REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MOHAMED KHIDER - BISKRA

الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
جامعة محمد خيضر بسكرة

Faculté des sciences exactes et des sciences de la nature et de la vie
Département d'informatique

THESE

Présentée pour l'obtention du diplôme de

DOCTORAT EN SCIENCES

Spécialité : Informatique

Présentée par :

Ahmed Bali

Thème

UNE APPROCHE BASÉE AGENTS POUR L'ADAPTATION DES SERVICES WEB

Soutenue publiquement le : **devant le jury composé de :**

Président :	**Pr. Laskri Mohamed Tayeb**	Université **d'Annaba**
Rapporteur :	**Pr. Kazar Okba**	Université **de Biskra**
Examinateurs :		
	Pr. Kimour Mohamed Tahar	Université **d'Annaba**
	Pr. Bousaid Oumar	Université Lyon2**, France**
	Dr Hamadi Bennoui	Université **de Biskra**
	Dr. Benabdelaziz Rachida	Université **de Biskra**

Je dédie cet humble travail

A mes très chers parents, que Dieu tout puissant les protège

A ma femme, mes enfants et toute ma famille qui sont toujours présents dans mes pensées

A mes amis ainsi que tous les gens qui m'ont aidé de près ou de loin à accomplir ce travail

A la mémoire de mon ami Adel Titous.

Remerciements

Je remercie **ALLAH** tout puissant de m'avoir donné la force et la patience nécessaire pour achever ce travail de thèse.

Ma profonde gratitude et mes sincères remerciements vont à mon directeur de recherche M. KAZAR Okba, professeur à l'université de Biskra, pour m'avoir proposé ce sujet et m'avoir dirigé, conseillé et encouragé tout au long de la réalisation de ce travail ainsi que pour sa patience et sa compréhension. Veuillez croire à l'expression de ma grande admiration et mon profond respect.

Je tiens à remercier également le codirecteur de ce travail, M. Abdelouahed Gherbi, professeur au département génie logiciel de l'École de Technologie Supérieure (ÉTS), au Canada, qui m'a honoré de sa confiance et du meilleur accueil qu'il m'a réservé malgré ses nombreuses occupations. J'ai particulièrement été impressionné par ses qualités scientifiques et humaines.

Je remercie beaucoup les membres du jury de m'avoir fait l'honneur d'accepter de participer à mon jury de thèse.

ملخص

في الوقت الحاضر، يتوقع مستخدمي تطبيقات الحوسبة (بما في ذلك خدمات الويب) أن تكون لها القدرة على التكيف مع سياق استخدامها، وذلك يعتمد أساسا على معرفة احتياجات المستخدم، وأجهزته (أجهزة الكمبيوتر الشخصية، والهاتف الذكي، الخ) وتغيرات بيئته (أساسا الشبكات).

في هذا العمل، ركزنا على مشكلة التكيف من أجل اقتراح حلين للإجابة على جزئيتين رئيسيتين: كفاءة إدارة السياق وتكييف تركيب خدمات الويب. وذلك انطلاقا من وجهة نظر هيكلية الخدمات الموجهة والمتمثلة في أن التطبيقات تتكون من الخدمات.

أولا، بيئات التطبيقات موجودة في كل مكان عموما. بحيث تكون متغيرة وذات طبيعة موزعة. ونتيجة لذلك، تواجه إدارة السياق مشكلة عدم التجانس، ومحدودية قدرة الأجهزة وعرض النطاق الترددي للشبكة. وللتغلب على هذه الصعوبة، يتطلب توفير إدارة سياق مخصصة لتجميعه وتحديثاته وتغيراته. لهذا السبب، اقترحنا مقاربة لإدارة السياق تعتمد على التمثيل الدلالي للسياق باستخدام لغات قياسية مثل RDF و RDF(S). كما تتميز مقاربتنا بأن لها قدرا كبيرا من المرونة والبساطة ما يسمح لمطور الخدمة، بالتعبير عن السياق دون الحاجة لنموذج إطار موحد. وبالإضافة إلى ذلك، أدخلنا فكرة سجل التغيرات، والتي تحافظ على التغيرات ذات معنى في السياق فقط. و لتحقيق الأجزاء النشطة من مخططنا المقترح، ونحن اعتمدنا على مفهوم الاعوان (agents) نظرا لفعاليتها في النظم الموزعة.
نتائج التجريب المستخرجة باستخدام نموذج لمنصة تدعم المقاربة المقترحة، كانت بشكل واضح تأكد فكرة مقاربتنا في حالة الأجهزة محدودة الموارد والشبكة.
ثانيا، اقترحنا مقاربة لتكييف تركيب خدمات الويب، حيث أن تركيب خدمات الويب يسمح بتوليد عمليات تجارية من أجل تلبية احتياجات طلبية المستخدم. ويكون التركيب ضروري عند عدم وجود خدمة ويب أحادية يمكن تلبية الطلب.
وفي هذا السياق، فقد ركزنا على مشكلة التركيب لتلبية جميع المتطلبات الوظيفية وغير الوظيفية (كنوعية جودة الخدمة QoS)، مع التقليل من عدد الخدمات الداخلة في التركيب.
مساهمتنا الأساسية هي تحسين بنية بيان شبكة التخطيط، فهو أحد تقنيات الذكاء الاصطناعي التي تعتبر الأكثر كفاءة في حل مشكلة تكوين خدمة ويب. وقد اظهرت النتائج تحسنا كبيرا في الوقت وحتى في عدد الخدمات الداخلة في التركيب، وذلك حتى في حالة العمل مع نطاق واسع من خدمات ويب.

كلمات البحث

التكيف، إدارة السياق، علم الوجود، عون، نظام متعدد الاعوان(SMA)، خدمة ويب، جودة الخدمة(QoS)،تركيب خدمة ويب الموجهة بنوعية الخدمة، تقنيات التخطيط

Abstract

Nowadays, users of computing applications (including web services) expect them to be adaptive to the context of use, which is consists essentially of information about user needs, devices (personal computer, SmartPhone, etc.) and the dynamics of the environment (mainly networks).

In this work, we are interested in this problem of adaptation in order to provide two solutions that answer two major sub-issues: efficient context management and adaptation of the Web service composition. Since in the perspective of service-oriented architecture, applications consist of services.

First, application environments are generally ubiquitous. So they are dynamic and distributed in nature. Therefore, the context management faces the problem of heterogeneity of the representations, the limitation of devices capacity and the network bandwidth. In this thesis, we propose an approach to the context management in ubiquitous environments based on the semantic representation of the context and its update.

For this reason, we proposed a context management approach based on a semantic representation of the context using standard languages such as RDF and RDF(S). Our approach is characterized by its great flexibility, simplicity and allows the developer of the service to express the context without the need for a unified context model. In addition, we introduced the notion of change log, which only keeps significant context changes. To realize the active parts of our architecture, we used the agent technique on account of their efficiency in distributed systems. The results of the experimentation we carried out, using a prototype implementation of middleware supporting the proposed approach, are clearly in favour of using our approach in the case of resource-constrained devices and network.

Second, to adapt the service, we proposed a new approach to the composition of Web service (WSC). The WSC allows to generate a business process in order to satisfy the needs of a user's request when a single atomic Web service cannot satisfy the request. In this context, we have focused on the composition problem satisfying both functional requirements and non-functional requirements (quality of service QoS), while minimizing the number of services of the composition.

Our basic contribution is the improvement of the planning graph structure. The planning graph is one of the artificial intelligence planning techniques that are considered to be the most efficient in solving a Web service composition problem. Experimental results show significant improvement in the runtime and even in the number of services when working with large-scale services.

Keywords

Web service Adaptation, Context management, Ontology, Agent, SMA, Quality of service (QoS), QoS-aware Web service composition, Planning Graph

Résumé

Aujourd'hui, les utilisateurs des applications informatiques (inclus les services Web) attendent qu'elles soient adaptatives au contexte d'utilisation qui est constitué essentiellement en des informations concernant les besoins d'utilisateur, ses dispositifs (ordinateur personnel, SmartPhone, etc.) et la dynamique de l'environnement (essentiellement les réseaux).

Dans ce travail, nous nous sommes intéressés à ce problème d'adaptation afin de fournir deux solutions qui répondent à deux sous-problématiques majeures, à savoir : la gestion efficace du contexte et l'adaptation de la composition de service Web. Puisque dans l'optique de l'architecture orientée service les applications se composent des services.

Premièrement, les environnements des applications sont généralement ubiquitaires. Ils sont, alors, dynamiques et de nature distribuée. Par conséquent, la gestion du contexte rencontre le problème de l'hétérogénéité des représentations, de la limitation de la capacité des dispositifs et de la bande passante du réseau. Afin de surmonter cette difficulté, une gestion particulière du contexte est requise pour sa collecte, sa mise à jour et son échange. Pour cela, nous avons proposé une approche de gestion de contexte basée sur une représentation sémantique du contexte en utilisant des langages standard tels que RDF, RDF(S). Notre approche est caractérisée par sa grande flexibilité, simplicité et permet au développeur du service de mettre son point de vue concernant le contexte que doit être utilisé tout en évitant l'exigence d'utiliser un modèle de contexte unifié. De plus, nous avons introduit la notion du journal de changements, qui garde seulement les changements de contexte significatifs. Pour réaliser les parties actives de notre architecture, nous nous basons sur la technique des agents grâce à leur efficacité dans les systèmes distribués.

Les résultats de l'expérimentation que nous avons réalisée en utilisant une implémentation de prototype d'un middleware supportant l'approche proposée sont clairement en faveur de l'utilisation de notre approche dans le cas des dispositifs et des réseaux à ressources limitées.

Deuxièmement, pour l'adaptation de service nous avons proposé une nouvelle approche de la composition de service Web (WSC). La WSC permet de générer un processus d'affaires afin de satisfaire les besoins d'une requête d'un utilisateur lorsqu'un seul service Web atomique ne peut pas satisfaire les besoins d'une requête donnée. Dans cette optique, nous nous sommes intéressés au problème de composition permettant de satisfaire à la fois les exigences fonctionnelles et les exigences non fonctionnelles (qualité de service QoS), tout en minimisant le nombre de services de la composition. Notre contribution de base est l'amélioration de la structure de graphe de planification. Le graphe de planification est l'un des techniques de planification en intelligence artificielle qui sont considérées comme les plus performantes pour résoudre le problème de la composition de service Web.

Les résultats expérimentaux montrent une amélioration significative du temps d'exécution de la composition et même du nombre de services lorsqu'on travaille sur une grande échelle des services (jusqu'au 20000 de services).

Mots Clés

Adaptation de services Web, Gestion de contexte, Ontologie, Agent, SMA, Qualité de. service (QoS), Composition de service Web sensible au QoS, Graphe de planification

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ASC	Automated Web Service Composition
BPEL	Business Process Execution Language
FTP	File Transfer Protocol
HTML	Hyper Text Mark-Up Language
HTTP	Hyper Text Transfer Protocol
IA	Intelligence Artificielle
IGP	Improved Graph's Propositions
ITPG	Improved Tracking Planning Graph
MPG	Minimal Planning Graph
NGP	New Graph's Propositions
OGP	Old Graph's Propositions
OWL	Web Ontology Language
PG	Planning Graph
QoS	Quality of Service
RDF	Resource Description Framework
RT	Response Time quality criterion
SMA	Système multi-agents
SMTP	Simple Mail Transfer Protocol
SOA	Service-Oriented Computing
SOAP	Service-Oriented Architecture
SOC	Service-Oriented Computing
TH	Throughput quality criterion
TPG	Tracking Planning Graph
UDDI	Universal Description, Discovery and Integration

URI	URI : Uniform Resource Identifier
W3C	World Wide Web Consortium
WS	Web Service
WSC	Web Service Composition
WSDL	Web Services Description Language
WSLA	Web Service Level Agreement
XML	eXtensible Markup Language

CHAPITRE 1

INTRODUCTION GÉNÉRALE

Au cours des dernières années, l'utilisation du Web s'est élargie très rapidement imprégnant de plus en plus le quotidien des utilisateurs. Ces derniers peuvent faire un grand nombre de leurs activités en ligne, telles que les achats, la consultation de météo, la lecture des news, la réservation d'un billet d'avion ou d'une table dans un restaurant.
L'apparition des technologies du Web permet donc aux organisations de proposer à l'utilisateur leurs services à travers l'Internet. Cependant, dans l'utilisation des services Web, et plus précisément quand un demandeur de service envoie une requête à un service Web, il attend souvent que le service soit adaptatif au contexte d'utilisation qui constitué essentiellement en des informations concernant les besoins d'utilisateur, ses dispositifs (ordinateur personnel, SmartPhone, etc.) et la dynamique de l'environnement (essentiellement les réseaux).

1.1 Contexte général

La prise en compte du contexte d'utilisation dans les applications est un domaine de recherche d'actualité connu sous le nom de « sensibilité au contexte » (context-awareness).
Nous présentons le contexte de notre travail, en expliquant brièvement les concepts suivants.

1.1.1 Service Web

La notion de service s'est fortement développée sur le Web où les services Web sont les composants de base de tout traitement dans une architecture orientée service (SOA). Cette dernière (SOA) est un style architectural, qui se préoccupe à faire communiquer des composants logiciels distants et hétérogènes afin d'atteindre un but précis. Elle se base sur le principe de la description, la publication, la recherche et l'invocation des services.

Grâce au mécanisme des services Web, les applications pourraient inter-opérer via le réseau (souvent Internet), indépendamment des plateformes sur lesquels ils tournent et indépendamment des langages de programmation avec lesquels ils sont codés. En plus, les services Web ont été dévelop-

pés en une technologie mature et sont basés sur des normes établies telles que SOAP[W3C (2007a)], WSDL[W3C (2007b)], UDDI[OASIS (2002)], WSLA[Keller & Ludwig (2003)] et BPEL[OASIS (2007)].
En raison de son importance, le concept de services Web a représenté, au cours de la dernière décennie, le noyau d'un domaine de recherche dynamique et actif, et a motivé plusieurs chercheurs.

1.1.2 Qualité des services (QoS)

L'utilisation des services web nécessite une étape préliminaire de découverte, qui consiste à rechercher et localiser des services web correspondants aux besoins du client qui sont exprimés au moyen d'une requête définissant les attentes fonctionnelles et non-fonctionnelles (qualité de service : QoS).

Plusieurs critères de qualités de services peuvent être prises en considération comme le temps de réponse (response time), le débit (throughput), le coût (cost), la réputation (reputation), la fiabilité (reliability), taux d'exécution réussie (successful execution rate) et la disponibilité (availability).

Ces différents critères de qualité de service sont divisés en deux catégories, à savoir les critères positives et ceux négatifs. Dans le cas d'un critère négatif, le service ayant la valeur la plus haute est le service ayant la qualité la plus basse. Par contre, dans le cas d'un critère positif, le service ayant la valeur la plus haute est le service ayant la qualité la plus haute.

Dans notre travail, nous avons choisi de chaque catégorie une qualité de services, à savoir le temps de réponse (Response time) comme un critère négatif de qualité et le débit (Throughput) comme un critère positif. Ce choix de deux critères de qualité de service, est motivé aussi par l'existence d'une base de données des services que la plupart de travaux voisins l'utilisent. Cette base de données utilise le temps de réponse et le débit comme critère de QoS.

1.1.3 composition de service Web (WSC- Web service composition)

Comme nous avons mentionné que les besoins du demandeur de service sont exprimés souvent en termes d'attentes fonctionnelles et de contraintes non-fonctionnelles (QoS), chose qui rend souvent difficile voire impossible de trouver un service Web atomique pouvant réaliser la tâche demandée

et satisfaire les contraintes de qualité. Dans ce cas-là, il est nécessaire de composer un service à partir de ceux qui sont déjà existants.

La composition qui vise à satisfaire les objectifs fonctionnels et à optimiser la QoS, est connue sous le nom de la composition de service sensible à la QoS (QoS-aware service composition) [Chen & Yan (2012)].

Du fait du nombre important des services Web disponibles sur le Web, il est difficile, voire impossible, de les composer de façon manuelle, c'est la raison pour laquelle la composition doit être automatique, cela est connu sous le nom « Automated Web Service Composition (ASC)".

Pour cela, le problème de la composition de service Web (WSC) est l'un des défis de recherche : lorsqu'un seul service Web ne satisfait pas à certaines exigences, l'objectif est alors de combiner automatiquement le service Web d'autres services pour répondre entièrement à ces exigences [Kil (2010)].

Or, les techniques de planification [LaValle (2006); Russell (2010); Ghallab *et al.* (2004)] en intelligence artificielle sont souvent considérées comme les plus efficaces et même prédominantes pour résoudre automatiquement des problèmes de compositions [Yan *et al.* (2012)].

Les techniques de planification que nous avons utilisée dans notre travail se basent sur l'idée de transformer un problème de planification en un problème d'exploration. L'espace d'exploration est défini via une structure de données spécifiques, intitulée graphe de planification permettant la définition d'un espace de recherche efficace, dont l'exploration permet de connaître l'existence de la solution et même son extraction. Toutefois, l'algorithme de l'extraction de la solution est souvent plus coûteux en terme de temps d'exécution. Pour cette raison, des techniques d'optimisation sont souvent utilisées.

1.1.4 Adaptation des services Web

L'utilisation croissante d'Internet par des utilisateurs ayant des profils variés a par conséquence un besoin croissant d'adaptation. Ce besoin d'adaptation avec le succès des services Web fait émerger

les travaux concernant l'adaptation des services Web.

Dans la littérature, les travaux sur l'adaptation s'articulent autour de quatre dimensions, à savoir :

- *L'adaptation du résultat de service Web*

  Vise à adapter le contenu final du résultat de l'invocation de service Web en ajoutant ou éliminant des données qui sont jugées pertinentes ou non respectivement. La détermination de la pertinence des données est reliée aux informations de contexte telles que les buts de l'utilisateur et ses caractéristiques.

- *L'adaptation de la présentation de service Web*

  Vise à effectuer des transformations au niveau de l'affichage. En suivant les préférences de l'utilisateur et son contexte, des adaptations sont appliquées sur les caractéristiques visuelles du document présenté à l'utilisateur tel que l'adaptation des caractéristiques graphiques, de la langue de présentation et de l'organisation du document.

- *L'adaptation du comportement*

  Vise à adapter le comportement des services suivant des changements de l'environnement d'exécution. L'adaptation du comportement du service consiste à changer la logique d'un service. Elle concerne l'occurrence et l'ordre des étapes d'exécution d'un service, ainsi que les relations logiques entre ces différentes étapes [Lemlouma (2004)].

- *L'adaptation de la composition de service*

  Vise à adapter la composition de service Web en se basant sur deux aspects, à savoir : le modèle de l'utilisateur et la tâche à effectuer.

Dans notre travail, nous allons nous concentrer sur l'adaptation de la composition de service Web. Cela, est motivé par plusieurs raisons, entre autres, l'importance de la composition de service Web, les deux premiers types d'adaptation (résultat et présentation) sont bien abordés grâce aux travaux de la communauté des hypermédias adaptatifs et solutions techniques telle que les feuilles de style (CSS- Cascading Style Sheets)[wikipedia (2017)] et pour le troisième type d'adaptation (comportement), la représentation interne du service (sa logique) est souvent indisponible.

Dans la littérature, la catégorie de l'adaptation de la composition de service Web, basée sur la réalisation d'une nouvelle composition, est plus favorisée que celle basée sur le changement de la composition actuelle. Cela est expliqué par le fait de recalculer, au besoin, la composition en

fonction de l'état actuel du contexte, permet de donner une composition plus pertinente que celle obtenue par un changement d'une composition déjà calculée pour un autre état de contexte qui est différent de l'état actuel. Sachant que théoriquement modifier une composition existante n'est pas efficace qu'une recomposition complète dans le pire des cas [Yan *et al.* (2010)].

Dans ce sens, tous les travaux sur la composition de service peuvent servir l'adaptation des services Web.

Dans la sous-section suivante, nous allons présenter les problèmes révélés pour l'adaptation des services et les limites présentes dans les travaux traitants ces problèmes.

1.2 Problématique et motivations

La problématique de notre travail rentre dans le cadre de la problématique globale de l'informatique sensible au contexte qui exige une adaptation permanente des applications au contexte d'utilisation c.-à-d. les applications s'adaptent selon plusieurs éléments comme la capacité des dispositifs qu'ils ont à disposition des utilisateurs et à leurs localisations.

Dû de l'importance du contexte dans l'opération de l'adaptation des services Web, nous allons présenter les limites actuelles reliées à la gestion du contexte. Par la suite, nous allons présenter celles qui concernent l'adaptation des services Web, plus précisément la composition de service Web.

1.2.1 Gestion de contexte

Actuellement, l'informatique ubiquitaire permet l'intégration des technologies informatiques au quotidien de l'homme le plus simplement possible, dans nombreux domaines tels que l'information, la formation, le transport ou encore la médecine. Cependant, l'environnement ubiquitaire est caractérisé par l'utilisation des divers types de dispositifs (PC, smartphone, etc.) et la mobilité des utilisateurs.

Toutefois, simplifier le développement d'applications sensibles au contexte est encore considéré comme un défi. Pour cette raison, il est préférable, comme dans [Ejigu *et al.* (2007), Jones (2008)],

que la gestion du contexte soit indépendante des autres acteurs (c'est-à-dire de mécanisme d'adaptation et des fournisseurs de services). De plus, cette séparation permet à l'adaptation de service d'être indépendant, le plus possible, du domaine tel que dans [Kapitsaki *et al.* (2008)].

La nature distribuée de l'environnement ubiquitaire et la mobilité des dispositifs utilisés qui sont souvent ont une capacité limitée (qui ne permet pas de gérer un grand volume de connaissances). Par conséquent, la prise en considération de ces contraintes, lance des défis supplémentaires que les travaux actuels sur la sensibilité au contexte ne se concentrent pas sur eux ou même sont ignorés.

Dans cette optique, les points suivants mettent en évidence, les problématiques concernant la gestion du contexte de plus les motivations comme objectives pour surmonter ces problématiques :

- *Capture de contexte :* Comment récupérer uniquement les éléments de contexte qui sont pertinents pour la détermination du comportement du service. En faisant cela, nous pouvons minimiser la taille des données requises pour représenter le contexte. Cette question est liée aux problèmes classiques de représentation de contexte, d'interprétation et de raisonnement qui fournissent un niveau d'abstraction des données brutes capturées. À ce niveau, le raisonnement consiste à dériver le contexte de celui qui est existant.
- *Détection de changement de contexte :* comment peut-on décider que le nouveau changement des valeurs des éléments de contexte est significatif ? C.-à-d. leurs nouvelles valeurs influencent sur le comportement du service (nécessitent une adaptation de service). En faisant cela, nous minimisons la taille des informations échangées de contexte entre la partie de gestion de contexte et la partie de fournisseur de service. Cette minimisation des données sera plus importante lorsque l'environnement change légèrement, car il empêche l'explosion du volume des données envoyées. Dans le travail [Devaraju & Hoh (2008)], les auteurs dans l'évaluation de leur système de maison intelligente démontrent que la taille des informations pertinentes est 70 fois plus petite que la taille de données sans faire l'exclusion des données en double. Malheureusement, cette exclusion des données en double se fait après sa réception via le réseau.
- *Représentation de changement de contexte :* Comment représenter le changement en format commun et formel, après la détection de changements significatifs. Le format commun permet à l'indépendance de la gestion du contexte de tout autre acteur, tandis que le format formel permet d'être traitable par la machine.

1.2.2 Composition de service Web

Plusieurs raisons nécessitant l'adaptation de la composition. Parmi ces raisons sont :

- Les services Web sont, le plus souvent, des ressources hors de contrôle de leurs utilisateurs. Ces ressources peuvent parfois devenir indisponibles, pour diverses raisons comme une panne matérielle, la congestion du réseau ou encore la suppression ou la restriction d'accès de la ressource par son fournisseur ;
- L'existence d'un autre service que son intégration (substitue d'autres services) dans la composition peut améliorer la QoS du service composé ;
- Le changement de besoin de l'utilisateur que ce soit fonctionnel ou non-fonctionnel.

Pour répondre à ces besoins, nous adaptons la composition de service. Comme nous avons mentionné avant que nous adoptons l'idée de produire une nouvelle composition (recomposer) à partir des nouveaux besoins de l'utilisateur et de l'état actuel de l'environnement comme les services actuellement disponibles.

Le problème de la composition de service Web en prenant en considération la QoS reste l'un des défis à relever. Or, les techniques de planification en intelligence artificielle sont souvent considérées comme les plus efficaces et même comme les techniques prédominantes pour résoudre automatiquement des problèmes de compositions [Yan *et al.* (2012)]. Cependant, les travaux utilisant les techniques de planification subissent encore deux problèmes importants :

- Temps de l'exécution de la composition : Les auteurs de ces algorithmes utilisent la méthode habituelle de graphe de planification, c'est-à-dire qu'ils utilisent une recherche en arrière pour extraire la solution après la construction du graphe de planification. Cependant, cette façon d'extraction de solutions prend une grande proportion du temps de l'exécution des algorithmes de la composition de SWs. L'algorithme de l'extraction de la solution peut avoir une complexité NP-complet [Yan *et al.* (2012)]. Pour cette raison, des techniques d'optimisation sont souvent utilisées.
- L'utilisation des techniques d'optimisation cause généralement le problème de la redondance des services Web dans la solution obtenue (le service composé). Sachant que, un service redondant est un service qui fait partie de la composition, mais sa suppression n'influence ni les exigences fonctionnelles ni la qualité de service [Chen & Yan (2012)]. Cela veut dire qu'après

la suppression d'un service redondant, les exigences fonctionnelles sont toujours atteintes et la qualité de service n'est pas dégradée. Les services Web redondants peuvent causer plusieurs problèmes dont : l'augmentation du coût de la solution (composition) obtenue, l'augmentation de la complexité de la solution et l'augmentation du risque de défaillance de la solution [Rodriguez-Mier *et al.* (2015)]. Le problème de la redondance est présent dans la plupart des approches permettant la composition de service Web, et il est possible d'avoir plus de 30% de services redondants dans certains cas [Chen & Yan (2012)]. Sachant que, ce problème ne touche pas uniquement les approches utilisant les techniques de planification [Chen & Yan (2012)].

1.3 Contributions

Suite aux deux problématiques majeures susmentionnées, la thèse que nous défendons s'articule autour de deux (2) contributions, à savoir :

1.3.1 Gestion sémantique du contexte

Dans ce travail, la gestion de contexte est l'ensemble des substrats (c.-à-d. les éléments élémentaires qui représentent différentes fonctionnalités/ressources [Kara *et al.* (2014)]) nécessaires pour que les applications soient sensibles au contexte, telles que la détection, la modélisation et la classification, le changement et l'échange de contexte. Ainsi, dans notre travail de recherche, pour surmonter les défis susmentionnés dans la section de la problématique qui concernent la gestion du contexte, nous proposons une approche de gestion de contexte s'appuyant sur une représentation sémantique du contexte en utilisant des standards établis tels que RDF, RDF(s) ou OWL si le contexte est compliqué.

Dans ce cadre, le contexte présente une instance du modèle de contexte présenté par l'outil d'ontologie. L'utilisation de l'ontologie permet une meilleure description sémantique formelle et un raisonnement efficace sur la connaissance modélisée (c'est-à-dire l'information contextuelle).

De plus, notre approche proposée est caractérisée par sa flexibilité, car elle permet au développeur de personnaliser le modèle du contexte de service en ne spécifiant que les éléments de contexte utiles dont le changement de leurs valeurs requiert une adaptation du service développé. Cela per-

met d'éviter l'utilisation d'un modèle de contexte général qui nécessite une quantité importante de ressources comme c'est le cas dans les approches existantes [Belhanafi *et al.* (2005)],[Tamura *et al.* (2013)][Gu *et al.* (2004)].

Afin de traiter les défis liés au changement de contexte, nous avons identifié les types de mises à jour qui peuvent être effectuées. Nous proposons de conserver ces mises à jour dans un journal de changements. L'utilité de l'utilisation et de la communication de ce journal de changements est double. Premièrement, il permet de surmonter les défis du changement de contexte et de l'échange avec les dispositifs et réseaux à ressources limitées, car il ne contient que les changements de contexte pertinents. Deuxièmement, il permet de gérer l'échange de la mise à jour du contexte entre les différentes parties principales du système sensible au contexte. Car nous proposons d'exprimer le journal de changement en utilisant un langage standard et est basé sur la sémantique.

D'un autre côté, l'architecture globale de notre intergiciel proposé est basée sur la notion d'agent et SMA (Système Multi-Agents). Cette utilisation est motivée par la nature distribuée et dynamique des systèmes sensibles au contexte.

1.3.2 Algorithme de la composition automatique de service Web basé sur un graphe de planification amélioré

Dans la deuxième problématique concernant la composition de service Web, nous avons mentionné deux défis, à savoir le temps d'exécution de la composition sensible à la QoS et les services redondants dans la solution.

Afin de minimiser le temps d'exécution de la solution d'extraction et, par conséquent, l'ensemble du processus ASC, nous proposons une heuristique qui nous permet de garder la trace durant la construction du graphe de planification. En utilisant les informations enregistrées, nous pouvons obtenir plus rapidement le résultat qui satisfait les exigences fonctionnelles, et qui a la meilleure qualité QoS. En outre, nous proposons une amélioration de la structure du graphe de planification, ce qui réduit le nombre de ses nœuds et arcs. Cette réduction de la structure du graphe de planification a pour objectif de récupérer le temps consacré à l'enregistrement des informations de la construction du graphe.

