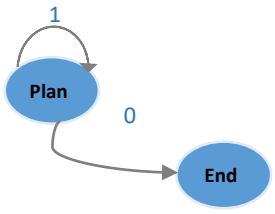
Tableau 7. 8: Description de l'entité agent Manager

L'agent manager dispose comme attributs des éléments ci-dessous :

```
public Operator[]  Op;// in, out, precond, effects
Conjecture  plan_partiel // le raffinement calculé dans un cycle
```

Le comportement représenté ci-après met en évidence la mission principale de raffinement de cet agent :

```
FSMBehaviour manager_beh= new FSMBehaviour();
manager_beh.registerFirstState(new plan(), "plan");
manager_beh.registerLastState(new end(), "end");
manager_beh.registerTransition("plan", "plan",1);
manager_beh.registerTransition("plan", "end",0);
```



L'état principal de cet agent (Plan) encapsule la boucle de raisonnement illustrée ci-après:

```
private class plan extends OneShotBehaviour{
@Override
public void action() {

/***************dans l'attente d'un appel**************/
ACLMessage messageRecu =blockingReceive();


/***************reception d'un CFP**************/

if (messageRecu.getPerformative() == ACLMessage.CFP) {
                                try {
plan_partiel= (Conjecture)messageRecu.getContentObject();

                                    }
...

/*************************Traitement de la conjecture*****************/
lg= plan_partiel.getConj();
result=new Vector<Noeud>();
fail=1;
for (  int i = 0; i < lg.size(); i++) {
if( refinable(lg.elementAt(i) )
{
fail=0;
```

```
refine (lg.elementAt(i), result); }
}
…
  }
/***************Proposition d'un plan partiel ************************/


if(fail==0)
{
plan_partiel.setConj(result);
ACLMessage propos = new ACLMessage(ACLMessage.PROPOSE);
propos.addReceiver(new AID("planner", AID.ISLOCALNAME));
try {
propos.setContentObject(plan_partiel)}
. . .
myAgent.send(propos);
      /***************Envoi d'une notification d'échec**************/

else if (fail==1) {
ACLMessage failureNotif = new ACLMessage(ACLMessage.FAILURE);
failureNotif.addReceiver(new AID("planner", AID.ISLOCALNAME));
try {
failureNotif.setContentObject(plan_partiel); }
….
myAgent.send(failureNotif); }
raisonne=1;  /**aller à "plan" et attendre un nouveau appel */
                        }
```

7.6.3.5. Outils de traduction

Description	Entité implémentant l'algorithme de traduction décrit dans le chapitre 4 (section 4.5.3.3)
Nature	Classe
Structure	La classe « tools » offre des outils qui explorent le contenu des fichiers owl-s et le convertient en operateurs PDDL, ces outils permettent également de traduire le plan solution en fichier owl-s

Tableau 7. 9: Description de l'entité Outils de traduction

La signature des principales méthodes formant la structure de cette entité est donnée ci-après avec des commentaires :

```
public class Tools {
      static public String fileToString (File filePath) {

      }
      static public String[] owlsToOperators (String owlsString) {
      }
      static public void planToOlwls (Plan plan) {
//plan est une structure formée par une liste des opérateurs, une liste des
contraintes d'ordre et une liste des liens de causalité (selon la
définition 5 du chapitre 4
                        }
```

7.6.3.6. Cycle de planification

En se basant sur les structures que nous venons de décrire, à chaque cycle de raffinement, le Planner envoie un message « CFP » pour les Mangers (*myAgent.send(cfp)*), les Managers raffinent la conjecture contenue dans le message d'une façon autonome et concurrente (*refine (conj, result))* et soumissionnent leurs contributions à travers un autre message FIPA-ACL (*myAgent.send(propos)),* l'occasion pour le Planner de mettre à jour le Proof board (*ProofBoard.addNode(goal.getConj())*) et de tester s'il y a un plan solution (*ProofBoard.isResolved()*) dans celle-ci, sinon un nouveau cycle de planification commence (*myAgent.send(cfp)*).

7.6.3.7. Résultat obtenu

Pour pouvoir interagir directement avec le moteur de planification SWS Composer, nous avons conçu une GUI (Graphical User Interface) intermédiaire à partir de laquelle nous soumissionnons la requête sémantique à résoudre, qui est, dans cette première version du livrable, directement traduite en opérateurs PDDL sans passer par l'état OWL-S. L'interface de communication du moteur permet également d'inscrire la description sémantique des services qui vont participer à la planification.

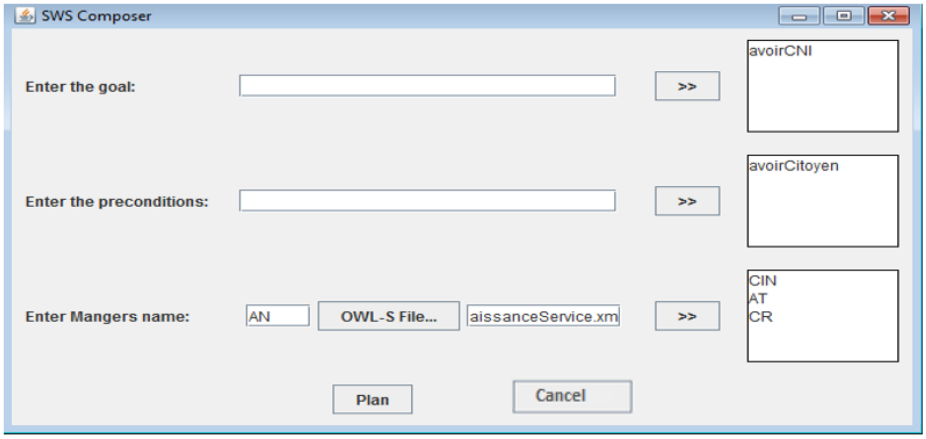


Fig.7. 19: L'interface de communication avec le moteur SWS Composer

Le lancement de planification se fait à travers le bouton « Plan » qui déclenche le débat du SMA généré automatique à partir des éléments entrés.

Le résultat obtenu, suite à l'introduction de la requête relative à l'étude de cas étudié ainsi que la liste des sévices abstraits y afférents, est illustré dans les figures ci-après :

Les éléments de sortie du moteur SWS Composer

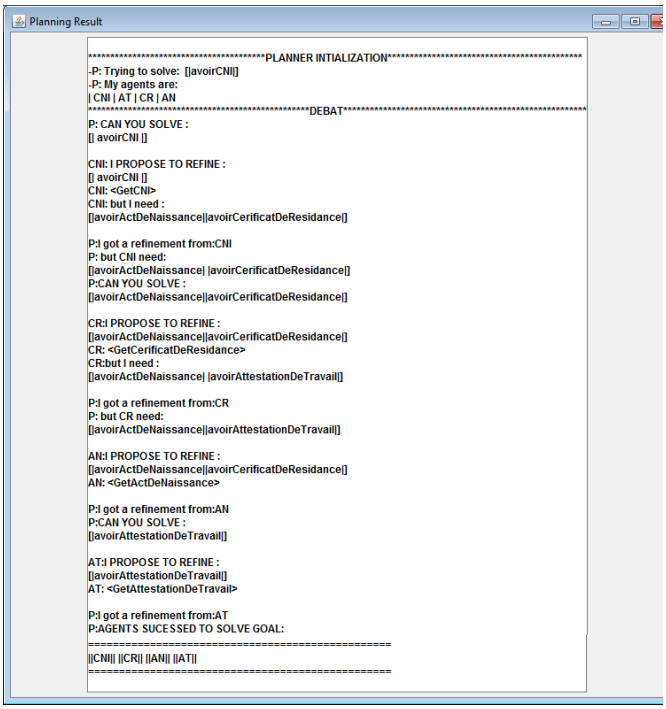


Fig.7.20.c: Journal du débat

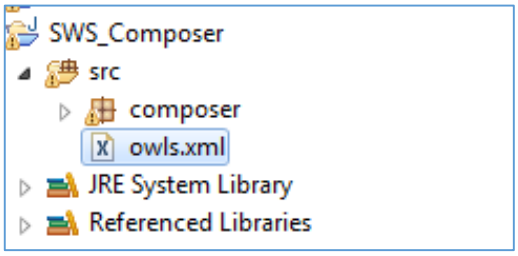


Fig.7. 20.a : Fichier owl-s généré

```
1
2  <process:CompositeProcess rdf:ID =" CNIomposite ">
3  <process:composedOf >
4  <process:Sequence >
5  <process:components rdf:parseType ="Collection">
6  <process:AtomicProcess rdf:about ="# CNI" />
7  <process:AtomicProcess rdf:about ="# CR" />
8  <process:AtomicProcess rdf:about ="# AN" />
9  <process:AtomicProcess rdf:about ="# AT" />
10 </process:components >
11 </process:Sequence >
12 </process:composedOf >
13 </process:CompositeProcess >
14 </rdf:RDF>
```

Fig.7.20.b : Extrait du contenu du fichier owl-s généré

Le moteur génère en sortie un fichier (Fig.7. 20.a) contenant la description sémantique du service composite (Fig.7.20.b). Ainsi qu'un journal traçant le déroulement du débat de planification et le plan retenu (Fig. 7. 20.c)

7.7. Vérification et analyse du prototype

7.7.1. Vérification du prototype

Le prototype exposé dans ce chapitre s'articule autour du moteur de planification appuyé par des entrées sémantiques et une référence de connaissance ontologique. Le modèle d'interaction entre ces trois éléments représente une première preuve de concept de notre approche.

Afin de montrer la pertinence du moteur de planification mis en place, nous l'avons testé sur l'étude de cas EC_CNI. Avant de dérouler le plan de composition généré par le moteur, résolvons (théoriquement) d'abord la requête en se basant sur la technique de modèle de planification.

Rappelons qu'il s'agit d'une demande de renouvèlement de la carte nationale d'identité. L'étude préétablie montre que cette requête implique des sous processus faisant appel à plusieurs instances administratives et ne peut donc pas être prise en charge par un seul service. Quatre services abstraits ont été identifiés pour contribuer à la résolution de ce problème, il s'agit notamment des services suivants :

- **ConstruireCNI (CNI)**
- **Livraison_AttestationDeTravail (AT)**
- **Livreaison_ActDeNaissance (AN)**
- **Livraison_CertificatDeResidance (CR)**

L'exercice consiste donc à identifier les cycles de raffinement entrepris par la communauté d'agents simulant les services identifiés et ce, à l'instar de l'exemple d'application présenté dans le chapitre 4. Le processus devrais donc se dérouler ainsi :

a) Initialisation : le Planner envoie à l'ensemble des mangers, qui simulent les quatre services découverts, la première conjecture qui représente le but à atteindre en l'occurrence « livraison de CNI » Fig.7. 21.
b) Premier raffinement : seul le service ConstruireCNI peut raffiner cette conjecture, mais la solution qu'il propose forme de nouvelles hypothèses qui devront être résolues Fig.7. 22.
c) Deuxième raffinement : les Mangers possèdent maintenant deux hypothèses à raffiner, AvoirCR et AvoirAN. La première hypothèse peut être résolue par l'agent AN, alors que la deuxième hypothèse peut être raffinée par l'agent CR, mais ce dernier a besoin d'un acte de naissance et d'une attestation de travail Fig.7. 23.
d) Raffinement solution : l'agent AN peut livrer l'acte de naissance à Mariam, pendant que l'agent AT peut fournir une attestation de travail au professeur, ainsi toutes les hypothèses seront vérifiées et nous obtenons le plan solution représenté dans la Fig.7.24.

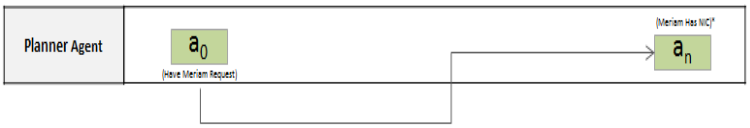


Fig.7. 21: La conjecture initiale [Adadi et al., 2015b]

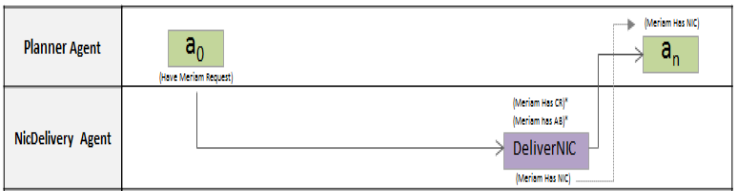


Fig.7. 22 : Premier raffinement [Adadi et al., 2015b]

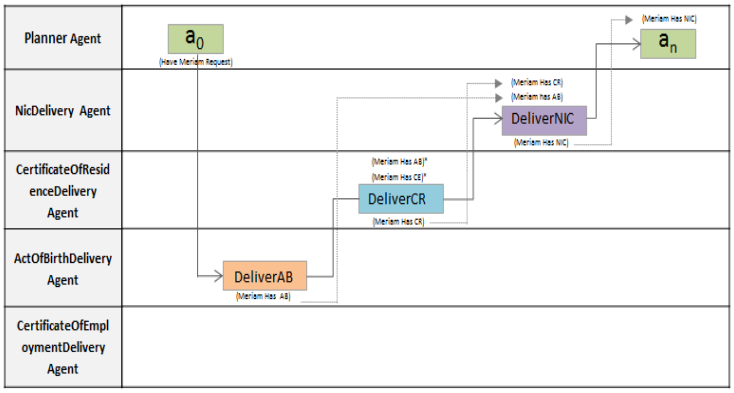


Fig.7. 23: Deuxième raffinement [Adadi et al., 2015b]

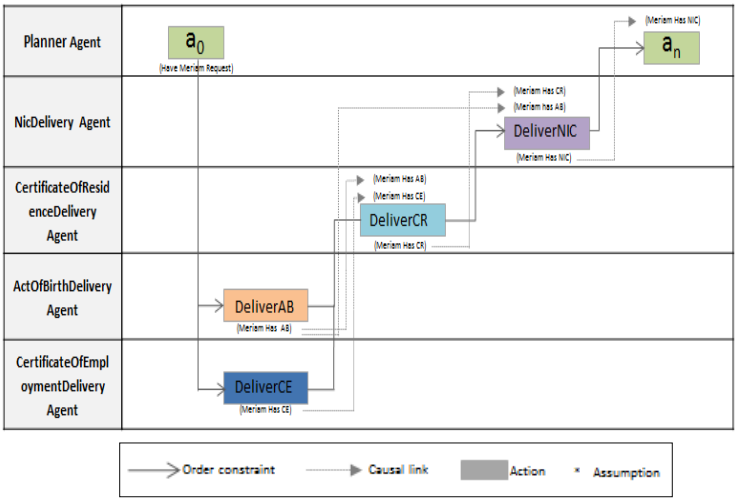


Fig.7. 24: Le plan solution [Adadi et al., 2015b]

Analysant à présent le degré de conformité des résultats générés par le moteur suite à l'injection de la requête de notre scénario d'illustration et vérifiant si ces résultats coïncident avec les résultats théoriques.

Ci-après un extrait commenté de la trace de débat de planification généré par le moteur :

```
**************************************PLANNER INTIALIZATION*****************************************
-P: Trying to solve:  [|avoirCNI|]
-P: My agents are:
| CNI | AT | CR | AN
```

- Lancement du débat avec l'envoi du but à résoudre : rappel des préconditions de la conjecture initiale représentent le but soumis à l'ensemble des agents impliqués dans ce cycle de composition.

```
************************************************DEBAT*************************
P: CAN YOU SOLVE :
[| avoirCNI |]

CNI: I PROPOSE TO REFINE :
[| avoirCNI |]
CNI: <GetCNI>
CNI: but I need :
[|avoirActDeNaissance||avoirCerificatDeResidance|]

P:I got a refinement from:CNI
```

- Premier raffinement : l'agent Manager CNI propose un raffinement et le soumis au Planner, ce dernier lance alors le deuxième raffinement

```
[|avoirActDeNaissance| |avoirCerificatDeResidance|]
P:CAN YOU SOLVE :
[|avoirActDeNaissance||avoirCerificatDeResidance|]

CR:I PROPOSE TO REFINE :
[|avoirActDeNaissance||avoirCerificatDeResidance|]
CR: <GetCerificatDeResidance>
CR:but I need :
[|avoirActDeNaissance| |avoirAttestationDeTravail|]
```

- Deuxième raffinement : l'agent Manger CR propose de raffiner une partie du plan en cours mais génère une nouvelle hypothèse

```
P:I got a refinement from:CR
P: but CR need:
[|avoirActDeNaissance||avoirAttestationDeTravail|]

AN:I PROPOSE TO REFINE :
[|avoirActDeNaissance||avoirCerificatDeResidance|]
AN: <GetActDeNaissance>
```

- Troisième raffinement : l'agent manger AN propose de raffiner l'autre partie du plan en cours, sans générer d'hypothèses, le Planner demande ainsi si un manager peut résoudre l'hypothèse généré par CR

```
P:I got a refinement from:AN
P:CAN YOU SOLVE :
[|avoirAttestationDeTravail|]

AT:I PROPOSE TO REFINE :
[|avoirAttestationDeTravail|]
AT: <GetAttestationDeTravail>
```

```
P:I got a refinement from:AT
P:AGENTS SUCESSED TO SOLVE GOAL:
================================================
||CNI|| ||CR|| ||AN|| ||AT||
================================================
```

- Raffinement final : l'agent AT propose de raffiner la dernière hypothèse et le Planner déclare le succès de planification sur un plan qui ordonne la participation des agents ainsi CNI-> CR->AN-> AT.
  Rappelons que le schéma généré définit l'ordre inverse de l'appel des services (du dernier au premier).

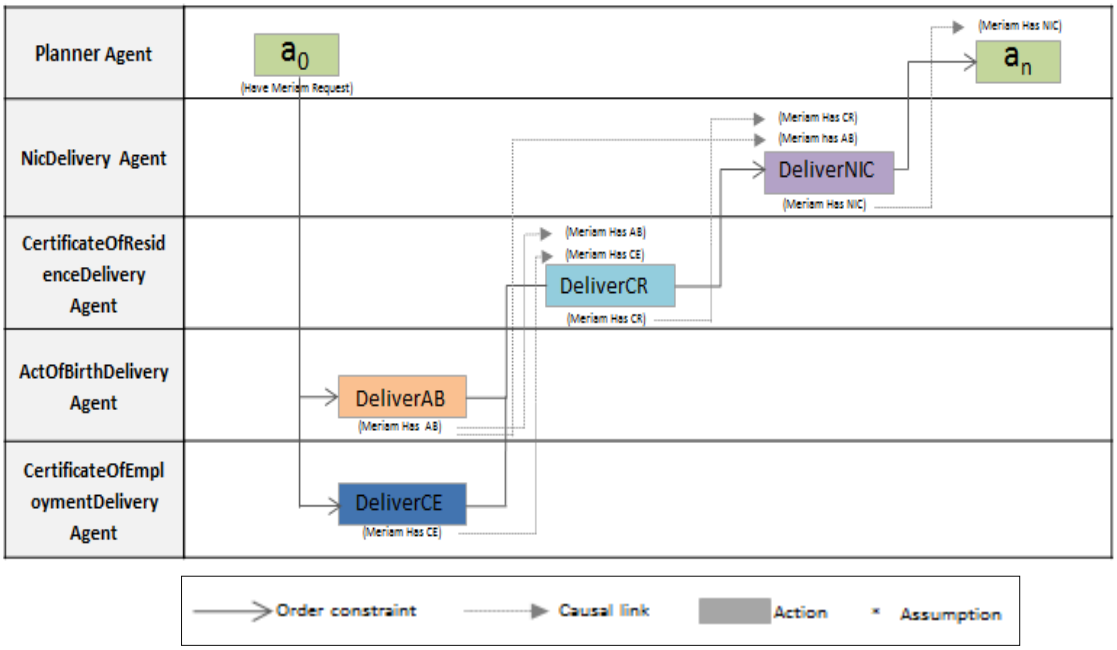


Fig.7. 25: Le plan solution (Rappel)

D'après la lecture des contraintes d'ordre et de causalité définies dans le plan solution rappelé dans la Fig.7. 25: l'exécution respecte l'ordre suivant : CNI->CR puis AN et CE en parallèle. Le résultat pratique et le résultat théorique sont donc identiques.