Les résultats de l'évaluation montrent la faisabilité et l'efficacité de cette amélioration par rapport à un graphe de planification classique en termes de temps de composition. Afin de réduire encore le temps de composition, nous proposons également un algorithme qui minimise l'espace de recherche présenté par le graphe de planification amélioré.

Pour l'élimination des services redondants, notre approche vise à réduire le nombre de services de la composition résultante. Cette réduction en nombre de services est obtenue grâce à la nouvelle structure proposée du graphe de planification, de l'algorithme de minimisation, et deux stratégies spécifiques de sélection de services et de solutions. Les résultats de l'évaluation montrent une amélioration significative en termes de nombre de services, par rapport aux travaux voisins.

1.4 Plan du document

Le reste de la thèse est structurée en quatre (4) chapitres principaux, plus d'une conclusion générale. Le deuxième chapitre est consacré à l'état de l'art, où nous présentons la notion de services Web et leurs définitions ; les architectures des services Web et la notion de SOA ; leurs différentes technologies comme les WSDL, SOAP, et UDDI ; leurs caractéristiques et la notion des services Web sémantiques ; la notion de la composition de service Web et ses différentes technologies ; et la qualité de service Web ; une section séparée est consacrée pour l'adaptation des services Web et ses dimensions ; une autre section est pour la présentation des techniques utilisés dans notre travail, comme les ontologies, les agents et les techniques de planification avec une description formelle dans son optique du problème de la composition de service Web. Finalement, nous présentons les différents travaux voisins de notre travail avec leurs comparaisons.

Le troisième chapitre est consacré à la présentation de notre approche de gestion sémantique du contexte. Nous présentons en premier notre architecture globale de notre intergiciel(middleware) qui se constitue essentiellement des éléments passifs et des éléments actifs. Les éléments passifs sont les documents communiqués entre les éléments actifs qui sont les agents de notre intergiciel. Notre contribution sur les éléments passifs est présentée à travers l'explication d'un exemple illustratif, la présentation formelle et commune du contexte, les opérations de changement, et le journal

de changements. Par la suite, nous expliquons les agents de notre système en présentant leurs interactions, structure et architecture interne.

Le quatrième chapitre concerne la présentation de notre approche d'adaptation de la composition de service Web. Dans cette approche, nous proposons une nouvelle structure de graphe de planification et la présentation formelle de notre approche de composition de service Web. Nous présenterons notre approche sous la forme de définitions et concepts proposés. Par la suite, une description de différents algorithmes et étude de leurs propriétés. Une sous-section a été rédigée pour la présentation de nos stratégies pour la minimisation de nombre de services dans la solution.

Le cinquième chapitre est consacré pour la validation de notre approche. Pour cela, il est divisé en deux sections globales qui valident nos deux contributions à savoir la gestion sémantique de contexte et la composition de service Web. La première section explique l'implémentation et l'évaluation de performance de notre middleware de gestion sémantique de contexte en expliquant les expérimentations de performance. La deuxième section montre les démonstrations des théorèmes de notre solution. De plus, une description de l'implémentation et l'évaluation de notre approche avec une discussion des résultats avec les travaux voisins.

La thèse s'achève avec une conclusion générale récapitulant l'objectif de recherche et nos contributions, et énonce un ensemble de perspectives de continuation.

CHAPITRE 2

ÉTAT DE L'ART

Ce chapitre est consacré à la présentation des concepts de base à savoir les services Web, nous y présentons leurs notions et définitions ; les architectures des services Web et la notion de SOA ; leurs différentes technologies comme les WSDL, SOAP, et UDDI ; leurs Caractéristiques et la notion des services Web sémantiques ; la notion de la composition de service Web et ses différentes technologies ; et la qualité de service Web ;
Une section séparée est consacrée pour l'adaptation des services Web et ses dimensions ; une autre section est pour la présentation des techniques utilisés dans notre travail, comme les ontologies, les systèmes multi-agents et les techniques de planification. Finalement, et après une description formelle du problème de la composition de service Web (WSC), nous présentons les travaux voisins de notre travail.

2.1 Services Web

Dans cette section, nous présentons un concept clé de notre travail à savoir les services Web.

2.1.1 Notions et définitions

Avant de présenter les services Web, nous allons présenter les différentes notions et technologies qui mènent à produire les services Web.

2.1.1.1 Composant

En informatique, la notion de composant logiciel est apparue pour aider à développer des systèmes informatiques à l'aide des composants réutilisables. Dans [Szyperski C. (2002)], un composant logiciel est définie comme une unité de composition qui a, par contrat, spécifié uniquement ses interfaces et ses dépendances explicites de contextes. Un composant logiciel peut être déployé indépendamment et est sujet à composition par de tierces entités.

2.1.1.2 Service

Un service représente certaines fonctionnalités (application fonctionnelle, transaction commerciale, service du système de base, etc.) exposées sous la forme d'un composant au sein d'un processus métier [Dodani (2004)].
En informatique, un service est vu alors, comme un composant. En particulier, dans le cadre du domaine de l'informatique orientée service (Service-Oriented Computing (SOC)), le service est vu comme une ressource logicielle découvrable grâce à une description fournie. Cette description de service est disponible pour la recherche, la liaison, et l'invocation par un consommateur de service.

M. P. Papazoglou et al. dans [Papazoglou (2003)] ont défini le SOC comme suit : «est un paradigme où les applications sont construites en utilisant des services comme blocs de construction. Comme composants, les services ont également mis en principes de modularité pratique et fournissent un bon niveau d'encapsulation.»

Dans la sous-section suivante, nous allons présenter les architectures orientées service (SOA) qui sont l'approche la plus connue sur la base SOC.

2.1.1.3 Les architectures orientées service (SOA)

L'architecture SOA est un moyen logique (loical way) de concevoir un système logiciel pour fournir des services soit à l'utilisateur final des applications ou à d'autres services distribués dans un réseau, via des interfaces publiées et découvrables [Papazoglou *et al.* (2008)]. L'objectif d'une architecture SOA est de décomposer une fonctionnalité en un ensemble de services et de décrire leurs interactions.

Par conséquent, l'architecture SOA permet la réutilisation des applications et des services existants, ainsi de nouveaux services peuvent être créés à partir d'une infrastructure informatique des systèmes déjà existants [Erl (2004)] [Erl (2005)]. En d'autres termes, SOA permet aux entreprises de tirer parti des investissements existants en leur permettant de réutiliser des applications existantes, en leur offrant une interopérabilité entre applications et technologies hétérogènes.

L'architecture SOA s'articule autour de trois concepts fondamentaux à savoir le fournisseur de services, l'annuaire de publication et le consommateur de services.

a. Le fournisseur de service : désigne une personne ou une organisation qui offre le service sur un serveur et publie sa description contenant les informations nécessaires afin de l'utiliser, ainsi que les opérations disponibles et leur mode d'invocation.

b. L'annuaire de publication : est un répertoire de description de services. Il représente aussi un médiateur entre les consommateurs et les fournisseurs de services. En fait, l'annuaire automatise les communications entre ces deux derniers en fournissant aux consommateurs les informations techniques et sémantiques sur le fonctionnement du service.

c. Le consommateur de service : ils invoquent les services offerts par les fournisseurs. Il peut s'agir soit d'applications clientes (par exemple un tableau de bord sur un PC de bureau, indiquant l'évolution des indices boursiers provenant en temps réel d'un serveur Web) ou bien soit de services qui s'appuient sur la fonctionnalité d'un autre fournisseur de service (ex. un service Web d'agrégation des indices boursiers, contactant les services de plusieurs bourses afin de calculer des indicateurs agrégés pour les publier auprès des clients).

Nous allons décrire davantage cette architecture en décrivant les interactions entre ces différents principaux acteurs, quand nous expliquerons le fonctionnement d'un service Web qui est basé sur l'architecture SOA, dans la sous-section qui suit.

2.1.1.4 Service Web

Le développement du Web a favorisé le dialogue entre les systèmes informatiques ainsi que le déploiement d'applications distribuées. Il a également révélé des manques évidents en matière d'intégration et d'interopérabilité entre services [Soukkarieh (2010)]. Dans cette optique, les services Web ont été proposés comme une clé pour l'interopérabilité entre systèmes indépendamment des langages de programmation.

Plusieurs définitions des services Web ont été mises en œuvre par différents auteurs. Ci-après, nous citons une définition généralement acceptée et fournie par le W3C (World Wide Web Consortium). "A Web service is a software system designed to support interoperable machine-to-machine inter-

action over a network. It has an interface described in a machine process- able format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards" [Hugo Haas (2017)].

Dans cette définition, un service Web est défini comme un système logiciel accessible sur le réseau. Il permet l'interopérabilité entre plusieurs systèmes en utilisant les standards du Web qui sont indépendants des langages de programmation.

En d'autres termes, les services Web dans la définition de leurs interfaces s'appuyant sur XML, et aussi quand ils interagissent dynamiquement entre eux et avec d'autres applications en utilisant les protocoles de transport Internet disponibles.

Dans la section suivante, nous allons présenter le principe du fonctionnement des services Web.

2.1.2 Architectures des services Web

On distingue deux types d'architecture des services Web : la première dite de référence, elle contient trois couches principales. La seconde architecture est plus complète, elle utilise les couches standards de la première architecture en ajoutant au-dessus d'autres couches plus spécifiques([David *et al.* (2004), Bhiri *et al.* (2005)]). Elle est appelée architecture étendue ou en pile [Kreger *et al.* (2001)].

2.1.2.1 Architecture de référence

Basée sur le modèle de référence du consortium OASIS pour les architectures orientées service (SOA). Ce modèle a pour objectif de définir un cadre conceptuel commun qui pourra être utilisé de façon consistante à travers les différentes implémentations [MacKenzie *et al.* (2006)].
L'architecture de référence des services Web vise trois objectifs importants : l'identification des composants fonctionnels, la définition des relations entre ces composants et l'établissement d'un ensemble de contraintes sur chaque composant de manière à garantir les propriétés globales de l'architecture.

La Figure 2.1 illustre les éléments fondamentaux de cette architecture, à savoir [Boudali (2007)] :

a. Le fournisseur de service : c'est le propriétaire du service. D'un point de vue technique, il est constitué par la plateforme d'hébergement du service.

b. Le client : c'est le demandeur de service. Techniquement, il est constitué par l'application qui va rechercher et invoquer un service. Une application cliente peut être elle-même un service Web.

c. L'annuaire des services : c'est un registre de descriptions de services offrant des facilités de publication de services pour les fournisseurs de services ainsi que des facilités de recherche de services pour les clients.

Ces trois éléments de l'architecture interagissent entre eux selon trois types d'opérations : les opérations de publication, de recherche et de liens.

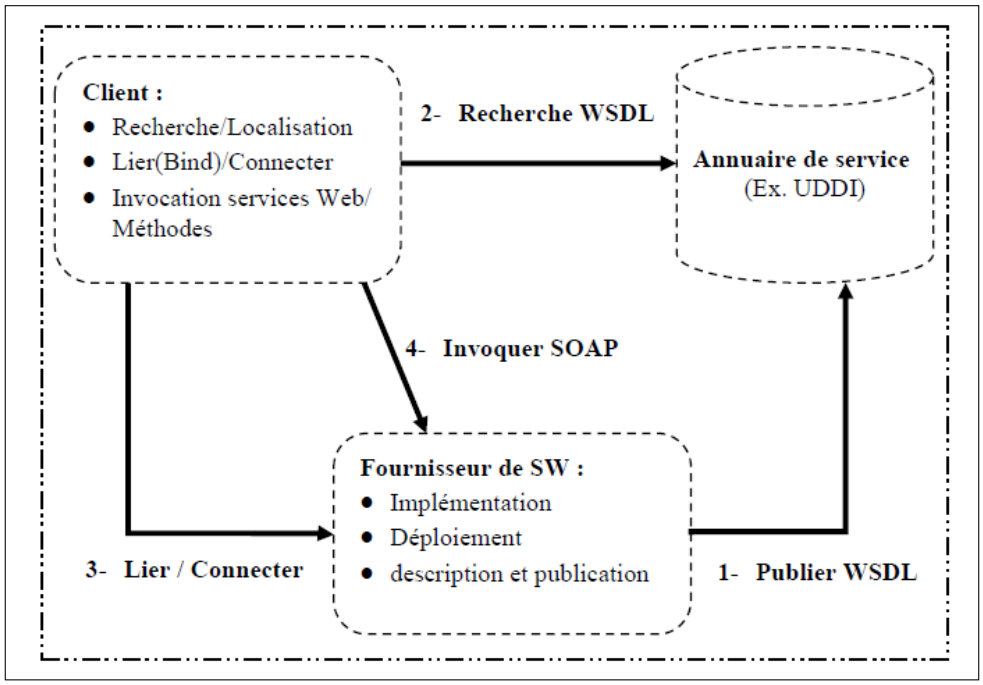


Figure 2.1 Architectural de référence des services Web [Boudali (2007)]

2.1.2.2 Architecture étendue

Une architecture étendue est constituée de plusieurs couches se superposant les unes sur les autres, d'où le nom de pile des services Web. La pile est constituée de plusieurs couches, chaque couche s'appuyant sur des standards particuliers. Les trois couches formant infrastructure de base décrite précédemment se retrouvent au-dessus de la couche de transport. Nous apportons une explication de la mise en relief des trois types de couches (la figure 2.2) :

a. L'infrastructure de base (Discovery, Discription, Exchange) : Ce sont les fondements techniques établis par l'architecture de référence. Nous distinguons les échanges des messages établis par SOAP, la description de service par WSDL et la recherche de services Web que les organisations souhaitent utiliser via le registre UDDI ;

b. Couches transversales (Security, Transactions, Administration, QoS) : Ce sont des couches qui rendent viable l'utilisation effective des services Web dans le monde industriel ;

c. La couche du processus métier (BusinessProcess) : Cette couche supérieure permet l'intégration de services Web, elle établit la représentation d'un « BusinessProcess » comme un ensemble de services Web. De plus, la description de l'utilisation de différents services composant ce service est disponible par l'intermédiaire de cette couche.

2.2 Technologies des services Web

L'architecture standard d'un service Web est implémentée à l'aide des diverses technologies et standards (tels que WSDL, SOAP et UDDI), organisés en quatre couches à savoir, Publication, Description, Message et Transport.

Alors, les différentes couches de l'architecture d'un service Web s'interfacent avec des standards, comme suit [Mahfoud (2011)] :

- *La couche de publication* : repose sur le protocole UDDI qui assure le regroupement, le stockage et la diffusion des descriptions des services Web.
- *La couche description* : est prise en charge par le langage WSDL qui décrit les fonctionnalités fournies par le service Web, les messages reçus et envoyés pour chaque fonctionnalité, ainsi que le protocole utilisé pour la communication.

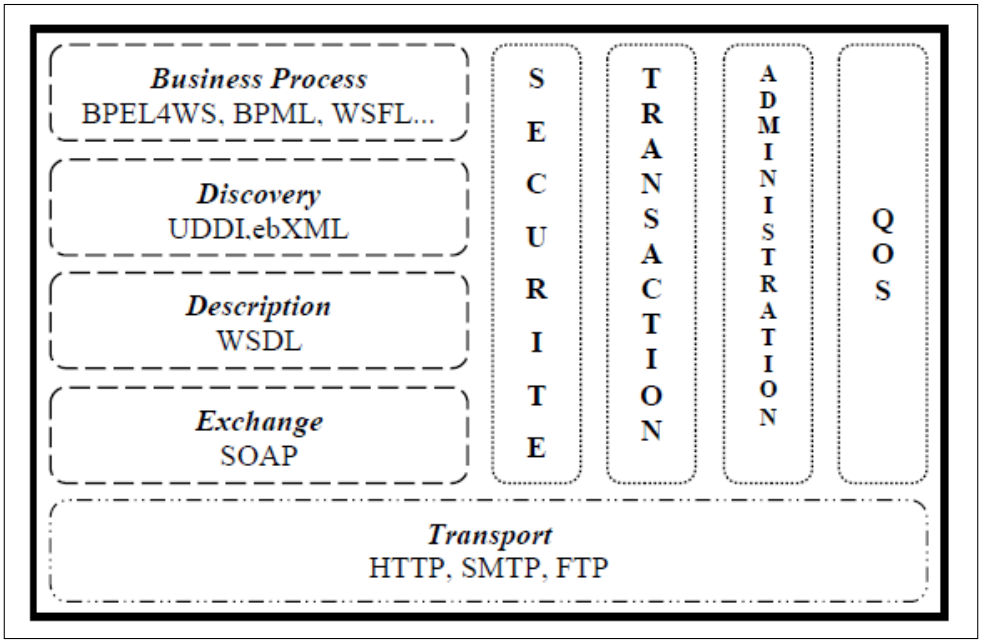


Figure 2.2 Architectural en pile (étendue) des services Web
[Boukhadra (2011)]

- *La couche message* : cette couche utilise des protocoles reposants sur le langage XML, car sa syntaxe unique résout les conflits syntaxiques lors de l'encodage des données. Actuellement, le protocole le plus utilisé est le SOAP.
- *La couche transport* : le protocole le plus utilisé dans cette couche est le HTTP. Cependant, d'autres protocoles peuvent être utilisés, tels que le SMTP ou le FTP, permettant ainsi aux services Web de rester indépendants du mode de transport utilisé.

Dans ce que suit, nous allons présenter ces différentes technologies.

2.2.1 XML

XML signifie langage à balises étendu, ou langage à balises extensible. Contrairement à HTML, le XML est un langage extensible qui a la possibilité de créer de nouvelles balises. Il s'agit effectivement d'un langage permettant de mettre en forme des documents grâce à des balises[PILLOU (2017)].

Son objectif initial est de faciliter l'échange automatisé de contenus entre systèmes d'informations hétérogènes. XML contient aussi un aspect supplémentaire appelé "schéma", qui précise la structure des tags pour qu'un document soit valide.

Il est ainsi possible de restreindre les données acceptées dans un document. Un document XML peut contenir d'autres documents XML. Pour éviter toute confusion entre les tags utilisés, chaque document contient un "espace de noms", ce qui rend ainsi les tags uniques à l'intérieur d'un document.

2.2.2 WSDL (Web Services Description Language)

WSDL est un langage, qui permet de fournir la description, en XML, des services indépendamment de la plateforme et du langage utilisés et sous une forme que des personnes ou des programmes peuvent interpréter. Il décrit les types de données supportés et les fonctions offertes par un service Web.

Un document WSDL est constitué d'une suite d'éléments décrivant un service sous forme d'un ensemble d'opérations exploitables depuis l'extérieur [W3C (2007b)]. Pour cela, il contient toutes les informations nécessaires pour contacter et utiliser un service, indépendamment de toute plateforme ou tout langage de programmation (figure 2.3) [Lopez-Velasco (2008)].

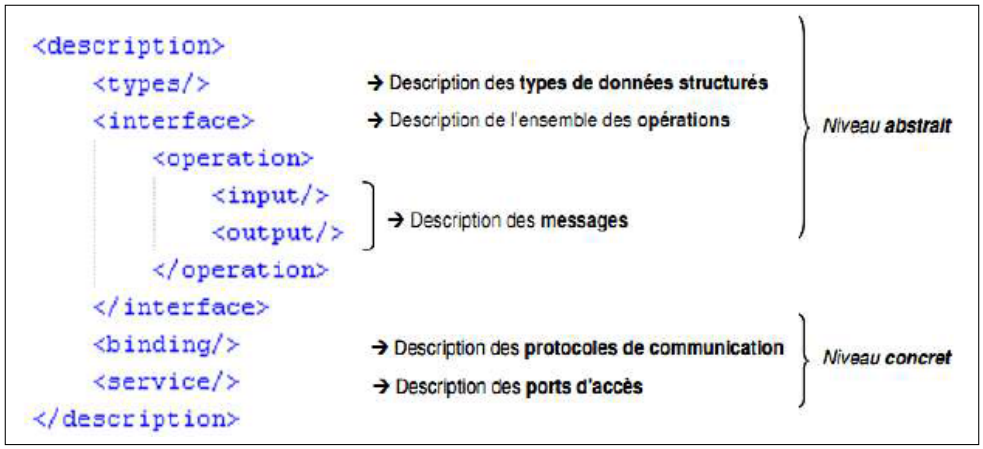


Figure 2.3 Structure générale d'un document WSDL.

Un document WSDL est divisé en deux parties, la partie abstraite et la partie concrète.

- *La partie abstraite :* elle décrit les informations propres aux méthodes proposées par le service, ainsi que les informations traitant des messages et des données échangées lors de l'invocation du service [Lopez-Velasco (2008)]. Ce niveau est composé des informations suivantes [Soukkarieh (2010)] :
  - *Types :* un conteneur de la définition de données utilisées.
  - *Messages :* description des types de données utilisées lors de l'invocation (et de la réponse) d'une opération.
  - *Opération :* description d'une des actions supportées par le service Web.
  - *Port Type :* description de l'ensemble des actions supportées.
  - *Port :* une combinaison d'une adresse réseau et d'un binding qui spécifie une liaison d'un « Port Type » à un protocole SOAP.
- *La partie concrète :* les informations décrites dans le niveau concret sont les suivantes [Soukkarieh (2010)] :
  - *Binding :* protocole et format de données spécifiques pour un type de port précis.
  - *Service :* une collection des ports d'accès du service.

L'intérêt d'avoir ces deux parties est que la partie concrète propose une ou plusieurs réalisations de la partie abstraite.

2.2.3 SOAP

C'est un protocole qui permet de faire communiquer des entités distantes en envoyant des messages via le réseau. Il est basé sur le langage XML et le protocole HTTP comme un mécanisme de transport des messages SOAP. Il est possible d'utiliser des protocoles autres que le HTTP comme FTP et SMTP [KARA (2014)].

Il permet ainsi l'accès aux services Web et l'interopérabilité des applications à travers le Web [la Vallée (2017)].

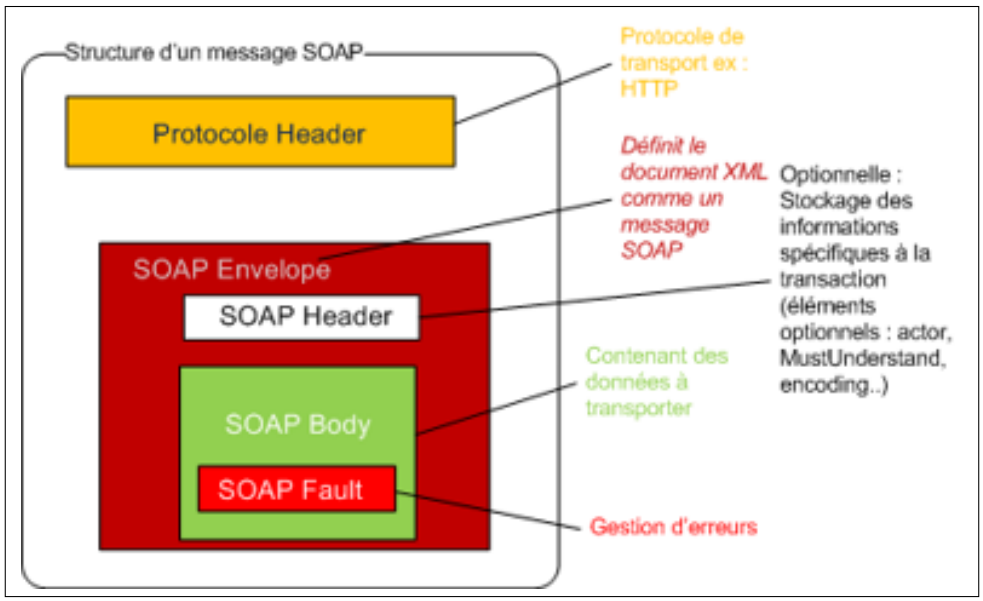


Figure 2.4 Structure d'un message SOAP [openclassrooms (2017)]

La figure 2.4 permet de montrer le format d'un message SOAP. Il comporte deux éléments obligatoires et un élément facultatif.

- *Éléments obligatoires :*
  - *SOAP Enveloppe :* l'enveloppe SOAP représente la racine d'un message SOAP. Il comporte des informations sur le format de données utilisées ainsi que les noms d'espaces [openclassrooms (2017)]. Il est composé d'un corps (Body) et d'un en-tête (Header).
  - *SOAP Body :* il contient l'appel de la procédure distante dans un sens et la réponse du serveur dans l'autre [Mahfoud (2011)].
- *Élément facultatif :*
  - *SOAP Header : il* permet de spécifier des données particulières concernant le message à envoyer (par exemple : des données d'authentifications) [openclassrooms (2017)].

2.2.4 UDDI (Universal Description, Discovery and Integration) :

UDDI [OASIS (2002)] est plus qu'un simple protocole : il fournit un protocole, une API et une plateforme permettant aux utilisateurs de services Web, depuis n'importe quel système, de localiser dynamiquement à travers Internet les services Web qu'ils désirent utiliser. UDDI est un modèle

d'information composé de structures de données persistantes appelées entités. Ces entités doivent être décrites en XML et sont stockées dans les différents nœuds UDDI [Rampacek (2006)]. Les différentes informations sont divisées en trois catégories (figure 2.5) :

a. *Les pages blanches :* Ces pages comprennent la liste des organisations, leurs informations de contacts et les services qu'elles fournissent.

b. *Les pages jaunes :* Ces pages classifient les compagnies et les services Web selon des taxonomies qui peuvent être standardisées ou définies par l'utilisateur.

c. *Les pages vertes :* Ces pages décrivent comment un service Web peut être invoqué.

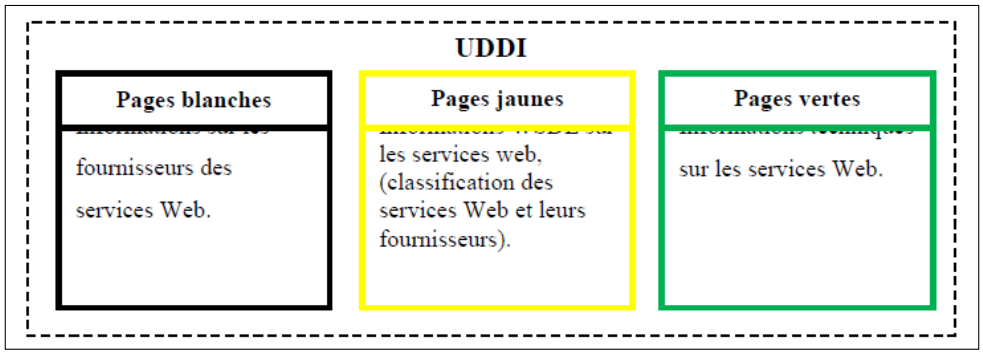


Figure 2.5 Les trois facettes de l'annuaire UDDI [Mahfoud (2011)].

À partir les architectures et les technologies utilisées des services Web, on peut révéler les caractéristiques des services Web dans la sous-section suivante.

2.3 Caractéristiques des services Web

Les services Web possèdent les caractéristiques suivantes [Cerami (2002)] [KARA (2014)] :

- *Couplage faible et interaction dynamique :* signifie que les services Web ont un degré de dépendance très faible. Le service Web et son consommateur sont indépendants l'un de l'autre. Si une modification est à faire sur le consommateur, on n'a pas besoin de connaitre la machine, le langage de programmation, le système d'exploitation ou autre paramètre, afin d'établir à nouveau une communication entre le service Web et son consommateur. Le consommateur possède une fonctionnalité qui correspond à faire une localisation et une invocation sur le service Web, au moment de l'exécution du programme de service Web de manière automatique. Cette interopé-

rabilité est basée sur l'utilisation du XML et les protocoles Internet standards tels que HTTP, SMTP et FTP.

- *Contrat de service :* indique que les échanges entre les services Web se font en respectant les spécifications définies dans les documents de descriptions.
- *Abstraction de service :* indique que la conception et l'implémentation des services Web ne sont pas visibles de l'extérieur.
- *Composition de service :* indique que les services Web peuvent être combinés afin d'en avoir qu'un seul.
- *Découverte de service :* montre que les services Web peuvent être découverts au moyen de leurs descriptions. En notant que chaque service Web est reconnu par un URI. URI est la façon d'identifier un point de contenu sur le Web comme un document tel qu'un texte, audio ou vidéo. Le service Web est donc accessible en spécifiant son URI.

2.4 Service Web sémantique

2.4.1 Définition

Le Web sémantique permet aux machines de comprendre la sémantique, la signification de l'information sur le Web. Il étend le réseau des hyperliens entre des pages Web classiques par un réseau de liens entre données structurées permettant ainsi aux agents automatisés d'accéder plus intelligemment aux différentes sources de données contenues sur le Web et, de cette manière, d'effectuer des tâches (recherche, apprentissage, etc.) plus précises pour les utilisateurs.

Un service Web sémantique est la combinaison des services Web avec des techniques du Web sémantique. Cette combinaison permet de prolonger les capacités d'un service Web en associant des concepts sémantiques au service Web afin de permettre l'automatisation des activités reliées aux services Web comme la découverte des services Web et la composition de service Web.