7.7.2. Analyse du prototype

L'objectif de cette première ébauche de prototype est d'aborder le volet « évaluation de l'approche » d'une façon simpliste mais efficace en s'attachant au contexte d'e-gouvernement pré-étudié.

Le présent PoC est considéré un jalon important dans le développement du prototype final de l'approche iDyCoS. En effet, en mettant l'accent sur la technique de planification, le produit proposé met également en pratique les aspects relatifs à l'ontologie, à la sémantique, au SMA, aux algorithmes de planification et plus important encore à l'interaction supposée être la plus harmonieuse possible afin que ces éléments appartenant à des domaines de recherche différents puissent collaborer.

Concernant le livrable relatif à l'ontologie de domaine, vue que les spécifications adoptées pour la simulation ne reflètent pas complètement la réalité, qui est évidemment beaucoup plus complexe à traduire. Ce livrable nécessite éventuellement plus de précision. La structure de WebGoV est toutefois évolutive et supporte tout éventuel raffinement, précision ou montée en charge.

Puisque la découverte ne fait pas partie du périmètre du prototype actuel, il a été important d'élaborer la description sémantique des services contributeurs à la main, les services développés sont ceux qui participent à la résolution de l'étude de cas sélectionnée. Cela ne pose aucune limite ou restriction par rapport aux inputs du moteur de planification, qui propose de synthétiser les offres de n'importe quelle collection de services décrits sémantiquement, le mécanisme de génération des services et de leurs descriptions exposé

précédemment reste le même pour la préparation des services à composer par le moteur. Le livrable relatif à cet aspect sera, par contre, abandonné une fois les techniques de la phase de découverte seront implémentées.

Le moteur de planification simule la sous architecture de planification, il prouve la viabilité de cette organisation piloté par une entité Leader (Planner) et des ouvriers (Managers) collaborant autour d'une unité de données (la conjecture) afin d'atteindre le but à résoudre. Les choix des structures algorithmiques semblent être adéquats, des optimisations seront, par contre, nécessaires pour traiter des questions avancées comme la scalabilité et l'efficience.

En ce qui concerne les travaux futurs, ce prototype devrait être intégré dans une approche globale de simulation de la composition abstraite, en mettant en œuvre l'ensemble de techniques de composition y afférents. La feuille de route intègre notamment :

- ➔ L'implémentation du mécanisme de découverte ;
- ➔ L'Optimisation de l'algorithme de l'exploration de la structure « Proof Board » et du choix des conjectures à raffiner ;
- ➔ La mise en place de l'interface de communication ontologique et implémentation des mécanismes de génération de la requête sémantique.

7.8. Conclusion

La réalisation d'un prototype est une étape importante dans la validation de notre approche. Ce chapitre présente en quatre parties principales les différentes briques implémentées dans le cadre d'une preuve de concept.

L'exercice pratiqué tente de concrétiser la notion d'ontologie, de manipuler des services sémantiques, d'implémenter la sous architecture de planification et d'intégrer ces trois éléments afin de former un tout indivisible.

La présentation du prototype ne fait que sommairement appel au fond théorique antérieurement détaillé dans les chapitres précédents, elle porte plutôt sur la description exhaustive des structures et des entités développées, sur les choix techniques et sur l'analyse des résultats.

Le chapitre se termine par les perspectives offertes par le développement de ce prototype et présente les futurs axes de développement.

Les contributions exposées dans ce manuscrit seront sujettes d'une évaluation au niveau du chapitre suivant (le dernier chapitre).

Chapitre 8

Evaluation et discussion

We drive into the future using only our rearview mirror.
Marshall McLuhan

8.1. Introduction

Afin de parvenir à tirer des conclusions fiables et valides par rapport à l'approche iDyCoS, il est essentiel de procéder à l'évaluation de cette proposition. L'évaluation a pour objectif d'identifier les points d'intérêt et d'envisager les pistes d'amélioration, c'est dans ce cadre que s'inscrivent les deux chapitres précédents qui ont discuté respectivement le champ d'exercice potentiel et la validité technique de l'approche. Pour compléter ce processus d'évaluation, une des démarches serait de confronter le système à des conditions réelles (des services et des utilisateurs réels) et de surveiller et mesurer les interactions. Toutefois une telle démarche d'évaluation quantitative risque, à ce stade précoce d'implémentation, d'être hâtive et de conduire à des fausses conclusions. Nous proposons dans ce chapitre une évaluation plutôt conceptuelle basée sur des critères qualitatifs qui consistent d'abord à s'assurer formellement que le résultat est conforme aux spécifications définies (une sorte de recette), et à positionner ce que nous avons conçu par rapports aux travaux connexes en s'appuyant sur une analyse comparative. En complément et parallèlement, la discussion des pistes d'extension et d'ouverture du modèle proposé, constitue un autre paramètre d'évaluation essentiel qui prendra place à la fin de ce chapitre.

8.2. Vérification des exigences

Maintenant que les différentes théories et techniques qui concrétisent les aspects caractérisant l'approche proposée ont été éclairées tout au long des chapitres de ce manuscrit. Nous pouvons, à présent, vérifier l'atteinte de la conception que nous avons imaginé au début et que nous l'avons formalisée sous formes d'exigences fonctionnelles et de qualité (cf. Chapitre 3).

Le tableau ci-après rappelle et illustre le résultat de conformité aux exigences fonctionnelles :

Référence	Exigence	Conformité
EF1	Le système doit assurer la traduction de la requête exprimée en langage naturel en un langage sémantique compréhensible par le système	Oui
EF2	Le système doit assurer la recherche sémantique des services Web élémentaires potentiels (susceptibles de résoudre la requête une fois assemblés)	Oui
EF3	Le système doit définir l'ordre de l'exécution des services utilisés dans la composition	Oui
EF4	Le système doit proposer la meilleure solution qui prend en compte les besoins fonctionnels ainsi que les préférences précisées par l'utilisateur	Oui
EF5	L'exécution du service produit doit être autonome et	Oui

	dynamique	
EF6	L'opération de la composition doit être effectuée de façon automatique et transparente vis-à-vis de l'utilisateur	Oui
EF7	Le système doit être générique et indépendant du domaine des services web (commerce, santé, biologie…)	Oui
EF8	Le système doit offrir des interfaces graphiques intuitives et adaptées pour faciliter l'interaction avec l'utilisateur d'une part et pour fluidifier l'opération de traduction d'autre part	Oui

Tableau 8. 1: Conformité aux exigences fonctionnelles

F1, F2, F3, F4 et F5: les quatre phases guidées par les techniques de composition proposées résolvent chacune l'un des sous problèmes de composition que représentent les cinq premières exigences.

EF6 : en s'appuyant sur l'H4F, les composants intervenant dans l'architecture iDyCoS et leurs interactions permettent de dégager l'aspect dynamique et autonome de la composition.

EF7 : l'ontologie de domaine, un élément pivot de l'approche, permet, en effet, d'assurer l'indépendance du système par rapport au domaine étudié. Dans le cadre de ce travail nous avons appliqué l'approche sur le domaine de l'administration électronique, dans ce sens et pour un besoin de prototypage, une ontologie du domaine d'e-gouvernement a été introduite. Pour gérer la composition des services d'un autre domaine il suffit donc de brancher le système avec l'ontologie de domaine adéquate.

EF8 : nous qualifions l'intermédiaire IHM (Interface Homme Machine) d'interface ontologique, puisqu'il permet de guider l'interaction utilisateur-système en se basant sur la couche ontologique pour faciliter l'opération de construction de la requête sémantique, une simulation de cette GUI (Graphical User Interface) a été présentée au niveau du chapitre 4.

Nous concluons alors que sur le plan fonctionnel le système répond peu ou prou aux spécifications définies. Sur le plan non fonctionnel, des scénarios de qualité ont été prédéfinis, et c'est principalement la définition des schémas d'interactions appropriés entre les éléments de l'architecture iDyCoS qui ont permis de concrétiser ces scénarios (cf. chapitre 3).

Scénario de qualité	Réponse attendue	Comportement implémenté
EQ1 : Demande d'un service complexe pour la première fois	Le service composite doit être sauvegardé pour une utilisation ultérieure, le système doit apprendre des expériences précédentes et améliorer ses performances lorsque des requêtes similaires sont soumissionnées	Au niveau de la dernière phase de composition, l'agent Executer sauvegarde le plan de composition sélectionné pour une éventuelle réutilisation
EQ2 : Demande d'un service simple	le système doit détecter qu'il s'agit d'une requête simple et n'effectuer que les étapes nécessaires pour résoudre la requête	Avant d'entamer l'opération de filtrage des services candidats, l'agent Discover vérifie d'abord si la phase de planification est nécessaire ou non, en vérifiant

		avant tout calcul de correspondance si un service existant est capable de satisfaire la requête, si c'est le cas, l'agent Evaluator est directement invoqué
EQ3 : Demande d'un service simple ou complexe	Les étapes de la composition doivent produire un service composite qui respecte les caractéristiques intrinsèques d'un service Web	Le modèle de planification utilisé, l'annotation sémantique utilisée (OWL-S), ainsi que les adaptateurs conçus garantissant le passage de la planification à la sémantique et vice versa, permettent de fournir ensemble les éléments nécessaires pour rendre le produit de la composition un service Web sémantique, utilisable et réutilisable
EQ4 : Préparation de la composition	En se basant sur les services Web disponibles (simples ou composés) le système doit chercher sémantiquement les services élémentaires potentiels	En s'appuyant sur l'algorithme de Matchmaking l'agent Discover scrute sémantiquement l'annuaire des SWS, alimenté constamment par les offres simples des fournisseurs et par les produits des cycles de composition antérieurs, pour trouver les services candidats. La recherche se base sur des paramètres sémantiques et du coup elle ne fait pas de différence entre le traitement d'un service simple ou composite, le seul critère pris en considération lors du filtrage est le potentiel du service à résoudre complétement ou partiellement la requête sémantique
EQ5 : Exécution du plan de composition	Le système doit assurer la détection et la correction des fautes d'exécution d'une façon proactive et il doit s'adapter aux aléas du temps réel	L'auto-guérison est la mission de l'agent Executer dont la structure interne lui permet de surveiller l'exécution et de réagir dynamiquement en cas de problèmes
EQ6 : Aucune correspondance n'a été trouvée	Le système retourne un échec de la résolution de la requête actuelle et propose l'alternative fructueuse la plus proche de la requête demandée	L'échec se calcule le plutôt possible pour optimiser le cycle de la composition : D'abord, suite à l'opération de filtrage des services candidats le Discover vérifie si la liste résultante est vide si c'est le cas

		celui-ci prononce l'échec (cf. chapitre 4 –Mécanisme de découverte). Si tous les managers confirment ne plus pouvoir raffiner les plans partiels, l'agent Planner arrête la planification sur un état d'échec (cf. chapitre 4 –Modèle de planification). Ce n'est qu'après l'aboutissement de la composition abstraite que l'Evaluator puisse procéder à la liaison des types services avec leurs instances. A ce niveau, le calcule de correspondance se base sur des paramètres non fonctionnels et le résultat renvoyé est celui le plus proche possible des attentes de l'utilisateur (cf. chapitre 5 – Stratégie d'évaluation).

Tableau 8. 2: Conformité aux exigences de qualité

Contrairement aux spécifications fonctionnelles dont la conformité se caractérise par son aspect binaire (Oui/Non), la conformité par rapport aux spécifications non fonctionnelles représente plutôt une satisfaction à maximiser. Selon les éléments présentés dans le tableau ci-avant le niveau de satisfaction actuel semble correct, la maximisation peut éventuellement porter sur le premier scénario de qualité en renforçant l'apprentissage des expériences antérieures. Cet aspect ouvre la discussion sur les pistes d'extensions de l'approche qui sont détaillées à la fin de ce chapitre.

8.3. Analyse comparative

La section précédente s'attache à démonter la conformité de ce qui est réellement conçu par rapport à ce qui est initialement perçu. Alors que la présente partie tente d'évaluer notre proposition en termes de perception et de conception en la confrontant à d'autres approches. Pour ce faire, nous utilisons une technique d'analyse comparative qualitative [Hofstee, 2006] qui consiste à positionner l'approche de composition proposée par rapport aux travaux similaires afin de noter qualitativement les avantages et les limites de la solution proposée par rapport aux solutions connexes.

Pour permettre une telle étude, une analyse exploratoire des déterminants de la qualité d'un modèle de composition dynamique et automatique des services Web est nécessaire. Cette section propose donc de définir d'abord les critères d'évaluation qualitatifs avant de présenter l'analyse comparative élaborée.

8.3.1. Identification des critères d'évaluation

En se basant sur une étude extensive des travaux les plus représentatifs qui tentent d'évaluer les approches de composition de services Web [Rao et al., 2004], [Khadka et al., 2004], [Dustdar et al., 2005], [Kauster et al., 2005], [Bucchiarone et al., 2006], [Beek et al., 2007], [Agarwal et al.,2008], [Pessoa et al., 2008], [H.M. Rusli et al., 2008], [Zuniga et al., 2010] , [Bartalos et al. , 2011], [Wang, 2013], [Lagares et al., 2016] nous proposons la liste des critères d'évaluation ci-après :

- **Aspect automatique**

Puisque nous nous intéressons à la composition automatique des services, le premier critère serait que la génération du schéma de composition doit être au moins partiellement (si ce n'est pas entièrement) automatisée. L'automatisation permet principalement de réduire le temps de génération du plan de composition, d'éliminer les erreurs humaines et de minimiser le coût du processus global. Aussi, nous devrions nous attendre à ce qu'une approche de composition de service Web réussie fournisse le niveau le plus élevé possible d'automatisation.

- **Aspect dynamique**

Une des caractéristiques définissant une approche de composition des services est la nature dynamique du schéma de composition produit. Une composition dynamique résiste (et s'adapte) mieux aux changements du temps réel qui rendent les schémas statiques de composition invalides, incohérents et impossibles.

- **Sémantique**

L'aspect sémantique est aussi un critère important à prendre en considération dans l'étude des modèles de composition. Comme il a été discuté dans les chapitres précédents, la couche sémantique permet aux services de prendre conscience les uns des autres et d'interagir sans intervention humaine, elle permet également d'augmenter le niveau de pertinence du choix des services participant à la réalisation du but demandé par l'utilisateur.

- **Support de QoS (QoS awareness)**

Un autre déterminant de la qualité d'une composition des services est la prise en charge des QoS. En considérant l'aspect non fonctionnel de la demande (requête) et de l'offre (les services), l'approche de composition s'engage non seulement à répondre aux fonctionnalités demandées mais elle garantit également la meilleure qualité possible.

- **Scalabilité**

Une autre exigence apportée par les applications industrielles est la scalabilité (la montée en charge), le fait qu'une approche fonctionne bien avec un ensemble donné de services ne garantit en rien le bon fonctionnement de celle-ci avec un autre ensemble de services plus large et plus complexe. Les modèles « sacalables » assurent les mécanismes nécessaires pour que le facteur taille n'impacte pas l'approche de composition. Lorsqu'on parle « scalabilité » un autre défi se pose, c'est le compromis entre la scalabilité et la performance. Pour une composition optimale, il est important de garder le niveau de performance stable lorsqu'on tente de maximiser la scalabilité d'une approche donnée et vice versa.

- **Indépendance du domaine**

Une approche de composition ne doit pas être limitée à un domaine spécifique. L'indépendance de la solution de composition par rapport au domaine traité, permet de celle-ci d'être utilisée pour résoudre un large éventail de problèmes.

- **Tolérance aux pannes**

Un des critères de distinction entre les modèles de composition des services est celui de la tolérance aux pannes (erreurs). On ne peut parler de ce critère que si le critère relatif à l'aspect dynamique est satisfait. En effet, un schéma de composition statique conçu en design time ne peut pas gérer les erreurs rencontrées en Runtime. Pour les approches dynamiques assurant une tolérance aux pannes, il existe une variation dans la manière de prise en charge des incohérences (proactive, réactive) qui impacte considérablement la qualité globale de la composition.

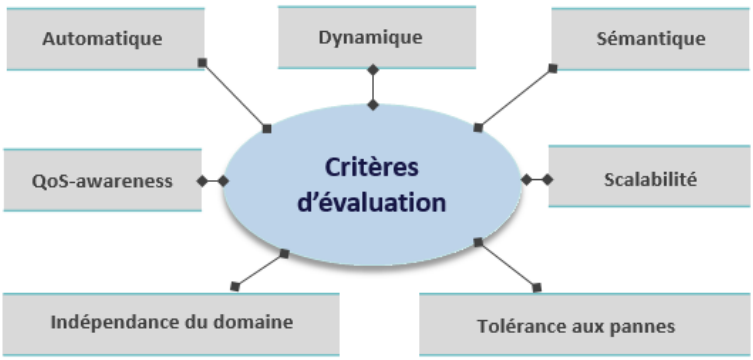


Fig.8. 1: Critères d'évaluation de l'analyse comparative

Les critères identifiés (Fig.8. 1) sont utilisés dans la suite de l'analyse pour examiner les approches de CSW. Le Tableau 8.3 présente la sémantique de symboles utilisée pour évaluer les approches de composition par rapport aux critères identifiés.

Symboles	+	+/-	-	?
Interprétations	Répond bien au critère	Répond partiellement au critère	Ne répond pas au critère	Inconnu

Tableau 8. 3: Sémantique des symboles utilisés dans l'analyse comparative

8.3.2. Comparaison qualitative

L'étude littéraire des travaux partageant des finalités similaires aux nôtres, nous a permis de repérer les travaux les plus significatifs qui nous aideront à positionner notre approche sur la sphère de recherche portant sur la composition dynamique et automatique des services Web. Une brève présentation de ces travaux, dont certains ont été introduits lors de l'étude de l'art présentée au niveau du chapitre 1, est donnée ci-après accompagnée de l'appréciation générale déduite.

Wu et al. [Wu et al., 2005] utilise un planificateur de type SHOP2 [Nau et al.,2003], un système de planification basé sur le modèle des Réseaux de Tâches Hiérarchiques (HTN) pour automatiser la composition des services Web décrit en langage DAML-S. Leur système ne

supporte pas la recomposition dynamique, ne découvre pas automatiquement les services et il n'est pas en mesure de repérer des erreurs de la composition en temps réel.

McIlraith et al. [McIlraith, 2005] utilise une version étendue de Golog [Levesque et al., 1997], un langage de programmation de haut niveau construit en dessus de la logique de calcul des situations qui permet de composer des services décrits en DAML-S. [Ankolenkar et al., 2002]. Les requêtes sont exprimées sous forme de Templates génériques de GonGolog [Giacomo et al., 2000] construites hors ligne qui sont ensuite complétées et adaptées en se basant sur les préférences de l'utilisateur. Ce travail manque de dynamique et ne permet pas une tolérance aux pannes.

Ponnekatni et al. propose SWORD [Ponnekanti et al., 2002] , une boite à outils pour la composition des services. SWORD utilise un système expert basé règles qui s'appuie sur l'algorithme de Rete [Forgy, 1982], il détermine automatiquement si le service demandé peut être produit à travers la composition de services existants prédéfinis. SWORD présente plusieurs limites, puisqu'il ne supporte pas la découverte automatique, le non déterminisme et ne prend pas en charge les aspects relatifs au QoS et à la tolérance aux pannes.

Berardi et al. [Berardi et al., 2005] considèrent le service Web comme un arbre de toutes les interactions possibles avec l'utilisateur et développent un composeur e-service (ESC). Ils utilisent le calcul de situation pour automatiser la composition. ESC ne permet pas lui aussi de gérer les erreurs en temps réel et ne traite pas le volet non fonctionnel de la requête.

Akkiraju et al. [Akkiraju et al., 2004] divisent leur architecture de composition en deux couches, la première se concentre sur la spécification abstraite du flux de processus d'affaires, tandis que la deuxième gère la partie découverte, composition et exécution. Le système n'offre lui aussi aucun mécanisme pour la tolérance aux pannes.

Pistore et al. [Pistore et al., 2004] proposent un Framework de composition basé sur un planificateur qui utilise le modèle de vérification (checking model), connu sous le nom de Model-Based Planning (MBP). Dans le cadre de ce travail un outil nommé ASTRO [Trainotti et al., 2000] a été développé qui supporte la composition, la vérification et l'exécution automatique des services. L'outil ne supporte, cependant, pas une tolérance dynamique aux pannes et ne prend pas en charge l'aspect non fonctionnel des services.

Les conclusions ci-avant sont récapitulées dans le tableau synthétique ci-après :

Travaux étudiés / Critères	**iDyCoS**	**Wu**	**mcIliarth**	**SWORD**	**ESC**	**Akkrijau**	**ASTRO**
Méthode(s) de base utilisée(s)	H4F	SHOP2	Golog	Algo de Rete	Situation calculus	State planner	MBP
Composition automatique	+	-/+	+	+	+	+	+
Composition dynamique	+	+	+	-/+	-/+	+	-/+
Langage de description des	OWL-S	DAML-S	DAML-S	XML	WSTL	OWL-S	BPEL4WS

services (Sémantique)							
Support de Qos	+	-	-	-	-	-	-
Scalibilité	-/+	-/+	-/+	-/+	-	-/+	+
Indépendance du domaine	+	-/+	-	-/+	+	+	+
Tolérance aux pannes	+	-	-	-	-	-	-

Tableau 8. 4: Analyse comparative

Les caractéristiques communes aux approches étudiées sont : des architectures centralisées guidées par des techniques d'intelligence artificielle, une génération (partiellement) automatique des schémas de composition et une prise en charge de l'aspect dynamique dans les mécanismes de composition. Toutefois elles présentent toutes des limites sur les aspects qui forment les points forts de notre approche, à savoir, l'auto-guérison (tolérance aux pannes) et la sélection basée sur les paramètres de QoS.

Il faut noter, par ailleurs, que la majorité des approches analysées ne priorisent pas la scalabilité. Un facteur dont notre approche tente d'améliorer en adoptant le style de composition par phases à deux niveaux, la réflexion sur l'optimisation de la faculté de montée en charge sans impacter la performance, est un autre élément qui alimentera la discussion autour de l'amélioration de l'approche iDyCoS ci-après.

8.4. Discussion

L'objectif de la discussion menée au niveau de cette section est de mettre sur table des propositions initiales pour prendre en charge les points à améliorer identifiés lors de l'évaluation de l'approche. En effet, même s'elles ne concernent que les aspects qualitatives, la vérification des exigences et l'étude comparative ont permis de dégager principalement deux paramètres à optimiser, il s'agit de la montée en charge (scalabilité) et de l'apprentissage des expériences antérieures.

L'approche telle qu'elle est proposée actuellement, attribue à l'agent Executer la mission de sauvegarde des plans de composition produits lors des cycles de composition pour une réutilisation ultérieure, elle s'appuie également sur l'ontologie de QoS et de profil utilisateurs pour une meilleure adaptabilité des services composites résultants. L'adaptabilité pourrait, toutefois, être agrémentée en utilisant un mécanisme d'apprentissage automatique des expériences antérieures.

Par ailleurs, la composition par phases à deux niveaux a été introduite comme une solution architecturale pour faire face au problème de la scalabilité. Au niveau de la présente discussion nous explorons une autre piste pour améliorer la montée en charge qui s'intéresse à la nature même des services Web plutôt qu'à la structure organisant ces services Web.

La discussion sera donc divisée en deux sujets et elle portera sur la possibilité d'intégration d'un mécanisme d'apprentissage automatique et d'étude de la composition d'autres types de services Web.

8.4.1. Utilisation de l'apprentissage automatique

Identifié comme l'une des pistes d'amélioration les plus prometteuses de l'approche de CSW proposée, l'apprentissage des expériences de composition antérieures fait appel à l'une des techniques d'intelligence artificielle la plus célèbre qui traite cet aspect, il s'agit de l'apprentissage automatique (Machine Learning) [Smola et al., 2008].

Apprendre est l'un des comportements le plus distinctif d'un être humain. Ce processus cognitif essentiel à l'adaptabilité, peut aujourd'hui être modélisé et reproduit artificiellement. L'apprentissage peut être appliqué à travers deux principales méthodes: l'apprentissage par analogie (memory-based learning) [Stanfill et al., 1986] et l'apprentissage par renforcement [Sutton et al., 1986].

Dans un contexte d'apprentissage par analogie, l'agent observe l'utilisateur en enregistrant tout ce qu'il fait en tant que couple situation-action. Ensuite, dans une situation donnée, l'agent cherche la situation enregistrée la plus proche et peut ainsi prédire l'action à effectuer. L'apprentissage par renforcement intervient dans le cas où la prédiction est incorrecte. Dans le cas d'une mauvaise prévision, l'agent explique à l'utilisateur la raison de son choix (la situation similaire dans la mémoire de l'agent) et lui offre la possibilité d'expliquer en quoi le cas courant diffère de la situation précédemment rencontrée. Par exemple, l'utilisateur peut avoir accepté une réunion sur un sujet pour lequel il avait précédemment refusé une réunion parce que cette fois l'initiateur est son patron. Si l'agent maintient un ensemble de poids associés aux initiateurs, participants et sujets des réunions, le poids du patron dans la liste des initiateurs sera alors augmenté d'une quantité, suite à ce feedback. C'est ainsi l'agent apprend à partir de son expérience.

L'apprentissage par renforcement est défini formellement comme étant une approche informatique de l'apprentissage dans laquelle un agent essaie de maximiser le montant total de la récompense qu'il reçoit en interagissant avec un environnement complexe et incertain [Sutton et al., 1986]. L'objectif de l'agent est d'apprendre, à partir d'expériences, ce qu'il convient de faire dans différentes situations. Les récompenses numériques reçues pour les essais sont utilisées dans ce but. Plusieurs algorithmes existent pour effectuer cet apprentissage ; ils se basent tous sur une représentation du problème sous forme d'un processus décisionnel de Markov [Barry et al., 2003]. L'utilisation de cette méthode d'apprentissage a connu beaucoup de succès dans le domaine de la robotique et des jeux.

Nous proposons dans le cadre de la présente discussion, d'analyser l'opportunité de combiner notre approche de composition avec cette méthode d'apprentissage en vue d'améliorer la qualité du résultat de la composition. Conceptuellement, nous proposons d'introduire une couche supplémentaire « Consulting » qui étend les capacités des modules de composition existants, qui va guider le système vers l'exécution optimale du service composite en se basant sur le feedback de l'utilisateur.

Notons que d'un point de vue apprentissage automatique, notre approche de composition est considérée une technique de composition symbolique [Todica et al., 2012], puisqu'elle s'appuie sur des données explicites. L'objectif des approches de composition orientées

apprentissage automatique est de remplacer les approches symboliques. Nous croyons, au contraire, que les deux approches sont davantage complémentaires que substituables. En effet, comme illustré dans la Fig.8. 2, nous envisageons une vision conceptuelle dans laquelle l'approche de composition actuelle sera responsable sur la production d'un service composite conforme aux spécifications fonctionnelles et non fonctionnelles. Quant à la couche « Consulting », elle aidera l'approche symbolique, en particulier l'agent Evaluateur, à décider sur la meilleure exécution en se basant sur sa technique d'évaluation qui maximise la qualité de service d'un point de vue utilisateur, mais aussi sur les informations implicites fournies par le Consultant qui formalisent le retour de l'utilisateur lors des cycles de composition précédents. Notons que la même interaction peut être envisagée pour aider l'Executer à décider sur l'action à entreprendre en cas d'erreurs d'exécution pour améliorer son aspect autonomique, comme précédemment énoncé au niveau du chapitre 5 (cf. Chapitre 5- Exécution autonomique).

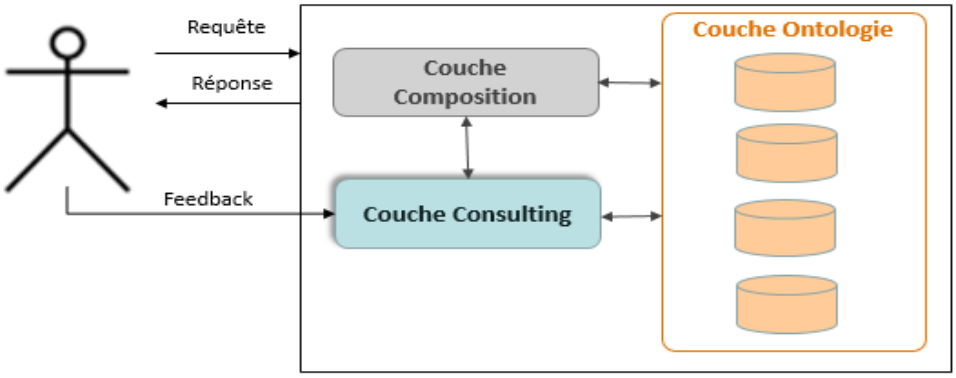


Fig.8. 2: Intégration de la couche « consulting » à l'approche proposée

L'objectif étant de proposer des réponses aux enjeux identifiés, la proposition discutée dans cette section se limite à une vision conceptuelle qui nécessitera évidement davantage d'éclaircissement, de rehaussement et éventuellement plus d'optimisation pour être mise en place. Notamment en termes de définition détaillée du schéma d'interactions et du partage de ressources entre l'approche symbolique et la couche « Consulting » (il conviendra, par exemple, de spécifier l'éventuel effet entrainés par la conjonction de l'ontologie de profil utilisateur et la nouvelle couche).

8.4.2. Composition des services REST

Dans notre contexte, le service Web est considéré comme une entité abstraite qui consomme des entrées, produit des sorties et s'associe à des pre-/post-conditions. Deux classes de services répondent à cette définition : les services SOAP de style RPC et les services REST [Fielding ,2000]. Même si les deux types de services peuvent être associés à des paramètres IOPEs, il existe une différence fondamentale entre les deux classes de services. Les services Web SOAP sont considérés comme des méthodes distribuées invocables à distance qui communiquent par des messages. Alors que dans un contexte REST, on gère plutôt des ressources distribuées sur lesquelles des opérations « http » sont exécutées.

En considérant la problématique de CSW, les travaux de recherche dans ce domaine sont presque systématiquement orientés vers les services SOAP, rares sont les travaux qui ont discuté la composition des services REST [Zhao et al., 2009]. Notons, à ce niveau, que même si le présent travail traite techniquement des services SOAP, il est important de noter que l'approche proposée est valable, au niveau conceptuel, pour le concept de service Web quelle que soit sa classe.

Toutefois, l'analyse des approches architecturelles de la CSW qui traitent les aspects de performance et de la scalabilité, montre que les principaux mécanismes conçus ou utilisés dans le cadre de ces approches s'inspirent du style REST des services. En effet, les principes du REST incluent :

(1) les entités conceptuelles et les fonctionnalités sont modélisées par des ressources identifiées par des URIs.
(2) les ressources sont accessibles et manipulables à travers des opérations http (GET, POST, PUT et DELETE).
(3) les composants du système communiquent via ces interfaces standards d'opérations et échangent des représentations de ces ressources (une ressource donnée peut avoir plusieurs représentations). Dans un système REST, le serveur et le client passent typiquement par plusieurs états de représentation de ressources en suivant les interrelations entre les ressources.

Ainsi, contrairement à la perception centrée opération des services SOAP, les services REST voient les applications d'un point de vue centré ressources, les atouts suivants sont les conséquences directe de cette perception :

- **Légèreté**: les services REST utilisent directement « http » comme protocole d'appel sans avoir besoin d'utiliser XML. La réponse est une représentation de la ressource et ne nécessite aucune encapsulation supplémentaire ou enveloppe. C'est pour cette raison que les services REST sont considérés plus faciles à développer et à consumer et délivrent une meilleure performance grâce à leur légèreté en comparaison avec les services SOAP.
- **Accessibilité:** les URIs sont utilisés pour identifier les services REST, ce qui facilite l'accessibilité à ce type de services. Ceci justifie le constat que, de plus en plus, ce sont les services REST qui sont utilisés pour construire des applications Web 2.0 et des Mashups.
- **Scalabilité:** REST fournit un ensemble de contraintes d'architecture qui, appliquées comme un tout, met en exergue la montée en charge des interactions de composants. Les contraintes forment un modèle «client/serveur sans état en couches avec cache». Ce modèle implique que la communication doit être par nature sans état, la possibilité de montée en charge est meilleure, car le fait de ne pas stocker l'état entre les requêtes permet aux composants du serveur de libérer rapidement les ressources. En plus le système en couches permet l'instauration des intermédiaires pour améliorer la montée en charge du système en offrant un équilibrage de charges pour les services. Finalement la contrainte de cache impose que les données d'une réponse à une requête peuvent être mises ou non en cache. Si une réponse est mise en cache, alors le cache du client obtient le droit de réutiliser ces données de réponse pour des demandes ultérieures équivalentes. Ceci a

l'avantage d'offrir la possibilité d'éliminer toutes ou une partie de certaines interactions, améliorant ainsi l'efficacité, la montée en charge et la perception qu'ont les utilisateurs de la performance et ce, en réduisant la latence moyenne dans une série d'interactions [Fielding ,2000].

- **Approche déclarative:** à contrario des services SOAP impératifs (orientés opérations), les services REST adoptent une approche déclarative qui s'intéresse à la description de la ressource elle-même en faisant abstraction de la façon avec laquelle les fonctions sont exécutées. Il est généralement reconnu et accepté que les approches de type déclaratif sont le meilleur choix pour la construction des systèmes flexibles et scalables.

Les caractéristiques précitées des services REST montrent bien que les services REST possèdent des avantages uniques et encourageants par rapport aux services traditionnels, en particulier en termes de scalabilité et de flexibilité. Pourtant, rares sont les travaux qui s'adressent à la composition de ce type de services [Zhao et al., 2009]. En effet, toute tentative d'étude de la composition dynamique des services REST est bloquée par des questions fondamentales qui nécessitent des réponses claires avant d'entretenir la composition des services de cette classe.

Nous soulignons ci-après les trois défis principaux auxquels la CSW REST est confrontée :

- la vision orientée ressources pour construire des services est relativement nouvelle, et la majorité des services existants n'adhèrent pas aux principes REST;
- le manque d'un langage sémantique (compris par la machine) pour décrire les services REST;
- pour intégrer des services REST (ressources) provenant de différents fournisseurs l'hétérogénéité des données peut représenter un obstacle majeur.

Nous retenons alors le potentiel des services REST comme concept, la composition de ce type de services est par contre contrainte d'être temporisée au risque de s'affronter à des contraintes de faisabilité techniques dû à la jeunesse de cette technologie. Il serait intéressant d'une vision futuriste d'étudier l'automatisation du processus de composition dans un environnement ressemblant les deux classes de services SOPA et REST.

8.5. Conclusion

L'emphase au niveau de ce chapitre est placée sur le volet d'évaluation qui a été détaillé au travers deux analyses; d'abord la vérification des exigences fonctionnelles et de qualité énoncées au début de ce manuscrit et ensuite la comparaison de l'approche proposée avec des travaux de recherche similaires en se basant sur des critères d'évaluation sélectionnées suite à une étude littéraire extensive.

Selon l'évaluation qualitative élaborée, l'approche proposée se démarque par la prise en charge des aspects de qualité de services et d'auto guérison, elle nécessite, toutefois, d'améliorer les aspects relatifs à la scalabilité et l'apprentissage des expériences antérieures. Les deux points d'amélioration ont été discutés à la fin du chapitre en introduisant deux nouveaux concepts prometteurs : les services REST et l'apprentissage automatique.

Conclusion Générale

To make an end is to make a beginning.
T. S. Eliot

Ce manuscrit conclut avec un récapitulatif des contributions avancées par cette thèse et du travail réalisé dans son ensemble, suivi des perspectives à court et à long terme que nous avons identifiées.

Synthèse des travaux

« Avoir un support conceptuel et technique qui sert à la composition AUTOMATIQUE et DYNAMIQUE de services Web», tel était le besoin pour lequel ce travail a été initié. Dans la quête d'atteindre ce but nous étions amenés à explorer plusieurs thématiques de recherches afin de repérer les acteurs adéquats qui animeront le processus de composition et d'identifier les synergies potentielles entre eux permettant de parvenir au niveau automatique et dynamique espéré. Cette démarche nous a conduits naturellement à proposer un modèle conceptuel sous forme d'architecture qui organise la contribution de ces acteurs issus de domaines différents, tout en définissant les mécanismes nécessaires à l'harmonisation de leurs interactions. Pour consolider l'édifice, des techniques de composition ont été conçues dans le but d'expliciter les aspects qui guident et distinguent notre approche.
En trois parties, ce manuscrit transcrit notre vision concernant la problématique traitée. Il introduit, explique et évalue la solution proposée d'une façon incrémentale et itérative:

I. L'approche proposée, qui se veut intelligente, s'appuie sur des concepts empruntés à des domaines très divers allant de l'intelligence artificielle jusqu'aux systèmes distribués. En effet, notre contribution repose sur l'hypothèse fondamentale stipulant que la synergie entre quatre forces permet de dégager l'aspect intelligent souhaité (H4F). Au travers une étude détaillée de chacune des quatre forces, la première partie de ce manuscrit, argue que :

    - le **Web sémantique** est la technologie adéquate pour assurer une composition automatique en se basant sur les descriptions formelles des ontologies ;
    - un **système multi-agents** dans lequel des agents raisonnent en utilisant des **algorithmes de planification,** est le cadre architecturale approprié pour composer dans un environnement dynamique et décentralisé ;
    - l'adoption et l'adaptation du paradigme de **l'Autonomic Computing** est nécessaire pour assurer une sûreté de fonctionnement du processus de composition.

II. Partant d'une conception générale, l'architecture proposée, dénommée iDyCoS, a été incrémentalement construite en faisant appel aux concepts clés de l'H4F. Des exigences fonctionnelles et de qualité ont été définies pour guider la distribution des rôles et la conception des interactions entre les composants de l'architecture. Ces exigences ont également provoqué l'introduction du style architectural distinguant notre approche, il

s'agit de la composition par phases à deux niveaux, dont les bienfaits ont été constatés et confirmés au fur et à mesure de l'avancement du travail exposé.

Le volet contribution ne se limite pas à l'instauration du cadre conceptuel de la composition dynamique et automatique de services Web, il introduit également les techniques de composition nécessaires pour que chaque phase puisse accomplir sa mission. Elles sont au nombre de cinq, ensemble elles rythment le processus de composition conçu :

- Au cœur du premier niveau de la composition : la composition abstraite, se trouve un modèle de planification entièrement distribué dans lequel les agents raisonnent de manière hypothétique sur leurs activités, en intégrant leurs compétences hétérogènes et leurs croyances partielles sur le monde. Il s'agit d'une méthode de planification, nommée synthèse dialectique des plans, qui a été adaptée et transposée au contexte de services Web. Une technique de traduction et une autre de filtrage ont été également introduites afin de préparer (optimiser) la phase de construction du plan abstrait.
- Au niveau de la composition concrète, les instances des services Web sont examinées sur la base de leurs QoS pour choisir les meilleures, il s'agit de la technique «Stratégie d'évaluation ». La dernière technique proposée : « Exécution autonomique » a pour mission de rendre l'exécution un mécanisme auto- guérissant, en faisant ainsi, elle renforce l'aspect autonomique du processus de composition global.