2.4.2 Description sémantique des services Web

Il existe deux méthodes permettant la description sémantique d'un service Web :

- La première méthode est basée sur l'annotation qui permet d'étendre les technologies des services Web en ajoutant de la sémantique. Cette méthode permet d'avoir un gain de temps puisqu'elle permet de rajouter des informations supplémentaires à ce qu'il existe déjà, par exemple, l'ajout des informations sémantiques à l'intérieur d'un fichier WSDL comme dans le travail [Lopez-Velasco (2008)] [Blake *et al.* (2010)].
- La deuxième méthode consiste à décrire les services Web à l'aide des technologies du Web sémantique (la description du service Web se fait de nouveau sans être passée par les descriptions existantes comme c'est le cas de la méthode précédente) [Chouchani (2010)]. Dans cette optique, le W3C a défini le OWL-S [David Martin (2004)] qui est une ontologie pour décrire les concepts de domaine des services Web. Il fournit des constructeurs afin d'indiquer des informations sur la façon de composer et d'accéder à des services. Plus de sa capacité à décrire sémantiquement les entrées-sorties (input/output), le OWL-S décrit aussi les conditions externes préalables requises par le service et les effets escomptés résultant de l'exécution du service. Par exemple, un service de vente peut exiger comme condition préalable une carte de crédit valide et comme entrée le numéro de carte de crédit et la date d'expiration. En tant que sortie, il génère un reçu, et comme effet, la carte est débitée [David Martin (2004)].

2.5 Composition de service Web

2.5.1 Définition

La composition de service Web est un processus d'affaire qui spécifie la façon dont plusieurs services Web collaborent, afin de réaliser une tâche qu'un seul service Web ne peut pas réaliser [Chouchani (2010)].

Exemple : supposons qu'on a un service Web d'échange de devise « CurrencyConverter», qui contient l'opération «converter» ayant comme entrées, le montant à convertir (Amount), devise de départ(FromCurrency), devise de destination (ToCurrency), et le taux de change (Rate) et comme sortie le montant converti (ConvertedAmount). Cependant, souvent l'utilisateur de ce service ne connait pas le taux de change du jour. Pour cela, on a besoin d'un autre service Web qui nous donne la valeur du taux d'échange. Pour la date du jour, on peut l'obtenir implicitement à partir de

la requête. Supposant qu'on a ce service dans le registre des services, le résultat de la composition des deux services est présenté dans la figure 2.6.

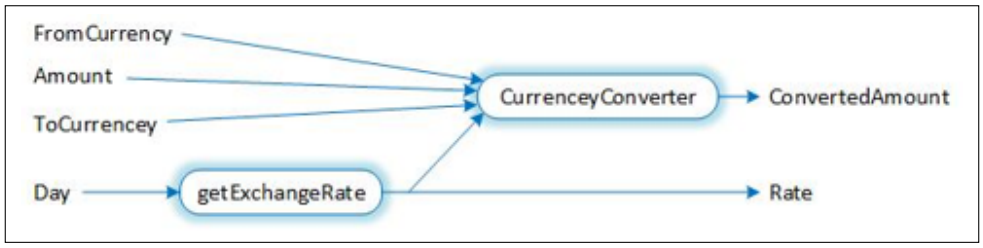


Figure 2.6 Example de composition de service Web

2.5.2 BPEL

BPEL est un langage basé sur XML qui permet de décrire des processus métiers en décrivant l'organisation (composition) des tâches (services Web). Un processus BPEL est composé de trois éléments principaux [TREMBLAY (2007a)] :

- *Les opérations :* elles décrivent les opérations offertes par le processus d'affaire aux partenaires (services Web avec lesquels interagit le processus), ou les opérations demandées par le processus [TREMBLAY (2007a)] [TREMBLAY (2007b)].
- *État interne :* il définit l'état interne des processus en utilisant des variables, afin de sauvegarder les échanges entre les différents partenaires (services Web et le processus d'affaires) [TREMBLAY (2007a)] [TREMBLAY (2007b)].
- *Comportement :* il définit les différentes activités au sein d'un processus d'affaires. Les deux activités principales qui ont été utilisées dans le cadre de notre travail sont : l'activité <sequence> et l'activité <flow>. La première activité (sequence) permet de définir un ordre séquentiel dans lequel des activités vont se dérouler, alors que la deuxième activité (flow) signifie qu'un ensemble d'activités se déroulent en parallèle [TREMBLAY (2007b)] [Baron (2011)].

2.6 Qualité de service (QoS)

2.6.1 Définition

Il existe trois types d'exigences logicielles qui sont énumérées ci-dessous [CHAMPAGNE (2014)].

- *Les exigences fonctionnelles :* elles représentent les fonctionnalités offertes par le logiciel.
- *Les exigences non fonctionnelles :* ce sont les attributs de qualités, des propriétés qui peuvent être mesurées ou testées. Elles reflètent la correspondance du logiciel aux besoins.
- *Les contraintes :* elles représentent des exigences qui ne peuvent pas être modifiées (par exemple, un client exige que son logiciel soit implémenté en utilisant le langage de programmation Java).

La qualité de service d'un service Web est une propriété qui fait partie des exigences non fonctionnelles.

Plusieurs critères de qualités de services peuvent être pris en considération comme le temps de réponse (response time), le débit (throughput), le coût (cost), la réputation (reputation), la fiabilité (reliability), taux d'éxecution réussie (successful execution rate) et la disponibilité (availability).

Ces différents critères de qualité de service sont divisés en deux catégories, à savoir les critères positifs et négatifs. Dans le cas d'un critère négatif, le service ayant la valeur la plus haute est le service ayant la qualité la plus basse. Par contre, dans le cas d'un critère positif, le service ayant la valeur la plus haute est le service ayant la qualité la plus haute.

Dans notre travail, nous avons choisi de chaque catégorie une qualité de services, à savoir le temps de réponse (Response time) comme un critère négatif et le débit (Throughput) comme un critère positif.

- *Le temps de réponse :* il correspond à la durée écoulée entre la réception de la fin de la transmission d'une requête et le début de la transmission de la réponse de cette requête [Zeng *et al.* (2003)].
- *Le débit :* il représente "le taux de transmission du message de succès sur un canal de communication" [wikipedia (2016b)]. Il informe sur le nombre de fois où un service Web est invoqué avec succès par unité de temps [Kona *et al.* (2009)].

2.6.2 WSLA

WSLA est un langage qui se base sur XML. Il permet de spécifier entre autres les valeurs de qualités de service d'un service Web (par exemple, le temps de réponse d'un service Web est spécifié à l'intérieur d'un fichier WSLA) [Kona *et al.* (2009)] [Ludwig *et al.* (2003)].

Le tableau 2.1 présente un extrait d'un fichier WSLA. Cet extrait détermine la valeur du temps de réponse du service Web nommé 'A' qui est inférieure à 200 millisecondes (l'unité de mesure a été spécifiée dans une autre section du fichier WSLA).

La ligne 8 définit le type du prédicat de type 'Less'. Il existe d'autres types comme le type 'greater' qui peut être utilisé dans le cas du débit.

Exemple : le débit d'un service Web 'A' est plus grand que 200 invocations par minute.

Tableau 2.1 Extrait d'un fichier WSLA [Kona *et al.* (2009)]

```
1  <ServiceLevelObjective name="ObjectiveAverageResponseTimeServiceA">
2     <Obliged>Vienna XPress Services </Obliged>
3     <Validity>
4        <Start>2008-01 -30 T14 :00 :00 </Start>
5        <End>2008 -05 -29T17 :29 :00 </End>
6     </Validity>
7     <Expression>
8        <Predicate xsi :type="Less">
9        <SLAParameter >AverageResponseTimeServiceA </SLAParameter>
10          <Value>200 </Value>
11       </Predicate>
12    </Expression>
13    <EvaluationEvent >NewValue <EvaluationEvent>
14 </ServiceLevelObjective>
```

2.7 Adaptation des applications et services Web

2.7.1 Contexte

Le mot contexte tel que trouvé dans le dictionnaire Larousse est défini comme suit : « un ensemble de circonstances dans lesquelles s'insère un fait, une situation globale où se situe un évènement». Les chercheurs dans le domaine de l'adaptation n'ont pas encore abouti à une définition générique de la notion de contexte. Mais parmi les différentes définitions qui existent, la définition de Dey [Dey (2000)] qui fait référence : « le contexte est construit à partir de tous les éléments d'information qui peuvent être utilisés pour caractériser la situation d'une entité. Une entité correspond ici à toute personne, tout endroit, ou tout objet (en incluant les utilisateurs et les applications) considéré(e) comme pertinent(e) pour l'interaction entre l'utilisateur et l'application ».

Selon Ferry et al. [Ferry *et al.* (2008)] il y a quatre catégories de contexte :

- *Contexte utilisateur :* qui concerne toutes les informations qui peuvent être relatives à un utilisateur comme sa localisation, son humeur, la tâche qu'il est en train de réaliser.
- *Contexte temporel :* qui traite de tout ce qui est relatif aux instants tels que la date et l'historique.
- *Contexte machine :* qui concerne l'environnement d'exécution comme les ressources disponibles.
- *Contexte environnemental :* qui est relatif à tout ce qui entoure le système, par exemple : luminosité, bruit, et température.

2.7.2 L'informatique sensible au contexte (Context-aware computing)

2.7.2.1 Notion

Caractérise un paradigme de l'informatique dans lequel les applications perçoivent la situation de l'utilisateur dans son environnement et adaptent ainsi leurs comportements [Abowd *et al.* (1999)]. L'objectif de la sensibilité au contexte est de fournir aux utilisateurs des applications adaptées qui prennent en compte le changement de leur contexte d'utilisation.

Avec le passage de l'informatique traditionnelle à l'informatique mobile et ubiquitaire, il y a un besoin accru de sensibilisation au contexte. L'objectif de la sensibilité au contexte est de fournir aux utilisateurs des applications adaptées qui prennent en compte le changement de leur contexte d'utilisation. La prise en conscience de contexte a gagné de l'attention grâce, entre autres, aux progrès techniques permettant une détection du contexte à faible coût.

2.7.2.2 Intergiciels pour la sensibilité au contexte (Context-awareness middleware)

Il n'est pas pratique de construire des applications sensibles au contexte à partir de zéro, car des éléments constitutifs de ces applications tels que la spécification de contexte, son acquisition et son traitement doivent être développés à chaque fois. Celles-ci doivent être séparées de l'application spécifique afin de réduire les coûts de développement. Ces éléments de construction constituent des infrastructures d'application qui sont communément appelées middleware (intergiciel) de contexte.

Dans le cadre de la sensibilité au contexte et prenant en compte le nouveau paradigme SOC (Service- Oriented Computing) et l'architecture SOA (l'architecture orientée service) que les services Web présente leur fameuse implémentation, nous devons nous intéresser à l'adaptation des services Web. Dans la sous-section suivante, nous allons présenter comment les travaux sur l'adaptation des services Web sont organisés.

2.7.3 Les différentes dimensions de l'adaptation des services Web

Dans les travaux relatifs à l'adaptation des services Web, il a y quatre dimensions d'adaptation.

2.7.3.1 L'adaptation de contenu de service Web :

L'adaptation du contenu est définie généralement comme le processus qui transforme un contenu de son état initial vers un état final afin de satisfaire un ensemble de contraintes (des caractéristiques, des buts de l'utilisateur, etc.). Le contenu final peut provenir du même contenu source ou d'autres ressources de contenu [Lemlouma (2004)]. D'après [Brusilovsky (1998)], il existe trois manières courantes d'adapter le contenu :

- Fournir un supplément d'information.

- Cacher des données jugées non pertinentes ou ne devant pas être délivrées.
- Choisir le contenu adéquat à l'utilisateur, si le système possède plusieurs alternatives de contenu.

2.7.3.2 L'adaptation de la présentation de service Web

L'adaptation de la présentation est l'un des objectifs de la communauté des hypermédias adaptatifs. Cette adaptation est appliquée sur les caractéristiques visuelles du document présentées à l'utilisateur. Elle vise à effectuer des transformations au niveau de l'affichage. Ces transformations suivent les préférences de l'utilisateur et son contexte. Nous décrivons les méthodes les plus courantes pour effectuer cette adaptation :

- L'adaptation des caractéristiques graphiques.
- L'adaptation de la langue de présentation.
- L'organisation du document.
- L'altération de médias et la substitution de médias.

2.7.3.3 L'adaptation de comportement

Le nombre d'utilisateurs du réseau Internet ne cesse de croitre depuis les années 90. Les utilisateurs du Web deviennent exigeants en termes d'attente et de qualité. C'est pourquoi des travaux essaient d'adapter le comportement des services afin qu'ils préviennent les changements de l'environnement d'exécution. L'adaptation du comportement du service consiste à changer la logique d'un service. Elle concerne l'occurrence et l'ordre des étapes d'exécution d'un service, ainsi que les relations logiques entre ces différentes étapes [Fouial (2004)].

2.7.3.4 L'adaptation de la composition de service

La composition a pour rôle l'assemblage de services Web basés sur leurs caractéristiques fonctionnelles afin de réaliser une tâche donnée et fournir un ordre d'interaction sur les services [Lécué (2008)]. La composition peut être statique ; cela veut dire que les services interagissent entre eux d'une manière prédéfinie. Elle peut être aussi dynamique ; c'est à dire que les services sont découverts les uns des autres. La composition de service peut s'adapter en se basant sur deux aspects : le

modèle de l'utilisateur et la tâche à effectuer. Le premier type d'adaptation choisit les services en adéquation avec un type d'utilisateur donné [Conlan *et al.* (2003)], tandis que le second choisit les services dont le comportement permet au mieux de réaliser la tâche principale [Amor *et al.* (2003)].

Dû de l'importance de la composition de service Web surtout dans la vision du paradigme SOC où une application est composée d'un ensemble de services, nous allons nous concentrer nos efforts sur l'adaptation de la composition de service Web.

Deux catégories d'adaptation de la composition de service Web :

- La première vise à changer la composition (plus précisément son plan) en répondant à un manque des services dans la composition actuelle, ou même à un changement au niveau de besoin de l'utilisateur. Le travail [Yan *et al.* (2010)] rentre dans cette catégorie. Il se base sur les graphes de planification où la composition se représente sous la forme d'un plan. Le travail essaie de réparer (modifier) le plan en réponse aux changements, en réutilisant autant que possible le plan original. Cette manière d'adaptation de la composition est motivée surtout si l'utilisateur final préfère utiliser un plan ressemblant du plan actuel et que le résultat de la re-planification est très différent de celui de la réparation du plan.
- La deuxième manière est d'effectuer une nouvelle planification (re-planification complète) à partir de l'état actuel. Cette catégorie qui est la plus adoptée dans la littérature (par exemple [Pistore *et al.* (2005)], [Zheng & Yan (2008)]) est basée sur le principe théorique qui affirme que modifier un plan existant n'est pas plus efficace qu'une re-planification complète dans le pire des cas [Yan *et al.* (2010)].

Dans notre travail, nous adoptons aussi la deuxième catégorie, car la génération d'un nouveau plan à partir de l'état actuel de l'utilisateur et des services permet d'obtenir une composition (plan) plus adéquate que celle obtenue à partir de la modification d'une ancienne composition qui était générée pour un autre état.

2.8 Techniques utilisées

2.8.1 Ontologie

Parallèlement à l'explosion de l'information dans plusieurs domaines qui a marqué ces dernières années, des progrès considérables ont été observés dans le développement de méthodes de modélisation génériques des données et dans les technologies de recherche de l'information afin de manipuler la diversité des données et leur ampleur [Jeannette (2006)].

Les ontologies apparaissent comme une clé pour la manipulation automatique de l'information au niveau sémantique.

Nous abordons le thème d'ontologie en deux sous-sections, l'une décrit la notion d'ontologie et l'autre concentre à sa représentation.

2.8.1.1 Description de l'ontologie

Notion et définition La notion d'ontologie est apparue en informatique (plus précisément en Intelligence artificielle) dans les années 90, le terme d'ontologie est cependant usité en philosophie depuis le XIXe siècle. Dans ce domaine, l'Ontologie désigne l'étude de ce qui existe, c'est-à-dire l'ensemble des connaissances existantes sur le monde [Welty & Guarino (2001)]. En IA, de façon moins ambitieuse, ne sont considérées que des ontologies, relatives aux différents domaines de connaissances. C'est à l'occasion de l'émergence de l'ingénierie des connaissances que les ontologies sont apparues en IA, comme réponses aux problématiques de représentation et de manipulation des connaissances au sein des systèmes informatiques [FÜRST (2004)].

Plusieurs définitions de l'ontologie ont été proposées ; la plus célèbre est celle de Gruber [Gruber (1993)] qui définit une ontologie comme « Une spécification explicite et formelle d'une conceptualisation partagée ».

- Le terme *Conceptualisation* réfère à un modèle (au sens d'ensemble structuré) de concept [Jeannette (2006)].

- Le terme *Explicite* signifie que les éléments de la conceptualisation (concepts, relations et contraintes de leurs utilisations) doivent être explicitement définis.
- Le terme *Formelle* réfère au fait qu'une ontologie doit être compréhensible et traitable par la machine, c'est-à-dire cette dernière soit capable d'interpréter la sémantique de l'information fournie. De l'importance de cette caractéristique, le degré de formalisation présente un critère de typage d'ontologie.
- Le terme *Partagée* indique que l'ontologie supporte la connaissance consensuelle, et elle n'est pas restreinte à certains individus, mais acceptée par un groupe [Broekstra *et al.* (2002)].

Constituants d'une ontologie Une ontologie définit les termes et concepts (signification) utilisés pour décrire et représenter un domaine de connaissance, ainsi que les relations entre eux.

Par conséquent, une ontologie contient :

- Les concepts du domaine qui sont appelés aussi termes ou classes d'ontologie, correspondent aux abstractions pertinentes d'un segment de la réalité, selon [Gaëlle (2002)] représente un type d'objet dans l'univers, un concept peut être tout ce qui peut être évoqué et, partant de là, peut consister en a description d'une tâche, d'une fonction, d'une action, d'une stratégie ou d'un processus de raisonnement, etc.
- Les propriétés des concepts [Stojanovic (2004)] distinguent deux genres de propriétés, qui sont les attributs (attribute properties) assignés aux concepts et les relations (relational properties) entre les concepts.
- Les valeurs des attributs caractérisant un concept sont de différents types : nombres, booléens, intervalles, mots, etc.
- Les relations représentent un type d'interaction entre les notions d'un domaine [Gaëlle (2002)]. Nous distinguons deux types de relations, relation inter-concepts et relation inter-relations. Les relations inter-concepts que nous pouvons les trouver sont l'abstraction, la subsomption, l'équivalence, la disjonction et bien d'autres relations définies par le concepteur de l'ontologie. Et pour les relations inter-relations, on a la subsomption, l'incompatibilité, l'inverse, l'exclusivité, et l'incompatibilité.

  FÜRST dans [Fürst (2004)] définit les propriétés intrinsèques d'une relation, qui sont :

- Les propriétés algébriques : symétrie, réflexivité, transitivité, antisymétrie, antiréflexivité.
- Et les cardinalités : qui précisent le nombre d'instances que la relation lie. Les instances (Objets) des concepts et ces valeurs, correspondent aux individus concrets dans le domaine [Stojanovic (2004)]. FÜRST dans [Fürst (2004)] considère l'ensemble d'objets (instances) comme l'extension du concept, qui regroupe les objets manipulés à travers le concept.

- Axiomes, constituent des assertions, acceptées comme vraies, à propos des abstractions du domaine traduites par l'ontologie. Ils spécifient la façon dont les primitives terminologiques du domaine (c.-à-d. les concepts et relations) peuvent être utilisées. Dans [Razika (2007)], l'utilisation des axiomes sert à définir le sens des entités, mettre des restrictions sur la valeur des attributs, examiner la conformité des informations spécifiées ou en déduire de nouvelles.

2.8.1.2 Classification d'ontologie

Cette section présente les types les plus généralement utilisés d'ontologies. Ces dernières peuvent être classifiées selon plusieurs dimensions. Parmi celles-ci, nous en examinons deux (2) classifications selon l'objet de conceptualisation et le niveau du formalisme de représentation.

a. *Niveau de formalisme*

Le niveau du formalisme de représentation du langage utilisé pour décrire l'ontologie qui peut être varié de façon continue depuis l'informel jusqu'au rigoureusement formel. Le travail [Lassila & McGuinness (2001)] définit une échelle des degrés de formalisation et du niveau d'engagement sémantique en représentation de connaissances, comme la montre la figure 2.7.

L'engagement sémantique désigne le niveau d'intégration des spécifications formelles intégrées à l'ontologie qui permettent de restreindre l'interprétation des différents concepts, et donc d'en donner la sémantique [Bachimont (2000)]. Cette échelle va du vocabulaire contrôlé, simple ensemble de mots appartenant à un domaine, aux ontologies lourdes (heavy-weight ontologies), comprenant les termes désignant les concepts et relations du domaine, leurs propriétés conceptuelles, et toutes les connaissances nécessaires à la description de la sémantique du domaine. Entre les deux, des glossaires aux ontologies légères, les représentations de connaissances offrent des possibilités croissantes d'intégrer la sémantique du domaine, en spécifiant toutefois toujours la terminologie [FÜRST (2004)].

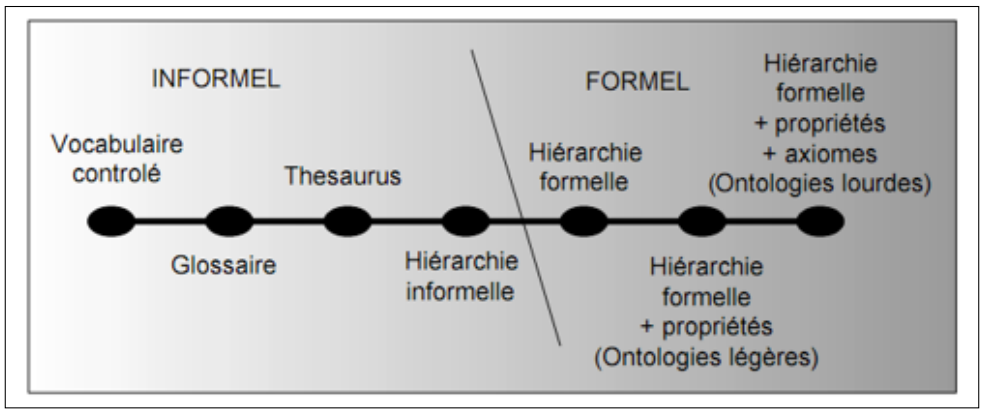


Figure 2.7 niveaux de formalisme et d'engagement sémantique d'ontologie [Lassila & McGuinness (2001)]

b. *Objet de conceptualisation*

Les ontologies peuvent être perçues à des niveaux d'abstraction variés, selon leur objet de conceptualisation (les connaissances à représenter). Guarino [Guarino *et al.* (1998)] distingue notamment :

- Les ontologies génériques, ou de haut niveau, contenant des conceptualisations générales valables dans différents domaines (ex. : temps, lieu, évènement, etc.).
- Les ontologies de domaines qui s'attachent à décrire le vocabulaire inhérent à un domaine (ex. la médecine, l'enseignement, l'e-commerce, etc.).
- Les ontologies de représentation, qui expriment les conceptualisations (primitives de représentation) des langages de représentation des connaissances, mais ne tiennent pas compte des connaissances sur le monde.
- Les ontologies d'applications (Application ontologies) et l'ontologie de tâches (Task ontologies) composées de concepts dérivés de tous les types d'ontologies évoqués précédemment. Ces ontologies émanent, donc, de groupes particuliers travaillant sur une application spécifique. Dans [Fensel & Groenboom (1997)] et [Studer *et al.* (1996)], elles sont nommées ontologies de tâches et ontologies de méthodes. Les ontologies de tâches fournissent des termes spécifiques pour des tâches particulières, et ontologies de méthodes fournissent des termes spécifiques à certains PSMs (Problem Solving Methods)). Les ontologies de tâches et les ontologies de méthodes fournissent un point de vue de raisonnement sur les connaissances de domaine [Fensel (2003)]. Certains auteurs emploient le nom « ontologie

du domaine de la tâche » pour faire référence à ce type d'ontologie comme dans le travail [Hernandez (2005)].

2.8.1.3 Langage de spécification d'ontologie

Plusieurs langages de spécification d'ontologies (ou langages d'ontologies) ont été développés pendant les dernières années, et ils deviendront sûrement des langages d'ontologie dans le contexte du Web sémantique, la plupart d'entre eux sont basés sur la syntaxe de XML. Dans cette section, nous allons présenter ceux que nous avons utilisés dans notre travail, qui sont des standards adoptés par W3C à savoir : RDF (Resource Description Framework), RDF Schéma et OWL (Web Ontology Language).

RDF (Resource Description Framework) RDF [Gruber (1995)] est un langage d'assertion et d'annotations. Les assertions affirment l'existence de relations entre les objets. Elles sont donc adaptées à l'expression des annotations que l'on veut associer aux ressources du Web. RDF est un langage formel qui permet d'affirmer des relations entre des « ressources ».
Le modèle RDF définit trois types d'objets :

- *Ressources :* les ressources sont tous les objets décrits par RDF. Généralement, ces ressources peuvent être aussi bien des pages Web que tout objet ou personne du monde réel. Les ressources sont alors identifiées par leur URI (Uniform Resource Identifier) ;
- *Propriétés :* une propriété est un attribut, un aspect, une caractéristique qui s'applique à une ressource. Il peut également s'agir d'une mise en relation avec une autre ressource ;
- *Valeurs :* les valeurs en question sont les valeurs particulières que prennent les propriétés.

Ces trois types d'objets peuvent être mis en relation par des assertions, c'est à dire des triplets (ressource, propriété, valeur), ou encore (sujet, prédicat, objet).

Exemple : L'auteur AuteurX qui a pour mail « AuteurX@mail.com », est l'auteur de la page d'adresse www.page.com.

- Sujet (ressource) : www.adresse de la page.com
- Prédicat (propriété) : créateur

- Objet (ressource) www. auteur de la page

La figure 2.8 montre un graphe RDF correspond aux assertions de l'exemple :

Figure 2.8 Un graphe RDF

Dans le graphe RDF :

- Une ressource est représentée par un cercle (ovale) ;
- Un prédicat (propriété) est représenté par un arc ;
- Un objet est représenté par un rectangle.

RDF(S) (Resource Description Framework Schema) Comme son nom l'indique, RDFS a pour but de définir des schémas de méta-données. Il définit le sens, les caractéristiques et les relations d'un ensemble de propriétés. La définition peut inclure des contraintes pour les valeurs potentielles et l'héritage des propriétés d'autres schémas. Il est, en effet, une extension sémantique de RDF afin de fournir un mécanisme pour décrire les groupes associés de ressources et les relations entre les ressources [Rada *et al.* (1989)].
L'intérêt de RDFS est qu'il facilite l'inférence sur les données et renforce la recherche sur ces données.

OWL OWL signifie Web Ontology Language. Il est défini par le W3C dans [Resnik (1995)], le langage OWL est basé sur la recherche effectuée dans le domaine de la logique de description. OWL permet de décrire des ontologies, c'est-à-dire qu'il permet de définir des terminologies pour décrire des domaines concrets. Une terminologie se constitue de concepts et de propriétés (aussi appelés rôles en logiques de description). Un domaine se compose d'instance de concepts.
OWL est un langage qui permet de construire des ontologies. Ses trois éléments principaux sont [Mbao (2007)] :

- *Classe :* c'est une entité qui permet de regrouper des éléments qui partagent les mêmes attributs. Par exemple, une classe Étudiant permet de regrouper tous les étudiants qui existent dans une école.
- *Instance :* c'est un élément appartenant à une classe donnée.
- *Propriété :* deux catégories de propriétés peuvent exister dans un fichier OWL : les propriétés entre les instances et les propriétés permettant de donner des valeurs aux instances.

Le langage OWL se compose de trois sous-langages qui proposent une expressivité croissante, chacun conçu pour des communautés de développeurs et des utilisateurs spécifiques : OWL Lite, OWL DL, OWL Full. Chacun est une extension par rapport à son prédécesseur plus simple.

- *OWL Lite* répond à des besoins de hiérarchie de classification et de fonctionnalités de contraintes simples de cardinalité 0 ou 1. Une cardinalité 0 ou 1 correspond à des relations fonctionnelles, par exemple, une personne a une adresse. Toutefois, cette personne peut avoir un ou plusieurs prénoms, OWL Lite ne suffit donc pas pour cette situation.
- *OWL DL* concerne les utilisateurs qui souhaitent une expressivité maximum couplée à la complétude du calcul (cela signifie que toutes les inférences seront assurées d'être prises en compte) et la décidabilité du système de raisonnement (c'est- à -dire que tous les calculs seront terminés dans un intervalle de temps fini). Ce langage inclut toutes les structures OWL avec certaines restrictions, comme la séparation des types : une classe ne peut pas aussi être un individu ou une propriété. Il est nommé DL car il correspond à la logique descriptive.
- *OWL Full* se destine aux personnes souhaitant une expressivité maximale. Il a l'avantage de la compatibilité complète avec RDF/RDFS, mais l'inconvénient d'avoir un haut niveau de capacité de description, quitte à ne pas pouvoir garantir la complétude et la décidabilité des calculs liés à l'ontologie [Mbao (2007)].

2.8.2 Agent et Système multi-agents (SMA)

Les travaux menés au début des années 70 sur la concurrence et la distribution ont contribué à la naissance d'une nouvelle discipline : l'Intelligence Artificielle Distribué (IAD). La technologie des systèmes multi-agents SMA est l'un des aspects de cette discipline.

Dans cette section, nous définissons, en premier, les notions d'agents et de système multi-agents, l'interaction et la communication entre agents.

2.8.2.1 Les agents

Définition d'un agent

L'une de premières définitions de l'agent et la plus admise couramment due à Ferber [Ferber (1995)] « Un agent est une entité située, réelle ou virtuelle, agissant dans un environnement, capable de le percevoir, d'agir sur celui-ci et d'interagir avec les différents composants l'entourant. Une entité est un agent si elle est capable d'exercer un contrôle local sur ses processus de perception, de communication, d'acquisition de connaissances, de raisonnement, de prise de décision et d'exécution ». Selon Wooldridge [Wooldridge (2009)] « Un agent est un système informatique capable d'agir de manière autonome et flexible dans un environnement changeant ».