Dans cette partie, nous avons tenu à respecter une certaine structure dans la description des différentes techniques de composition afin de garantir la cohérence des explications fournies. Ainsi, chaque technique conclut avec une sorte d'évaluation marquant ses points forts et ses points d'amélioration, ensuite elle est appliquée à un cas d'utilisation de planification de voyage, qui s'est développé au fil de l'exposition des techniques. En effet, l'exemple illustratif de voyage a constitué de par ses caractéristiques intrinsèques un véritable support à la démonstration des différentes techniques décrites dans cette partie.

III. Au niveau de la dernière partie, plusieurs pistes ont été empruntées afin d'étayer la validation de nos contributions. D'abord nous avons discuté comment l'approche proposée peut-elle contribuer à la résolution de problématiques réelles comme celles relatives au domaine du gouvernement électronique. Ensuite, une preuve de concept couvrant un périmètre significatif de l'architecture iDyCoS a été mise en place pour aborder la validité de l'approche sur le plan technique. Enfin, une évaluation à caractère qualitatif a pu dégager les caractéristiques distinctives de l'approche et instaurer une feuille de route vers le perfectionnement de nos contributions.

Partageant l'objectif de la composition dynamique et automatique avec plusieurs autres travaux de recherche, nous estimons que notre approche se démarque d'abord par sa perception globale et holistique de la problématique. Nous visons une solution de « bout en bout » qui ne restreint pas la composition à l'une de ses sous-problématiques, les mécanismes

proposés tâchent alors de définir un processus complet de composition depuis la demande de l'utilisateur jusqu'à sa satisfaction. Il est également important de souligner que notre proposition ne rompt pas avec les approches déjà existantes, bien au contraire, nos contributions sont à même de compléter efficacement ces approches traditionnelles pour améliorer la capacité dynamique et la capacité automatique. Nous avons ainsi exploité des concepts phares qui ont prouvé leur efficacité dans ce sens (comme le Web sémantique et les algorithmes de planification) tout en se basant sur le principe de composition comme finalité et outil. Finalement, notre approche se distingue par les aspects qu'elle développe pour répondre aux besoins d'automatisation et de la dynamique, les mécanismes présentés tendent tous vers la concrétisation de trois principaux aspects autour desquels s'articule notre démarche: l'autonomie (autonomique), l'adaptation et l'intelligence.

- **Composition autonomique :** que ce soit au niveau du mécanisme de la découverte, de la planification, de l'évaluation ou de l'exécution on retrouve toujours l'une des quatre caractéristiques d'un système autonomique vérifiée, le processus dans sa globalité jouit alors d'un niveau important d'autonomique.
- **Composition adaptative :** à partir du moment où l'approche proposée s'adapte dynamiquement au domaine, à l'utilisateur (son contexte et ses préférences) et aux changements en temps réel lors de l'exécution, alors elle peut être également considérée adaptative**.**
- **Composition intelligente :** Puisque nos contributions sur la composition autonomique et adaptative de services, prennent tout leur sens lorsqu'elles sont déployées conjointement au sein d'une même approche. La contribution intelligente se présente alors comme la « résultante » des deux précédentes. Il s'agit, en outre, de la manipulation habile de composants connus par leurs capacités de générer un comportement « intelligent » comme les agents intelligents, les algorithmes de planification et la boucle intelligente (MAPE).

Il reste cependant de nombreuses pistes à explorer au-delà de ce travail pour maximiser ces trois caractéristiques extrêmement prometteuses.

Perspectives

A court terme, différentes perspectives de travail, pour lesquelles des réflexions préliminaires ont été menées et exposées au cours de ce manuscrit, s'offrent à nous :

- Comme indiqué au niveau du chapitre 4, l'aspect relatif à la contextualisation des besoins de l'utilisateur selon son profil est à évoluer. Dans ce sens l'effort doit s'orienter, dans une prochaine étape, vers la conception d'une ontologie de profil utilisateur et la définition des mécanismes d'extraction des données pour alimenter cette ontologie;
- Sur le plan technique, le but est d'étendre la validation expérimentale de nos contributions, en élargissant le périmètre du prototype pour qu'il couvre l'ensemble du processus de la composition abstraite. Notons que l'optimisation du

moteur de planification mis en place est toujours en cours selon les recommandations de la dernière section du chapitre 7, c'est principalement la qualité modulaire de l'architecture qui fait en sorte que la construction incrémentale et même parallèle du prototype soit possible.

- Il convient également d'étudier d'autres scénarios de prestation des services d'e-gouvernement et de procéder à leur test et implémentation afin de soutenir notre proposition qui présente l'approche iDyCoS comme une solution potentielle pour la résolution des problématiques de l'intégration et d'hétérogénéité liées à ce domaine. Evidemment, cela n'empêche pas de s'ouvrir, à long terme, vers d'autres domaines d'application.

Par ailleurs les perspectives présentées ci-avant, des perspectives plus larges, qui s'inscrivent dans la prolongation de nos contributions, mais pour lesquelles une étude plus approfondie doit être menée, sont aussi envisageables. Elles ouvrent, nous l'espérons, de nouveaux angles de recherche. Il s'agit principalement des deux pistes de recherches représentant la synthèse de l'évaluation de l'approche proposée et discutée en détail au niveau du dernier chapitre. La finalité de ces perspectives d'amélioration est de fortifier la caractéristique « intelligente » de notre approche en la faisant évoluer sur les deux axes « adaptation » et «autonomique». Nous rappelons ci-après les deux extensions suggérées :

a) l'étude de la composition des services REST et l'analyse de l'impact sur la performance du processus de composition.
b) l'étude de l'intégration des mécanismes d'apprentissage automatique, en particulier les algorithmes de l'apprentissage par renforcement, et l'analyse des retombés positifs sur le niveau d'adaptation de l'approche.

Annexe A : Principaux planificateurs

Il existe une multitude d'algorithmes qui s'adresse au problème de planification, nous présentons ci-dessous, les principaux planificateurs (algorithmes de planification) qui sont utilisés pour la résolution du problème de CSW, ainsi que les algorithmes y afférents. Cette étude est importante pour choisir l'algorithme de planification le plus adapté à notre approche.

A.1. Planification classique : Espace d'états et espace de plans

Les algorithmes de planification classique peuvent être regroupés en deux grandes familles. D'un côté, les algorithmes de planification dans un espace d'états (state space planning) qui sont les algorithmes les plus simples. De l'autre côté, les algorithmes de planification dans un espace de plans (plan space planning).

A.1.1 Planification dans un espace d'états

Les algorithmes de planification dans un espace d'états sont des algorithmes de recherche dans lesquels l'espace de recherche est un sous-ensemble de l'espace d'états : chaque nœud correspond à un état du monde, chaque arc correspond à une transition entre deux états, et le plan courant correspond au chemin courant dans l'espace de recherche. Les principaux algorithmes de planification dans un espace d'états sont la recherche par chainage avant, la recherche par chainage arrière et l'algorithme STRIPS [Fikes et al. , 1971] :

- la recherche par chainage avant : partant de l'état initial, l'objectif est de trouver un état qui satisfait le but. L'algorithme prend en entrée un problème P. Si P est résolvable, alors l'algorithme retourne un plan solution. Le plan retourné à chaque invocation récursive est appelé plan partiel car c'est une partie du plan solution qui sera retourné par l'algorithme ;
- la recherche par chainage arrière : l'idée est de partir du but et d'appliquer l'inverse des opérateurs de planification pour produire des sous-buts jusqu'à arriver à l'état initial. Si l'algorithme permet de remonter à cet état initial, alors il retourne le plan solution trouvé ;
- l'algorithme STRIPS : le principal inconvénient des algorithmes de planification dans un espace d'états est la taille de l'espace de recherche. L'algorithme STRIPS fonctionne de la même façon que l'algorithme de recherche par recherche arrière, mais son avantage réside dans sa capacité à réduire l'espace de recherche. Cette optimisation est caractérisée par les deux points suivants :
  - à chaque appel récursif de l'algorithme, les sous-buts qui doivent être satisfaits sont les préconditions du dernier opérateur ajouté dans le plan, ce qui a comme conséquence de réduire substanciellement le facteur de branchement de l'algorithme ;
  - si l'état courant satisfait toutes les préconditions d'un opérateur, alors STRIPS marque l'opérateur. Ainsi, en cas d'échec, le retour arrière se fera à partir de cet opérateur.

A.1.2 Planification dans un espace de plans

Les algorithmes de planification dans un espace de plans travaillent sur un espace de recherche plus sophistiqué : les nœuds représentent des plans partiellement spécifiés et les

arcs sont des opérations de raffinement de plans, i.e., qui permettent de réaliser un but ouvert ou d'enlever une inconsistance potentielle (par exemple lorsque deux actions sont en conflit). Une autre différence fondamentale concerne la définition du plan solution : la planification dans un espace de plans utilise une structure de plans plus générale que la séquence. Un plan est défini comme étant un ensemble d'opérateurs reliés par des contraintes d'ordre et des contraintes d'instanciation. Afin de limiter la complexité, la recherche dans un espace de plans peut être guidée par des heuristiques. Une des heuristiques les plus efficaces est de décomposer les tâches : c'est la planification hiérarchique.

Malgré qu'il soit les bases sur lesquelles sont construite la grande majorité des algorithmes de planification actuels, les algorithmes de planification classique souffrent de quelques hypothèses restrictives qui ont motivé l'émergence de d'autres types de planificateurs, ces limitations sont principalement: (1) Transitions instantanées, pas de durée, pas de parallélisme (2) Monde statique, pas de transitions contingentes (3) Connaissances complètes (4) Actions déterministes.

A.2. Problèmes de satisfaction de contraintes

Une avenue possible de la résolution d'un problème de planification est la traduction de ce dernier en un problème de satisfaction de contraintes (Constraint Satisfaction Problem, ou CSP). Un CSP consiste en la définition d'un problème par un ensemble de variables soumises à un domaine de valeurs possibles et à un ensemble de contraintes sur ces variables. Cette technique de résolution est efficace lorsque le problème de planification est borné, c'est-à-dire qu'il s'agit de trouver un plan de longueur déterminée. Parmi les travaux les plus significatifs traitant ce type d'algorithme, on trouve celui de [Kautz et al. , 1996] qui ont présenté une méthode de satisfiabilité booléenne (SAT) connue sous le nom de SATPLAN pour la résolution de problèmes de planification. un autre travail qui mérite également d'être cité est celui de [Do et al. , 2001] qui ont proposé le système GP-CSP, permettant de traduire un graphe de planification en problème de satisfaction de contraintes.

A.3. Réseaux hiérarchisés de tâches (Hierarchical Task Networks : HTN)

Une approche différente à la résolution de problèmes de planification est la décomposition du problème de manière hiérarchique. La planification hiérarchique [Erol, 1994] diffère des autres approches présentées jusqu'ici par le fait qu'elle ne considère pas les actions comme un ensemble fini d'éléments atomiques, mais plutôt en décrivant des actions de haut niveau (High Level Actions, ou HLA). Celles-ci correspondent à un ensemble d'affinements, consistant en des actions simples, dites actions primitives, ou à d'autres actions de haut niveau. Ceci permet d'établir une hiérarchie dans les actions et ainsi de décomposer le plan en une arborescence d'actions de deux types. Cette approche de planification permet l'introduction de concepts d'implémentations d'actions différentes des approches exploratives simples comme la récursivité, cependant la planification hiérarchique diffère des autres planificateurs sur la façon de planifier. Dans un planificateur HTN l'objectif n'est pas

d'atteindre un ensemble de buts (exemple : porte ouverte) mais de résoudre un ensemble de tâches (exemple : ouvrir porte).

A.4. Planification conditionnelle : sous incertitude (basé sur le modèle de contrôle, MDP)

Des incertitudes peuvent survenir à la conception du problème de planification, c'est-à- dire être issue d'une limitation dans la description du monde, où être propres à l'exécution, i.e., dépendre de l'application des actions. Les structures de contrôle que nous recherchons dans les plans ont du sens uniquement lorsque le monde est observable, au moins partiellement, à l'exécution. En effet, sans observation, il n'y a pas de choix. Aussi, les incertitudes dans la description des états, doivent pouvoir être résolues à l'exécution : les plans présents ainsi nécessairement des structures de contrôles selon les observations effectuées à l'exécution. Nous distinguons plusieurs types d'actions [Guere et al. , 1999] liés aux incertitudes discutées:

- les actions déterministes, qui n'ont qu'un seul comportement possible ;
- les actions conditionnelles pour lesquelles le contexte à l'exécution conditionne le comportement ;
- les actions non déterministes qui ont un ensemble de comportements possibles, indépendamment du contexte ;
- les actions de perception, ou senseurs, qui lèvent des incertitudes sur l'état du monde : les incertitudes sont alors modélisées comme une valeur inconnue d'une propriété du monde (par exemple son affirmation ou sa négation), et les senseurs, par perception du monde à l'exécution, donnent l'unique valeur de l'inconnue parmi toutes les valeurs possibles (les senseurs produisent de la connaissance).

Lorsque le domaine de planification contient l'une de ces incertitudes, c'est-à-dire lorsque les actions peuvent avoir plusieurs effets différents selon le contexte à l'exécution, alors les planificateurs doivent prendre en compte chacun des effets. Il est difficile d'être exhaustif en planification conditionnelle, car de nombreux travaux ont été mené pour pallier aux différentes incertitudes et avec des approches très différentes comme la planification partiellement ordonnée et la planification probabiliste qui utilise notamment POMDP (Partially Observable Markov Decision Process).

Annexe B : E-gouvernement: Etat des lieux

Au niveau de cette annexe nous dressons un état des lieux sur les travaux et les initiatives réalisées dans le domaine d'e-gouvernement à l'échelle national et international. L'objectif étant de prouver la pertinence du traitement de la problématique de l'intégration des services d'e-gouvernement pour le contexte marocain.

B.1. L'E-gouvernement dans le monde

Depuis quelques années, l'e-Gouvernement a pris de l'ampleur dans la plupart des pays, aujourd'hui les initiatives et les programmes e-gouvernement sont inscrits en tête des agendas des gouvernements des quatre coins du monde, tous stades de développement confondus.

Les résultats de l'enquête de l'e-gouvernement, réalisée tous les deux ans depuis 2003 par l'ONU [UN, 2016], sont particulièrement pertinents pour dresser le bilan de la situation de l'e-gouvernement à l'échelle international.

Dans son dernier rapport établi en 2016 [UN, 2016], l'enquête intitulée « e-government in support of sustainable development » présente l'e-gouvernement essentiellement comme un outil efficace pour soutenir l'implémentation de l'agenda 2030 de développement durable et de ses 17 objectifs [UN, 2015]. La même enquête révèle que l'e-gouvernement entre dans une nouvelle phase avec l'utilisation des médias sociaux et les mobiles par les gouvernements.

Ses résultats montrent, par ailleurs, que c'est le Royaume Uni, l'Australie, la Corée du Sud, Singapour et la Finlande qui occupent les cinq premiers rangs au niveau mondial du classement calculé à la base de plusieurs critères qui forment l'indice de développement de l'e-gouvernement (E-Governement Developement Index – EGDI).

E–Government Development Index - Top 10 Countries	
Country	*Index*
United Kingdom	0.9193
Australia	0.9143
Republic of Korea	0.8915
Singapore	0.8828
Finland	0.8817
Sweden	0.8704
Netherlands	0.8659
New Zealand	0.8653
Denmark	0.8510
France	0.8456

Fig.B. 1: Les leaders internationaux en matière d'e-gouvernement [UN, 2016]

On constate par ailleurs, que le premier pays africain au classement, l'île Maurice, se situe à la 58ème place mondiale qui a cependant gagné 17 places par rapport au précédent classement publié en 2014.

Suivent dans le classement, la Tunisie, l'Afrique du Sud, le Maroc et les Seychelles qui complètent le Top 5 africain et sont respectivement 72ème, 76ème, 85ème, et 86ème au niveau mondial.

Le Cap-Vert, l'Égypte, le Botswana, la Lybie et le Kenya occupent les 5 places suivantes du Top 10 africain.

Top 10 countries for e-government in Africa	
Country	*Rank*
Mauritius	58
Tunisia	72
South Africa	76
Morocco	85
Seychelles	86
Cape Verde	103
Egypt	108
Botswana	113
Libyan	118
Kenya	119

Fig.B. 2: Le classement des 10 premiers pays africains en matière d'e-gouvernement [UN, 2016]

Le rapport note, cependant, les grandes disparités en matière de l'e-gouvernement entre les régions et les pays en développement et ceux développés.

En effet, si l'e-gouvernement constitue un défi majeur dans les économies développées, c'est encore plus le cas dans les pays en développement, ces derniers rencontrent principalement des contraintes en termes d'infrastructures, de ressources financières et de compétences humaines.

Cette situation exige que l'on s'intéresse à l'application des modèles d'e-gouvernement dans le contexte des pays en développement et en particulier à des stratégies de développement où on tire parti le maximum possible du potentiel offert par les technologies électroniques dans le processus de développement.

Bien qu'impressionnante, la situation n'est pas totalement morose. Même si le secteur de l'e-gouvernement est beaucoup plus évolué dans les pays du premier monde, une remarque qui mérite d'être citée c'est que « les utilisations les plus innovantes de l'Internet en matière de gouvernance apparaissent dans le monde en développement » [InfoDev, 2002]. Par ailleurs, beaucoup de pays moyennement développés ont entrepris d'ambitieux programmes visant à instaurer l'e-gouvernement en tant qu'élément moteur dans leur développement national [Kettani ,2014] [Ben Chouikha, 2008].

Nous focalisons ci-après sur le cas du Maroc qui, selon le dernière enquête de l'ONU d'e-gouvernement, occupe la position de leader africain en matière d'e-participation et pointe à la 17ème place du classement mondiale [UN, 2016]. Cette place lui accorde une note de

performance «très élevée». Ainsi, parmi les 50 pays performeurs en 2016, le Maroc est classé au même rang que la Lituanie, le Monténégro et la Serbie.

Rank	Country
17	Morocco
17	Lithuania
17	Montenegro
17	Serbia

Fig.B. 3: Classement du Maroc parmi les pays leaders en matière d'e-participation [UN, 2016]

A.2. L'e-gouvernement au Maroc

Au Maroc comme ailleurs dans les pays en développement, le déploiement du gouvernement électronique constitue l'un des fondements du chantier de modernisation de l'Etat. Dans cette optique la stratégie Maroc Numeric 2013, lancée en 2009, a érigé le programme «e-Gouvernement» en tant qu'axe prioritaire. Cet axe a pour objectif de rapprocher l'administration des besoins du citoyen et de l'entreprise, en termes d'efficacité, de qualité et de transparence à travers un ambitieux programme, visant la mise en œuvre de 89 services et projets répartis en 3 catégories [Egov.ma, 2011]:

- Services améliorant l'efficience et réduisant les coûts de l'administration ;
- Services simplifiant les procédures administratives des entreprises ;
- Services mettant en place une administration au service des citoyens.

Pour servir cette vocation des instances de pilotage dédiées ont été mises en place afin de coordonner et harmoniser la réalisation de ce programme [Egov.ma, 2011].