Caractéristiques

Tout agent possède un ensemble de propriétés qui permettent de le caractériser, les principales caractéristiques des agents sont présentées dans les points suivants :

a. *L'autonomie* : autonome signifie que l'agent est capable d'agir sans l'intervention directe d'un humain (ou d'un autre agent) et qu'il a le contrôle de ses actions et de son état interne.

b. *La situation* : les agents sont situés dans un environnement contenant également des entités passives, manipulées par les agents (par exemple : des ressources, des données, des objets physiques. . .) et communément appelées objets. L'agent est capable d'agir sur son environnement qu'il peut percevoir grâce à ses entrées sensorielles. L'agent doit s'adapter sans cesse aux changements de son entourage qui pourraient modifier de façon pertinente son comportement à tous les niveaux (objectif, plan, action. . . etc.).

c. *La flexibilité* : la flexibilité signifie que l'agent soit :

   - *Réactif :* l'agent doit réagir rapidement à une modification de son environnement.
   - *Proactif :* il peut prendre des initiatives afin de satisfaire ses buts. Pour se faire, il n'est pas soumis à l'invocation d'une autre entité pour agir, mais peut agir sur sa propre initiative.

- *Social :* il a la capacité d'interagir pour atteindre ses buts ou pour aider d'autres agents dans leurs activités.

d. *La mobilité* : Un agent mobile est une application logicielle capable de se déplacer au sein d'un réseau de communication pour aller effectuer un traitement auprès d'un fournisseur de service puis de revenir à son point de départ avec le résultat [Nwana (1996)].

   Dans ce contexte, il y a un autre type d'agent mobile, qui est l'agent stationnaire, un agent mobile qui n'exécute en général qu'une seule migration. Cette migration s'effectue depuis la station de départ vers un nœud bien défini au lancement. À l'arrivée sur le nœud, l'agent mobile ne se déplace plus, mais exécute une tâche prédéfinie [Boukhadra (2011)].

2.8.2.2 Les systèmes multi-agents

Définition

Ferber [Ferber (1995)] définit un système multi-agents (SMA) comme étant un système composé des éléments suivants (figure 2.9) :

- Un environnement E, c'est-à-dire un espace disposant généralement d'une métrique.
- Un ensemble d'objets O situé dans E. Ces objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, créés, détruits et modifiés par les agents.
- Un ensemble A d'agents, qui représentent les entités actives du système.
- Un ensemble de relations R qui unissent des objets (et donc des agents) entre eux.
- Un ensemble d'opérations Op permettant aux agents de A de percevoir, produire, consommer, transformer et manipuler des objets de O.
- Des opérateurs chargés de représenter l'application de ces opérations et la réaction de l'environnement envers les tentatives de modification.

L'interaction entre agents de SMA

L'interaction est une mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques. Les interactions s'expriment ainsi à partir d'une série d'actions dont les conséquences exercent en retour une influence sur le comportement futur des agents [ferber1995systemes]. Les interactions inter-agents et la manière dont celles-ci sont organisées per-

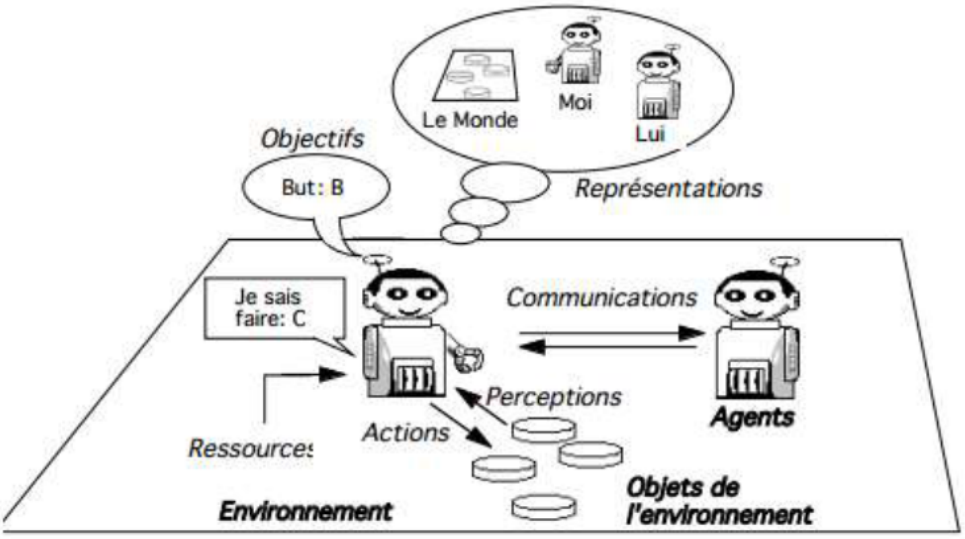


Figure 2.9 Représentation d'un agent en interaction avec son environnement et les autres agents [Ferber (1995)]

mettent aux agents de communiquer et de se coordonner, de coopérer ou encore de négocier.
Les points suivants tentent d'expliquer ces différents modes d'interaction :

- *La coordination* : La coordination est définie par Thomas Malone et Krevin Crowston comme l'acte de gérer les interdépendances des différentes activités exécutées pendant la réalisation d'un but [Bouron (1992)].
- *La coopération* : La coopération peut être considérée comme une attitude intentionnelle adoptée par les agents qui décident de travailler ensemble. On dit que plusieurs agents coopèrent ou encore qu'ils sont dans une situation de coopération si l'une des deux conditions est vérifiée :
  a. L'ajout d'un nouvel agent permet d'accroître différentiellement les performances du groupe.
  b. L'action des agents sert à éviter ou à résoudre des conflits potentiels ou actuels» [Ferber (1995)].
- *La négociation* : La négociation est un processus au cours duquel les agents échangent et évaluent des propositions (des contrats) dans le but de maximiser leurs intérêts individuels ou d'atteindre un accord sur un plan ou un but. La mise en œuvre de la négociation entre agents doit respecter les caractéristiques suivantes [Mathieu & Taquet (2000)] :
  - Les intervenants doivent pouvoir être indifféremment humains ou logiques.
  - Ils doivent tous utiliser le même langage pour communiquer.

- La communication doit être asynchrone et banalisée.
- Un agent ne doit pas avoir accès à la totalité de la connaissance des autres agents.
- Si une proposition de contrat n'est pas acceptée, l'agent doit être capable de modifier d'autres contrats et de négocier la modification de contrats des autres interlocuteurs.
- Plusieurs négociations simultanées doivent être possibles.
- Certains contrats doivent être plus prioritaires que d'autres.
- Certains participants doivent être plus prioritaires que d'autres.
- L'humain peut commencer la négociation, laisser son agent continuer un certain temps puis reprendre la négociation sans que les autres intervenants ne se rendent compte du changement.
- Le système ne doit pas se bloquer.

La communication entre les agents

La communication est une forme d'interaction entre agents. Il existe principalement deux modes de communication dans un SMA [Labidi & Lejouad (2004)] : par partage de ressources et par envoi de message.

- *Communication par partage de ressources (indirect) :* Toute l'information sur le système est centralisée dans une structure de données globale. Les agents viennent lire et écrire dans cette base de données pour faire évoluer le système qui contient initialement les données du problème. Ce type de communication, appelé «*tableau noir*» ou «*blackboard*».
- *Communication par envoi de messages (direct) :* Se caractérise par le fait que chaque agent possède une représentation propre et locale de l'environnement qui l'entoure. Chaque agent va alors interroger les autres agents sur cet environnement ou leur envoyer des informations sur sa propre perception des choses.

  La communication se fait soit en mode point à point, soit en diffusion.

Pour exprimer les communications inter-agents, des langages de communication sont proposés comme *KQML* (Knowledge Query and Manipulation Language) [Finin *et al.* (1994)]et *FIPA ACL* (Agent Communication Language of the Foundation for Intelligent Physical Agents)[Poslad (2007)]. Ces deux langages sont largement utilisés.

Les plateformes de SMA

Le meilleur moyen pour construire un SMA est d'utiliser une plateforme multi agents.
Une plateforme multi-agents est un ensemble d'outils nécessaires à la construction et à la mise en service d'agents au sein d'un environnement spécifique. Il existe actuellement de nombreuses plateformes multi-agents qu'on peut classer en plusieurs catégories :

- *Les plateformes pour agents mobiles :* comme voyager, Odissey, et Aglet, qui fournissent la mobilité à des agents.
- *Les plateformes pour agents cognitifs :* comme AgentBuilder, dans lesquels on trouve les plateformes FIPA compliant (Jade, FIPA-OS, etc.) ou KQML compliant (JAT, JAT-Lite, etc.).
- *Les plateformes pour agents collaboratifs :* comme Zeus, JAFMAS, KAoS, et JAFIMA.
- *Les plateformes pour la simulation multi-agents :* comme Cormas, Swarm, et Netlogo.

2.8.3 IA planification et Graphe de planification

En intelligence artificielle, la planification est un domaine qui recherche un enchaînement d'actions qui permet d'atteindre un état final (ensemble des buts) en partant d'un état initial [Ghallab *et al.* (2004)]. En intelligence artificielle, la planification est une partie cruciale de l'IA qui vise la détermination d'un plan d'action qui permet d'atteindre un but [Russell (2010)]. Elle est un domaine qui recherche un enchaînement d'actions qui permet d'atteindre un état final (ensemble des buts) en partant d'un état initial [Ghallab *et al.* (2004)].

2.8.3.1 Problème de planification

Avant de définir un problème de planification, il est nécessaire de présenter brièvement la définition d'une proposition.
En logique mathématique, une proposition est un énoncé qui peut prendre soit la valeur vrai ou faux [wikipedia (2016a)]. Par exemple, « le nombre deux est un entier » est une proposition vraie.
Soit $L = \{p_1, \ldots, p_n\}$ un ensemble contenant des propositions [Ghallab *et al.* (2004)].
Un problème de planification est un triplet représenté de la façon suivante [Ghallab *et al.* (2004)] :

$$P = (\Sigma, s_0, g)$$

P : désigne un problème de planification pour un domaine donné.

Σ : est un triplet composé de (S, A, γ) :

- $S \subseteq 2^L$: c'est un ensemble qui regroupe tous les états possibles (un état $s \subseteq S$ est un sous-ensemble de l'ensemble de propositions L. Un état ne retient que les propositions qui lui appartiennent) [Ghallab *et al.* (2004)].
- A : c'est un ensemble d'actions. Une action $a \in A$ est représentée comme suit :
  - $a = (precond(a), (effet^-(a)), effet^+(a))$
  - $precond(a)$: est un ensemble contenant des propositions qui sont considérées comme une précondition pour que l'action a soit applicable. (par exemple, dans un état s, si $precond(a) \subseteq s$ alors l'action a peut être appliquée) [Ghallab *et al.* (2004)].
  - $effet^-(a)$: c'est un ensemble qui représente l'effet négatif produit après l'application de l'action a (c'est-à-dire les propositions à supprimer dans le nouvel état généré après l'application de l'action a) [Ghallab *et al.* (2004)].
  - $effet^+(a)$: c'est un ensemble qui représente l'effet positif produit après l'application de l'action a. (les propositions à rajouter dans le nouvel état) [Ghallab *et al.* (2004)].
  - γ est la fonction de transition d'état $\gamma(s,a) = (s - effet^-(a)) \cup effet^+(a)$[Ghallab *et al.* (2004)]. Cette fonction permet de passer d'un état s à un autre et cela si les préconditions de a font partie de l'état s ($precond(a) \subseteq s$).

s_0 : représente un état initial.

g : représente le but à atteindre (état final).

Il est à noter que, l'intersection entre l'ensemble des effets positifs et l'ensemble des effets négatifs est un ensemble vide. Le résultat de la fonction de transition ($\gamma(s,a)$) produit un état qui appartient à l'ensemble des états (S) [Ghallab *et al.* (2004)].

2.8.3.2 Graphe de planification

Un graphe de planification est un graphe orienté structuré en niveaux qui permet d'organiser d'une manière efficiente l'espace de recherche d'un problème de planification. La solution de ce dernier est obtenue en explorant son graphe de planification [Ghallab *et al.* (2004)] [Russell (2010)].

Formellement, un graphe de planification est défini comme suit [Ghallab *et al.* (2004)] :

$$G = (L_0, L_1, \ldots, L_n)$$

G : est un graphe de planification.
n : est le nombre de niveaux que comporte le graphe de planification G.
L_0 : représente le premier niveau (level) du graphe qui est composé d'une seule couche nommée P_0. Cette couche comporte toutes les propositions de l'état initial s_0 d'un problème de planification.
$L_{1 \leq i \leq n}$: représente un niveau ayant le rang i, chacun est composé de deux couches : A_i suivie par P_i. A_i : est l'ensemble des actions (section 2.8.3.1), dont leurs préconditions sont incluses dans la couche de propositions P_{i-1}. P_i : est l'union de l'ensemble P_{i-1} et les ensembles des effets positifs de chaque action a de la couche A_i.

Les effets négatifs ne sont pas supprimés lors de la création d'une nouvelle couche de propositions. Les actions appartenant à la couche A_i sont reliées par des arêtes à leurs préconditions qui appartiennent à la couche P_{i-1}. Il existe aussi des arêtes qui partent de la couche A_i vers la couche P_i. Il y a donc deux types d'arêtes, des arêtes entrantes (de P_{i-1} vers A_i) et des arêtes sortantes (de A_i vers P_i). Le développement du graphe de planification s'arrête si le but est atteint ($g \subseteq P_i$) ou si deux niveaux consécutifs sont égaux (c'est-à-dire $P_{i-1} = P_i$ et $A_{i-1} = A_i$) [Chen (2015)].

2.9 Formalisation du problème de composition de service Web

Dans notre travail, le problème de composition de service Web consiste en la satisfaction des exigences fonctionnelles et des exigences non fonctionnelles (QoS) en même temps. Ce type de problème est appelé « QoS-aware service composition ». Plusieurs méthodes permettant la résolution de problèmes de composition de service Web ont été proposées dans la littérature. Celle qui est la

plus populaire et la plus efficace est l'application des techniques de planification en intelligence artificielle [Chen (2015)].

Il est nécessaire de transformer le problème de composition de service Web en un problème de planification, afin de pouvoir le résoudre en appliquant les techniques de planification en intelligence artificielle.

Il existe plusieurs algorithmes qui permettent de résoudre un problème de planification. Parmi ceux-ci, l'algorithme « Graphplan » permet la construction d'un graphe de planification et la recherche de la solution [Russell (2010)]. La construction (extension de graphe niveau par niveau) et la recherche se font d'une manière itérative. Dans notre travail, l'approche proposée garde la trace de la construction du graphe de planification pour les exploiter dans la phase de l'extraction de la solution.

2.9.1 Définition formelle des services Web

Dans cette section, nous adoptons la définition d'un service Web donnée dans le travail [Chen (2015)]. Cette définition permet de modéliser un service Web comme une boîte qui requiert des entrées pour fournir des sorties. Cela facilite la transformation d'un problème de composition de service Web en un problème de planification.
La définition formelle est définie comme suit [Chen (2015)] :
$D = \{c_1, \ldots, c_n\}$ est un ensemble de concepts d'une ontologie. Un service Web w est un triplet :

$$w = (input(w), output(w), Q(w))$$

$input(w)$: il représente les entrées du service w. Ces entrées sont un ensemble de concepts appartenant à l'ensemble D.
$output(w)$: il représente les sorties du service w. Ces sorties sont un ensemble de concepts appartenant à l'ensemble D.
$Q(w)$: c'est un ensemble de qualité de services (QoS) comme les temps de réponse et débit. $Q(w)$: c'est un ensemble de qualité de services (QoS) comme les temps de réponse et débit. Cette défi-

nition considère qu'un service Web est composé d'une seule opération. Dans le cas où un service possède plusieurs opérations, il est considéré comme étant plusieurs services Web [Chen (2015)].

2.9.2 Formalisation d'un problème de composition de service Web

Un problème de composition de service Web est formulé de la façon suivante [Chen (2015)] :

$$(W, D_{input}, D_{output}, Q)$$

W : il représente un ensemble de services Web. D_{input} : il représente un ensemble fourni de concepts d'entrées. D_{output} : il représente un ensemble attendu de concepts de sorties. Q : il représente un ensemble de qualités de services (temps de réponse, débit, etc.).

2.9.3 Transformation d'un problème de composition de service Web

Un problème de composition de service Web peut être transformé en un problème de planification en suivant les étapes suivantes [Chen (2015)] :

$$composition = (W, D_{input}, D_{output}, Q) \Longrightarrow planification = (\Sigma, s_0, g)$$

D_{input} : cet ensemble correspond à l'ensemble s_0 (état initial) d'un problème de planification. D_{output} : cet ensemble correspond à l'ensemble g (état final) d'un problème de planification. W : cet ensemble de services Web correspond à l'ensemble d'actions A d'un problème de planification. D : cet ensemble de concepts correspond à l'ensemble de propositions L. Par conséquent, l'ensemble $S \subseteq 2^L$ (section 2.8.3.1) correspond à un ensemble nommé $P \subseteq 2^D$. w : est un service Web qui correspond à une action a. $input(w)$: il correspond à $precond(a)$. $output(w)$: il correspond à $effets^+(a)$.

Trois manières de représenter un problème de planification classique ont été proposées [Russell (2010)] : la théorie des ensembles (en utilisant les propositions), la représentation classique (qui utilise des prédicats) et la représentation avec les variables d'états.

Dans notre travail, nous avons utilisé l'approche qui considère que les états d'un problème de planification sont des propositions (la représentation en théorie des ensembles) puisque dans le domaine de services Web il n'y a pas la notion des prédicats [Chen (2015)]. De plus, l'invocation d'un service Web ne génère pas des effets négatifs [Chen (2015)]. Donc, lors de la résolution d'un problème de composition de service Web il n'y a que les effets positifs (output(w), sont les sorties générées après l'invocation d'un service Web).

2.9.4 Exemple d'un graphe de planification

Tableau 2.2 Un ensemble de services Web

Service	**Input**	**Output**	**QoS (temps de réponse)**
w_1	p_0	p_3, p_4	100
w_2	p_1, p_2	p_4, p_5, p_6	150
w_3	p_3, p_4	p_7	200
w_4	p_5	p_8, p_9	250
w_5	p_6	p_10, p_{11}	300

Soient A, P et Q, trois ensembles. A représente un ensemble de services Web, $A = \{w_1, \ldots, w_5\}$, P est un ensemble de concepts (l'ensemble peut être vu comme un ensemble de propositions) $P = \{p_0, \ldots, p_{11}\}$ et Q, un ensemble contenant une seule qualité de service Q=temps de réponse. Le tableau 2.2 présente les différents inputs et outputs de chaque service Web, ainsi que la valeur de la qualité de service (temps de réponse).

$(A, \{p_0, p_1, p_2\}, \{p_7, p_9, p_{11}\}, \{tempsderponse\})$, est un problème de composition de service Web tel que $\{p_0, p_1, p_2\}$ est l'état initial et $\{p_7, p_9, p_{11}\}$, le but.

Le graphe de planification de ce problème est réalisé en utilisant les correspondances entre un problème de planification et un problème de composition de service Web (section 2.9.3).

La figure 2.10 montre un nouveau type d'actions baptisées les actions no-op qui sont représentées par des arêtes en pointillés. Ces actions permettent de garder les propositions d'une couche précédente dans une nouvelle couche et cela sans coût (le coût d'une action no-op est égal à 0) [Chen (2015)].

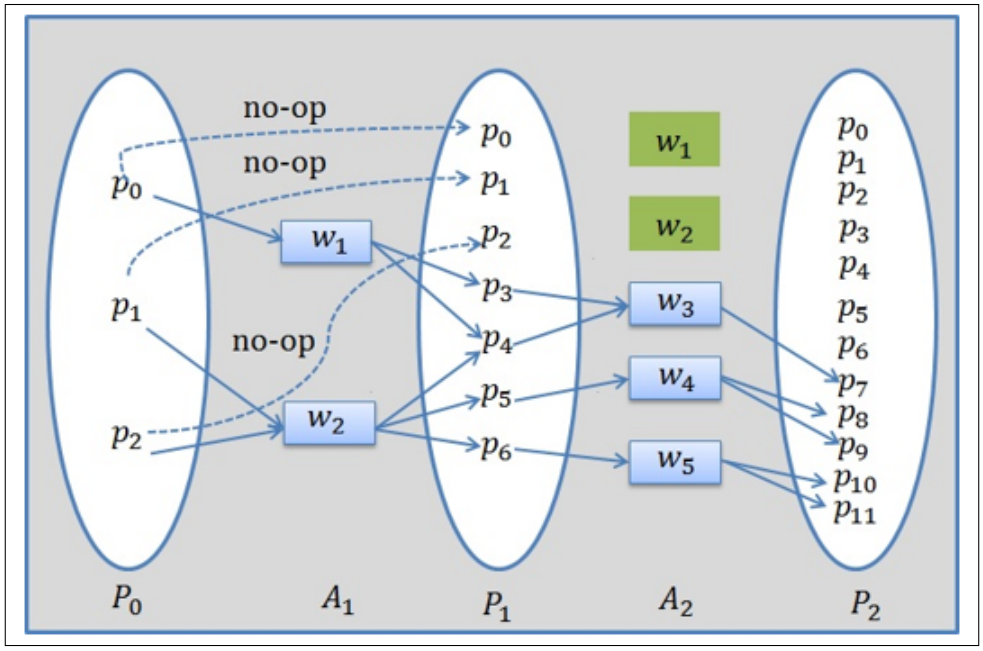


Figure 2.10 Exemple de graphe de planification

Par exemple, les propositions de la couche $P_0 = \{p_0, p_1, p_2\}$ ont été gardées dans la couche P_1 en utilisant des actions no-op.

Seules les actions no-op de la couche A_1 ont été représentées pour ne pas encombrer le graphe avec trop d'arêtes. Pour la même raison, les inputs et les outputs des services Web w_1, w_2 n'ont pas été représentés dans la couche A_2.

La solution du problème de composition de service Web précédent peut être tirée du graphe de planification ci-dessus. Cette solution est composée de cinq services Web exécutés comme suit :
$(w_1||w_2; w_3||w_4||w_5)$ signifie que les deux premiers services Web s'exécutent en premier, suivis par les trois derniers services Web.
$(w_1||w_2)$ signifie que ces deux services Web s'exécutent en parallèle. Il en est de même pour $(w_3||w_4||w_5)$.
L'orchestration des services Web (la manière dont les services Web collaborent : en série, en parallèle, etc.) se fait à l'aide du langage BPEL (section 2.5.2).

2.9.5 Services Web redondants

La solution obtenue lors de la composition de service Web contient généralement des services Web redondants. Ces services sont parmi ceux qui composent la solution, mais leur suppression n'impacte ni les exigences fonctionnelles ni les exigences non fonctionnelles.

Voici un exemple qui permet d'illustrer le concept de la redondance des services Web.

Tableau 2.3 Exemple pour illustrer les services Web redondants

Service	**Input**	**Output**	**Temps de réponse (ms)**
w_1	p_0	p_2	50
w_2	p_1	p_2, p_3	100
w_3	p_2	p_4	100
w_4	p_3	p_5	150

Le tableau 2.3 regroupe quatre services Web avec leurs inputs et outputs, ainsi que leurs temps de réponse et leurs coûts. À partir de ce tableau, un problème de composition de service Web est défini comme suit : $(A = \{w_1, \ldots, w_4\}, \{p_0, p_1\}, \{p_4, p_5\}, \{tempsderponse\})$.
L'ensemble p_0, p_1 représente l'état initial et l'ensemble p_4, p_5 représente l'état final (le but). La figure 2.11 montre le graphe de planification de ce problème de composition.
Il est clair que la proposition p_2 peut être générée par deux services Web : w_1 ou w_2. Cependant, le service w_1 la génère en 50 ms, alors que le service w_2 la génère en 100 ms. Si le service w_1 est sélectionné pour générer la proposition p_2, la solution est $(w_1||w_2; w_3||w_4)$ et la qualité de service globale de la solution est égale à 250 ms.
Dans ce cas, p_2 est atteinte en 50 ms (après l'invocation du service w_1) et la proposition p_4 qui fait partie de l'ensemble des buts est atteinte en 150 ms (valeur en rouge sur le graphe de planification).

Si le service w_2 est sélectionné pour générer la proposition p_2, la solution est $(w_2; w_3||w_4)$ et la qualité de service globale de la solution est égale à 250 ms. Dans ce cas, p_2 est atteinte en 100 ms (après l'invocation du service w_2) et la proposition p_4 qui fait partie de l'ensemble des buts est atteinte en 200 ms (valeur en vert sur le graphe de planification).

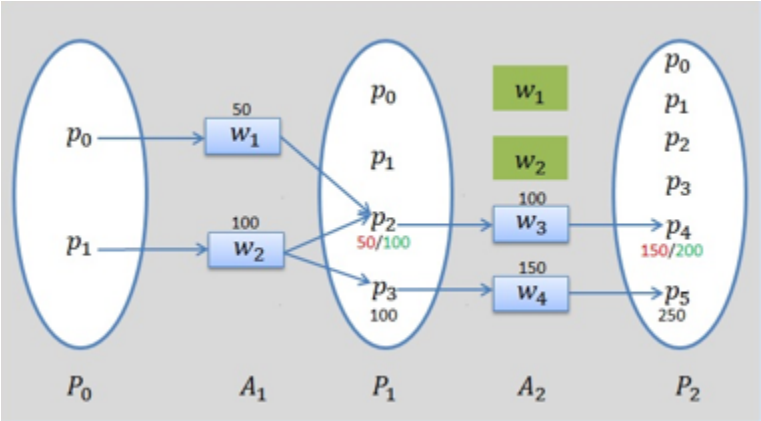


Figure 2.11 Illustration de la redondance de services

Tableau 2.4 Récapitulatif des résultats de l'exemple des services redondants

Service sélectionné	**Solution**	**Valeur globale de la QoS (ms)**	p_4 **est atteinte en (ms))**	p_5 **est atteinte en (ms))**
w_1	$w_1 \| \| w_2; w_3 \| \| w_4$	250	150	250
w_2	$w_2; w_3 \| \| w_4$	250	200	250

Le tableau 2.4 montre que les deux solutions permettent d'atteindre le but (p_4, p_5) avec la même valeur de la qualité de service globale (250 ms).

Ainsi, atteindre la proposition p_4 en 150 ms (le cas ou w_1 est sélectionné) n'a aucune influence sur la qualité de service globale, puisqu'il est nécessaire d'attendre jusqu'à 250 ms pour atteindre la proposition p_5 qui fait partie de l'ensemble des buts et doit être obligatoirement atteinte. Cela montre que le service w_1 est un service redondant, puisque sa suppression ne dégrade pas la qualité de service globale (pas de dégradation au niveau des exigences non fonctionnelles) et le but est toujours atteint même après sa suppression (les exigences fonctionnelles sont satisfaites).

Comme un exemple de l'intérêt de l'élimination de la redondance : si le coût de chaque service Web $(w_1,\ldots,w_4)$ est égal à 100\$, la solution $(w_1||w_2;w_3||w_4)$ coûte 400\$, alors que la solution $(w_2;w_3||w_4)$ coûte 300 \$.

2.10 Travaux voisins

2.10.1 Travaux voisins pour la gestion de contexte

Dans cette section, nous nous limitons aux travaux qui utilisent les ontologies dans leur proposition pour la sensibilité au contexte. Dans ce domaine de la sensibilité au contexte Le travail [Ejigu *et al.* (2007)] (intitulé "An ontology-based approach to context modeling and reasoning in pervasive computing") est motivé par la demande croissante pour intégrer la sensibilité du contexte dans les applications omniprésentes. Leur objectif dans ce travail est de proposer et d'étudier un modèle de gestion de contexte sémantiquement riche, réutilisable et évolutif basé sur l'ontologie qui prend en charge le raisonnement collaboratif dans une application omniprésente multi-domaines. Ils proposent un modèle de contexte réutilisable à base d'ontologie. Le modèle facilite le raisonnement du contexte en fournissant une structure pour les contextes, les règles et leur sémantique.

Cependant, il est difficile d'anticiper un ensemble complet de types de contexte qui convient tout système sensible au contexte surtout que le processus d'évolution du modèle n'est pas décrit dans ce travail. De plus, l'existence des dispositifs ayant des ressources restreintes contrainte l'utilisation d'un modèle de contexte générique.

Le travail [Xiao *et al.* (2010)] (intitulé "An Approach for Context-aware Service Discovery and Recommendation") est motivé par la grande quantité de services existants et les besoins diversifiés de nos jours, ce qui rend difficile aux utilisateurs finaux de trouver des services appropriés. Dans cette optique, les systèmes sensibles au contexte offrent un moyen prometteur de rechercher et de recommander automatiquement les services en utilisant le contexte de l'utilisateur.
Ils proposent une approche de modélisation de contexte qui peut gérer dynamiquement différents types et valeurs de contexte. Plus précisément, ils utilisent des ontologies pour améliorer la signification des valeurs de contexte d'un utilisateur et identifier automatiquement les relations entre différentes valeurs de contexte. Sur la base des relations entre les valeurs de contexte, ils captent les services potentiels dont l'utilisateur pourrait avoir besoin.

Le travail [Nijhuis *et al.* (2007)] (intitulé "Context-Aware Services for Constrained Mobile") se concentre sur les services de données pour les dispositifs mobiles limités qui ont généralement

un petit écran et un clavier gênant. Ils essaient de montrer comment le paradigme de sensibilisation au contexte (context-awareness) peut aider à découvrir des informations qui pourraient être intéressantes pour un utilisateur en fonction de sa situation.

Pour définir les différents types de contexte, ils proposent une ontologie (AWARENESS Ontology) qui basée sur le méta-modèle défini dans [Costa *et al.* (2006)]. Ils proposent aussi d'utiliser les hiérarchies des sources de contexte pour inférer des connaissances de haut niveau à partir des types de contexte élémentaires. Les applications clientes qui ont souscrit à des informations d'un type de contexte commenceront alors à recevoir des mises à jour de contexte.

Cependant la présentation et la communication de changements de contexte ne sont pas présentées. De plus, l'auteur a présenté l'utilisation de cette approche pour la recommandation de service.

Le travail [Jones (2008)] (intitulé "Building a context-aware service architecture") examine certaines approches de l'architecture des services sensibles au contexte, y compris la livraison et l'enrichissement du contexte, la découverte dynamique des services orientée contexte et l'invocation.

L'architecture principale qui représente le scénario général se base sur l'isolement de la logique de traitement de contexte à d'autres composants. C.-à.-d. séparer le fournisseur de contexte, la logique du consommateur du contexte et les composants du fournisseur de services lors de la mise en œuvre du scénario général. Cette séparation permet de :

- L'enrichissement futur de l'information sur le contexte physique et logique au cours du temps.
- Un raisonnement plus sophistiqué sur le contexte.
- Extension du nombre et de la nature des fournisseurs de services de soutien.

Dans notre travail [Ahmed *et al.* (2017a)], nous avons adopté le même principe de séparation. De plus, notre approche vise à minimiser la communication des changements du contexte en utilisant un format standard du document de changement qui ne contient que les changements significatifs.

Le travail [Devaraju & Hoh (2008)] (intitulé "Ontology-based Context Modeling for User-Centered Context-Aware Services Platform") tente de résoudre les difficultés des applications omniprésentes dans l'interprétation et l'intégration de connaissances ou d'informations contextuelles venant de différentes sources. C'est un complément d'un autre travail de développement d'une architecture

de middleware appelée Context-aware Service Platform (CASP), pour alléger les services sensibles au contexte de la charge de l'intégration, de la gestion et de l'interrogation d'informations contextuelles provenant de diverses sources. Ce travail se concentre principalement sur la modélisation de l'information contextuelle basée sur l'ontologie utilisant le langage OWL pour résoudre les problèmes de représentation sémantique sur le middleware de contexte.

Le framework ontologique fournit un mécanisme permettant aux services sensibles au contexte de récupérer les données contextuelles soit de les recevoir périodiquement (mode push), soit par interrogation (mode pull).

En notant que les ontologies proposées utilisent trois niveaux, à savoir ontologie fusionnée supérieure (Upper level) qui fournit des définitions aux termes généraux, représentation d'entité contextuelle (Middle level) qui fournit des concepts et relations au Lower Level qui contient les ontologies spécifiques au domaine.

Cependant, cette utilisation de différents niveaux et ontologies rend la taille du contexte plus grand. En conséquence, sa communication (pour les deux modes) soit aussi plus couteuse en terme de quantité. Dans le mode pull, quand le service interroge le contexte, il recevra une information complète de la situation actuelle du contexte que sa taille est proportionnée de la taille du contexte. Dans le mode push, le client envoie toujours les informations des capteurs (sonsors) chose qui pose un souci de la quantité d'information à communiquer puisqu'il n'y a pas un système de filtrage qui permet d'envoyer seulement les informations de contexte pertinentes c.-à-d. celles qui ont été changées et leur changement à une valeur comme le cas de notre approche. Pour la représentation du contexte, notre approche est flexible pour utiliser une ontologie de contexte déjà définie comme CONON [Wang *et al.* (2004)] ou SOUPA [Chen *et al.* (2004)], ou bien une ontologie personnalisée par le développeur.

Le travail [Van Nguyen *et al.* (2010)] (intitulé "Context Ontology Implementation for Smart Home") Dans cet article, ils présentent la réalisation d'un système sensible au contexte de la maison intelligente basé sur l'ontologie. Ce travail repose sur l'idée de découvrir que la relation entre les données d'utilisateur, d'activité et de contexte dans l'environnement domestique (home environment) est sémantique. Par conséquent, ils publient une ontologie pour modéliser ces relations et les rai-

sonner comme information sémantique. L'ontologie proposée est spécifique pour l'environnement domestique (la maison intelligente) qui aide les développeurs d'applications dans la création des applications informatiques omniprésente à domicile.

Cette ontologie est prévue qu'elle soit partageable entre le middleware et les applications afin que les développeurs d'applications puissent établir des règles pour définir le contexte de l'application sans dépendre d'un modèle de contexte spécifique de middleware. De plus, ils mentionnent que pour s'adapter dynamiquement aux changements du contexte, les applications devraient pouvoir recevoir des informations dynamiques en fonction des changements de contexte.

Dans notre approche [Ahmed *et al.* (2017a)], on adopte ce même principe de donner au développeur une flexibilité dans la définition du modèle de contexte et aussi pour la notification des changements du contexte. Cependant, cette approche n'a pas utilisé une phase de filtrage de données du contexte avant leurs envois. Les auteurs ont découvert que la quantité d'informations après avoir exclu les informations non pertinentes est 70 fois plus petite que sans l'exclusion. Malheureusement, cette exclusion est faite après l'envoi de données.

Le travail [Guermah *et al.* (2014)] (intitulé "An Ontology Oriented Architecture for Context Aware Services Adaptation") vise un défi important dans l'informatique omniprésente qui est le traitement du contexte. Dans lequel, les auteurs ont bien motivé le choix de l'ontologie pour la représentation et la modélisation du contexte en raison de leur haute expressivité formelle et des possibilités d'appliquer des techniques de raisonnement ontologique sur le contexte. De plus, les ontologies offrent un outil efficace de partage des connaissances. Ils présentent une approche basée sur l'ontologie pour le développement de services sensible au contexte. L'architecture globale proposée est composée de quatre composants principaux :

- Dispositif utilisateur : c'est un périphérique intelligent équipé des capteurs de contexte. Il permet d'entrer la requête de l'utilisateur et acquérir via les capteurs les informations contextuelles.
- La récupération et l'assimilation du contexte : Capture du contexte et son traitement.
- Le traitement et le raisonnement du contexte : raisonnement et déduire de nouvelles situations à partir des représentations OWL et de règles d'inférence.

- L'adaptation des services : à partir de situations dérivées par l'étape précédente un outil basé sur WSMO (Web Server Modelling Ontology)[Roman *et al.* (2005)] permet d'adapter les services.

Cependant, l'adaptation ici est seulement de choisir les services les plus appropriés dans une situation. De plus, nous voyons que l'utilisation d'une ontologie de contexte général a des soucis. Cette utilisation a une exige en terme de capacité et cela ne convient pas les dispositifs limités. De plus, toute ontologie utilisée, car toute ontologie proposée ne peut pas (dans la pratique) couvrir toutes les connaissances du domaine. D'ailleurs, le raisonnement contextuel basé sur l'ontologie est utilisé dans cette approche (et dans autres approches sensibles au contexte) pour dériver de nouvelles informations contextuelles sur la base des concepts définis OWL et des règles définies par l'utilisateur, en conséquence, il existe un souci du temps et de capacité nécessaire pour découvrir les changements dans le contexte.

Le tableau 2.5, représente un récapitulatif et une comparaison des travaux voisins basés sur les ontologies. Nous avons choisi comme critères la flexibilité de modèle du contexte, l'interprétation et le raisonnement du contexte, la séparation entre la gestion du contexte et l'adaptation, le changement du contexte, la communication du contexte, l'adaptation de services (Web), et la convenabilité aux dispositifs limités.

- La flexibilité de modèle du contexte : par ce critère, on distingue entre les travaux qui utilisent un modèle générique destiné à tous les cas d'utilisation (on note **-**) et un modèle spécifique et adapté au cas d'utilisation (**+**).
- Interprétation et raisonnement du contexte : désigne si le travail fait l'abstraction des données brutes capturées et le raisonnement sur ces données pour extraire de nouvelles connaissances concernant le processus d'adaptation du service. On note (**++**) si le travail fait aussi le filtrage de données.
- Séparation entre la gestion du contexte et l'adaptation : ce critère est basé sur l'importance du principe de séparation entre la gestion du contexte et les autres acteurs (c'est-à-dire du mécanisme d'adaptation et des fournisseurs de services). On note « **++** » si le module de la gestion du contexte est totalement indépendant et utilise des langages standards pour communiquer avec les autres acteurs. Et l'on note (**-**), même si l'approche sépare dans l'architecture entre le

module de la gestion du contexte, et ce de l'adaptation si elle ne fournit pas une interface qui permet l'utilisation et l'intégration facile d'un autre mécanisme d'adaptation.

- Représentation et Communication des changements du contexte : Par ce critère, nous voulons savoir si le travail communique le contexte (instance) total (par exemple, avec la requête) ou bien il communique seulement les changements significatifs du contexte.
- Adaptation de services (Web) : ce critère est pour évaluer si l'approche prend en considération, plus de la gestion du contexte, l'adaptation des applications. On note (**++**) si l'approche se concentre sur l'adaptation de services Web qui est notre objectif de ce travail.
- Convenable aux dispositifs et réseaux à ressources limités : En général, ce critère est relié aux critères précédents. On vérifie si l'approche prend en considération les limites des dispositifs et la bande passante dans tout le processus de l'adaptation (représentation, interprétation et raisonnement, communication du contexte et l'adaptation de services). Cette prise en considération est désignée par la minimisation et l'optimisation de la taille de données à traiter et communiquer et par conséquent la minimisation de la capacité et du temps nécessaire.

La comparaison révèle l'importance de la contribution présentée dans cette thèse par rapport aux autres approches.

D'autres travaux utilisant la technologie des agents dans le processus de l'adaptation comme le travail [Amor *et al.* (2003)] et [Luo *et al.* (2006)] :

Le modèle de [Amor *et al.* (2003)], les fonctionnalités et la coordination des échanges de messages sont séparées. En effet, les services Web sont vus comme des « Boites noires». Les agents interviennent dans les appels aux services afin d'adapter les invocations selon le contexte. L'adaptation est rendue possible par le recours d'un agent coordonnant les appels en fonction des besoins.
Le modèle proposé par l'auteur se base sur les entités suivantes :

- *Un (ou des) connecteur (s)(Coordination) :* cette classe coordonne les interactions entre les agents. Ces interactions sont établies à partir d'un protocole d'interaction suivant des règles, contraintes, etc.

Tableau 2.5 Comparaison entre notre approche proposée de gestion de contexte et les autres approches de domaine

Les approches	**Flexibilité de modèle du contexte**	**Interprétation et raisonnement du contexte**	**Séparation entre la gestion du contexte et l'adaptation**	**Représentation et communication du changement de contexte**	**Adaptation de services (Web)**	**Adéquation aux dispositifs limités**
[Ejigu *et al.* (2007)]	(-) Modèle générique	(+)	(+)	(-) le contexte total	(-) comme perspective	(-)
[Xiao *et al.* (2010)]	(+) il pourrait qu'on a besoin de plusieurs ontologies de domaines pour détecter les relations du contexte actuel avec les ontologies de domaines	(+)	(-) séparé dans le processus, mais il n'y a pas une interface pour intégrer un autre mécanisme ou module d'adaptation	(-) le contexte total	(+) sélection et recommandation des services	(+) le contexte est bien réduit, mais le raisonnement sur lui nécessite plusieurs ressources
[Nijhuis *et al.* (2007)]	(+) une partie est générique et l'autre est flexible selon les sources de contexte	(+) contexte Provider component	(-) inclus dans le même composant CMF container.	(-)	(+) sélection et recommandation des services	(+)
[Jones (2008)]	(+) sépare le modèle en nombre de sources de contexte	(+) via les requêtes de contexte	(+)	(-)	(+) sélection et recommandation des services	(+) Cette architecture est destiné le contexte des utilisateurs mobiles.
[Devaraju & Hoh (2008)]	(+) La spécificité est seulement au niveau du choix d'ontologie de domaine	(+)	(+)	(-)	(+) basé sur l'architecture CASP	(-) plusieurs niveaux et ressources sont nécessaires pour représenter le contexte.
[Van Nguyen *et al.* (2010)]	(+) spécifique pour les applications de maison intelligente	(+)	(+), mais dans l'implémentation ils l'ont intégré dans le système sensible au contexte	(-) c'est annoncé, mais n'est pas présenté	(-) le travail vise seulement à proposer une ontologie	(+) c'est testé sur un système de maison intelligente.
[Guermah *et al.* (2014)]	(-) Modèle générique	(+)	(+)	(-) contexte total, est pour détecter le changement du contexte il faut appliquer des raisonnements sur le contexte	(+) sélection et recommandation des services	(-)
Notre approche proposée [Ahmed *et al.* (2017a)]	(++) le modèle du contexte est spécifique pour chaque service en utilisant la notion de l'ontologie de tâche ou d'application	(+) abstraction et raisonnement simplifié par sa séparation pour chaque élément de contexte	(+) grâce aux langages standards utilisés on peut utiliser la gestion du contexte avec tout autre acteur	(++) notre approche a bien traité ce point et est basée sur le changement du contexte et sa communication	(+) adaptation de la composition de service Web présenté dans [Ahmed *et al.* (2017b)]	(++) minimisation de la taille nécessaire de données pour la représentation de son contexte et la communication de son changement

- *Un médiateur :* l'agent issu de cette classe (*AgentCompositionalCore*) est l'entité principale de l'architecture. Il gère la composition dynamique des composants et des connecteurs. Il invoque

selon les besoins le service/composant adéquat et peut, si nécessaire, brancher de nouveaux composants.

- *Un (ou des) composant(s) :* le composant est dans ce cas un service Web. Il est invoqué par le médiateur afin d'exécuter la fonctionnalité qui lui est propre.
  Les descriptions des composants se font par l'intermédiaire des ontologies. Ce modèle est basé sur le langage DAML-S.

- *Des interfaces (Interface) :* L'interface de cette architecture est soutenue par deux agents distincts : l'*AgentInterface* (AI) et l'*agentEternalCommunication* (AEC). Le AEC traite les messages d'entrée et les renvoie ensuite à la plateforme d'agents. L'AI est une interface publique de l'agent (il s'agit d'une extension du traditionnel IDL (*Interface Description Language*), d'un composant logiciel).

Le travail [Luo *et al.* (2006)], présente une plateforme permettant la composition de service Web sensible au contexte lors de la définition de la composition. Cette plateforme exécute des compositions de services orientées vers les buts. Les auteurs modélisent le service composé avec un réseau de Pétri et peuvent ainsi valider son exactitude. Trois agents logiciels sont développés dans la plateforme pour soutenir efficacement les besoins à la sensibilité au contexte dans les applications SOC (Service Oriented Computing) : l'agent Utilisateur «User Agent», l'agent Courtier «Broker Agent» et l'agent d'exécution du service «Service Execution Agent».

- *User Agent :* est responsable de la réception de la requête de l'utilisateur cherchant un service et lui restitue l'exécution finale du résultat. Il est aussi responsable de la collecte des informations contextuelles de l'utilisateur contenant un identifiant de l'utilisateur, sa localisation, etc. Cet agent sert de mémoire pour enregistrer les préférences de l'utilisateur et son contexte.

- *Broker Agent :* reçoit la requête envoyée par User Agent et extrait les informations contextuelles. Il cherche dans l'annuaire, le service répondant à cette requête. S'il n'a pas dans l'annuaire un service pertinent, il décompose la requête en sous-requêtes afin de trouver les services pertinents à chaque sous-requête. Puis, ces services sont filtrés selon l'évaluation de la fonctionnalité du contexte afin de garantir que l'utilisateur reçoive la meilleure instance du service. Enfin, une séquence d'exécution est générée.

- *Service Execution Agent :* est responsable de la génération des services impliqués dans la séquence d'exécution retournée selon le résultat composé qui est généré par Broker Agent en utilisant deux parties : le vérificateur du processus et le moteur d'exécution.
- Le vérificateur du processus a pour but de vérifier si les séquences d'exécution du service généré par Broker Agent sont possibles.
- Le moteur d'exécution est utilisé pour vérifier l'exécution du service. L'exécution peut se faire à distance ou en local.

Ces travaux ont essayé à intégrer le service Web et les technologies des agents pour l'adaptation des services Web selon le contexte d'utilisation. Ils se basent sur la composition dynamique de services (composants). Cependant, ils ne détaillent pas les techniques utilisées pour la composition. En d'autres termes, ils se concentrent beaucoup plus sur la modélisation globale des différents éléments et de leurs fonctionnalités.

Dans notre travail[Ahmed *et al.* (2017b)], et pour la composition, nous nous basons sur la technique de planification qui est considérée comme la technique la plus efficace en ce sujet [Yan *et al.* (2012)].

Dans la section qui suit, nous allons montrer des travaux voisins sur la composition de service Web qui se basent spécifiquement sur la technique de planification.

2.10.2 Travaux voisins pour la composition de service Web

Comme nous avons mentionné dans la section de l'adaptation des services Web, que nous travaillons sur la dimension de l'adaptation de la composition. Aussi, nous avons mentionné que dans cet objectif, il y a deux catégories principales, l'une se base sur le changement de la composition actuelle et l'autre est basée sur la génération d'une nouvelle composition. Pour qu'on puisse répondre mieux au besoin actuel du client exprimé en requête (et pour d'autres raisons bien mentionnées), nous nous concentrons nos efforts à la deuxième catégorie, à savoir la génération d'une nouvelle composition.

Dans la littérature, plusieurs travaux de recherche ont été consacrés à la composition de service Web en général et à la composition de SW sensible à la QoS (QoS-aware WSC). Dans ces travaux, deux types de problèmes de composition sont étudiés [Chen & Yan (2012)]. Le premier type est un problème de sélection de service, dans lequel un modèle de processus métier est prédéfini. C.-à-d. les activités dans le processus métier sont prédéfinies, alors l'objectif est de sélectionner les services adéquats pour les tâches dans le processus métier afin que le résultat processus métier peut avoir des valeurs QoS optimales.

En cas d'optimisation multicritères de QoS, ce problème est combinatoire (de type NP-complet) [Chen & Yan (2012)]. La programmation linéaire en nombres entiers (Integer programming) est un outil puissant de résolution [Zeng *et al.* (2004)]. Des techniques de la recherche heuristique peuvent être appliquées afin de rechercher partiellement dans l'espace du problème [Berbner *et al.* (2006)], [Yu *et al.* (2007)]. D'ailleurs, les algorithmes génétiques (AGs) présentent une autre façon de réduire cet espace de recherche [Canfora *et al.* (2005)]. De plus, les AGs ont l'avantage par rapport à la programmation linéaire en nombres entiers de traiter les contraintes non linéaires des critères de QoS [Chen & Yan (2012)].

Or, il n'existe aucun problème de redondance de service dans ce type de problèmes. Sachant qu'un service redondant est un service qui fait partie de la composition et que sa suppression n'a aucune influence sur les exigences fonctionnelles et celles de QoS (voir la section 2.9.5 pour plus de détail).

Cependant, ce type de problème exige l'existence permanente du processus métier. De plus, même avec l'obtention d'une composition optimale (en terme de QoS) correspondante au processus métier donné, il y a toujours la possibilité d'avoir une autre composition meilleure en QoS et satisfaite les exigences fonctionnelles visées par le processus métier, mais elle ne lui correspond pas. Pour cette raison, dans notre travail nous n'exigeons pas l'existence du processus métier prédéfini pour obtenir la composition adéquate. Partant de ce fait, nous considérons seulement dans cette section des travaux voisins, les travaux ayant le même principe c.-à-d. ceux qui considèrent la génération du processus métier fait partie de l'opération de la composition.

Dans ce contexte, la technique de planification apparaît comme la solution prédominante qui permet la détermination du processus métier et l'optimisation des critères de QoS en même temps

[Chen & Yan (2012)]. Par conséquent, dans cette section, nous nous concentrons sur les travaux de recherche existants qui utilisent la technologie de planification pour la composition de service Web, y compris [Yan *et al.* (2012); Rodriguez-Mier *et al.* (2015); Chen & Yan (2012, 2014); Zheng & Yan (2008); Poizat & Yan (2010); Pistore *et al.* (2005)].

Dans [Yan *et al.* (2012)], les auteurs proposent d'appliquer l'algorithme de Dijkstra sur le graphe de planification étendue qui est généré par l'approche GraphPlan avec un changement minimum. En particulier, ils utilisent deux types de graphes de planification, à savoir le graphe de planification partiellement labellisé (PLPG) et le Graphe pondéré en couches (LWG). Dans ce travail, les auteurs proposent une extension de l'algorithme de Dijkstra pour permettre aux nœuds de traitement ayant des sources multiples. Le GraphPlan proposé comporte deux phases, la phase de construction du graphe de planification et la phase de la solution d'extraction. Cette dernière phase consiste essentiellement, dans une recherche en arrière à partir de la couche d'objectif, qui utilise l'information obtenue dans la première phase (c'est-à-dire la phase de construction).

Dans une publication plus récente [Chen & Yan (2014)] des auteurs de [Yan *et al.* (2012)], l'approche GraphPlan, combinée à l'algorithme de Dijkstra, est également utilisée. Les auteurs proposent un moyen de modifier l'accessibilité du graphe pour faciliter l'utilisation de l'algorithme de Dijkstra et réduire la possibilité d'une explosion combinatoire. L'avantage significatif de cette approche est de trouver la solution globale optimale pour tous les types de critères de qualité de service, mais pour un seul critère. Cependant, pour atteindre ses résultats, cette approche réalise un ensemble d'étapes, à savoir la génération d'un marquage de graphe de planification (PLPG), la conversion en un graphe pondéré en couches et la génération de plan. Ces étapes prennent un temps considérable, qui pourrait même être en un temps d'explosion combinatoire.
Dans notre travail[Ahmed *et al.* (2017b)], nous nous sommes concentrés sur le temps de réponse et le débit dans la mesure de la QoS. Nous visons à réduire le temps d'exécution de la composition en sauvegardant les meilleurs chemins à travers la structure améliorée du graphe de planification (ITPG) et sa minimisation (MPG). Lorsque nous faisons la composition en une seule phase, nous terminons la construction du graphe de planification avec les meilleurs chemins sauvegardés. Ainsi,

pour l'extraction d'une solution, on ne fait que multiplier les chemins des propositions du but. En combinant cela avec le processus de minimisation (qui nécessite plus d'une phase), on réduit le temps d'exécution de la composition comme indiqué dans le chapitre de la validation(Chapitre 5).

Le travail [Zheng & Yan (2008)] est basé sur l'idée d'un graphe de planification, qui modélise le problème de composition du service Web et fournit une solution triviale au problème. Les auteurs n'utilisent pas de recherche en amont pour trouver la solution, mais proposent de supprimer les services Web redondants contenus dans le graphe de planification. Cependant, cette approche ne tient pas compte de la QoS dans le problème de composition du service Web. De plus, la solution obtenue est une solution réalisable qui peut contenir certains services Web redondants. Sachant que la suppression des services Web redondants n'affecte pas les valeurs fonctionnelles et QoS de la solution. En outre, les stratégies d'élimination des services redondants proposées dans le cadre de ce travail, d'autres travaux intéressants ont également été envisagés : l'élimination des services redondants [Yan *et al.* (2012); Rodriguez-Mier *et al.* (2015)], ou même se concentrer uniquement sur l'élimination des services redondants, tels que [Chen & Yan (2012)]. Les résultats présentés dans ces travaux démontrent leur efficacité. Cependant, le temps d'exécution reste à améliorer.

Dans [Rodriguez-Mier *et al.* (2015)], les auteurs proposent deux méthodes d'ASC, de recherche locale et de recherche globale. La recherche globale cherche à obtenir la composition avec un nombre minimum de services. Cependant, cela consomme beaucoup de temps et peut ne pas renvoyer aucun résultat dans un délai raisonnable (5 minutes dans ce travail). Pour la recherche locale qui vise d'avoir un bon rapport temps d'exécution/service redondants, peut également contenir des services redondants.

Dans notre travail[Ahmed *et al.* (2017b)], la structure du graphe de planification amélioré avec sa minimisation et les stratégies de sélection des services et du chemin de solution que nous allons présenter dans la section 4.6, réduisent le nombre de services redondants dans la solution, ainsi que le temps d'exécution de la composition. Il convient de rappeler que l'objectif principal de notre travail est d'améliorer le temps d'exécution de l'ASC QoS-aware tout en minimisant le nombre

de services de la composition autant que possible. Pour cela, nous avons proposé des stratégies de minimisation de services dans la composition. Ces stratégies sont subdivisées en deux catégories, à savoir les stratégies globales (amélioration du graphe de planification classique et sa minimisation) qui vise à minimiser l'espace de la recherche et les stratégies spécifiques (sélection de services et de solutions) qui aide à obtenir une solution ayant moins de services redondants.

Dans le cas où l'élimination du service redondant est plus importante que le temps d'exécution du processus de composition, d'autres travaux d'élimination des services redondants peuvent être appliqués en complément de notre travail, c'est-à-dire qu'ils éliminent les services redondants du résultat obtenu par notre approche ASC. Dans [Pistore *et al.* (2005)], une approche de composition automatisée de services Web est proposée en utilisant la technique de planification dans les domaines asynchrones. La particularité de cette approche consiste à décrire le comportement interne des services Web à l'aide du modèle BPEL4WS. Cette description des services Web est traduite en un système d'état de transition(STS).
Cependant, la description du comportement interne n'est pas toujours disponible. Pour cela, dans le problème de la composition des SW, il est plus pratique de considérer les services comme des boites noires ayant des entrées et sorties. De plus, cette approche ne tient pas compte de la QoS dans le processus de composition.

2.11 Conclusion

Dans ce chapitre, nous avons présenté les notions concernant les services Web qui sont considérés comme des composants logiciels représentant une fonction applicative. Ils représentent un mécanisme de communication entre applications distantes à travers le Web en utilisant différents standards XML : SOAP pour l'échange de messages, WSDL pour la description de services et UDDI pour la publication et la découverte de services. Ils reposent sur une architecture orientée service (SOA) et correspondent à des composants logiciels qui peuvent être combinés, pour former de nouveaux services composés plus élaborés. Nous avons montré notre motivation pour le choix du type de l'adaptation des services Web, à savoir l'adaptation de la composition de service. De plus, nous avons présenté les techniques les plus utilisées à savoir les ontologies et la planifica-

tion en intelligence artificielle. En adoptant la technique de planification, nous avons montré une représentation formelle du problème de la composition de service Web.

Par ailleurs, une section a été consacrée pour la présentation des travaux voisins organisés en deux sous-sections, à savoir, la gestion de contexte et la composition de service Web. Pour la gestion de contexte nous avons constaté le problème de l'utilisation d'un modèle générique du contexte et l'absence de la séparation entre la gestion de contexte et le mécanisme de l'adaptation. Concernant la composition de service Web, nous avons montré le problème de l'optimisation de temps du processus de la composition de services Web tout en prenant en considération le nombre de services dans la composition.

Le chapitre suivant est dédié à la présentation de notre approche pour la gestion de contexte d'utilisation dans un environnement ubiquitaire.

CHAPITRE 3

GESTION SÉMANTIQUE DU CONTEXTE DANS UN ENVIRONNEMENT UBIQUITAIRE

3.1 Introduction

L'informatique ubiquitaire permet une bonne intégration des technologies informatiques dans divers domaines tels que l'éducation, la santé, le transport en formant la vie quotidienne de la société moderne. Cependant, l'environnement ubiquitaire est caractérisé par l'utilisation des divers types de dispositifs (PC, smartphone, etc.) et la mobilité des utilisateurs. Chose qui nécessite une adaptation permanente des applications au contexte d'utilisation c.-à-d. les applications s'adaptent selon plusieurs éléments comme la capacité des dispositifs qu'ils sont à la disposition des utilisateurs et à leurs localisations.
La nature distribuée de ce genre d'environnement et la mobilité des dispositifs utilisés qui sont souvent ont une capacité limitée (qui ne permet pas de gérer un grand volume de connaissances), exige une gestion particulière de contexte soit pour sa collection, représentation et son changement afin de répondre au problème d'hétérogénéité des représentations, les limites des dispositifs et de la bande passante du réseau.

Le travail présenté dans ce chapitre s'inscrit dans un cadre d'une solution globale pour l'adaptation sémantique des services au contexte d'utilisation sous la forme d'un nouvel intergiciel (middleware) pour les services sensibles au contexte. Cependant, nous avons vu qu'avant d'adapter un service au contexte d'utilisation, nous devons avoir une gestion efficace du contexte qui prend en considération les contraintes susmentionnées. En notant qu'il est préférable que la gestion du contexte soit indépendante des autres acteurs (c'est-à-dire du mécanisme d'adaptation et des fournisseurs de services) comme dans [Ejigu *et al.* (2007), Jones (2008)]. De plus, cette séparation permet à l'adaptation de service d'être indépendant, le plus possible, du domaine tel que dans [Kapitsaki *et al.* (2008)]. Dans ce chapitre, nous nous limitons, alors, de ne présenter que notre approche sémantique de représentation et gestion du contexte d'utilisation.

Pour faire face aux défis de la gestion du contexte dans un système ubiquitaire, et compte tenu des limitations des solutions existantes, nous proposons une approche de gestion de contexte basée sur une représentation sémantique du contexte en utilisant des langages standards tels que RDF, RDF(s). Notre approche est caractérisée par sa grande flexibilité, simplicité et permet au développeur de personnaliser le modèle du contexte d'utilisation de service. Cela nous permet d'éviter à gérer un modèle global de contexte qui a besoin plus de capacité, comme le cas dans les travaux [Tamura *et al.* (2013)] [Belhanafi *et al.* (2005)] [Gu *et al.* (2004)].