Ainsi, plusieurs projets ont vu le jour dans le cadre de l'initiative marocaine de l'e-gouvernement, le tableau ci-après, représente un échantillon des services e-gouvernement en ligne illustrant la diversité et la multitude de ces projets :

Service	Type d'interaction géré	Description
watiqa.ma [Watiqa, 2016]	G2C	Le guichet électronique de commande de documents administratifs permet aux citoyens de commander leurs documents administratifs en minimisant voire supprimant le déplacement physique à l'administration
fikra.egov.ma [Fikra, 2016]	C2G	Le service d'e-Participation Fikra eGov permet aux citoyens de participer au programme eGov à travers la proposition de nouveaux services électroniques. Le service est ouvert à tous (citoyens, fonctionnaires, entreprises)
Service-Public [Service-public, 2016]	G2C	Service-Public.ma, portail public permettant de s'informer sur les procédures, démarches administratives et services e-gouvernement disponibles en ligne
Marchés publics [Marchespublics,	G2B	Plateforme de consultation et de soumission électroniques aux appels d'offres publics, permet

2016]		aussi bien aux entreprises qu'aux administrations de réaliser des gains dans le traitement et de renforcer la mise en concurrence et la transparence du processus de l'achat public
GID.gov.ma [GID, 2016]	G2G	GID est un système d'information budgétaire et comptable unifié et commun à l'ensemble des acteurs de la dépense
E-invest [Invest, 2016]	G2B	Un applicatif web dont la vocation est de renforcer la collaboration entre les départements impliqués dans l'instruction des projets d'investissement. En permettant aux investisseurs de s'informer en ligne sur l'avancement de leurs dossiers

Tableau B. 1: Exemple d'e-services marocains

La richesse et l'amélioration notable de la qualité des services proposés expliquent l'avancement réalisé sur l'index des Nations unies. En effet, comme mentionné auparavant, selon le dernier barème de la dématérialisation des Nations unies [UN, 2016], le Maroc s'est classé 85ème sur 193 pays avec un EGDI de 0.5186. Certes le Maroc a perdu trois places par rapport au dernier classement, publié en 2014, où il avait progressé de 38 places d'un coup, mais il est toujours considéré comme l'un des pays possédant un indice "élevé" de développement de l'e-gouvernement.

Il est à préciser que l'indice EGDI couvre trois indices clés : les services en ligne, les infrastructures de télécommunication et, enfin, le capital humain. Comme le montre le tableau ci-après construit à base des résultats des quatre dernières enquêtes des Nations unies en matière d'e-gouvernement [UN, 2016][UN, 2014][UN, 2012][UN, 2010]. Sur les deux premiers critères le Maroc est largement distancé. En revanche, les services en ligne est le critère qui progresse le plus vite.

	Indice global	Infrastructure IT	Capital humain	Services en ligne
2010	0.3287	0.1769	0.5739	0.2381
2012	0.4209	0.2772	0.4430	0.5425
2014	0.5060	0.3350	0.4901	0.6929
2016	0.5186	0.3429	0.4737	0.7391

Tableau B. 2: Evolution des indices de développement d'e-gouvernement du Maroc

Cependant, en raisonnant en termes de modèle d'évolution, les services électroniques gouvernementaux actuels ne dépassent pas la phase de transaction, ce qui explique pourquoi ils évoluent indépendamment des autres critères. Dans une logique d'intégration caractérisée par l'offre d'un service « one stop shop », la transformation concernera aussi bien la dimension humaine et infrastructure que la dimension technologique [St-Amant, 2004] et par conséquent, la progression des critères pourrait être plus équilibrée.

Annexe C : Outils de mise en place du prototype

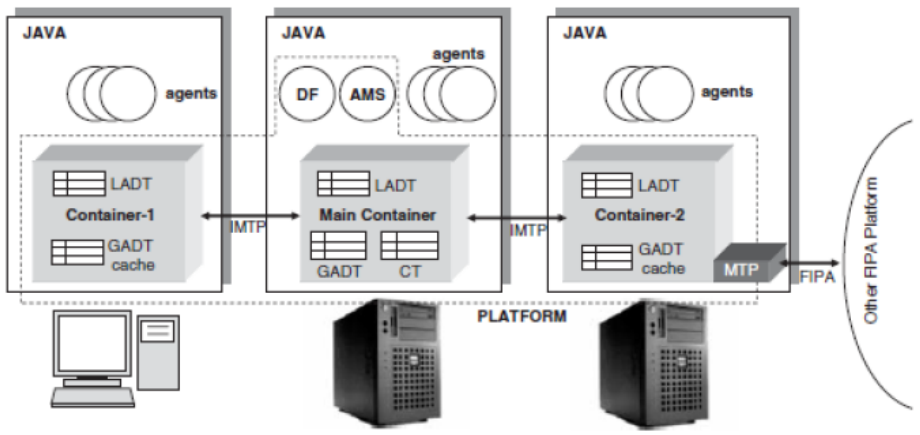
La mise en place des différents livrables du prototype élaboré dans ce travail a nécessité de faire des choix judicieux de langages, d'outils et d'environnements de développements en privilégiant certains critères de facilitation de développement, cet annexe a pour objectif de détailler quelques choix technologiques. Il s'agit, en l'occurrence du langage de communication des agents FIPA-ACL, du langage des contraintes SWRL et de la plateforme des SMA JADE.

C.1. Plateforme multi agents : JADE

JADE est une plateforme logicielle implémentée en JAVA permettant le développement et la gestion de SMA en respect des normes FIPA [Rimassa et al., 1999] . Sa création a été motivée par le besoin de disposer d'un outil répondant aux normes FIPA et permettant de valider les spécifications de cette norme relative aux SMA. Il est open source depuis 2000 et est distribué sous la licence LGPL (Library Gnu Public Licence). Ses contributeurs sont très nombreux, assurant ainsi son évolution et un support en cas de problèmes.

C.1.1 L'architecture de JADE

La figure ci-après présente l'architecture d'une plateforme JADE. Elle est composée d'un ensemble de conteneurs qui sont des instances JADE. Ces conteneurs peuvent être sur la même machine, comme elles peuvent être réparties sur plusieurs machines d'un réseau.
Chaque instance ou conteneur peut héberger/contenir plusieurs agents. Chaque plateforme JADE contient obligatoirement un conteneur principal MC (Main Container) auprès duquel s'enregistrent tous les autres conteneurs dès leur lancement. Chaque conteneur est identifié par un nom logique qui lui est associé.



CT =Container Table
GADT = Global Agent Descriptor Table
LADT = Local Agent Descriptor Table
AMS= Agent Management System
DF= Directory Facilitator
IMTP = Internal Message Transport Protocol
MTP = Message Transport Protocol

Fig.C. 1: Architecture d'une plateforme JADE [Bellifemine et al, 2001]

Cette architecture répond aux exigences de la norme FIPA [FIPA, 2016] relative à la conception de SMA. En effet, les spécifications FIPA 97 de la norme FIPA, établissent les

règles normatives permettant aux agents d'inter-opérer dans un SMA. Ces spécifications décrivent le modèle de référence d'une plateforme multi-agents où ils identifient les rôles de certains agents clés nécessaires à la gestion de la plateforme, et spécifient le contenu du langage de gestion des agents et l'ontologie du langage.

(a) Le MC a en charge de :
- la gestion de la table des conteneurs enregistrés (CT) ;
- la gestion de la table de description globale des agents (GADT) ;
- l'hébergement du AMS et du DF.

En effet, le MC, au-delà des autres agents qu'il peut héberger, contient toujours deux agents spéciaux AMS et DF qui sont lancés automatiquement à son démarrage.

Chaque conteneur dispose localement de sa propre table locale de description de ces agents (LDAT) de sorte que lors de la recherche d'un agent, le conteneur consulte d'abord sa table LDAT et c'est seulement en cas d'échec qu'il se réfère au MC pour obtenir de la GADT les références distantes de l'agent puis les enregistre dans son cache (GADT Cache) pour un éventuel usage futur.

(b) Le AMS fournit un service de nommage unique et représente l'autorité qui supervise la plateforme en assurant la gestion du cycle de vie des agents (création, attente, destruction, etc.). Chaque agent, à sa création, doit s'enregistrer à son niveau pour obtenir un identifiant unique et valide qui lui servira d'authentifiant lors de ces accès et utilisations du système.

(c) Quant au DF, il fournit un service de pages jaunes permettant aux agents d'enregistrer leurs services ou de retrouver les agents fournisseurs de services dont ils ont besoin.

Le système étant dynamique, la gestion des éventuelles migrations, terminaisons et/ou apparitions de nouveaux agents est prise en compte et les différentes mises à jours des tables sont effectuées pour garantir la continuité de service voire la tolérance aux pannes en mettant en place au besoin un mécanisme de réplication.

Une représentation de la plateforme, mettant en exergue les mécanismes de communication et d'interaction entre agents, est présentée dans la figure ci-dessous.

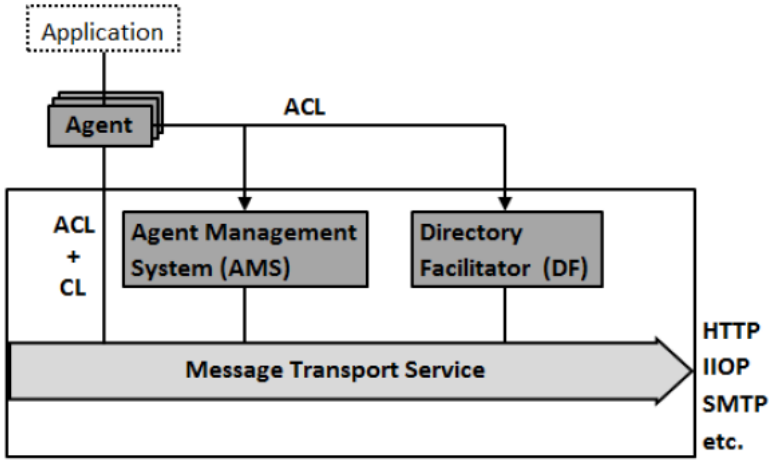


Fig.C. 2: Plateforme multi-agents aux normes FIPA [Rimassa et al., 1999]

JADE contient les librairies de Java (classes) qui sont requises pour le développement des applications à base d'agents. Un focus sur le modèle agent est donné dans la suite.

C.1.2 Le modèle Agent

La classe Agent représente une superclasse commune pour tous les agents définis par l'utilisateur. D'un point de vue programmation, cela veut dire que l'agent JADE est simplement une classe Java qui étend la classe de base Agent. Cela permet à l'agent d'hériter un comportement fondamental caché (qui traite toutes les tâches liées à la plateforme, telles que l'enregistrement, la configuration, la gestion à distance, etc.), et un ensemble de méthodes qui peuvent être appelées pour implémenter les tâches spécifiques à l'agent, par exemple envoi des messages, utilisation des protocoles d'interaction standard, enregistrement sur plusieurs domaines, ou encore des méthodes pour gérer la file de comportements d'agents. Le modèle d'exécution d'un agent est multitâches dans lequel les comportements (behaviour) d'un agent sont exécutés en parallèle (Fig.C. 3).

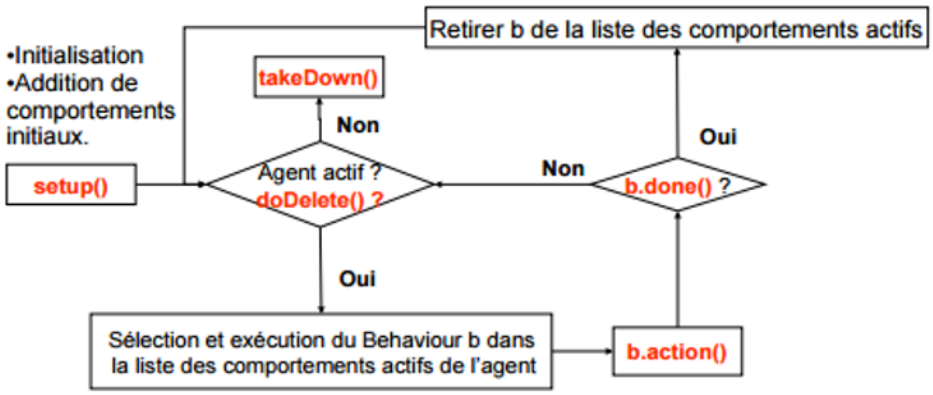


Fig.C. 3: Modèle d'exécution d'un agent [JADE, 2016]

Durant l'exécution d'un agent, son état peut être modifié:

- INITIATED : l'agent est lancé mais non enregistré auprès de l'AMS, aucun nom, aucune adresse.
- ACTIVE : l'agent est répertorié auprès de l'AMS et peut accéder aux services.
- SUSPENDED : tous les behaviours de l'agent sont suspendus.
- TRANSIT : l'agent migre vers une autre plateforme.
- WAITING : tous les behaviours de l'agent sont temporairement interrompus.
- DELETED : l'exécution de l'agent est terminée et n'est plus répertorié au sein de l'AMS.

JADE permet de contrôler le passage d'un agent d'un état à l'autre avec les méthodes doXXX, exemple doWait(), doActivate() (Fig.C. 4).

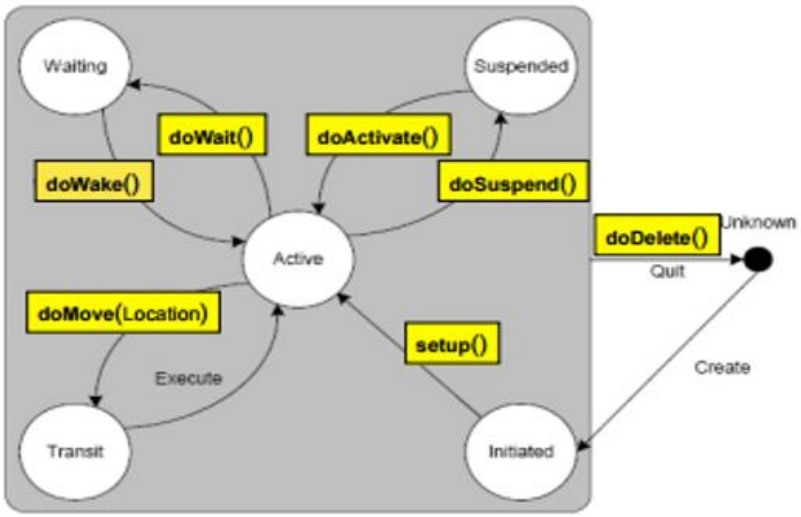


Fig.C. 4: Cycle de vie d'un agent [JADE, 2016]

C.1.3 Les comportements

Tout système SMA est mis en place pour résoudre au moins un problème. Il a donc des objets à satisfaire ou des buts à atteindre. Pour atteindre ces buts, chaque agent se voit attribuer des rôles. Pour jouer un rôle donné, un agent doit avoir un comportement et sa capacité se mesure à l'ensemble des tâches (rôles) qu'il peut accomplir ou comportements qu'il peut avoir. JADE permet d'implémenter des agents cognitifs ou réactifs et chaque agent peut se voir ajouté un ou plusieurs comportements (behaviours) [Bellifemine et al, 2001]. En effet, à la création les agents sont comme des coquilles vides auxquelles il faut ajouter des comportements implémentant des services/fonctionnalités. Les comportements dans JADE sont classifiés en plusieurs catégories.

Ainsi distingue-t-on aussi bien des comportements simples que des comportements composés. La figure suivante présente la hiérarchie entre les comportements en termes de généralisation et de spécialisation.

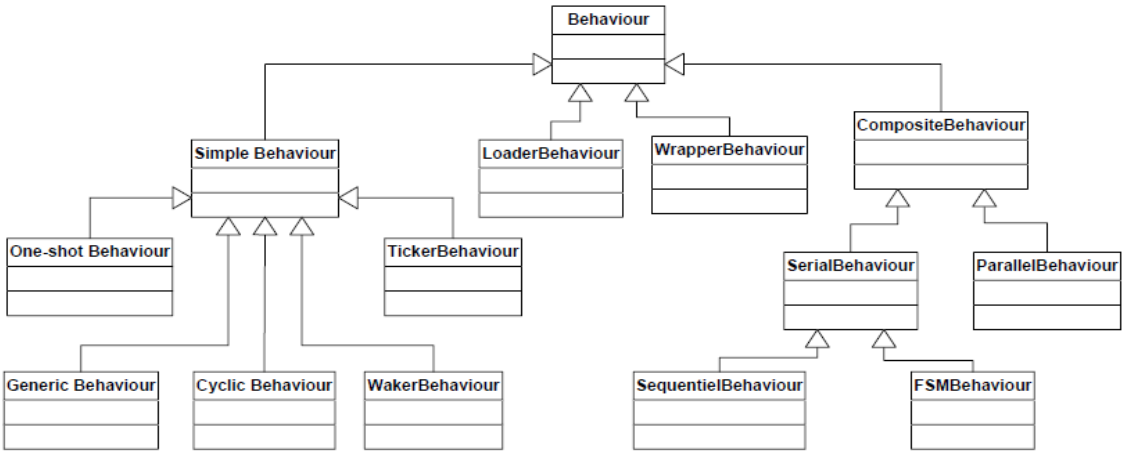


Fig.C. 5: Modélisation UML de la classification hiérarchique des comportements JADE [Bellifemine et al, 2001]

- **Les Behaviours simples :** la plate-forme JADE offre trois types de Behaviours simples:

  a) *One-shotBehaviour* : un one-shotBehaviour est une instance de la classe *jade.core.behaviours.OneShotBehaviour***.** Il a la particularité d'exécuter sa tâche une et une seule fois puis il se termine. La classe *OneShotBehaviour* implémente la méthode *done()* et elle retourne toujours *true*.
  b) *CyclicBehaviour* ***:*** un cyclic Behaviour est une instance de la classe *jade.core.behaviours.CyclicBehaviour*. Il a la particularité d'exécuter sa tâche d'une manière répétitive. La classe *CyclicBehaviour* implémente la méthode *done()* qui retourne toujours *false*.
  c) *GenericBehaviour* ***:*** un Generic Behaviour est une instance de la classe *jade.core.behaviours.Behaviour*. Le Generic Behaviour vient entre le *OneshotBehaviour* et le *Cyclic Behaviour* de faite qu'il n'implémente pas la méthode *done()* et laisse son implémentation au programmeur, donc il peut planifier la terminaison de son Behaviour selon ses besoins.

- **Les Behaviours planifiés :** pour planifier une tâche d'un agent JADE, il existe deux types de Behaviours :

  a) *WakerBehaviour (timeouts)* ***:*** le *WakerBehaviour* est implémenté de façon à exécuter la méthode *onWake()* après une période passée comme argument au constructeur. Cette période est exprimée en millisecondes. Le Behaviour prend fin juste après avoir exécuté la méthode *onWake()*.
  b) *TickerBehaviour (timeouts répétés):* le *TickerBehaviour* est implémenté pour qu'il exécute sa tâche périodiquement par la méthode *onTick()*. La durée de la période est passée comme argument au constructeur.

- **Les Behaviours composés :** les Behaviours (simples et planifiés) ne peuvent pas répondre d'une manière efficace à tous les besoins d'un développeur d'un système multi-agents. Pour cela JADE offre un ensemble de Behaviours composés pour présenter des tâches complexes. La classe mère de toutes les autres complexes est la classe *jade.core.behaviours.CompositeBehaviour*. Notons que l'ordre d'exécution des Behaviours est à la charge des deux méthodes *scheduleFirst()* et *scheduleNext()* que les classes héritant de la classe *CompositeBehaviour* doivent implémenter. Le programmeur n'est pas dans l'obligation d'utiliser directement la classe *CompositeBehaviour* car il dispose de trois classes filles.
  a) *SequentielBehaviour* ***:*** dans le *SequentielBehaviour*, le Behaviour commence par exécuter le premier sous-Behaviour et lorsqu'il termine son exécution (sa méthode *done()* retourne *true*), il passe au Behaviour suivant, et ainsi de suite. Les sous-Behaviours sont ajoutés au *sequentielBehaviour* par la méthode *addSubBehaviour()* et l'ordre de l'ajout détermine l'ordre d'exécution.
  b) *FSMBehaviour* ***:*** *FSMBehaviour* est une sorte de Behaviour qui implémente un automate à états finis dont chaque état correspond à l'exécution d'un sous-Behaviour.
  c) *ParallelBehaviour* ***:*** le *parallelBehaviour* est un Behaviour qui permet d'exécuter plusieurs Behaviours en parallèle. L'ajout de sous-Behaviour se fait par la méthode *addSubBehaviour()*. Si on veut que le *parallelBehaviour* termine dès qu'un de ses sous-Behaviours termine alors on doit passer à son constructeur l'argument *WHEN_ANY* et pour attendre la fin de tous les sous-Behaviours on doit lui passer l'argument *WHEN_ALL*.

d) *Combiner plusieurs Behaviours composés :* il est possible de combiner les différents comportements composés pour créer des comportements plus complexes.