Dans notre travail de ce chapitre et pour la modélisation sémantique, nous nous basons sur l'utilisation de la notion d'ontologie qui est devenue une clé pour la gestion de l'information au niveau sémantique. Dans cette optique, nous voyons que le développeur propose son point de vue concernant le modèle du contexte d'utilisation de son service sous forme d'une ontologie spécialisée qui peut utiliser les concepts d'autres ontologies de domaine. Cette ontologie spécialisée peut être vue comme ontologie d'application qui définit des concepts très spécifiques à un domaine et une tâche particulière, comme indiqué dans la section 2.8.1. De ce fait, chaque utilisation du service engendre une instanciation de l'ontologie d'application. Concernant le changement de contexte, nous avons déterminé quels sont les types de changement qui peuvent être appliqués et nous proposons de garder ces changements dans ce que nous baptisons journal de changements (Change log).
L'utilisation et la communication du journal permettent de faire face aux limites de capacité des dispositifs utilisés et aussi il assure la séparation entre la gestion du contexte et les autres acteurs (le mécanisme d'adaptation et le fournisseur du service). Car il ne contient que les opérations de changement appliquées et nous proposons que le journal soit écrit en langage commun et standard.

Or, notre approche se base essentiellement sur l'outil de système multi-agents (SMA) qui présente un aspect de l'Intelligence Artificielle Distribuée (IAD). Un SMA se base sur l'interaction et la coopération entre les différents agents qui peuvent même être mobiles. Chose qui peut contribuer davantage sur les contraintes des systèmes ubiquitaires dues à leur nature distribuée et mobile.

Ce chapitre est organisé comme suit : la section 3.2 montre comment ces notions (ontologie et SMA) s'inscrivent dans une architecture globale contenant les différents acteurs afin d'assurer la gestion sémantique de contexte. Dans la section 3.3, nous détaillons l'aspect sémantique de notre

approche, qui est présenté sous la forme des éléments passifs dans notre architecture. Tandis que, la section 3.4, montre les éléments actifs de notre architecture par la modélisation des différentes interactions entre les agents et leurs structures. En finissant par une conclusion qui résume le travail.

3.2 Architecture globale de notre approche de gestion de contexte

Comme déjà mentionné dans l'introduction, nous proposons une approche de gestion sémantique de contexte. Cette approche est basée sur l'utilisation d'ontologie et la technique d'agent.

Dans cette section, nous présentons l'architecture conceptuelle globale de la partie de la gestion sémantique de contexte du notre intergiciel (middleware).

L'architecture de notre middleware combine les différents éléments logiciels afin de permettre aux systèmes (clients et fournisseurs) de communiquer et d'interagir entre eux d'une manière sensible au contexte c.-à-d. il permet au demandeur de services à présenter son contexte actuel et aussi permet au fournisseur de service à l'adapter selon l'état du contexte. La figure 3.1 schématise, alors, la partie de la gestion sémantique du contexte. Où chaque service a son propre modèle du contexte (sous la forme d'une ontologie spécifique de contexte).

L'architecture est divisée en trois couches, à savoir la couche des sources, de contexte et de l'application. La couche des sources de contexte (context source level) s'occupe essentiellement à la gestion de sources de contexte qui peuvent être des senseurs physique (comme un thermomètre) ou logique (comme une base de données). Le rôle principal de la couche de contexte (Context level) est de gérer le contexte et le communiquer aux autres composants, en se basant sur les lectures obtenues de la couche de sources de contexte. La couche applicative (Application level) assure une consommation du service adaptatif au contexte d'utilisation géré et communiqué par la couche du contexte.

On associe un agent (Agent Client) à chaque client qui consomme un service. Cet agent permet de présenter le client et son contexte dans le système, il crée une instance du modèle de contexte. Cette instance, qui présente l'état actuel du contexte de client, va être exploitée par le fournisseur du service afin de fournir des résultats et un comportement adaptés à cet état. En rappelant que

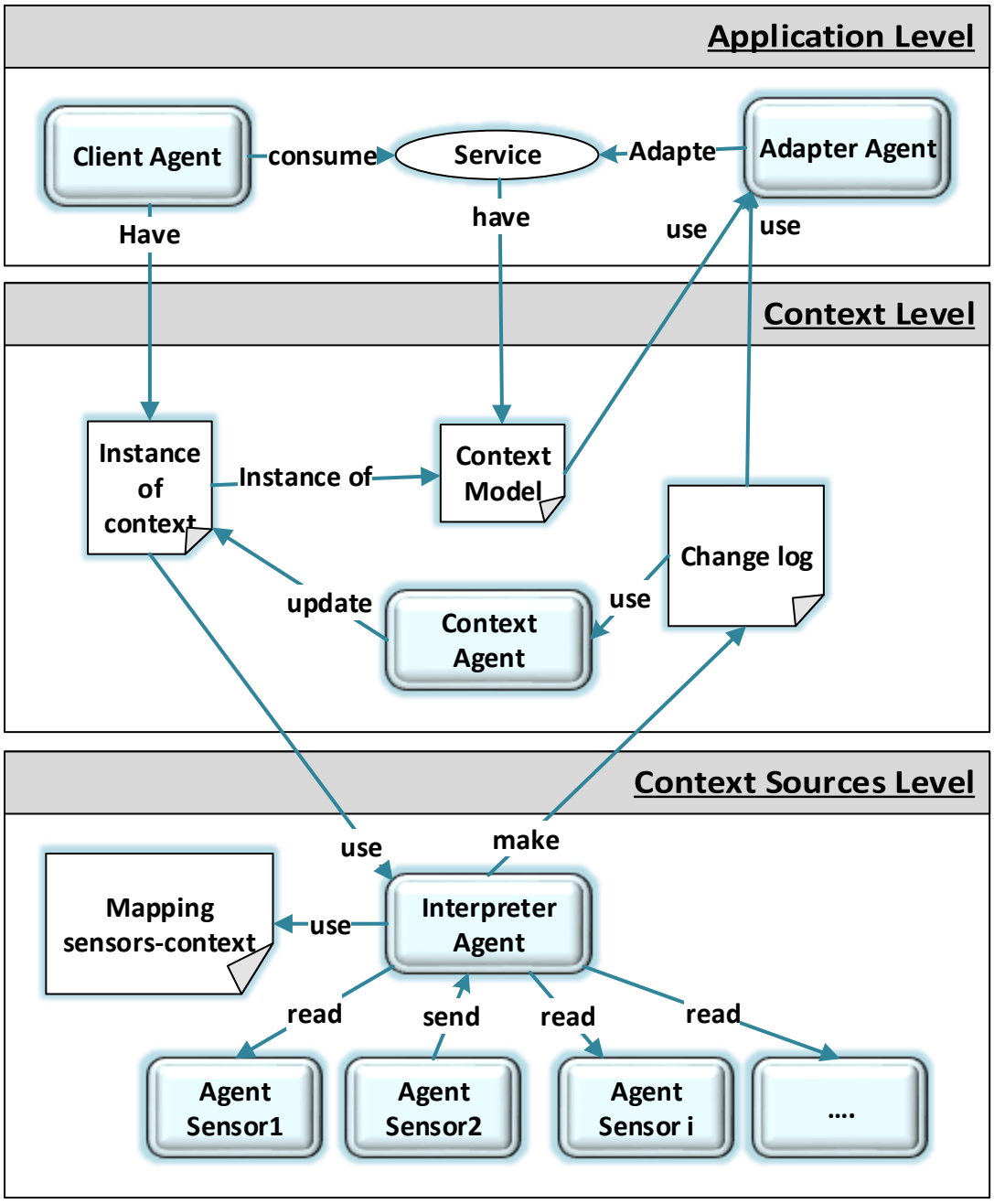


Figure 3.1 Architecture globale de notre approche de gestion de contexte

l'instance contient des informations concernant la consommation de service qui sont jugées par le développeur (à travers le modèle du contexte) comme pertinentes à être utilisées pour l'adaptation du service.

D'un autre côté, on a les Agents Capteurs (Sensor Agents) qui assurent la lecture des valeurs obtenues par des senseurs (capteurs) de contexte, ces senseurs assurent la capture de l'état actuel de l'environnement de l'exécution. Plus précisément, ils surveillent les valeurs des éléments du contexte. Pour cela, généralement, on affecte un capteur à un élément spécifique du contexte (correspondance capteurs-contexte dans la figure 3.1). Par exemple, si on a un concept « Température » alors on peut avoir un senseur qui mesure la température actuelle. Le senseur peut être physique ou logique. Après la lecture des valeurs obtenues par leurs senseurs, les Agents Capteurs envoient ces valeurs à l'agent interpréteur.

S'il est nécessaire, l'agent interpréteur construit un journal de changement de contexte (change Log, qui contient seulement les changements significatifs) et l'envoie à l'agent de contexte sous une forme plus adéquate au contexte d'utilisation. Par exemple, en prenant le concept «Température», l'agent interpréteur peut mettre la valeur «hot» si le senseur mesure une température plus de 35. Nous notons que nous pouvons utiliser la logique floue afin de caractériser les valeurs obtenues à partir de senseurs.

Si l'interpréteur détecte un changement du contexte, il crée le journal de changements. Ce dernier va être exploité par l'agent de contexte qui à son tour assure la mise à jour du contexte actuel. En notant que les différents constituants des couches sont éventuellement distribués entre les différents acteurs, par exemple, l'instance du modèle de contexte peut être enregistrée au niveau du dispositif de client ou bien au niveau du serveur de service. Et cela dépend des caractéristiques techniques comme la capacité des dispositifs et aussi à la stratégie suivie (adaptation cotée client, côté proxy, ou côté serveur).

3.3 Architecture détaillée des éléments passifs de notre approche de gestion de contexte

Nous voulons dire ici par les éléments passifs, les éléments de base communiqués entre agents à savoir, le modèle de contexte, l'instance du contexte et le journal de changement.
Dans la section qui suit, nous allons présenter l'utilisation de la notion d'ontologie dans notre proposition concernant la gestion de contexte dans un environnement ubiquitaire.

3.3.1 Une représentation basée sur l'ontologie

L'ontologie pour décrire un domaine de connaissance définit les concepts du domaine et les relations entre eux.
Le succès et la large utilisation de l'ontologie dans plusieurs domaines comme des spécifications partagées de connaissances invitent les chercheurs dans le domaine de la sensibilité au contexte à exploiter cet important outil. Cependant, nous avons remarqué que cette utilisation à exiger aux développeurs des services et utilisateurs (compris les agents machines) à n'utiliser qu'un seul modèle commun de contexte présenté par une ontologie de domaine. Il est connu que l'objectif essentiel et initial de l'ontologie est le partage des connaissances d'un domaine. C'est le cas de l'ontologie de domaine, mais il y a d'autres raisons pour lesquelles les ontologies peuvent être utilisées d'une manière personnalisée comme les cas de l'ontologie de tâches (Task ontology) et l'ontologie d'application (Application ontology).

L'ontologie d'application qui définit des concepts très spécifiques à un domaine et une tâche particulière. Selon Gruber ces concepts correspondent souvent aux rôles joués par les entités du domaine lors de l'exécution d'une certaine activité [Guarino *et al.* (1998)]. Nous adoptons ce genre d'ontologie dans notre proposition de gestion de contexte, où nous voyons que le développeur du service doit proposer son ontologie qui modélise son point de vue concernant le modèle du contexte d'utilisation de l'application [Ahmed *et al.* (2017a)].

Donc, pour la représentation du contexte nous proposons d'utiliser un modèle ontologique, qui permet de le représenter en manière sémantique. Par cette modélisation et par l'utilisation des langages standards tels que RDF et RDF(S), nous pouvons résoudre le problème d'hétérogénéité des représentations et des protocoles de communication utilisés (qui assure la séparation entre différents acteurs de l'adaptation). Ainsi, dans notre solution nous avons laissé au développeur la possibilité de personnaliser (présenter son point de vue concernant) le contexte utilisé par son service fourni. De ce fait, chaque utilisation du service correspond une instanciation du modèle de contexte défini. Pour la modélisation et la présentation graphique de contexte, nous proposons d'utiliser le modèle des réseaux sémantiques qui permet de modéliser les concepts et ses relations entre eux. Et lorsqu'il y a une utilisation du contexte, il modélise aussi les liens entre les instances des concepts spéci-

fiques du cas actuel de contexte. Les concepts utilisés dans la présentation du contexte peuvent être obtenus à partir de l'ontologie du domaine.
Le choix du paradigme réseau sémantique est motivé par sa capacité de la représentation sémantique, sa grande flexibilité et simplicité. De plus, ce paradigme est très proche au modèle de graphe RDF chose qui facilite la transformation de ses modèles au langage RDF(S).
Alors, pour définir un contexte, nous devons définir trois choses :

- Sujet : le sujet de la modélisation par exemple Utilisateur ;
- Concept : tous les domaines contiennent de nombreux concepts, et tout concept peut prendre une valeur (instances du concept) ;
- Relation : grâce aux relations, nous présentons les relations sémantiques entre les concepts du domaine. Les instances des concepts impliqués par une relation peuvent avoir également des liens.

3.3.2 Exemple d'illustration

Pour la description de notre approche, nous donnons un exemple afin d'illustrer comment il peut être utilisé pour la gestion de contexte selon notre approche.
La figure 3.2 présente l'exemple sous la forme du graphe réseau sémantique.
On peut remarquer que le concept racine (le sujet) présente le type de contexte comme "user" pour désigner le contexte de l'utilisateur et "environement" pour le contexte de l'environnement.
Par cette représentation, on peut distinguer les utilisations du concept "Name" (qui prend une chaine de caractères comme valeur) pour désigner le nom de l'emplacement de l'utilisateur (user.environment.place.name) et le nom d'un utilisateur (user.name).
Cet exemple présente alors une ontologie correspond à un modèle global du contexte proposé par le développeur d'un service. Ce modèle définit les concepts et relations que leurs changements peuvent entraîner un changement de comportement du service. Ce modèle doit décrire tous les éléments de base du contexte (ex. user, environment, etc.). Lors de l'utilisation du service on fait l'instanciation de ce contexte c.-à-d. on met les différentes valeurs des instances de ces concepts et aussi on crée les liens (instances des relations) entre les différentes instances.
Pour cette raison, nous pouvons avoir beaucoup d'instances qui ne sont pas liées. Cela permet de

distinguer les utilisateurs dans les différents cas. Par exemple, on prend l'élément "Mobile" lorsque l'utilisateur est connecté par un mobile, nous relions une instance du concept "Device" avec une instance du concept "mobile", et par la même manière et dans la même instance du contexte, nous pouvons représenter que l'utilisateur est connecté avec un ordinateur.

Aussi, par ce modèle de représentation on peut même présenter la qualité de contexte (QoC). C.-à-d. la qualité exigée aux informations du contexte et qui doivent toujours les assurer comme la certitude (une instance du concept peut accepter une valeur incertaine ou non).

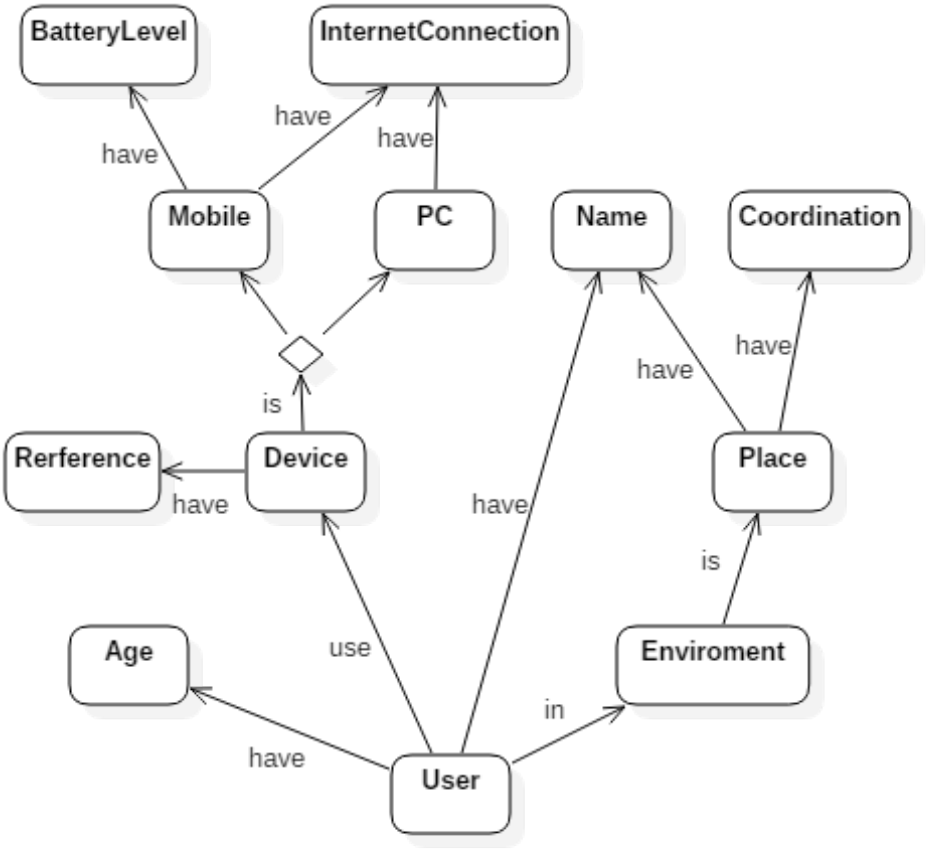


Figure 3.2 Example de contexte

3.3.3 Présentation formelle et commune

Vu de problème de l'hétérogénéité de la représentation du contexte et sa compréhension, surtout si le contexte de service est spécifié par le développeur. Il est, alors, nécessaire de le présenter dans un format commun pour élargir la portée de son utilisation. À cet effet, nous utilisons, pour la description du contexte, le langage RDF(S), adopté par le W3C comme formalisme standard de représentations. Ce langage est, expressivement, riche de représenter toutes les informations disponibles au niveau de contexte.
Le tableau 3.1 présente une partie de contexte présenté dans la figure 3.2 en langage RDF(S) :
Le formalisme XML contribue à l'interopérabilité et aussi la limite de la taille du fichier décrivant le contexte, car le développeur est limité à ne décrit que les concepts et relations du contexte que leur changement a une influence sur le service développé. Cela présente une réponse aux dispositifs ayant des capacités limitées.
Chaque consommation du service ou application engendre une utilisation du modèle de contexte, chose qui s'interprète par la création d'un document RDF contenant les instances des concepts utilisés et leurs valeurs plus les liens entre elles.
Le tableau 3.2 présente un document RDF décrivant un utilisateur X a 24 ans utilise un PC de type HP.

Après la description de la présentation sémantique de l'instance du contexte, nous allons voir comment présenter son changement.

3.3.4 Opérations de changement

À partir de la représentation de contexte (instance du modèle de contexte) proposée au-dessus, nous identifions les types d'opérations de changement suivant, qui soient l'ajout, la suppression ou bien la modification de l'un de ses constituants.
L'ensemble (la taxonomie) d'opérations de changement est représenté dans le tableau 3.3.

En notant que ces opérations de changement sont détectées par l'interprétation des valeurs obtenues à partir des senseurs.
Ainsi, l'application d'une opération de changement peut entrainer l'application d'autres opérations

Tableau 3.1 Représentation sémantique et formelle du contexte

```
< ?xml version="1.0" ?>
<rdf :RDF
xmlns :rdf="http ://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns :xsd="http ://www.w3.org/2001/XMLSchema#"
xmlns :rdfs="http ://www.w3.org/2000/01/rdf-schema#"
xmlns :DO="http ://localhost/DomainOntology#">
....
<rdfs :Class rdf :ID="User"/>
<rdfs :Class rdf :ID="Name"/>
<rdfs :Class rdf :ID="Age"/>
<rdfs :Class rdf :ID="Environement"/>
<rdfs :Class rdf :ID="PC"/>
<rdfs :Class rdf :ID="Mobile"/>
<rdfs :Class rdf :about="#Device">
<rdf :Alt>
      <rdf :li><rdf :Description rdf :about=”PC”></rdf :li>
      <rdf :li><rdf :Description rdf :about=”Mobile”></rdf :li>
</rdf :Alt>
<rdfs :Class rdf :ID="Place">
      <rdfs :subClassOf rdf :resource="#Environement"/>
</rdfs :Class>
<rdfs :Class rdf :ID="Connect"/>
<rdfs :Class rdf :ID="Bluetooth"/>
.....
<rdf :Property rdf :ID="have">
      <rdfs :domain rdf :resource="#User"/>
      <rdfs :range rdf :resource="#Name"/>
      <rdfs :range rdf :resource="#Age"/>
.....
</rdf :Property>
<rdf :Property rdf :ID="in">
      <rdfs :domain rdf :resource="#User"/>
      <rdfs :range rdf :resource="#Environement "/>
</rdf :Property>
<rdf :Property rdf :ID="value">
      <rdfs :domain rdf :resource="#Name"/>
      <rdfs :range rdf :resource="rdfs :Literal"/>
</rdf :Property>
</rdf :RDF>
```

de changement pour conserver la consistance de l'instance du modèle de contexte.

Par exemple, si on suppose que les senseurs ont détecté que l'utilisateur a changé son dispositif de PC à mobile c.-à-d. il n'utilise plus son PC. Alors on a comme opérations directes :

Tableau 3.2 Exemple d'utilisation du modèle de contexte

```
<rdf :RDF
xmlns :rdf="http ://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns :OE=http ://localhost/evolutionontology/
xmlns :CM="http ://localhost/ContextModel/"
 <rdf :Description rdf :ID="X">
  <rdf :type rdf :ressource="CM :User"/>
  <CM :have>
   <CM :Age>24<CM :Age/>
  </CM :have>
  <CM :use>
   <rdf :Description rdf :resource="Device. . . ">
    <rdf :type rdf :ressource="CM :Device"/>
    <CM :is rdf :type=”CM :PC”>
     <rdf :Value>HPxyz</red :value>
    </CM :is>
   </rdf :Description>
  </CM :use>
 </rdf :Description>
</rdf :RDF>
```

Tableau 3.3 Opération de changement du contexte

Element|Operation	**Add**	**Delete**	**Update**
Instance	AddInstance	RemoveInstance	IDChange
Value	AddValue	EraseValue	ValueChange
Link	AddLink	RemoveLink	DomainChange
			RangeChange

AjouterLien('is', Dispositif, mobile) et SupprimerInstance(PCi). Alors la suppression de la relation 'is' entre PC et Dispositif (SupprimerLien('is', Dispositif,PCi)) est une relation générée.
C'est pour cette raison, nous introduisons un type de relation 'Entraîne' (generate) entre les concepts représentant les opérations de changement pour noter qu'une opération de changement peut entraîner l'exécution d'une autre opération.
La figure 3.3 présente la relation 'Entraine' utilisée dans l'exemple.

Ces opérations de changement et relations peuvent être vues comme une ontologie de changement. Cette dernière va être exploitée par le journal de changement qui va être décrit dans la section qui suit.

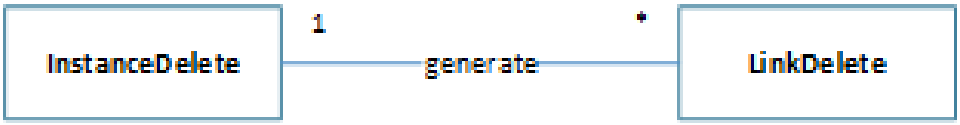


Figure 3.3 Generate relationship

3.3.5 Journal de changements

Le premier rôle du journal de changement est de garder les traces de changement du contexte, c.-à-d. l'ensemble des opérations de changement appliquées lors de changement de contexte. Et comme deuxième rôle, ce journal se considère comme un moyen de communiquer le changement du contexte d'un service ou une application.

Pour cela, il présente une séquence d'instances de l'ontologie de changement. Ces instances correspondent aux opérations de changement appliquées.

On reprend l'exemple, de changement de dispositif PC au dispositif mobile. Le journal d'évolution doit garder l'ensemble des opérations appliquées directement et aussi les opérations générées. Alors le journal soit :

JE=AjouterLien('is', Dispositif, mobile), SupprimerInstance(PCi), SupprimerLien('is', Dispositif, PCi) Vu l'importance du journal de changement, il est nécessaire de le présenter dans un format commun pour élargir la portée de son utilisation. De ce fait, nous utilisons, pour la description du journal, le langage RDF, adopté par le W3C comme formalisme standard de représentations.

Le RDF associe trois types d'objets : une ressource «resource» définie par des propriétés «proprerties» ; l'attribution d'une valeur à une propriété d'une ressource est une déclaration « statement».

Alors, pour représenter le journal de changement avec le langage RDF, nous considérons que les opérations de changement comme des ressources, les paramètres de ces opérations comme des propriétés prenant leurs valeurs dans l'ensemble des éléments du contexte changé.

La figure 3.4 représente le modèle de données pour l'opération 'AjouterLien('is', Dispositif, mobile) de l'exemple cité précédemment.

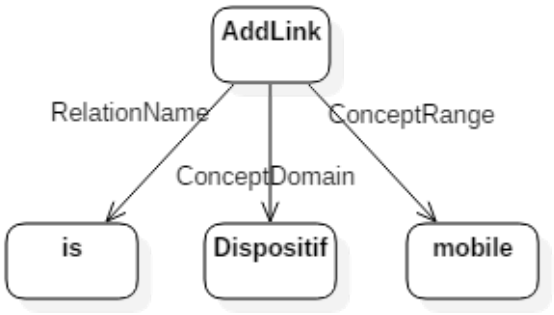


Figure 3.4 Le graphe RDF de l'opération AddLink('is', Dispositif, mobile)

La représentation du modèle de donnée (graphe RDF) de la figure 3.4 en langage RDF/XML est dans le tableau 3.4.

Tableau 3.4 Exemple en format RDF du journal de changement

```
<rdf :RDF
xmlns :rdf="http ://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns :CM="http ://localhost/ContextModel/"
xmlns :CCO="http ://localhost/ContextChangeOntology/"
  <rdf :Description about="AddLink">
    <CCO :RelationName rdf :resource="is"/>
    <CCO :ConceptDomain rdf :resource="Dispositif"/>
    <CCO : ConceptRange rdf :resource="mobile"/>
  </rdf :Description>
</rdf :RDF>
```

Tel que, CCO est un espace de nom qui permet de rendre disponible le vocabulaire de propriétés des opérations de changement définies dans l'ontologie de changement de contexte.
En raison de simplicité, on n'a pas présenté les valeurs des instances.

3.4 Architecture détaillée des éléments actifs de notre approche de gestion du contexte

Dans la première partie de ce chapitre, nous avons détaillé les éléments de représentation (modèle de contexte, instance du contexte, ontologie de changement et journal de changement). Ces éléments sont des éléments passifs dans notre architecture globale présentés dans la figure 3.1.

Dans cette section, nous allons présenter les éléments actifs, c.-à.-d, les différents agents du système (intergiciel). Notre modélisation de ces éléments se déroule en deux phases, à savoir la modélisation des interactions au sein de notre système qui nous mène à modéliser leur structure via le diagramme de classes et la description de leurs architectures internes.

3.4.1 Interactions dans le système

Nous présentons ci-après le protocole d'interaction entre les différents composants du système. Un protocole d'interaction est défini par une séquence d'actes de communication échangés entre les agents du système.

3.4.1.1 Actes échangés

Nous définissons les types d'actes échangés suivants :

- Service_query : demande de service à consommer.
- Consume_service : service à consommer.
- Context_model : modèle du contexte.
- Inst_context : Instance du contexte d'utilisation actuel de service.
- Change_log : rapport de changement dans le contexte d'utilisation (journal de changement).
- Read_context_val : demande de lire la valeur d'un paramètre du contexte.
- Send_context_val : nouvelle valeur lue d'un paramètre du contexte.
- update_inst_context : application des mises à jour dans l'instance du contexte.
- adapte_service : adaptation du service selon le nouveau changement du contexte.
- adapted_service : service à consommer après l'adaptation.

Le tableau 3.5 explique pour chaque acte échangé : l'émetteur, le récepteur, contenu et son code (pour l'identifier sur le diagramme de séquence du protocole d'interaction).

Tableau 3.5 Actes échangés entre les Agents du système

N	Code Acte	Emetteur	Récepteur	Explication
1	Service_query	ACl	AA	Requête de service
2	Consume_service	AA	ACl	Consommation (invocation) du service
3	context_modèle	AA	AC	Modèle du contexte
4	Inst_context	AC	AA, AI	Instance du modèle de contexte
5	Read_context_val	AI	AS	Demande à l'agent AS de fournir la valeur actuelle du contexte
6	Send_context_val	AS	AI	Envoi de la valeur actuelle du contexte
7	Change_log	AI	AC, AA	Le journal de changements
8	update_inst_context	AC	AC	Mettre à jour le contexte
9	adapte_service	AI	AI	Adapter le service
10	adapted_Service	AA	ACl	Le service adapté

3.4.1.2 Diagramme d'interactions

La figure 3.5 présente un diagramme de séquences UML qui modélise les différentes interactions entre agents qui sont modélisés sous la forme des acteurs.

3.4.1.3 Structure du système

Le diagramme de classe UML défini dans la figure 3.6, présente la structure globale du notre intergiciel. Il décrit les différents éléments et leurs liens qui sont déduits, essentiellement, à partir du diagramme d'interactions. Les différents agents dans notre système héritent les caractéristiques et comportements de la classe abstraite « Agent ». Cette classe présente l'architecture interne et générique de chaque agent de notre système. Les éléments essentiels de cette architecture comme présentée dans la figure 3.7 sont :

- *État d'agent* : il mémorise son état actuel. Son état est défini par son sous-objectif actuel et l'état de l'exécution de son plan d'action.

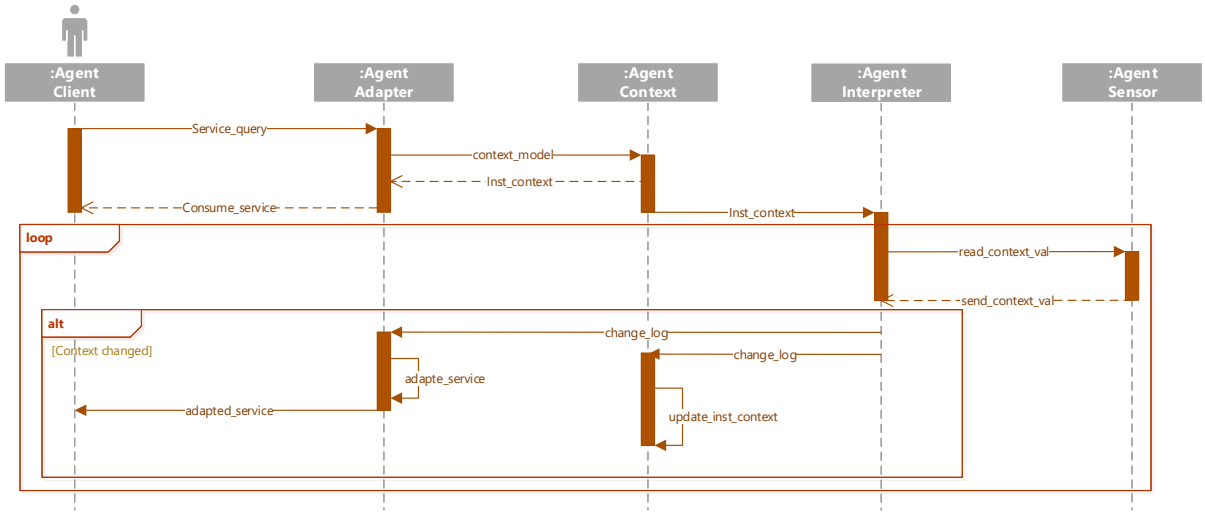


Figure 3.5 interaction entre agents

- *Mobilité* : Cette propriété définie si l'agent est mobile ou non.
- *Perception* : Le composant de perception permet à l'agent de visualiser son environnement.
- *Analyser* : ce composant permet d'analyser les informations de perception.
- *Raisonner* : ce composant permet de raisonner sur les informations obtenues du composant d'analyse. Par le raisonnement, l'agent va déduire les différentes actions (ou même plan) possibles qui correspond sa perception et son objectif actuels. Par la suite, l'agent sélectionne la meilleure action à exécuter parmi l'ensemble des plans obtenus, cette sélection est basée sur des mesures multicritères.
- *Action* : C'est la partie responsable de l'exécution de l'action sélectionnée dans la phase de raisonnement.

Dans notre architecture, chaque agent personnalise ces différents éléments de comportement et en ajoutant d'autres selon la mission de chaque agent comme indiqué dans le tableau 3.6.

3.5 Conclusion

Dans le cadre des applications sensibles au contexte, nous avons proposé une approche pour la gestion du contexte dans un environnement ubiquitaire caractérisé par la dynamique et la distribu-

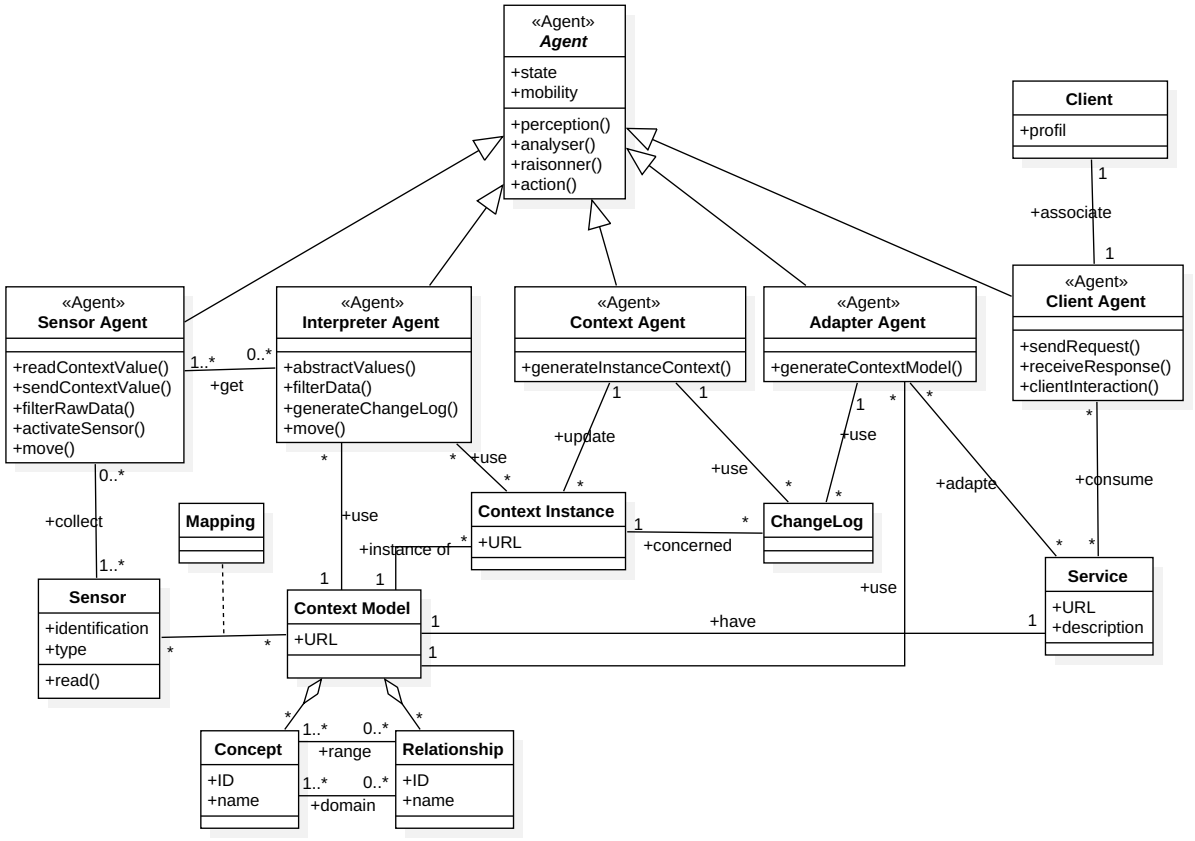


Figure 3.6 Diagramme de classes décrivant la structure des agents

tion. Notre proposition est basée sur la notion d'ontologie qui apporte une représentation formelle et sémantique. Chose qui a contribué à la résolution du problème de l'hétérogénéité dans ce genre d'environnement.

Aussi, nous avons proposé l'utilisation du journal de changements du contexte, qui garde les opérations de changements appliqués et minimise le coût de la communication.

Ces différents éléments sont manipulés par des agents. Chose qui a apporté plus d'efficacité à notre approche grâce à leur capacité de fonctionnement dans les systèmes. Nous avons décrit notre système multi-agents en présentant essentiellement leurs interactions et structures.

Concernant l'agent d'adaptation (Adapter Agent), nous avons proposé notre approche d'adaptation de la composition de service, que nous allons présenter dans le chapitre qui suit.

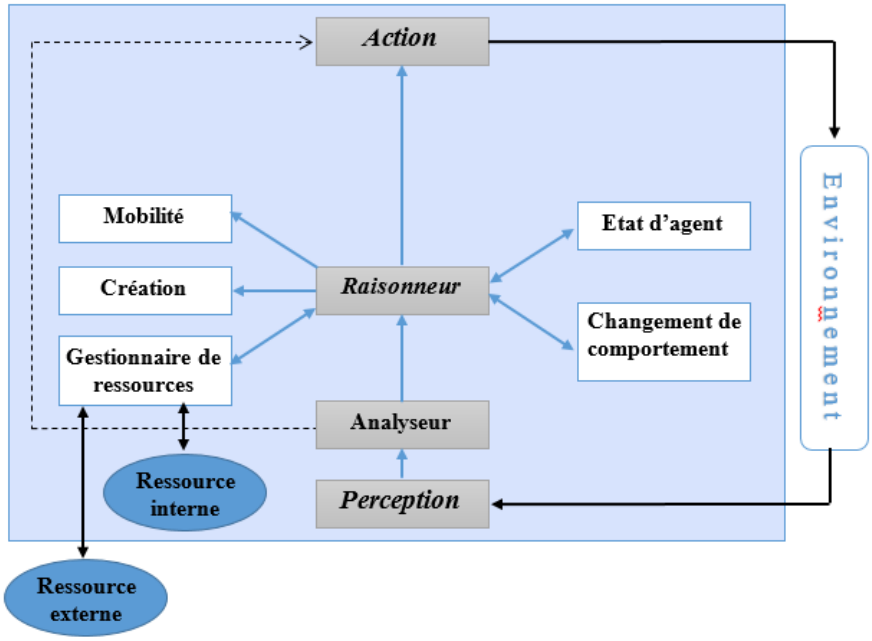


Figure 3.7 Architecture interne d'agent

Tableau 3.6 Détails des architectures internes des Agents du système

Élement-Agent	**Sensor Agent**	**Interpreter Agent**	**Context Agent**	**Adapter Agent**	**Client Agent**
Mobilité	Oui (en cas de besoin)	Oui (en cas de besoin)	Oui (en cas de besoin)	Non (coté service ou proxy)	Non (coté service)
Perception	readContextValue() lookupSensor()	interpretValue()	receiveChangeLog()	receiveChangeLog()	receiveClientReq() receiveResponse()
Analyse	filterRawData()	detectChange()	readChangeLog()	readChangeLog()	readClientReq() readResponse()
Raisonnement	s'il y a de nouvelles valeurs	S'il y a des changements significatifs.	En basant sur l'ontologie et le journal de changement, on déduit les mises à jour nécessaires pour l'instant du contexte.	En basant sur le journal de changements, il adapte le service consommé par le client.	La formulation de la requête et résultat.
Actions	sendContextValue() activateSensor() move()	generateChangeLog() move()	updateContext() move()	apdateService() generateContexteModel()	sentRequest() sentResponse()

CHAPITRE 4

APPROCHE BASÉE SUR UN GRAPHE DE PLANIFICATION POUR L'ADAPTATION AUTOMATIQUE DE LA COMPOSITION DE SERVICE WEB

4.1 Introduction

L'importance des services Web est primordiale pour l'interopérabilité entre les applications sur le Web, indépendamment de leurs plates-formes et de leurs langages d'implémentation. En outre, les services Web ont été développés en une technologie mature et sont basés sur des normes établies telles que SOAP[W3C (2007a)], WSDL[W3C (2007b)], UDDI[OASIS (2002)], WSLA[Keller & Ludwig (2003)] and BPEL[OASIS (2007)]. En raison de son importance, le concept de services Web a représenté, au cours de la dernière décennie, le noyau d'un domaine de recherche dynamique et actif, et a motivé plusieurs chercheurs.

Dans notre travail qui concerne l'adaptation des services Web, et comme nous avons déjà mentionné, nous nous concentrons sur la dimension de l'adaptation de la composition de service. De plus, nous avons expliqué comment la réalisation d'une nouvelle composition est plus favorisée que celle basée sur le changement de la composition actuelle (section 2.7.3). La nouvelle composition permet de donner une composition plus pertinente que celle obtenue par un changement d'une composition déjà calculée pour une autre situation différente.

Le problème de la composition de service Web (WSC) reste un des défis de la recherche : lorsqu'un seul service Web ne satisfait pas à certaines exigences, l'objectif est alors de combiner automatiquement le service Web d'autres services pour répondre pleinement à ces exigences [Kil (2010)].

Une telle exigence par le demandeur peut être fonctionnelle ou de qualité (QoS). Cela est connu sous le nom de composition de service sensible à la qualité (QoS-aware service composition).

Plusieurs critères de qualités de services peuvent être prises en considération comme le temps de réponse (response time), le débit (throughput), le coût (cost), la réputation (reputation), la fiabilité (reliability), taux d'éxecution réussie (successful execution rate) et la disponibilité (availability). Dans notre travail, nous avons choisi de chaque catégorie une qualité de services, à savoir le temps

de réponse comme un critère négatif de qualité et le débit comme un critère positif. Ce choix de deux critères de qualité de service, est motivé aussi par l'existence d'un ensemble de données (Data set) des services que la plupart de travaux voisins l'utilisent. Cet ensemble de données que nous allons utiliser pour l'évaluation de notre approche(section 5.3.2), utilise ces deux critères, à savoir le temps de réponse et le débit.

Par ailleurs, les techniques de planification 2.8.3 sont souvent considérées comme les plus efficaces et même comme prédominantes pour résoudre automatiquement des problèmes de compositions [Yan *et al.* (2012)].

Pour cette raison, la plupart des travaux existants couvrant le problème de la composition de service, tels que [Zheng & Yan (2008); Poizat & Yan (2010); Pistore *et al.* (2005)], se concentrent sur l'utilisation des techniques de planification ; d'autres travaux sont présentés dans [Peer (2005); han *et al.* (2007)].

Dans ces travaux, le graphe de planification est la technique de planification la plus utilisée. Le graphe de planification modélise l'espace de recherche du problème et se base sur l'idée de transformer un problème de planification en un problème d'exploration, dans lequel l'espace d'état est exploré à partir de l'état initial en cherchant le but. Ses notions sont très compatibles avec le problème de la composition du service ; ils incluent le concept d'action et de proposition et les concepts d'entrée/sortie. Le type de problème couvert par les approches citées consiste à déterminer le processus metier des services et à optimiser les QoS. Par d'autres termes, le problème considère qu'aucun modèle de processus métier prédéfini qui sera rempli par les services sélectionnés. Beaucoup des algorithmes de composition proposés cherchent à générer un service composé qui satisfait les contraintes fonctionnelles et possède des valeurs de QoS optimales.

Les auteurs de ces algorithmes utilisent la méthode habituelle de graphe de planification, c'est-à-dire qu'ils utilisent une recherche en arrière pour extraire la solution après la construction du graphe de planification. Cependant, cette façon d'extraction de la solution prend la grande partie du temps de l'exécution de ces algorithmes de composition.

De plus, la plupart des approches de la composition des SWa ont le problème des services redondants dans le résultat [Chen & Yan (2012)]. Sachant que l'élimination des services redondants n'entraîne pas de diminution de la valeur QoS de la solution ainsi que les exigences fonctionnelles restent toujours satisfaites. L'élimination des services redondants réduit le nombre de services de la solution, ce qui présente une motivation qui reste un défi.

Afin de minimiser le temps d'exécution de l'extraction de la solution et, par conséquent, l'ensemble du processus de la composition, nous proposons une heuristique qui nous permet de garder la trace pendant la construction du graphe de planification. En utilisant les informations enregistrées, nous pouvons obtenir plus rapidement le résultat qui satisfait les exigences fonctionnelles, et qui a la meilleure qualité QoS. En outre, nous proposons une amélioration de la structure du graphe de planification, ce qui réduit le nombre de ses nœuds et arcs. Cette réduction de la structure du graphe de planification a pour objectif de compenser le temps consacré à l'enregistrement des informations concernant la construction du graphe.
Afin de réduire encore le temps de composition, nous proposons également un algorithme qui minimise l'espace de recherche présenté par le graphe de planification amélioré. Grâce à la nouvelle structure et à sa minimisation, le nombre de services de la composition résultante a été réduit. Pour le même objectif, nous proposons également deux stratégies spécifiques de sélection de services et de solutions.

Ce chapitre est organisé comme suit : la section 4.2 contient notre formulation des différentes notions de base du domaine, telles que les concepts de service Web, la composition du service sensible à la qualité et la technique de planification pour ASC. La section 4.3, modélise notre approche de composition qui est basée sur notre graphe de planification amélioré et des autres concepts introduits d'une manière formelle. La section 4.4 présente nos algorithmes de base, proposés pour la mise en œuvre de notre approche. Notre algorithme de minimisation est présenté dans la section 4.5. La section 4.6 aborde notre proposition d'élimination des services redondants sous la forme des stratégies. En finissant par une conclusion qui résume le travail.

4.2 Notre formalisation du problème de la composition

Dans cette section, nous présentons nos définitions formelles des différents notions et concepts de base du domaine de la composition de service. Cette présentation formelle assure la cohérence dans la présentation des définitions, équations, algorithmes de notre contribution. Par conséquent, cette section aide à la compréhension de notre contribution.

4.2.1 Service Web et Qualité de Service(QoS)

Dans le domaine du Service-Oriented Computing (SOC) [Papazoglou (2003)], un service est considéré comme une ressource logicielle découverte à travers une description fournie. Cette description de service est disponible pour la recherche, la liaison et l'invocation par un consommateur de service. Les principaux acteurs impliqués dans les approches basées sur les services sont le registre des services, le fournisseur de services et le consommateur du service. Dans le contexte du service Web et de l'architecture SOA (architecture orientée service) [Papazoglou *et al.* (2008)], le protocole standard SOAP [W3C (2007a)] est utilisé pour la communication entre ces acteurs. Pour la découverte des services Web, c'est le registre des services qui contient les descriptions des différents services. Ces registres sont souvent basés sur le Standard UDDI [OASIS (2002)].

Pour la description des services Web, le standard WSDL (Web Services Description Language) [W3C (2007b)] est souvent utilisé, qui ne fournit qu'une description de la fonctionnalité offerte par le service. À cette fin, nous pouvons ajouter des informations sémantiques à la description des entrées et des sorties du service en utilisant OWL [W3C (2012)] comme utilisé dans de nombreux travaux, y compris [Chen & Yan (2012)]. Pour la description de la qualité des services, le langage WSLA [Keller & Ludwig (2003)] peut être utilisé ; c'est le cas dans les ensembles de données (data sets) utilisés dans la section d'évaluation (Section 5.3.2).

La plupart des définitions formelles des services Web proposées telles que celles qui apparaissent dans [Chen & Yan (2012, 2014); Yan *et al.* (2012); Kil & Nam (2011); Li *et al.* (2014); Yan *et al.* (2008)]considèrent le service comme une boîte noire sans état (pas de conversation) qui reçoit un message d'entré et génère un message de sorti. Dans le problème de l'ASC (Automatic Services

Composition), il est plus pratique de supposer que les services sont apatrides (stateless) plutôt que des processus étatiques (stateful, c'est-à-dire qu'ils ont un comportement interne). En effet, il n'est souvent pas possible de connaître le comportement interne d'un service.
Dans cette thèse et pour la simplicité, nous adoptons les définitions formelles suivantes :

Définition 1. *Concepts et paramètres :*
Soit un ensemble C de concepts d'une ontologie O *qui est utilisé pour décrire un ensemble de services* W*, et un ensemble* P *de paramètres de services typés.*
Chaque paramètre $\forall p \in P$ *qui peut être un paramètre d'entrée ou de sortie d'un service et a un type* c*, tel que c est un concept dans l'ontologie, c'est-à-dire* $type(p) \in C$*. Nous pouvons aussi dire que* p *est une instance de concept* c*. Tel que la fonction mathématique du type (il peut être indiqué comme application) est définie comme suit :*

$$type : P \longrightarrow C$$
$$p \longmapsto type(p) = c.$$

Définition 2. *entrée et sortie (input and output)*
Nous définissons également deux fonctions : input(entrée) et output(sortie), telles que :
$input : W \longrightarrow P$
$w \longmapsto input(w)$*. Par conséquent,* $input(w) \subseteq P$ *est un ensemble fini de paramètres d'entrée pour* w*. Un service Web peut être invoqué lorsque tous ses paramètres d'entrée,* input(w)*, sont disponibles. En outre, de la même manière pour la fonction de sortie :*
$output : S \longrightarrow P$
$w \longmapsto output(w)$*. Par conséquent,* $output(w) \subseteq P$ *est un ensemble fini de paramètres de sortie pour* w*. L'invocation du service génère l'* output(w)*.*

Dans notre travail, nous nous concentrons davantage sur le type de paramètres de service que sur les valeurs des paramètres ; Cela est motivé par la description des services et leurs comparaisons, qui nécessitent le type de paramètre, alors que les valeurs sont nécessaires pour l'exécution du service. Sur cette base, dans le reste de ce chapitre, nous nous référons à un paramètre (entrée ou sortie) à travers son **type**, qui est un concept dans l'ontologie utilisée.

En outre, si on considère la relation de la subsomption (c'est-à-dire une relation de type est-un ou sous-classe) entre les concepts, nous pouvons également accepter des sous-concepts pour les entrées d'un service. Par exemple, supposons qu'un service de voyage prend un paramètre *"Consumer identification"* et la classe *"Consumer identification"* a une sous-classe *"Passport ID"* indiquée dans l'ontologie OWL. Alors, prenant le paramètre *"Passport ID"* comme paramètre désigné pour *"Consumer identification"*, est considéré comme un cas valide.

Définition 3. *Service Web*
Un service Web w est un tuple $(input(w), output(w), Q(w))$ *où* Q(w) *est un ensemble fini de critères de qualité pour w*$(i.e., Q(w) = \{Q_i(w) | 1 \leq i \leq k\})$.

Comme la plupart des travaux en rapport avec la composition automatique des services (ASC), nous supposons que les services sont apatrides. Par conséquent, les services sont décrits par leurs paramètres d'entrées / sorties à travers des messages dans des descriptions WSDL, qui n'incluent pas de description comportementale spécifiant l'ordre dans lequel les opérations doivent être appelées. Par conséquent, pour une raison de simplicité, nous supposons que chaque service n'a qu'une seule opération ; si un service a *n* opérations, nous supposons que nous avons *n* services.

Pour prendre en considération l'état du contexte dans la définition de service Web, on étend le tuple avec deux autres paramètres à savoir, les Préconditions et Effets. C.-à-d. le tuple devient $(input(w), output(w), Preconditions(w), Effets(w), Q(w))$.

Les Préconditions signifient que le service ne peut pas être exécuté avec succès à moins que les conditions préalables soient vraies. Par ailleurs, l'exécution du service peut entraîner des changements dans le contexte (effets).
Par exemple, un service de vente peut exiger comme précondition une carte de crédit valide et comme entrée le numéro de carte de crédit et la date d'expiration. En tant que son exécution génère un reçu, et comme effet, la carte est débitée [David Martin (2004)].

En raison de simplicité, nous omettons l'intégration de la description de l'état du contexte (préconditions et effets) dans la description du service. Dans ce que suit, nous allons supposer que le

contexte est modélisé et qu'un service pour être invocable, il doit satisfaire la situation actuelle du contexte.

Exemple 1. *Prenant le service « CurrencyConverter » de l'exemple de WSC dans la section 2.5.1, qui contient l'opération «convert».*
Inputs :

- *Amount : concept = Money*
- *FromCurrency : Concept = Currency*
- *ToCurrency : Concept = Currency*
- *Rate : Concept = Exchange rate.*

Outputs :
ConvertedAmount : Concept = Money

Concernant la qualité de service (QoS), comme nous l'avons mentionné, nous nous concentrons uniquement sur les critères de qualité les plus couramment utilisés, à savoir le temps de réponse et le débit :

- *Response time* $Q_{RT}(w)$: l'intervalle de temps entre le moment où une requête est envoyée et le moment où les résultats sont reçus [Zeng *et al.* (2003)].
- *Throughput* $Q_{TH}(w)$: est le nombre d'invocations réussies sur une période spécifiée [Kona *et al.* (2009)].

Ce sont des modèles de deux types de critères QoS (positifs et négatifs). La qualité du temps de réponse est un critère négatif, c'est-à-dire que, plus la valeur est élevée, plus la qualité est faible. La qualité de débit est un critère positif, plus la valeur est élevée, plus la qualité est élevée [Yan *et al.* (2012)]. Par conséquent, plusieurs travaux, comme as [Yan *et al.* (2012); Zeng *et al.* (2003); Bouguettaya *et al.* (2011)], cherchent à mettre à l'échelle les valeurs de qualité pour être utilisées uniformément, en utilisant une fonction d'utilité croissante ou même décroissante. Dans notre travail, nous utilisons également la fonction utilitaire pour la mise à l'échelle d'une valeur de qualité.
Afin de simplifier les exemples utilisés dans ce travail, nous utilisons la qualité du temps de réponse

au lieu d'utiliser la qualité de débit. Pour cela, nous choisissons une fonction d'utilité décroissante pour sa commodité à la qualité utilisée dans les exemples, à savoir la qualité du temps de réponse (critère négatif).

En conséquence, pour la mise à l'échelle des valeurs de qualité du service unique, nous utilisons les équations suivantes [Yan *et al.* (2012)] :

- For response time quality (RT) :

$$U_{RT}(w_i) = \left\{ \begin{array}{cc} \frac{Q_{RT}(w_i) - \min Q_{RT}}{\max Q_{RT} - \min Q_{RT}} & \text{if } \max Q_{RT} - \min Q_{RT} \neq 0 \\ 1 & \text{if } \max Q_{RT} - \min Q_{RT} = 0 \end{array} \right\} \tag{4.1}$$

- For throughput quality (TH) :

$$U_{TH}(w_i) = \left\{ \begin{array}{cc} \frac{\max Q_{TH} - Q_{TH}(w_i)}{\max Q_{TH} - \min Q_{TH}} & \text{if } \max Q_{TH} - \min Q_{TH} \neq 0 \\ 1 & \text{if } \max Q_{TH} - \min Q_{TH} = 0 \end{array} \right\} \tag{4.2}$$

Avant de commencer par la notion de composition, nous devons expliquer la requête de service du demandeur (qui peut être une application ou un utilisateur). Une requête *r* peut être vue comme un modèle du service demandé.

Définition 4. *Requête*

Une requête r *est un tuple* (input(r), output(r), Q(r)) *avec les composants suivants :*

- Input(r) *(resp.* output(r)*) : comme pour la définition du service, il s'agit d'un ensemble fini de paramètres d'entrée (resp. sortie) pour* r*. Nous sommes intéressés par ses types pour découvrir un service correspondant, c.-à-d. un service* $w \in W$ *que ses entrées (resp. les sorties de* r*) sont incluses dans les entrées de* r*(resp. les sorties* w*). Les valeurs des paramètres d'entrée de* r *sont utilisées pour l'invocation du service* w*.*
- Q(r) *: est un ensemble fini de critères de qualité que doit être prise en compte lors de l'optimisation de la qualité. Cela signifie que nous retournons une solution qui satisfait simultanément aux exigences fonctionnelles et qui possède la qualité la plus élevée. S'il y a une valeur de* Q(r) *qui présente un seuil de qualité que doit être garanti par un service trouvé à retourner au demandeur.*

Nous pouvons formaliser ces notions comme suit :

a. Pour les exigences fonctionnelles :

$$functional\_solution(r) = \{w \in W | input(w) \subseteq input(r) \text{ et } output(r) \subseteq output(w)\} \quad (4.3)$$

b. De plus, pour la qualité

$$optimalQ\_Solution(r) = \{w \in functional\_solution(r) \mid \forall w' \in functional\_solution(r) \wedge w' \neq w \rightarrow U_Q(w) \leq U_Q(w')\} \quad (4.4)$$

C'est-à-dire w est la solution optimale des solutions acceptables dans $functional\_solution(r)$. Si les valeurs $Q(r)$ sont définies, nous ajoutons la condition suivante :

c. La solution optimale w doit vérifier la condition $U_Q(w) \leq U_Q(r)$ (nous utilisons la fonction d'utilité décroissante), C'est-à-dire que la qualité du service retournée doit être supérieure au seuil spécifié dans la requête r.

Dans la définition précédente de la requête, nous avons expliqué que pour répondre à la requête, nous devons découvrir le service le plus approprié. Cependant, si aucun service ne peut répondre à la requête, nous avons besoin de la composition de service Web présentée dans la section suivante.

4.2.2 Composition de service Web

Si nous ne pouvons pas trouver un service qui correspond aux exigences de la requête, pour des exigences fonctionnelles ou de qualité, nous devons essayer de trouver une composition de service existant. Dans ce qui suit, nous présentons la définition formelle de la composition de service Web.

Définition 5. *Problème de composition de service*
Un problème de composition de service Web peut être présenté par un couple (W, r) *où :*

- W *est un ensemble fini de services.*
- r *est une requête, c'est-à-dire* $r = (input(r), output(r), sortie(r), Q(r))$.

Exemple 2. *Supposant qu'on cherche un service de type «CurrencyConverter». Donc, sa requête soit* $r = (input = \{Amount, FromCurrency, ToCurrency\}, output = \{ConvertedAmount, Rate\}, \{Q\})$.
Le service qu'on a requiert quatre (04) entrées, à savoir Montant, FromCurrency, ToCurrency et

Rate. Puisque l'utilisateur ne connait pas le taux du jour, on a besoin, alors, d'un autre service dédié à cela. Le résultat de cette requête qui soit une composition de service est expliqué dans l'exemple de la section 2.5.1.