Pour notre cas c'est les comportements FSMBehivour et OneShotBehaviour qui ont été utilisés le plus pour implémenter l'automate représentant le comportement des agents Manger et Planner précédemment décrits dans cette thèse.

Pour le comportement FSM, l'introduction de l'automate se fait de la manière suivante :

- L'ajout d'un nouveau état se fait par la méthode registerState (Behaviour state, String name) ;
- L'ajout de l'état initial (il n'existe qu'un seule état initial) se fait par la méthode registerFirstState (Behaviour state, String name);
- L'ajout d'un état final (il est possible d'en avoir plusieurs) se fait par la méthode registerLastState (Behaviour state, String name).
- L'ajout d'une nouvelle transition se fait par la méthode registerTransition(String s_1, String s_2, int event);
- L'ajout d'une transition par défaut (la seule transition entre deux états ou bien la transition à prendre si aucune autre n'est prise) se fait par la méthode registerDefaultTransition(String s_1, String s_2, String[] toBeReset)

Avec :

- *state : le Behaviour qui représente l'état;*
- *name : le nom de l'état*
- *s_1 : l'état source;*
- *s_2 : l'état destination;*
- *event: l'étiquette de la transition;*
- *String[] toBeReset : l'ensemble d'états pour lesquels on doit faire un reset() avant de les ré-exécuter parce qu'on ne peut pas re-éxecuter un Behaviour une autre fois avant de lui faire un reset().*

C.2. Langage de communication des agents : FIPA-ACL

Pour que des agents autonomes soient capables d'interagir entre eux, sans se soucier du langage de programmation utilisé, il faut que ces agents soient dotés d'un langage de communication commun. Il existe deux principaux langages de communication -Agent Communication Language (ACL)- : KQML (pourKnowledge Query and Manipulation Language) [Finin et al., 1994] et FIPA-ACL [FIPA-ACL, 2016] . Ils sont tous les deux basés sur la théorie des actes de langages du philosophe américain John Searle. Dans un contexte JADE, la communication inter-agents est probablement la caractéristique la plus fondamentale de cette plateforme et elle est mise en œuvre conformément aux spécifications FIPA. C'est donc le langage FIPA-ACL qui a été utilisé dans les messages inter-agent dans le prototype élaboré.

Le langage FIPA ACL est le langage standard des messages selon les spécifications de FIPA, il impose le codage, la sémantique et la pragmatique des messages. La norme n'impose pas de mécanismes spécifiques pour le transport interne de messages. Plutôt, puisque des agents différents pourraient s'exécuter sur des plateformes différentes et utiliser des technologies différentes d'interconnexion, FIPA spécifie que les messages transportés entre les plateformes devraient être codés sous forme textuelle. On suppose que l'agent est en mesure de transmettre

cette forme textuelle. La norme FIPA préconise des formes communes pour les conversations entre agents par la spécification de protocoles d'interaction, qui incluent des protocoles simples de type requête-réponse.

C.2.1 Actes de communication

Ayant une syntaxe similaire à KQML le langage s'appuie sur la définition de deux ensembles:

- un ensemble d'actes de communication primitifs, auquel s'ajoutent les autres actes de communication pouvant être obtenus par la composition de ces actes de base ;
- un ensemble de messages prédéfinis que tous les agents peuvent comprendre.

FIPA-ACL possède 21 actes communicatifs, exprimés par des performatives, qui peuvent être groupés selon leur fonctionnalité de la façon suivante :

- passage d'information : ***inform*, inform-if (macro act), inform-ref (macro act), confirm*, disconfirm*, proxy***
- réquisition d'information : ***query-if, query-ref, subscribe***
- négociation : ***accept-proposal, cfp, propose, reject-proposal***
- distribution de tâches (ou exécution d'une action) : ***request*, request-when, request-whenever, agree, cancel, refuse***
- manipulation des erreurs : ***failure, not-understood***

En FIPA-ACL il n'existe pas de primitives de gestion ni de facilitation.

Les actes communicatifs peuvent être primitifs ou composés. Les actes communicatifs primitifs sont définis de façon atomique, c'est-à-dire qu'ils ne sont pas définis à partir d'autres actes (dans la classification ci-dessus ils sont suivis d'une étoile - "*"). En revanche, les actes communicatifs composés sont définis à partir d'autres actes par l'une des opérations suivantes:

- un acte fait partie du contenu d'un autre acte à travers l'opérateur de composition " ; " pour indiquer une séquence d'actions.
- à travers l'opérateur de composition " | " pour indiquer un choix non déterministe de l'action.

C. 2.2 Structure d'un message

Un message comprend plusieurs éléments qui sont présentés dans le tableau suivant [FIPA-ACL, 2016] :

Elément	**Signification**
performative	type de l'acte communicatif
sender	l'émetteur du message
receiver	le destinataire du message
reply-to	participant à l'acte de communication
content	le contenu du message (l'information transportée par la performative)
language	le langage dans lequel le contenu est représenté
encoding	décrit le mode d'encodage du contenu du message
ontology	le nom de l'ontologie utilisée pour donner un sens aux termes utilisés dans le *content*
protocol	contrôle la conversation
conversation-id	identificateur de la conversation

reply-with	identificateur unique du message, en vue d'une référence ultérieure
in-reply-to	référence à un message auquel l'agent est entrain de répondre (précisé par l'attribut reply-with de l'émetteur)
reply-by	impose un délai pour la réponse

Tableau C. 1: Structure d'un message FIPA-ACL

On peut observer que le contenu des deux premières catégories des performatives, passage et réquisition d'information exprime en fait une proposition, le contenu de la catégorie 3, une négociation, exprime une action et une proposition, alors que la catégorie 4, une distribution de tâches, est certainement une action.

❖ **Example d'un message**

```
(inform
        :sender A
        :receiver B
        :reply-with laptop
        :language KIF
        :ontology ordinateurs
        :content (=(prix HP-Jet) (scalar 1500 USD))
        :reply-by 10
        :conversation-id conv01
        )
```

C. 2.3 Sémantique du modèle

SL (Semantic Language) [FIPA-ACL, 2016] est le langage formel employé pour définir la sémantique des performatives de FIPA-ACL. **SL** utilise une logique multi-modale avec les opérateurs modaux pour les croyances (B), les désirs (D), les croyances incertaines (U) et les intentions (buts persistants, PG). **SL** peut représenter des propositions, des objets, et des actions. Une description détaillée de **SL**, y compris sa propre sémantique peut être trouvée dans les spécifications de FIPA 97.

Le modèle mental d'un agent est basé sur la représentation de trois attitudes primitives : croyance, incertitude et choix (ou, dans une certaine mesure, but). Elles sont respectivement formalisées par les opérateurs modaux B, U, et C.

B_i(p) : "i croit (implicitement) que p est vrai"
U_i(p) : "i pense que p est plutôt vrai que faux "
C_i(p) : "i souhaite que p soit vrai. Pour ce faire, l'agent établit un plan d'actions lui permettant d'atteindre le but désiré p"

Les composantes des actes de communication planifiés au sein d'un agent caractérisent à la fois le but pour lequel l'acte est choisi et les conditions qui doivent être satisfaites pour que l'acte soit exécuté. Pour un acte donné, le premier est désigné sous le nom de l'effet rationnel (*rational effect*), et le dernier comme préconditions de faisabilité (*feasibility preconditions*) qui sont les qualifications de l'acte. Ainsi, chaque acte de communication est défini de cette manière, ce qui permet aux agents de raisonner et d'avoir un comportement rationnel cohérent.

L'intention d'un agent de réaliser un but donné se traduit par l'intention que l'un des actes connus par l'agent soit exécuté de telle sorte que son effet rationnel corresponde exactement au but de l'agent.
L'ensemble des préconditions de faisabilité pour un acte de communication peut être décomposé en deux sous-ensembles : (i) les préconditions de capacité et (ii) les préconditions du contexte dans lequel se trouve l'agent.
Les préconditions de capacité caractérisent les capacités intrinsèques d'un agent dans l'exécution d'un acte donné. Par exemple, pour affirmer sincèrement une certaine proposition P, un agent doit croire que P est vraie.
Les préconditions associées au contexte caractérisent la pertinence de l'acte par rapport au contexte dans lequel il est exécuté. Par exemple, un agent peut faire une promesse tout en croyant que l'action promise est nécessaire pour le destinataire.
Un agent ayant observé (reçu) un acte de communication doit croire que les conditions préalables de faisabilité sont vraies et que l'agent qui a exécuté l'acte (envoi de message) a l'intention de provoquer l'effet rationnel (effet intentionnel).

C.3. Langage des règles : SWRL

Afin de décrire sémantiquement les services sujets de composition dans le cadre du cas étudié, nous avons utilisé OWL-S comme langage sémantique. Pour augmenter l'expressivité des contraintes, nous avons eu besoin de primitives additionnelles non fournis par OWL-S. Nous nous sommes alors servis de SWRL (Semantic Web Rule Language) [Horrocks et al., 2004]. Dans cette section nous présentons brièvement ce langage de règles.

SWRL est un langage de règles pour le Web sémantique, combinant le langage OWL-DL et le langage RuleML « Rule Markup Language (Unary/Binary Datalog) ».

Avec SWRL, les connaissances peuvent être exprimées sous la forme : antécédent → conclusion. Les axiomes du langage OWL-DL sont étendus à l'aide des clauses de Horn réduites aux prédicats unaires et binaires.

Ce langage est indécidable, les règles SWRL sont appliquées même si les individus ne sont pas présents dans la base. Une restriction de SWRL, appelée DL-safe rules, a été conçue pour conserver la décidabilité, c'est-à-dire que les règles écrites ne portent que sur des individus explicitement présents dans la base. Autrement dit, les règles ne peuvent s'appliquer que si l'identité de toutes les instances présentes est connue.

De nombreux moteurs d'inférences commencent à supporter SWRL : Bossam, Hoolet, KAON2, Pellet, RacerPro, R2ML « REWERSE Rule Markup Language » et Sesame [Horrocks et al., 2004]. Ils suivent trois types d'approches :

- Traduire SWRL en logique du premier ordre (Hoolet) et appliquer un algorithme de chaînage avant ;
- Traduire OWL-DL en règles et appliquer un algorithme de chaînage avant (Bossam) ;
- Intégrer les règles SWRL dans le moteur d'inférences OWL-DL fondé sur les algorithmes des tableaux sémantiques (Pellet).

L'intérêt de l'utilisation de SWRL repose sur le fait qu'il permet d'exprimer des connaissances qu'on ne peut pas exprimer en OWL (et en OWL-S). Par exemple, avec OWL (version 1) on ne peut pas exprimer le fait que x est l'oncle de y parce que z est à la fois parent de x et le frère de y. Cette règle est exprimée en SWRL comme suit :

estParentde(?z,?y) ^ estFrerede(?x,?z) → estOnclede(?x,?y).

En OWL (version 2), il n'est pas possible d'exprimer le fait que x est l'enfant d'un couple marié (y,z) car OWLne permet pas d'exprimer des relations entre des instances qui sont liées à une instance donnée. En SWRL on peut exprimer cette relation comme suit :

Personne(?x) ^ aPourPere(?x, ?y) ^ aPourMere(?x, ?z) ^ estMariea(?y, ?z) → estFilsduCouple(?x).

Une autre fonctionnalité puissante du langage SWRL réside dans sa capacité à supporter une large gamme de compléments définis par l'utilisateur appelés aussi builts-ins. Ces compléments permettent d'étendre significativement l'expressivité du langage SWRL. Certains compléments sont basés sur des fonctions de comparaison (equal, lessThan, etc.), des fonctions mathématiques (add, multiply, sin, etc.), des fonctions booléennes (booleanNot), des fonctions de traitement des chaines de caractères (stringLength, substring, etc.) et d'autres fonctions.

Dans OWL-S, les conditions (formules logiques) sont représentées sous forme de chaînes de littéraux ou des littéraux XML. Pour ce dernier cas on peut utiliser des langages dont le codage standard est en XML, comme SWRL. Le corps et l'entête des règles SWRL sont des formules logiques, les conditions OWL-S peuvent donc être exprimées par le corps ou l'entête de règles SWRL. En plus, puisque ces conditions sont exprimées en utilisant les entrées et les sorties. Elles seront également exprimées en termes d'ontologie de domaine et auront donc une expressivité sémantique adéquate.

Bibliographie

[ActiveBPEL, 2016]: ActiveBPEL Engine - the open source BPEL engine, http://www.activebpel.org, consulté en 2016.

[Adadi et al , 2016]: A. Adadi, M.Berrada, D.Chenouni, A phased two-stage approach for dynamic Web service composition, The international conference on Information Technology for Organization Development (IT4OD-2016), 2016.

[Adadi et al, 2014]: A. Adadi, M.Berrada, D.Chenouni, A Multi-Agent Planning Architecture for Semantic Web Service Composition, International Review on Computers and Software (IRECOS), 2014.

[Adadi et al., 2015]: A. Adadi, M.Berrada, D.Chenouni, B.Bounabat A semantic Web service composition for e-Government services, Journal of Theoretical and Applied Information Technology (JATIT), 2015.

[Adadi et al., 2015b]: A. Adadi, M.Berrada, D.Chenouni, B.Bounabat, Ontology based composition of e-Government services using AI Planning, the 10th International Conference on Inelligent Systems : Theories and Applications (SITA'2015), 2015.

[Agarwal et al.,2008]: V. Agarwal, G. Chafle, S. Mittal, B. Srivast ava, Understanding Approaches for Web Service Composition and Execution, The First Bangalaore Annual Compute conference, Article no. 1, 2008.

[Akkiraju et al., 2004] : R. Akkiraju, K. Verma, R. Goodwin, P. Doshi, J. Lee, Executing abstract Web process flows, The international conference on automated planning and scheduling (ICAPS).Workshop on planning and scheduling for Web and grid services, Whistler, BC, Canada, 2004.

[Akkiraju et al., 2005]: R. Akkiraju, J. Farrell, L. Miller, M.Nagarajan, M. Schmidt, A. Sheth, K.Verma, Web Service Semantics – WSDL-S, Technical note,http://lsdis.cs.uga.edu/library/download/WSDL-S-V1.html, 2005.

[AlSedrani et al., 2016] A. AlSedrani, A. Touir, Web Service Composition In Dynamic Environment: A Comparative Study, Computer Science & Information Technology, 2016

[Angelis, 2009]: F. Angelis, Interoperability in e-Government Services. School of Advanced Studies-Doctorate Course in Information Science and Complex Systems, 2009.

[Ankolekar et al, 2008]: A. Ankolekar, M. Krotzsch, T. Tran, D. Vrandecic, The two cultures: mashing up Web 2.0 and the semantic web, The 16th International World Wide Web Conference, pp.70–75, 2008.

[Ankolekar et al., 2002]: A. Ankolekar, M.Burstein, J.R. Hobbs, O.Lassila, D.Martin, D.McDermett, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara,DAML-S : Web Service Description for Semantic Web, Semantic Web conference, Italy, 2002.

[ApacheODE, 2016]: ApacheODE: The Orchestration Director Engine executes business processes written following the WS-BPEL standard, http://ode.apache.org/, consulté en 2016.

[Arenaza, 2006] : N. Arenaza, Composition semi-automatique de Services Web, 2006.

[Arkin et al. , 2002]: A. Arkin, S. Askary, S.Fordin, Web Service Choreography Interface (WSCI) 1.0, Editors. Note W3C, http://www.w3.org/TR/2002/NOTE-wsci-20020808, 2002.

[Arkin, 2002]: A. Arkin, Business Process Modeling Language, 2002.

[Austin, 1962]: J. Austin, How to do things with words, 1962.

[Baldon et al., 2007]: M. Baldon, C. Baroglio, A. Martelli, V. Patti. "Reasoning about interaction protocols for customizing web service selection and composition". The Journal of Logic and Algebraic Programming, Elsevier, No. 70, pp. 53 – 73, 2007.

[Banerji et al., 2002]: A.Banerji, C. Bartolini, D. Beringer,V. Chopella, K.Govindarajan, A.Karp, H.Kuno, M.Lemon, G.Pogossiants, S.Sharma, S.Williams, Web Services Conversation Language (WSCL). Note W3C, http://www.w3.org/TR/wscl10,2002.

[Barry et al., 2003]: J.Barry, L.P. Kaelbling, T.L. Pérez, Hierarchical Solution of Large Markov Decision Processes, AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 2003.

[Bartalos et al. , 2011] : P.Bartalos, M.Bielikova , Automatic Dynamic Web Service Composition: A Survey And Problem Formalization, COMPUTING AND INFORMATICS, Volume 30, No 4, 2011.

[Bass et al., 2003]: L. Bass, L., P. Clements, R. Kazman, Software Architecture in Practice, 2nd Addison-Wesley, 2003.

[Beek et al., 2007] : M. Beek, A. Bucchiarone, S. Gnesi, Formal Methods for Service Composition, Annals of Mathematics, Computing & Teleinformatics, vol. 1, no. 5, pp.1-10, 2007.

[Bellifemine et al, 2001]: F. Bellifemine, A. Poggi, and G. Rimassa, "Developing multi-agent systems with JADE," in Intelligent Agents VII Agent Theories Architectures and Languages, Springer, pp. 89–103, 2001.

[Ben Chouikha, 2008] : M. Ben Chouikha, B. Bounabat, Méthodologie pour l'évaluation qualitative pour le gouvernement électronique local dans les pays en voie de développement, Colloque e-management AFME'2008, France. 2008.

[Benatallah et al. , 2003] B.BENATALLAH, Q.Z. SHENG, M. DUMAS, The self-serv environment for web services composition, IEEE Internet Computing, pp.40–48, 2003.

[Bent et al., 1999]: S.Bent, K. Kernaghan, B.D Marson, Innovations and Good Practices in Single Window Service. Canadian Center for Management Development, Canada, 1999.

[Berardi et al., 2005]: D. Berardi, Automatic service composition. Models, techniques, tools. PhD thesis, University of Rome "La Sapienza", Rome, Italy, 2005.

[Berners-Lee et al. ,2001]: T. Berners-Lee, J. Hendler, et O. Lasilla, The Semantic Web, Scientific American, 2001..

[Berrada et al. , 2006]: M. Berrada, B. Bounabat, M. Harti, M. Qualitative Verification of Multi-Agents Reactive Decisional System Using Business Process Modeling Notation, Intelligent Agent Technology (IAT '06), 2006.

[Berrada, 2008] : M. Berrada Elaboration d'une Méthode De Modélisation Et De Vérification Orientée Management Par Processus Pour Les Systèmes Multi-Agents, 2008.

[Blázquez et al., 1998]: M. Blázquez, M. Fernández , J.M García-Pinar, A. Gómez-Pérez, Building Ontologies at the Knowledge Level using the Ontology Design Envirnoment, The Eleventh Knowledge Acquisition Workshop, KAW98, 1998.

[BnF, 2010]: Bibliothèque nationale de France, « Web sémantique, Web de données », .2010

[Bond et al., 1988]: A. Bond, et L. Gasser, A Survey of Distributed Artificial Intelligence, 1988.

[Bosnjak et al., 2010]: S. Bosnjak, M. Maric, Z. Bosnjak, The Role of Web Usage Mining in Web Applications Evaluation, Management Information Systems, Vol. 5, No. 1, pp. 31-36, 2010.

[Bruaux et al. 2014] : S. Bruaux et G. Kassel, OntoKADS : une ontologie générale de la résolution de problèmes, 2014.

[Bucchiarone et al., 2006]: A. Bucchiarone, S. Gnesi, A Survey on Services Composition Languages and Models, Int. Workshop on Web Services Modeling and Testing , pp. 51-63, 2006.

[Cardoso et al., 2004]: J. Cardoso, A. P. Sheth, J. A. Miller, J. Arnold,K. J. Kochut, Modeling quality of service for work-flows and web service processes, Web Semantics Journal: Science, Services and Agents on the World Wide Web Journal, pp. 281–308, 2004.

[Carman et al ., 2003] : M. Carman, L. Serafini, P. Traverso, Web service composition as planning. ICAPS'03 Workshop on Planning for Web Services, Trento, Italy, 2003.

[Casati et al., 2000]: F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy, and M.C. Shan. Adaptive and dynamic service composition in EFlow, The 12th Conference on Advanced Information Systems Engineering (CAISE), S. Verlag, Ed., pp. 13–31, 2000.

[Casati et .al, 2001]: F. Casati, M. Sayal, and M.C. Shan. Developping e-services for composing e-services, The 13th International Conference on Advanced Information Systems Engineering (CAiSE), S. Verlag, Ed, 2001.

[CEFACT-ONU, 2005] : Centre des Nations Unies pour la facilitation du commerce et les transactions électroniques, Recommandation et lignes directrices en vue de la mise en place d'un guichet unique, 2005.

[CGI ,2013] : CGI, business Consulting, L'entreprise étendue, réponse aux enjeux d'un monde nouveau, Livre blanc édition 2013, 2013.

[Chandrasekaran et al., 1998]: B. Chandrasekaran, J. Josephson, V.R. Benjamins, The ontology of tasks and methods. Knowledge Acquisition Workshop, KAW98, 1998.

[Chen et al., 2003]: L. Chen, N.R. Shadbolt, C. Goble, F. Tao, S.J. Cox, C. Puleston, , P. Smart, Towards a Knowledge-based Approach to Semantic Service Composition, 2nd International Semantic Web Conference, 2003.

[Chen, 2002]: H. Chen, Digital Governement: technologies and practices, Decision support Systems (special issues), pp. 223–227, 2002.

[Cheng et al ., 2002]: Z. Cheng, M. Singh, et M. Vouk, Composition constraints for semantic web services, The WWW-2002 workshop ,2002.

[Christensen et al., 2001]: E. Christensen, F. Curbera, G. Meredith & S. Weerawarana. Web services description language (wsdl) 1.1. Technical report, W3C, http://www.w3.org/TR/wsdl/, 2001.

[Clement et al., 2004]: L. Clement, A. Hately, C. von Riegen & T. Rogers. Uddi version 3.0.2, Technical report, OASIS, 2004.

[cnie, 2016] : http://www.cnie.ma/, consulté en 2016.

[Commonwealth of Australia, 2003]: E-Governement benefit study, Commonwealth of Australia, 2003,

[Comuzzi et al., 2009] : M. Comuzzi, B. Pernici. A Framework for QoS-Based Web. New York : ACM Transactions on The Web, 2009.

[Constantinescu et al., 2005]: I.Constantinescu, B. Faltings, W. Binder, Type based service composition, The 13th international World Wide Web Conference on Alternate Track, New York, USA, ACM Press, pp. 268–269, 2004.

[Curbera et al., 2001]: F. Curbera, I. Sliva-Lepe & S. Weerawarana, On the integration of heterogeneous web services partners, 2001.

[Curbera et al., 2003]: F. Curbera ,F. Leymann, R. Khalaf, BPEL4WS (Business Process Execution Language for Web Services) ,IBM, http://www.daml.org/services/swsl/materials/bpel11.pdf, 2003

[D.S Weld, 1999]: D.S Weld, Recent Advantages in AI Planning, AI Magazine, 1999.

[Delgrande et al., 2013] : J.P. Delgrande, R.Wassermann, Horn Clause Contraction Functions, Journal of Artificial Intelligence Research, pp. 475-511, 2013.

[Deloitte Research, 2000]: Deloitte research,. At the Dawn of e-Government: the citizen as Customer, A global public sector study by Deloitte & Touche, 2000.

[Do et al. , 2001]: M. Do, et S. Kambhampati, SAPA: A domain independent heuristic metric temporal planner, In Proceedings of ECP-01, 2001.

[Dobson et al., 2005]: G. Dobson, R. Lock, I. Sommerville, QoSOnt : a QoS ontology for service-centric systems, The 31st EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 80–87. IEEE Computer Society, 2005.

[Domingue et al., 2004] : J. Domingue and S. Galizia, Towards a Choreography for IRS-III, the Workshop on WSMO Implementations (WIW 2004), 2004.

[Dong-Hoon et al., 2009]: S. Dong-Hoon, L. Kyong-Ho, S.Tatsuya, Automated generation of composite web services based on functional semantics. Journal of Web Semantics: Science, Services and Agents on the World Wide Web, vol:7, pp. 332 – 343, Science Direct 2009.

[Dustdar et al., 2005]: S. Dustdar, W. Schreiner, A survey on web services composition, Int. Journal of Web and Grid Services, vol. 1, no.1, pp. 1–30, 2005.

[Eckerson, 2002]: W.W. Eckerson, Data Quality and Bottom Line: Achieving Business Success Through High Quality Data, TDWI Report Series, The Data Warehousing Institute, Seattle, WA, 2002.

[Eclipse, 2016]: https://eclipse.org, consulté 2016.

[Egov.ma, 2011] : Programme eGouvernement, http://www.egov.ma/sites/default/files/Programme eGov%20Morocco.pdf, 2011.

[Eistert et al., 1995]: T. Eistert, R. O'Callaghan, R, An association's leadership for industry-wide EDI, EDI in Europe: how it works in practice (Wiley series on information systems), pp. 277-297, 1995.

[EL Falou et al., 2010]: M. EL Falou , M.Bouzid , A.Mouaddib ,T.Vidal, A Distributed Planning Approach for Web Services Composition, Web Services (ICWS), IEEE International Conference, pp. 337 – 344, 2010.

[Emani, 2006]: S. Emani, A Comparative Evaluation Of Semantic Web Service Discovery Algorithms And Engines, University of Georgia in Partial, India , 2006.

[Englemore , 1988]: R. Englemore, T.Morgan , Blackboard Systems, Addison Wesley, 1988.

[Entreprise,2015]:The Enterprise Ontology, http://www.aiai.ed.ac.uk/project/enterprise/enterprise/ ontology.html, consulté en 2015.

[Erol, 1994]: K. Erol, J. Hendler et D. Nau. Semantics for hierarchical task-network planning, Rapport technique, University of Maryland at College Park, College Park, MD, USA, pp. 27-33, 1994.

[European Commission, 2005]: Commission européenne, Transforming government, http://europa.eu.int/information_society/doc/factsheets/010-egovernment.pdf, 2005.

[Farrell et al., 2007]: J. Farrell et H. Lausen, Semantic annotations for WSDL and XML schema, W3C recommendation, W3C, http://www.w3.org/TR/2007/REC-sawsdl-20070828/, 2007

[Feng et al., 2012]: Y. Feng, A. Veeramani, R. Kanagasabai, Automatic DAG-Based Service Composition: A Model Checking Approach, IEEE 19th International Conference on Web Services (ICWS), 2012

[Ferber, 1995]: J. Ferber, Les systèmes multi-agents : vers une intelligence collective, 1995.

[Ferber,1999]: J. Ferber, Multi-Agent Systems - an introduction to distributed artificial intelligence, Addison-Wesley, pp. 509,1999.

[Fernández et al., 1997]: M. Fernández-López, A.Gómez-Pérez, N.Jurist, METHONTOLOGY: From Ontological Art Towards Ontological Engineering. Spring Symposium on Ontological Engineering of AAAI. Stanford University, California, pp 33–40 10, 1997.

[Fielding, 2000]: R. T. Fielding, Architectural Styles and the Design of Network-based Software Architecture, thèse, University of California, 2000.

[Fikes et al. , 1971]: R.E.Fikes, N.J.Nilsson., STRIPS: A new approach to the application of theorem proving to problem solving, Artificial Intelligence 2, pp. 189−208; 1971.

[Fikra, 2016] : http://fikra.egov.ma/, consulté en 2016.

[Finin et al., 1994]: T. Finin, Rich Fritzson, D. McKay, R. McEntire, KQML as an Agent Communication Language, Conference on Information and Knowledge Management (CIKM), 1994.

[FIPA, 2016] : http://www.fipa.org/, consulté en 2016.

[FIPA-ACL, 2016] : http://www.fipa.org/repository/aclspecs.html, consulté en 2016.

[Forgy , 1982]: C. Forgy, Rete: A fast algorithm for the many patterns/manyobjects match problem. Artif Intell, pp. 17–37, 1982.

[Frei et al., 2013]: R. Frei, R. McWilliam, B. Derrick, A. Purvis, A. Tiwari, G. Serugendo, Self-healing and self-repairing technologies, International Journal of Advanced Manufacture Technologies DOI 10.1007/s00170-013-5070-2 ,2013

[Friese et al., 2005]: T. Friese , J.P. Müller , and B. Freisleben, Self-healing Execution of Business Processes based on a Peer-to-peer Service Architecture, ARCS 2005, pp. 108-123, 2005.

[Gagnon ,2013] O. Gagnon, Indexation de Documents Web à l'aide d'ontologie, Université De Montréal, Département De Génie Informatique Et Génie Logiciel, Ecole Polytechnique De Montréal, 2013

[Galilée, 1982] Le thermomètre de Galilée, http://essorensen.com/galileiglassreadings_fr.html, consulté en 2016.

[Garson et al., 2008]: G.D.Garson, M. Khosrow-Pour, Handbook of research on public Information Technology, livre, 2008

[Gartner, 2002]: Groupe Gartner, http://www.gartner.com, 2002.

[Ghallab et al. , 2004]: M.Ghallab, et P.Traverso, Automated planning: Theory and practice. San Francisco, CA: Morgan Kaufmann Elsevier, 2004.

[Ghallab, 2001]: M.Ghallab. Planification et décision, 2001.

[Giacomo et al., 2000]: GD. Giacomo, Y. Lesperance, HJ. Levesque, Congolog, a concurrent programming language based on the situation calculus, Artif Intell , pp. 109–169, 2000.

[GID, 2016] : https://www.gid.gov.ma/GID.Web/, consulté en 2016.

[Gomez-Perez et al., 1999]: A. Gomez-Perez et V.R. Benjamins, Overview of knowledge Sharing and Reuse Components: Ontology and Problem-Solving Methods, The IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm, Sweden, pp. 1-15, 2 1999.

[Gottschalk et al., 2002]: K. D. Gottschalk, S. Graham, H. Kreger & J. Snell, Introduction to Web services architecture, IBM Systems Journal 41(2), pp.170-177, 2002

[GRAF, 1996] : Lexique de philosophie, Éditions du Seuil, 1996.

[Gruber, 1992]: T.R. Gruber, Ontolingua : A mechanism to support portable ontologies. tech. report. Rapport technique, Stanford University, Knowledge Systems Laboratory, 1992.

[Gruber, 1993]: T.Gruber,Toward Principles for the Design of Ontologies Used for Knowledge Sharing, KSL 93-4, Computer Science Department, Stanford University, 1993.

[Grüninger et al., 1995]: M. Grüninger, MS. Fox , Methodology for the design and evaluation of ontologies InSkuce D (ed) IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing, pp. 6.1–6.10, 1995.

[Gudgin et al., 2003]: M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau & H. Frystyk, Soap version 1.2 part 1: Messaging framework, Technical report, W3 Consortium, 2003.

[Guere et al. , 1999]: E.Guere et R.Alami, A possibilistic planner that deals with non-determinism and contingency, The 16th international joint conference on Artificial intelligence - Volume 2, IJCAI'99, pages 996– 1001, San Francisco, CA, USA, 1999.

[Guitton, 2006] : J. Guitton, Planification multi-agent pour la composition dynamique de services web, Rapport de stage, Master 2 Recherche, Grenoble 1, 2006.

[Gurguis et al., 2005]: S. Gurguis , A. Zeid, Towards autonomic web services: Achieving self-healing using web services. SIGSOFT Softw. Eng. Notes. Vol. 30, No 4, pp.1-5, 2005.

[H.M. Rusli et al., 2008] : H.M. Rusli, M. Puteh, S. Ibrahim, S. G. H. Tabatabaei, A Comparative Evaluation of State-of-the-Art Approaches for Web Service Composition, The Third International Conference on Software Engineer ing Advances, pp. 488-493, 2008.

[Harmelen et al. , 2001]: V. Harmelen, F., Peter F. Patel-Schneider, P., Horrocks, I.: Reference description of the DAML+OIL ontology markup language. http://www.daml.org/2001/03/ reference.html, 2001.

[Hatzi et al.,2010]: O.Hatzi, D. Vrakas, N. Bassiliades, D. Gnostopoulos, I. Vlahavas, The PORSCE II Framework: Using AI Planning for Automated Semantic Web Service Composition, Cambridge University Press, The Knowledge Engineering Review, pp:1–24. 2010,

[Heeks, 2001]: R. Heeks, iGovernment : Understanding e-Governance for Development, Working Paper Series : Paper No. 11, Institute for Development Policy and Management, 2001.

[Heflin et al., 1999]: J.Heflin, J.Hendler et S.Luke, S. SHOE: A Knowledge Representation Language for Internet Applications. Technical Report CS-TR-4078 (UMIACS TR-99-71), Dept. of Computer Science, University of Maryland at College Park, 1999.

[Hegel, 1991]: G.W.F.HEGEL, Phénoménologie de l'Esprit (1807), Traduction de Jean Pierre Lefebvre, Paris, pp. 565, 1991.

[Helmert, 2008]: M. Helmert, Understanding Planning Tasks. Springer Verlag, Berlin, Germany, 2008.

[Hofstee, 2006]: E. Hofstee, Constructing a good dissertation: a practical guide to finishing a finishing a master's, MBA, or PhD on schedule, 2006.

[Horn, 2001]: P. Horn, Autonomic Computing: IBM’s Perspective on the State of Information Technology, IBM T.J. Watson Labs, NY, 2001.

[Horrocks et al., 2004]: I. Horrocks, P. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Submission, 2004.

[IBM, 2002]: IBM, International Business Machines Corporation, http://www.ibm.com, 2006.

[IBM, 2003]: IBM, Automating problem determination: A first step toward self-healing computing systems, http://www.ibm.com/autonomic/pdfs/Problem_Determination_WP_Final_100703.pdf, 2003.

[IBM,2004]: IBM, Patterns: ServiceOriented Architecture and Web Services, http://www.immagic.com/eLibrary/ARCHIVES/GENERAL/IBM/SG246303.pdf, consulté en 2016

[IBM, 2016]: IBM, http://www.software.ibm.com/wsdd/zones/was/wpc.html, consulté en 2016.

[IEEE, 1996]: S. IEEE, IEEE standard for developing software life cycle process, IEEE Computer Society, New York (USA), 1996.

[IEEE, 2005] : IEEE, IEEE 1471-2000 IEEE Recommended Practice for Architectural Description of Software-Intensive Systems –Description, http://standards.ieee.org/reading/ieee/std_public/description/se/1471-2000_desc.html, 2005.

[InfoDev, 2002]: The Center for Democracy & Technology, Washington: The World Bank, The e-government handbook for developing countries, 2002.

[Invest, 2016] : http://www.invest.gov.ma/, consulté en 2016.

[IPCS, 2003]: IPCS newsletter, 2003

[ISO, 2005] : ISO. Qualité, Normes et ISO 9000.
http://www.jalix.org/ressources/miscellaneous/~vrac/SI/ISO/#_Toc411087330, 2005.

[JADE, 2016] : http://jade.tilab.com/, consulté en 2016.

[Jarras et al., 2002] : I. JARRAS, B. CHAIB-DRAA, Aperçu sur les systèmes multiagents, http://www.cirano.qc.ca/pdf/publication/2002s-67.pdf , 2002.

[Kauster et al., 2005] : U. Kauster, M. Stern, B.Kaonig Ries, A Classification of Issues and Approaches in Automatic Service Composition, 2005.

[Kautz et al. , 1996]: H. Kautz, et B. Selman, Pushing the envelope: planning, propositional logic, and stochastic search, The 13th National Conference on Artificial Intelligence and the 8th Innovative Applications of Artificial Intelligence Conference, pp. 1194–1201, 1996.

[Kellert et al., 2003] : P. Kellert, F. Toumani. Les web services sémantiques, Web sémantique, Action spécifique 32 CNRS/STIC, 2003.

[Kephart et al., 2003]: J. O.Kephart et D. M.Chess. The vision of autonomic computing, IBM Thomas J.Watson Research Center, 2003.

[Kettani ,2014] : D. Kettani et B. Moulin, L'e-gouvernement pour la bonne gouvernance dans les pays en développement : l'expérience du Projet eFez, 2014.

[Khadka et al., 2004] : R. Khadka et B. Sapkota, An Evaluation of Dynamic Web Service Composition Approaches, The 4th International Workshop on Architectures, Concepts and Technologies for Service Oriented Computing, 2010.

[Khalaf et al., 2003]: R. Khalaf, N. Mukhi, and S. Weerawarana. Service-oriented composition in BPEL4WS, The 12th International World Wide Web Conference, 2003.

[Kim et al., 2009]: A. Kim, M. Kang, C. Meadows, A Framework for Automatic Web Service Composition,Naval Research Laboratory, Center for High Assurance Computer Systems, USA 2009.

[Ko et al. , 2009]: R. K.L. Ko, S.S.G. Lee, E. Wah Lee, Business process management (BPM) standards: a survey, Business Process Management Journal, Vol. 15 No. 5, pp. 744-791, 2009.

[Kumar et al., 2009]: K. Kumar , K. Maralla , R. Kumar , M. Thirumaran, A greedy approach with criteria factors for QoS based web service discovery: Proceedings of the 2nd Bangalore Annual Compute Conference on 2nd Bangalore Annual Compute, 2009.

[Kuyoro et al., 2012]: S.Kuyoro, F.Ibikunle, S.Okolie, Security Issues in Web Services, IJCSNS International Journal of Computer Science and Network Security, VOL.12 No.1, 2012

[Lagares et al., 2016] : L.A. Lagares, D. Florian, B. Boualem , Web Service Composition: A Survey of Techniques and Tools,,ACM Computing Surveys (CSUR), Volume 48, No 3, 2016.

[Lambert et al., 2000]:D.M. Lambert, M. C.Cooper, Issues in supply chain management," Industrial Marketing Management, vol. 29, pp. 65-83, 2000.

[Laukkanen et al., 2003]: M. Laukkanen, and H. Helin. Composing workflows of semantic web services, The Workshop on Web-Services and Agent-based Engineering, 2003.

[Layne et al., 2001]: K. Layne, J. Lee, Developing Fully Functional e-Government : A Four Stage Model , Government Information Quarterly, Vol. 18, No. 2, 2001.

[Lenat et al., 1990]: D. B. Lenat, et R .V. Guha, Building large knowledge based systems. Reading, Massachusetts : Addison Wesley, 1990.

[Lenk, 1998]: K. Lenk, Information as a key concept in the theory of public administration: Preliminary remarks. Discussion Paper, University of Oldenburg, Germany, 1998.

[Levesque et al., 1997] : H. J.Levesque, R. Reiter, Y. Lesperance, F. Lin, RB. Scherl GOLOG: a logic programming language for dynamic domains. J Logic Program, pp. 59–83, 1997.

[MacKenzie et al., 2006]: M. MacKenzie, K. Laskey, F. McCabe, P. Brown, R. Metz, Reference Model for Service Oriented Architecture 1.0. Technical Report wd-soa-rm-cd1, OASIS, 2006.

[Madhusudan et al., 2006]: T. Madhusudan , N. A. Uttamsingh, declarative approach to composing Web services in dynamic environments, Decision Support System, Vo. 21, No 2, pp. 325-357, 2006.

[Manning et al. ,2009]: C. D. Manning, P.Raghavan, H.Schütze, An Introduction to Information Retrieval Cambridge University Press, 2009.

[Marchespublics, 2016] : https://www.marchespublics.gov.ma/ , consulté en 2016.

[Martin et al. ¸ 2004]: D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan et K. Sycara, OWL-S: Markup for Web services, W3C Member, http://www.w3.org/Submission/OWL-S/, 2004.

[Martin, 2003]: R. Martin, Agile Software Development: Principles, Patterns, and Practices, Prentice Hall, 2003.

[McDermott, 1998]: McDermott D.V., PDDL, « The Planning Domain Definition Language », Technical Report, Yale University, 1998.

[McDermott, 2002] : D. McDermott, Estimated-regression planning for interactions with web services. the 6th International Conference on AI Planning and Scheduling, Toulouse, France, AAAI Press, 2002.

[McIlraith , 2005] : S. McIlraith, Son TC (2002) Adapting golog for composition of semantic Web Services, The eighth international conference on knowledge representation and reasoning (KR2002),Toulouse, France, 2002.

[McIlraith et al., 2002]: S. McIlraith, and T. Son. Adapting Golog for composition of semantic web services. the 8th International Conference on Knowledge Representation and Reasoning (KR ’02), Toulouse, France, Morgan Kaufmann Publishers, pp. 482–493, 2002.

[Medjahed et al ., 2003]: B. Medjahed, A. Bouguettaya, et A. Elmagarmid. Semantic web enabled composition of web services. The VLDB Journal 12, 4, pp. 333–351, 2003.

[Mikic-Rakic et al., 2002]: M. Mikic-Rakic , N. Mehta , N. Medvidovic, Architectural style requirements for self-healing systems, The 1st Workshop on Self-healing Systems, pp. 49-54, 2002.

[Moser et al., 2010]: O.Moser, F.Rosenbeg, S.Dustdar, Event Driven Monitoring for Service Composition Infrastructures, Web Information Systems Engineering (WISE), 2010.

[Mühl et al., 2006]: G. Mühl,, L. Fiege, P. Pietzuch, Distributed Event Based Systems, Springer, 2006.

[Muller et al ., 2005]: I. Muller, R. Kowalczik, Service composition through agent-based coalition formation, WWW Service composition with Semantic Web Services, Compiegne, France, pp. 34–43, 2005

[Naccache et al., 2007]: H. Naccache, G.C. Gannod, A Self-healing Framework for Web Services, The 2007 IEEE International Conference on Web Services. pp. 398-345, 2007.

[Narayanan et al., 2002]: S. Narayanan, and S. McIlraith. Simulation, verification and automated composition of web services, the 11th international conference on World Wide Web Honolulu, USA, 2002.

[Nau et al,2003] : D.Nau, O.Ilghami, U.Kuter, W.Murdock, D.Wu, F.Yaman, SHOP2: An HTN Planning System, Journal of Artificial Intelligence Research, pp379-404, 2003.

[Newcomer, 2006]: E. Newcomer, Understanding Web Services: XML, WSDL, SOAP, and UDDI, ed. Addison-Wesley, pp.36, 2006.

[Noy et al., 2001]: N. Noy , D. McGuinness , Ontology Development 101: A Guide to Creating Your First Ontology, 2001.

[OCDE, 2015] : Organisation de Coopération et de développement économiques (OCDE), perspective de l'économie numérique de L'OCDE 2015, 2015

[oegov ,2016]: Ontologies for e-Government, http://oegov.org/, consulté en 2016.

[OMG, 2006]: OMG, Meta Object Facility (MOF) Core Specification, 2006.

[Osman et al., 2005]: T. Osman, D. Thakker, and D. Al-Dabass. Bridging the gap between workflow and semantic-based web services composition. In Proceedings of WWW Service composition with Semantic Web Services, pp. 13–23, 2005.

[O'sullivan et al., 2002]: J. O'sullivan, D. Edmond et A.T. Hofstede. What's in a service? Towards accurate description of non-functional service properties, Distributed and Parallel Databases Journal, 2002.

[Paolucci et al. , 2002]: M. Paolucci, T.Kawamura, T.Payne, K.Sycara, Semantic Matching of Web Services Capabilities, First International Semantic Web Conference, Italy, pp. 333-347, 2002.

[Patil et al., 1992]: R.S. Patil, R.E. Fikes, P.F. Patel-Schneider, D. MacKay, T. Finin, T. Gruber, & R. Neches, The DARPA Knowledge Sharing Effort: Progress Report, KR'92 - The Annual International Conference on Knowledge Representation, 1992.

[Pednault ,1989]: E. Pednault, ADL: Exploring the middle ground between STRIPS and the situation calculus, in: Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning (KR−89), pp. 324−332,Toronto, Canada, 1989.

[Peer, 2004]: J. Peer. A PDDL based tool for automatic web services composition, The Second Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR 2004) at the 20th International Conference on Logic Programming, Springer Verlag, pp. 149–163, 2004.

[Peer, 2005]: J. Peer, Web service composition as planning - a survey. Tech. rep., Univ. Of St.Gallen, 2005.

[Pellier, 2005] : D. Pellier, Modèle dialectique pour la synthèse de plans, thèse de doctorat, IMAG - LEIBNIZ, 2005.

[Peltz, 2003]: C.Peltz, Web services orchestration - a review of emerging technologies, tools, and standards. Tech. rep., HP, 2003.

[Pessoa et al., 2008]: R. Mantovaneli Pessoa, E. Silva, Marten van Sinderen, D. A. C. Quartel, L. Ferreira Pires, Enterprise Interoperability with SOA: a Survey of Service Composition Approaches, in Workshop on Enterprise Interoperability, pp. 32-45, 2008.

[Pistore et al., 2005]: M. Pistore, F. Barbon, P. Bertoli, D. Shaparau, P. Traverso ,Planning and Monitoring Web Service Composition, the International Conference on Artificial Intelligence, Methodology, Systems, and Applications, 2005.

[Pistore et al., 2004]: M. Pistore, F.Barbon , P. Bertoli, D. Shaparau, P.Traverso , Planning and monitoring Web service composition, The artificial intelligence: methodology, systems, and applications, 11th international conference (AIMSA), Varna, Bulgaria pp. 106–115, 2004.

[Ponnekanti et al., 2002]: SR. Ponnekanti, A. Fox, Sword: A developer toolkit for Web service composition, The 11th World Wide Webconference (Web Engineering Track), Honolulu, Hawaii, USA, 2002.

[Popper, 1963] : K.Popper, Conjectures and Refutations, 1963.

[Poslad,2006]: S. Poslad, Review of FIPA Specifications, IEEE FIPA Revision of FIPA Specifications Group, Foundation for intelligent Physical Agents http://www.fipa.org , 2006.

[Protégé, 2016] : Protégé, The Protege Project, http://protege.stanford.edu, consulté en 2016.

[Puttonena et al., 2015] J.Puttonena, A. Lobova, M. Sotob, J. Lastraa, Planning-based semantic web service composition in factory automation, Advanced Engineering Informatics, Vol:29, Num: 4, pp:1041–1054, 2015

[Québec,2014] : Architecture d'entreprise gouvernementale 3.0, Bibliothèque et Archives nationales du Québec , 2014.

[Rao et al ., 2003]: J. Rao, P. Kungas, M. Matskin, Application of linear logic to web service composition, The 1st International Conference on Web Services, Las Vegas, USA, 2003.

[Rao et al., 2004]: J. Rao et X. Su, A Survey of Automated Web Service Composition Methods, The First International Workshop on Semantic Web Services Web Process Composition, 2004.

[Rimassa et al., 1999] : G.RIMASSA, F.BELLIFEMINE et A.POGGI A, JADE - A FIPA Compliant Agent Framework, PMAA`99, pp. 97-108, 1999.

[Rodriguez-Mier et al., 2011]: P. Rodriguez-Mier, M. Mucientes, M. Lama, Automatic web service composition with a heuristic-based search algorithm, IEEE International Conference on Web Services, 2011

[Roman et al., 2005]: D. Roman, U. Keller, H. Lausen, J. Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, et D. Fensel: Web Service Modeling Ontology, Applied Ontology, pp.77 - 106, 2005.

[Rosenberg, 2009]: F. Rosenberg, QoS-Aware Composition of Adaptive Service-Oriented Systems, 2009.

[Russell, 2009] : S. J Russell , et P. Norvig. Artificial intelligence : a modern approach, Prentice hall, 2009.

[Schneider et al., 2014]: C. Schneider, A. Barker, S. Dobson ,A survey of self-healing systems frameworks, SOFTWARE –PRACTICE AND EXPERIENCE, publié dans Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/spe.2250, 2014

[Scholl et al., 2007]: H.J. Scholl, R. Klischewski, E-Government Integration and Interoperability: Framing the Research Agenda, International Journal of Public Administration, pp :889−920, 2007.

[Segond, 2006]: M. Segond, Algorithmes biomimétiques pour la reconnaissance de formes et l'apprentissage, 2006.

[Seiffert, 2002]: J.W. Seiffert, A primer on e-governement: sectors, stages opportunities, and challenges of online governance. CRS report for Congress, 2002.

[Service-public, 2016] : http://www.service-public.ma/ , consulté en 2016.

[Shalini et al., 2014] L. Shalini, R. Femila Goldy, Efficient Discovery of Semantic Web Services, International Journal of Innovative Research in Science, Engineering and Technology, 2014

[Sharrock, 2010] : R. Sharrock Gestion autonomique de performance, d'énergie et de qualité de service. Application aux réseaux filaires, réseaux de capteurs et grilles de calcul, 2010.

[Sheshagiri et al ., 2003]: M. Sheshagiri, M. Desjardins, and T. Finin, A planner for composing services described in DAML-S, The AAMAS Workshop on Web Services and Agent-based Engineering, 2003.

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

[Sirin et al ., 2004]: E. Sirin, and B. Parsia. Planning for semantic web services. Semantic Web Services Workshop at 3rd International Semantic Web Conference, 2004.

[Sirin et al., 2004]: E. Sirin, B. Parsia, Planning for Semantic Web Services, in: Semantic Web Services Workshop at 3rd International Semantic Web Conference, 2004.

[SL, 2016] : http://www.fipa.org/specs/fipa00037/SC00037J.html#_Toc26729715, consulté en 2016.

[Smola et al., 2008]: A. Smola et S.V.N. Vishwanathan, Introduction to machine learning, thèse, Cambridge University, 2008.

[Sowa, 1999]: J.F. Sowa, Knowledge3, CA, 1999.

[Srinivasan et al., 2004]: N. Srinivasan, M.Paolucci, and K. Sycara. An efficient algorithm for owl-s based semantic search in uddi. In SWSWPC, pp. 96–110, 2004.

[Srinivasan, 2006]: N. Srinivasan, M.Paolucci, K.P. Sycara, Semantic Web Service Discovery in the OWL-S IDE. HICSS, 2006.

[Staab et al., 2001]: S. Staab, HP. Schnurr , R. Studer , Y. Sure, Knowledge Processes and Ontologies.IEEE Intelligent Systems 16(1), pp. 26–34, 2001.

[St-Amant, 2004] : G. St-Amant, Gouvernement en ligne : cadre d'évolution de l'administration électronique, Université du Québec à Montréal, 2004.

[Stanfill et al., 1986]: C. Stanfill et D. Waltz, Toward memory-based reasoning, Communications of the ACM, pp. 1213–1228, 1986.

[Steinmetz, 2008]: N. Steinmetz, Simplifying the Web Service Discovery Process. 1st Workshop on Semantic Metadata Management and Applications (SeMMA '08), 2008.

[Sterritt et al. , 2003]: R. Sterritt, et D. Bustard, Autonomic Computing-a means of achieving dependability ?, The10th IEEE International Conference and Workshop on the, pp. 247–251, 2003.

[Sterritt, 2005]: R. Sterritt, Autonomic computing, Innovations in systems and software engineering, pp.79–88, 2005.

[Sun et al., 2003]: H.Sun, X.Wang, B. Zhou, P.Zou, Research and Implementation of Dynamic Web Services Composition, APPT 2003, LNCS 2834, Springer-Verlag Berlin Hbidelberg, pp.457—466, 2003

[Sutton et al., 1986] : R. S . Sutton, & A.G Barto , Reinforcement learning: An introduction, Cambridge, MA: MIT Press, 1998.

[Swarnamugi et al., 2010]: M. Swarnamugi, M. Sathya, P. Dhavachelvan, A Negotiation Model for Web Service Selection, International Conference on Recent Trends in Soft Computing and Information Technology, pp. 251–256, Bhopal, India, 2010.

[Talantikite et al., 2008]: H. Talantikite, D. Aissani, N. Boudjlida.“ Semantic annotations for web services discovery and composition”. Journal of Computer Standards & Interfaces, Elsevier, pp. 1 – 10, 2008.

[Tang et al., 2013]: X.Tang, F. Tang, L. Bing et D. Chen , Dynamic web service composition based on service integration and HTN planning, IEEE Computer Society, pp: 307–312, 2013.

[Taylor et al., 2003]: J. TAYLOR, Matthew S. SHIELDS, I. WANG, R.PHILP : Distributed p2p computing within triana : A galaxy visualization test case, 17th International Parallel and Distributed Processing Symposium (IPDPS 2003), pp.22-26, 2003

[Thatte, 2001]: Thatte, S. XLANG - web services for business process design, 2001.

[Tian et al., 2003]: M. Tian, A. Gramm, T. Naumowicz, H. Ritter, J. Schiller, A concept for QoS integration in web services. In Proceedings of WQW2003, pp.149–155, 2003.

[Tianfield, 2003]: H. Tianfield, Multi-agent based autonomic architecture for network management, Industrial Informatics, 2003. IEEE International Conference on, pp. 462–469. 2003.

[Todica et al., 2012]: V. Todica, M. Cremene, M.Vaida, Using Machine Learning in Web Service Composition, The Fourth International Conferences on Advanced Service Computing, 2012.

[Tong et al., 2011]: H.Tong , J.Cao , S.Zhang , M.Li, A Distributed Algorithm for Web Service Composition Based on Service Agent Model, IEEE Transactions on Parallel and Distributed Systems, pp.2008 – 2021, 2011.

[Trainotti et al., 2000]: M. Trainotti, M. Pistore , G. Calabrese, G. Zacco, G. Lucchese, F. Barbon, P. Bertoli, T, ASTRO: supporting composition and execution of Web services, The international conference on automated and planning sheduling (ICAPS). Demo, Monterey, CA, USA, 2000.

[UK, 2005]: e-Government Unit London, UK: Cabinet Office, E-Government interoperability framework, 2005.

[UN, 2008]: United Nations, Department of economic and Social Affairs, Division for Public Administration and Development Management, The Global e-Government Survey 2008, 2008

[UN, 2010]: United Nations, E-government survey 2010: Leveraging e-government at a time of financial and economic crisis, https://publicadministration.un.org/egovkb/Portals/egovkb/Documents/un/2010-Survey/Complete-survey.pdf, 2010,

[UN, 2012]: United Nations, E-government survey 2012: E-Government for the People, https://publicadministration.un.org/egovkb/Portals/egovkb/Documents/un/2012-Survey/Complete-Survey.pdf, 2012,

[UN, 2014]: United Nations, E-government survey 2014: E-Government for the Future We Want, https://publicadministration.un.org/egovkb/portals/egovkb/documents/un/2014-survey/e-gov_complete_survey-2014.pdf, 2014,

[UN, 2015]: United Nations, Transforming Our World: The 2030 Agenda For Sustainable Development, https://sustainabledevelopment.un.org/content/documents/21252030%20Agenda%20for%20Sustainable%20Development%20web.pdf, 2015,

[UN, 2016]: United Nations, E-government survey 2016: E-Government In Support Of Sustainable Development, http://workspace.unpan.org/sites/Internet/Documents/UNPAN96407.pdf, 2016.

[Urs, 2012] : Nicolae Urs, E-government, thèse, université de Babes-Bolyai , Romania, ,2012

[Uschold et al.,1995]: M. Uschold M, M. King, Towards a Methodology for Building Ontologies. In: Skuce D (eds) IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing. Montreal, Canada, pp. 6.1–6.10, 1995.

[Uschold, 1996]: M. Uschold, Building Ontologies: Towards a Unified Methodology, 16th Annual Conference of the British Computer Society Specialists Group on Expert Systems, Cambrige, UK, 1996.

[Van Heijst et al., 1997]: G.Van Heijst, A.Schreiber, B.Wielinga, Using explicit ontologies in kbs development. Int. J. of Human-Computer Studies, 1997.

[VELASCO, 2008]: C.LOPEZ-VELASCO, Sélection et composition de services Web pour la génération d'applications adaptées au contexte d'utilisation, 2008

[Vintar et al., 2002]: M.Vintar, M.Kunstelj, A.Leben ,Delivering better quality public services through life-event portals, NISPAcce Annual Conference, 2002

[W3C, 2002]: World Wide Web Consortium, http://www.W3C.com, 2002.

[W3C, 2003]: OWL Web Ontology Language Reference, http://www.w3.org/TR/owl-ref/, 2003.

[W3C, 2004]: Rdf primer, http://www.w3.org/TR/rdf-primer/,2004.

[W3C, 2005]: Semantic Web Services Description Based on Web Services Description, https://www.w3.org/2005/04/FSWS/Submissions/5/SZTAKI_FSWS_050420.htm, 2005.

[W3C, 2007]: W3C, Semantic Web Layer Cake, http://www.w3C.org/2007/03/layerCake.png, 2007.

[W3C, 2008]: World Wide Web Consortium, https://www.w3.org/TR/2008/REC-xml-20081126/, 2008.

[W3C2, 2004]: W3C, OWL Web Ontology Language- Overview, https://www.w3.org/TR/owl-features/, 2004.

[W3C2, 2008]: W3C, SPARQL Query Language for RDF (W3C Recommendation 2008-01-15), http://www.w3.org/TR/rdf-sparql-query/, 2008.

[Wache et al., 2001]: H. Wache , T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, S. Hübner , Ontology-Based Integration of Information –A Survey of Existing Approaches. Proc. IJCAI-01 Workshop: Ontologies and Information Sharing, Seattle, WA, pp. 108-117, 2001.

[Waltinger et al., 2000]: R. Waltinger. Web agents cooperating deductively. FAABS'00, Greenbelt, Ed., vol. 1871 of Lecture Notes in Computer Sciences, Springer-Verlag, pp. 250–262, 2000.

[Wang, 2013]: Y.Wang, A Survey on Formal Methods for Web Service Composition, Software Engineering, 2013.

[Watiqa, 2016] : https://www.watiqa.ma/, consulté en 2016.

[Weiss, 2004]: G.Weiss, Multiagent systems, In Morgan kaufman Publishers. 2004.

[Whatis, 2016]: Information society, http://whatis.techtarget.com/, consulté en 2016.

[Wielinga et al., 1992]: B.J. Wielinga, A. Schreiber et J. Breuker, Kads : A modelling approach to knowledge engineering. Knowledge Acquisition, Special issue "The KADS Approach to Knowledge Engineering", 1992.

[Wikipédia, 2016] : Wikipédia, https://fr.wikipedia.org/wiki/Service_Web, consulté en 2016.

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

[Wu et al., 2005] : D. Wu, E. Sirin, J. Hendler, D. Nau , B. Parsia , Automatic Web Services composition using SHOP2, The 13th international conference on automated planning and scheduling. Workshop on planning for Web Services, Trento, Italy, 2003.

[Yu et al., 2011]: F. Yu, Z. Chen, D. Xie, A Method for Semantic Web Service Selection Based on QoS Ontology, JOURNAL OF COMPUTERS, VOL. 6, NO. 2, 2011.

[Zhao et al., 2009]: H. Zhao et P.Dosh, Towards Automated RESTful Web Service Composition, IEEE International Conference on Web Services (ICWS), 2009.

[Zhou et al., 2004]: C. Zhou, L.Chia.,B. Lee, Daml-qos ontology for web services. In Proceedings of ICWS, pp. 472–479, 2004.

[Zuniga et al., 2010] : J.Zuniga, J. Alcazar, L. Digiampietri, Implementation Issues for Automatic Composition of Web Services, Workshops on Database and Expert Systems Applications, pp. 201-205, 2010.