Dans [Cui *et al.* (2011); Yan *et al.* (2012)], des formules ont été proposées pour calculer la qualité du réseau de services Web comme suit :

- *Response Time (RT)* :

$$Q_{RT}(w_1;...;w_n) = \sum_{i=1}^{n} Q_{RT}(w_i)$$
$$Q_{RT}(w_1||...||w_n) = \max_{i=1}^{n} Q_{RT}(w_i) \tag{4.5}$$

- *Throughput (TH)* :

$$Q_{TH}(w_1;...;w_n) = \min_{i=1,n} Q_{TH}(w_i)$$
$$Q_{TH}(w_1||...||w_n) = \min_{i=1,n} Q_{TH}(w_i) \tag{4.6}$$

Tel que ";" et "||" signifient que les services sont en séquence et en parallèle, respectivement. Les équations pour la fonction d'utilité décroissante sont les mêmes que ces équations (en changeant, seulement Q à U). Sauf dans Équation 4.6, *max* remplace *min*, car U_{TH} augmente lorsque Q_{TH} diminue.
Cependant, un service Web composé est principalement constitué de services parallèles et séquentiels. Dans la section décrivant notre approche (Section 4.3), nous présentons notre méthode de traitement et de mise à l'échelle de la qualité d'une composition de service Web.

4.2.3 Technique de planification pour ASC

Les techniques de planification en intelligence artificielle IA [Ghallab *et al.* (2004)] sont souvent considérées comme les plus efficaces pour résoudre le problème ASC [Pistore *et al.* (2005); Poizat & Yan (2010); Zheng & Yan (2008)]. La planification en IA est appliquée avec succès en représentant le problème de WSC en tant que problème de planification [Yan *et al.* (2010)].

Tout d'abord, nous présentons la définition formelle d'un problème de planification global, qui est modifié de [Yan *et al.* (2010); Russell (2010)].

Définition 6. *Un problème de planification est un tuple* (P,A,s_0,g) *où :*

- P *est un ensemble fini de propositions.*
- A *est un ensemble fini d'actions, qui définit le domaine, avec une action* a *étant un triplet* (PRECOND, effects-, effects +) *où* PRECOND(a) *désigne les conditions préalables de l'action* a *et* $effects^-(a)$ *and* $effects^+(a)$ *(*$effects^-(a) \cap effects^+(a) = \emptyset$*), désignent les effets négatifs et positifs de l'action* a *respectivement.*
- $s_0 \subseteq P$ *est l'état initial.*
- $g \subseteq P$ *est un ensemble de propositions appelées propositions du but.*

Notez que bien que P puisse être obtenu implicitement à partir de l'ensemble A, nous l'avons ajouté pour clarifier la définition.

L'algorithme de planification repose sur l'idée de transformer un problème de planification en un problème d'exploration, c'est-à-dire en explorant l'espace d'état à partir de l'état initial en recherchant le but. Sachant que, les quatre éléments nécessaires pour définir le problème d'exploration sont : l'état initial, les actions disponibles dans un état, le résultat de la mise en œuvre d'une action et le test du but [Russell (2010)].

L'ensemble d'actions A définit implicitement les fonctions $ACTIONS(s)$ et $RESULT(s,a)$ nécessaires à l'exploration de l'espace d'états. La fonction $ACTIONS(s)$ fournit les actions applicables dans un état s et est définie de la manière suivante :

$$(a \in ACTIONS(s) \leftrightarrow s \models PRECOND(a)) \tag{4.7}$$

C'est-à-dire pour être exécutable, s doit contenir toutes les propositions positives $PRECOND(a)$ et doit ne contenir aucune des propositions négatives.

La fonction $RESULT(s,a)$ qui exprime le résultat de l'application d'une action dans un état s est :
$RESULT(s,a) = (s - effects^-(a)) \cup effects^+(a)$

L'exécution de toutes les actions applicables $(RESULT(s,ACTIONS(s)))$ est définie par un nouvel état *s'*, et pour cette raison, la fonction RESULT peut également être appelée une fonction de

transition comme dans [Yan *et al.* (2010)].

Le problème ASC a la particularité de ne pas avoir des effets négatifs des actions, c'est-à-dire qu'il n'y a que des effets positifs [Yan *et al.* (2012)]. Cela minimise la complexité des algorithmes de planification. Il en est de même pour la condition préalable, qui ne contient aucune proposition négative. Ainsi, le problème de planification ASC est formellement défini comme suit :

Définition 7. *problème de planification ASC.*

Le problème de planification ASC est une instanciation du problème de planification :

- *Un ensemble de paramètres typés* P *peut également être utilisé comme un ensemble de propositions. Parce qu'une proposition* p *est un nom de concept* c *dans l'ontologie utilisée,* $P = \cup_{i=1}^{|W|} (input(w_i) \cup output(w_i))$. *Telle que* W *est un ensemble de services, et comme nous l'avons mentionné, nous sommes intéressés par le type de paramètre ;*
- $A = W$, *où* W *est l'ensemble de services (voir définition 5). Ainsi, une action* a *est un service* w *avec* $PRECOND(w) = input(w)$ *(contient uniquement des propositions positives), et nous n'avons que des effets positifs,* $effects^{+}(w) = output(w)$. *Pour les* $ACTIONS(s)$: $(w \in ACTIONS(s) \leftrightarrow PRECOND(w) \subseteq s)$, *c'est-à-dire* $(w \in ACTIONS(s) \leftrightarrow Input(w) \subseteq s)$.
  Pour la fonction de transition $RESULT(s,w)$: $RESULT(s,w) = s \cup effects(w)$ *c.-à-d.* $RESULT(s,w) = s \cup output(w)$;
- $s0 = input(r)$, *tel que r est la requête du demandeur de l'utilisateur ;*
- *et* $g = output(r)$.

Le graphe de planification est une structure de données spécifique qui définit un espace d'état. En l'explorant, nous pouvons identifier l'existence de la solution, et même l'extraction de cette dernière.

Voici sa définition formelle spécifique pour l'ASC, comme mentionné dans [Yan & Chen (2015)] :

Définition 8. *Le graphe de planification (classic Planning Graph)*

Le graphe de planification $PG = (V,E)$ *est un graphe orienté acyclique (GOA). Il a deux types de sommets* $V = V_A \cup V_P$ *où* V_A *sont les sommets représentant les actions et* V_P *sont les sommets représentant les propositions. Les arcs* $E = (V_P \times V_A) \cup (V_A \times V_P)$ *connectent les sommets. Les arcs*

$(V_P \times V_A)$ connectent les paramètres d'entrée avec les actions, tandis que $(V_A \times V_P)$ connectent les actions avec leurs paramètres de sortie.

Parce que chaque état contient un ensemble de propositions qui sont un ensemble de paramètres de services, nous désignons l'état *i* par P_i.

Ainsi, le graphe de planification *PG* est organisé en couches : $P_0, A_1, P_1, ...P_i, A_{i+1}, P_{i+1}, ...A_n, P_n$, où : P_i : l'état P_i contient toutes les propositions vérifiées pour la couche i.

A_{i+1} : toutes les actions ayant les préconditions vérifiées dans P_i, c'est-à-dire $A_{i+1} = ACTION(P_i)$.

P_{i+1} : en plus des propositions de l'état P_i, il contient les propositions générées par A_{i+1}, c'est-à-dire $P_{i+1} = P_i \cup RESULT(P_i, A_{i+1})$. La construction du graphe s'arrête lorsque le but est atteint, c'est-à-dire $g \subseteq P_n$ ou un point fixe est atteint ($P_i = P_{i+1}$ and $A_i = A_{i+1}$).

Pour illustrer le graphe de planification et ses améliorations, nous proposons l'exemple abstrait suivant.

Exemple 3. *Le tableau 4.1 montre un ensemble de services disponibles avec leurs paramètres d'entrée et de sortie et les valeurs de qualité de réponse de temps. La requête utilisateur* r *est (input(r),output(r), Q (r)) = ({A, B, C }, {G, K }, {RT}). Le graphe de planification construit est présenté dans la figure 4.1.*

L'action no-op (non-opération) permet d'hériter des propositions d'une couche de proposition précédente sans encourir aucun coût. Pour rendre la figure plus illustrative, nous présentons les actions non-op uniquement dans la première couche ; de plus, nous ne dessinons pas les arcs d'entrée/sortie qui relient les actions répétées dans les couches précédentes (les actions ombrées de la figure 4.1). Dans l'état des propositions de la couche 2, le but défini {G, K } est inclus, ce qui indique l'existence d'une solution, par exemple, $(w1, w5||w6)$, $w1$ suivi de $w5$ et $w6$ en parallèle.

Tableau 4.1 Un ensemble de services

Service	Input	Output	Q_{RT}	Service	Input	Output	Q_{RT}
w1	A	E,F	120	w5	F	K,M	40
w2	A, B	E	90	w6	A,E	G	80
w3	C	J,D	50	w7	J,D	E,K	30
w4	C	L	40	w8	L	M	60

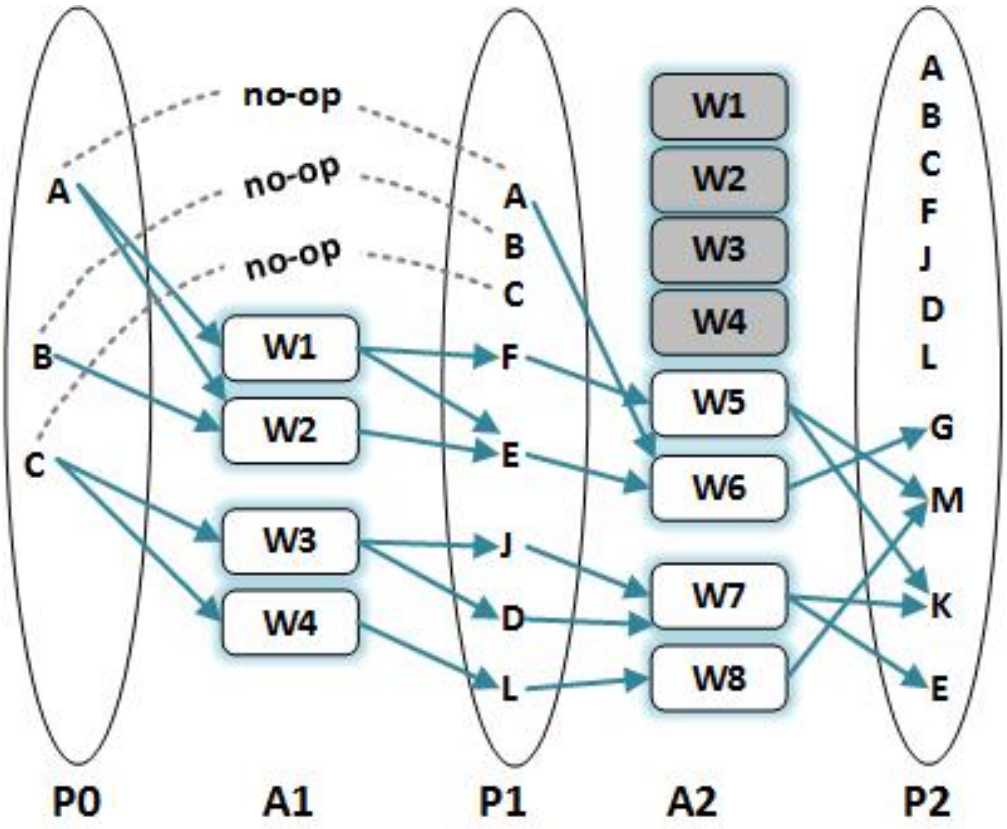


Figure 4.1 Exemple de graphe de planification

En utilisant l'algorithme GraphPlan [Russell (2010)], une solution peut être trouvée dans l'espace défini par le graphe de planification. Cet algorithme peut également être utilisé pour construire le graphe de planification, avant la phase d'extraction de la solution.

Bien que la complexité de l'algorithme de construction du graphe soit polynomiale [Blum & Furst (1997)], l'extraction de la solution est plus coûteuse dans la plupart des cas ; en effet, sa complexité peut même être NP-complete [Yan *et al.* (2012)], en particulier si le problème a des effets négatifs [Yan & Chen (2015)]. Dans la section suivante, nous présentons l'heuristique utilisée par notre approche de composition automatique de services et qui est sensible à la qualité de service (ASC QoS-Aware) pour minimiser le temps d'exécution de l'extraction de la solution en tenant compte du nombre de services de la solution.

4.3 Modélisation formelle de notre approche

Dans cette section, nous présentons les détails de notre approche sensible à la qualité de service et qui est basée sur la technique de planification (QoS-aware planning-based approach). Cette approche permet d'automatiser la composition de service Web. Plus précisément, nous décrivons formellement notre algorithme GraphPlan qui permet la construction d'un graphe de planification amélioré et montre comment une solution peut être atteinte en utilisant le graphe obtenu.

Afin de réduire le temps d'exécution du processus d'extraction de la solution, nous proposons d'exploiter le processus de la construction de graphe de planification. À cette fin, nous proposons de garder la trace des propositions générées lors de la construction du graphe de planification. Pour chaque proposition, nous enregistrons le meilleur chemin pour générer cette proposition dans le graphe de planification.

Pour chaque proposition générée, nous enregistrons son meilleur chemin (Path) dans le graphe de planification, ainsi que sa meilleure qualité (QoS) obtenue par ce chemin.
Par conséquent, en utilisant les informations enregistrées, l'extraction de la solution devient moins complexe et, par conséquent, nécessite un temps d'exécution plus court. La composition automatique des services (ASC) a donc besoin de moins de temps d'exécution.

Pour réaliser l'idée ci-dessus, nous commençons par introduire deux concepts qui sont : Chemin (Path) et Niveau (Level)pour la description de la composition du service. La proposition de ces deux concepts est motivée par la technique utilisée pour la composition du service, c'est-à-dire le graphe de planification (Section 4.2.3). Il est à noter que ces deux concepts peuvent rester abstraits pour toute autre technique de représentation utilisée.

Définition 9. *Path (Chemin)*
Un chemin Pth *représente une composition de service.* $Pth = (L1, L2, ..., Li, ...Ln)$ *est un tuple de niveaux et* n *est le nombre de niveaux, c'est à dire* Pth *est une liste finie et ordonnée de niveaux,*

la manière dont une liste est ordonnée permet de spécifier la séquence d'exécution des niveaux qui composent cette liste.

Définition 10. *Level (Niveau)*
Un niveau $L = \{w_i, i = 1, m\}$ *est un ensemble fini de services qui peuvent être exécutés en parallèle.*

Exemple 4. *Soit* $L = \{w1, w2\}$*, c.-à-d. un niveau contenant deux services* $w1$ *et* $w2$ *qui peuvent s'exécuter en parallèle. Nous pouvons noter cela comme suit :* $L = (w1||w2)$.

Exemple 5. *Le chemin* $Pth = ((w_{1a}||w_{1b}); w_2; (w_{3a}||w_{3b}||w_{3c}); w_4)$ *contient 4 niveaux :*

- $L_1 = (w_{1a}||w_{1b})$.
- $L_2 = (w_2)$ *ou simplement* w_2.
- $L_3 = (w_{3a}||w_{3b}||w_{3c})$.
- $L_4 = w_4$.

En plus de la définition d'un niveau (Level) et d'un chemin (Path), nous introduisons d'autres concepts, qui sont des opérations (dans un sens mathématique) dans l'ensemble des chemins *Path*.

Définition 11. *Opération de concaténation (Concatenation operation)*
Nous choisissons l'opérateur "+" pour désigner l'opération de concaténation, où ses opérandes sont dans l'ensemble de chemins Path*. Ainsi, nous définissons l'opération de concaténation comme suit :*

$+ : Path \times Path \longrightarrow Path$

$pth_1 = (L_{11}, \ldots, L_{1n}) + pth_2 = (L_{21}, \ldots, L_{2m}) \longmapsto p_{th} = (L_{11}, \ldots, L_{1n}, L_{21}, \ldots, L_{2m})$

Tel que L_{ij} *désigne le niveau* j *du chemin* i.

Définition 12. *Opération de multiplication (Multiplication operation)*
Soit l'ensemble des chemins Path*, nous définissons l'opération de multiplication comme suit :*

$\times : Path \times Path \longrightarrow Path$

$pth_1 = (L_{11}, \ldots, L_{1n}) \times pth_2 = (L_{21}, \ldots, L_{2m}) \longmapsto p_{th} = (L_1, L_2, \ldots, L_t)$ *tel que,* $t = \max(n, m)$
$= \sum_{i=1}^{t=max(n,m)} L_{1i} \cup L_{2i}$ *tel que,* $L_{ij} = \emptyset$*, if* $j > |pth_i|$

Où L_{ij} désigne le niveau j du chemin i, L_j signifie le niveau j du chemin obtenu, et $|pth_i|$ est la cardinalité de pth_i, c'est-à-dire le nombre de ses niveaux. Il convient de rappeler qu'un niveau est un ensemble de services qui peuvent s'exécuter en parallèle.
Ainsi, la multiplication des chemins est la concaténation de l'union dans l'ordre de leurs niveaux.

La définition suivante généralise la définition précédente pour les chemins multiples :

Définition 13. *Multiplication des chemins*
$\times : Path \times Path \times \ldots \times Path(n\ times) \longrightarrow Path$
$pth_1 \times pth_2 \times \ldots \times pth_n \longmapsto pth = \prod_{i=1}^{n} pth_i = (L_1, L_2, \ldots, L_t)$ *tel que,* $t = max_{i=1}^{n}(|pth_i|)$
C'est à dire $pth = \sum_{j=1}^{t} \bigcup_{i=1}^{n} L_{ij}$ *tel que,* $L_{ij} = \emptyset$, *si* $j > |pth_i|$

Ici, nous présentons la qualité globale du chemin, qui présente une composition de service Web. Comme nous l'avons dit précédemment, nous nous concentrons uniquement sur les qualités de services(QoS) les plus utilisées, à savoir le temps de réponse et le débit. Ces derniers représentent les deux types de critères de qualité, à savoir le critère négatif et le critère positif.

Dans la section préliminaire (Section 2.9), nous présentons des équations qui couvrent la qualité d'un réseau de services Web. Sur la base de ces équations, nous pouvons déduire intuitivement la qualité du chemin et du niveau comme suit :

- *Response Time (RT)* : soit Pth un chemin,

$$Q_{RT}(Pth) = \sum_{i=1}^{n} Q_{RT}(L_i)$$
$$Q_{RT}(L_i) = \max_{k=1,m} Q_{RT}(w_k) \tag{4.8}$$

- *Throughput (TH)* : soit Pth un chemin,

$$Q_{TH}(Pth) = \min_{i=1,n} Q_{TH}(L_i)$$
$$Q_{TH}(L_i) = \min_{k=1,m} Q_{TH}(w_k) \tag{4.9}$$

  Tel que n est le nombre de niveaux, m est le nombre de services du niveau L_i, et w_k est k^{eme} service du niveau L_i.

Comme nous l'avons mentionné dans la section 2.6, la qualité du temps de réponse est un critère négatif, alors que la qualité du débit est un critère positif. Afin d'utiliser notre approche uniformément pour chaque critère de qualité, nous définissons la fonction d'utilité décroissante suivante qui est basée sur les équations 4.1 et 4.2

.Pour la qualité du temps de réponse U_{RT} (RT) : soit un chemin Pth,

$$U_{RT}(Pth) = \sum_{i=1}^{n} U_{RT}(L_i)$$
$$U_{RT}(L_i) = \max_{k=1,m} U_{RT}(w_k) \tag{4.10}$$

Pour le critère du débit (TH) : soit un chemin Pth,

$$U_{TH}(Pth) = \max_{i=1,n} U_{TH}(L_i)$$
$$U_{TH}(L_i) = \max_{k=1,m} U_{TH}(w_k) \tag{4.11}$$

Tel que n est le nombre de niveaux, m est le nombre de services du niveau L_i, et w_k est le k^{eme} service dans le niveau L_i.

Ces différentes définitions concernant la notion de chemin nous permettent de structurer et d'organiser notre approche. Cependant, la qualité du chemin présentée dans l'équation 4.8 et par conséquent l'équation 4.10 (contrairement aux équations 4.9 et 4.11) n'est pas toujours correcte. Cela est présenté à l'aide de l'exemple de la figure 4.2.

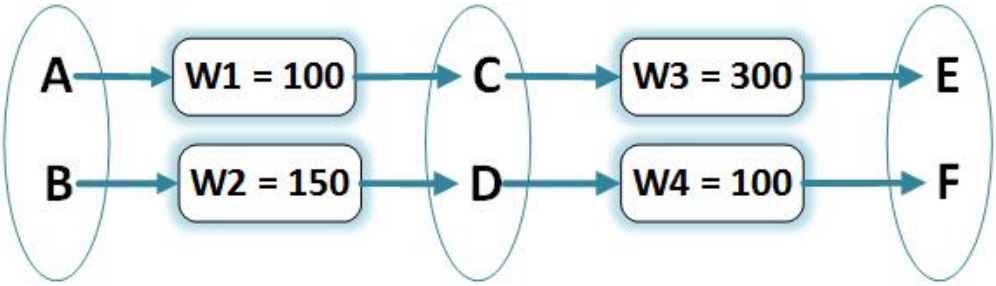


Figure 4.2 Exemple d'évaluation de la qualité d'un chemin

En appliquant l'équation 4.8, nous assignons la valeur 450ms au chemin $(w1||w2;w3||w4)$ présentée dans l'exemple. Cependant, le service w3 peut être exécuté directement après l'obtention de la proposition C sans attendre que l'exécution du service w2 soit terminée. Ainsi, la qualité du temps de réponse est 400ms au lieu de 450ms.

Dans notre approche, pour résoudre ce problème, nous avons proposé que notre méthode de calcul de la valeur de qualité soit basée sur la valeur de qualité unique de chaque proposition. Pour cette raison, nous proposons d'enregistrer, en plus du meilleur chemin, la meilleure qualité de chaque proposition générée.

Pour supporter la trace de la construction du graphe, nous améliorons le *graphe de planification classique* (définition 8) à ce que nous avons nommé un graphe de planification de suivi (*Tracking Planning Graph (TPG)*). *TPG* associe à chaque proposition (nœud, sommet) le meilleur chemin ($bestPath$) qui génère la proposition et sa valeur de meilleure qualité ($bestQ$) et associe également à chaque action (service) sa valeur d'utilité de qualité U_Q

Définition 14. *Graphe de planification de suivi (Tracking Planning Graph).*
Graphe de planification de suivi (TPG) est un graphe de planification $G = (V_{Au} \cup V_{Pth}, E)$ *et chaque sommet* $au = (a,u) \in V_{Au}$, *tel que* a *est une action,* $a \in A$ *et* $u = U_Q(a)$, *et* u *est la valeur d'utilité de qualité de l'action a. De plus, chaque sommet* $vp \in V_{Pth}, vp = (p, bestPath, bestQ)$, *tel que p est une proposition, bestPath est le meilleur chemin qui peut générer la proposition* p, *et bestQ est la meilleure valeur de qualité qui peut être obtenue en générant la proposition* p.

Avant de présenter comment calculer la valeur de la meilleure qualité et le meilleur chemin, nous expliquons les notations suivantes qui sont utilisées dans les équations du calcul :

- $bestQ(p)$: obtient la meilleure qualité enregistrée, $bestQ$,de la proposition p.
- $bestPath(p)$: obtenient le meilleur chemin enregistré, $bestPath$, de la proposition p.
- $U_Q(p)$: calcule la nouvelle qualité de la proposition p qui peut être obtenue à partir des actions générant la proposition p dans la couche courante du graphe de planification.
- $Chemin(p)$: calcule le nouveau chemin de la proposition p qui est obtenue à partir des actions générant la proposition p dans la couche courante du graphe de planification.

Il convient de noter que nous mettons à jour la valeur $bestQ$ de la proposition p en choisissant la valeur de la meilleure qualité entre l'ancienne enregistrée ($bestQ$) et celle nouvellement calculée ($U_Q(p)$). Dans la première génération de p, nous enregistrons la valeur de $U_Q(p)$ directement.
Ainsi, pour enregistrer la meilleure valeur de qualité $bestQ$ d'une proposition p, nous nous basons sur la fonction d'utilité décroissante ($U_Q(p)$) qui est calculée comme suit :

- S'il n'y a qu'une seule action a générant p, pour les deux critères de qualité RT et TH :

$$U_Q(p) = U_{Qa}(p) = \left\{ \begin{array}{ll} RT: & \left(\max_{j=1}^{|Input(a)|} bestQ\left(Input_j(a)\right) + U_Q(a)\right) \\ TH: & \max\left(U_Q(a), \max_{j=1}^{|Input(a)|} bestQ\left(Input_j(a)\right)\right) \end{array} \right\} \tag{4.12}$$

  C'est-à-dire s'il s'agit du temps de réponse (RT), la valeur $bestQ$ de p est égale à l'addition de la qualité de service généré et le maximum des valeurs $bestQ$ des entrées du service qui sont déjà enregistrées. Le maximum de ces valeurs est la meilleure valeur de qualité $bestQ$ pour la qualité TH.

- S'il existe plus d'une action pour générer p, c'est-à-dire les actions qui peuvent générer p sont gA où $|gA| > 1$, nous avons l'équation suivante :

$$U_Q(p) = U_{QgA}(p) = min_{i=1}^{|gA|} U_{Qai}(p) \tag{4.13}$$

C'est-à-dire $U_Q(p)$ est la valeur de la meilleure qualité qui peut être obtenue à partir des actions qui peuvent générer p dans la couche courante du graphe de planification. $U_{Qai}(p)$ calcule la valeur de la qualité obtenue par l'action ai, tel que présenté dans l'équation 4.12.

Le calcul de la meilleure valeur de la qualité $bestQ$ de la proposition p sert à mettre à jour son meilleur chemin qui a permet de la générer $bestPath$. Si le $bestQ$ est mis à jour, nous mettons à jour le $bestPath$ par le nouveau chemin obtenu par $Path(p)$, qui est calculé comme suit :

- S'il n'y a qu'une seule action a qui génère p :

$$Path(p) = Path_a(p) = \left(\prod_{j=1}^{|Input(a)|} bestPath\left(Input_j(a)\right) \right) + (a) \tag{4.14}$$

  C'est-à-dire un chemin de p est la multiplication (définition 13)des meilleurs chemins des propositions d'entrée (inputs) de l'action a qui génère p, et nous ajoutons (concaténer) le chemin résultant avec un chemin contenant uniquement l'action a.

- S'il existe plus d'une action qui génère p, c'est-à-dire les actions qui peuvent générer p sont gA ou $|gA| > 1$, nous avons l'équation suivante :

$$Path(p) = Path_{gA}(p) = Path_b(p) \in \bigcup_{i=1}^{|gA(p)|} Path_{ai}(p),$$

  Tel que :

$$\forall a_i \in gA, U_{Qb}(p) \leq U_{ai}(p) \tag{4.15}$$

  C'est-à-dire à partir d'un ensemble d'actions génératrices, nous sélectionnons l'action b qui peut générer p avec la meilleure valeur de la qualité de service.

  Le $bestPath_{ai}(p)$ est calculé par l'équation 4.14.

Pour simplifier la comparaison entre les chemins, nous utilisons la fonction suivante :
$Best(path_1, path_2, \ldots, path_n)$, sans citer U_Q.

Exemple 6. *Dans l'exemple précédent présenté dans la figure 4.1, le chemin de la proposition E dans la couche 1 est* $Path(E) = Best((bestPath(A) + (w1)), (bestPath(A) \times bestPath(B)) + (w2) = Best(w1 = 120, w2 = 90) = (w2)$. *Alors, nous enregistrons ce chemin dans le bestPath de la proposition E.*
Dans la couche 2, le bestPath de E sera le meilleur chemin entre l'ancien chemin enregistré $(w2)$ *et qui a été obtenu par l'action* $w7, Path_{w7}(E)$, *tel que* $Path_{w7}(E) = (bestPath(J) \times bestPath(D)) + (w7) = (w3||w3) + (w7) = (w3, w7)$. *Ainsi, le nouveau* $bestPath(E) = Best((bestPath(E), Path_{w7}(E)) = Best((w2) = 90, ((w3, w7) = 80)) = (w3, w7)$. *Pour simplifier davantage, nous avons utilisé directement les valeurs de la qualité RT plutôt que les valeurs mises à l'échelle (the scaled values).*

Généralement, l'ensemble du but contient plus d'une proposition. Pour cela, nous devons aussi savoir comment calculer U_Q et $Path$ d'un ensemble de propositions. Soit SP un ensemble de propositions, nous proposons les équations suivantes :

- $U_Q(SP)$: la meilleure qualité de toutes les propositions SP générées est calculée de la manière suivante, et cela pour les deux qualités de service RT et TH :

$$U_Q(SP) = \max_{i=1}^{|SP|} bestQ(p_i) \tag{4.16}$$

  Nous choisissons le maximum des valeurs enregistrées des propositions, car nous utilisons une fonction d'utilité décroissante.

- $Path(SP)$: pour toutes les propositions SP générées, le chemin se calcule de la manière suivante, et cela pour les deux qualités de service RT et TH :

$$Path(SP) = \prod_{i=1}^{|SP|} bestPath(p_i) \tag{4.17}$$

  C'est-à-dire la multiplication de tous les meilleurs chemins enregistrés de l'ensemble des propositions. La multiplication se fait en se basant sur l'opération de la multiplication (définition 13).

Comme nous l'avons vu précédemment, nous avons ajouté un mécanisme de suivi au TPG (Tracking Planning Graph), qui permet d'enregistrer la meilleure valeur pour chaque proposition. Pour

compenser le temps passé à calculer et enregistrer les informations de suivi, nous avons proposé de minimiser le Tracking Planning Graph (TPG) en utilisant les informations enregistrées. Nous avons appelé cette nouvelle structure le graphe de planification de suivi amélioré (Improved Tracking Planning Graph (ITPG)).

L'amélioration apportée à l'ITPG a impliqué la minimisation du nombre d'actions dans le graphe. Nous ajoutons seulement des actions bénéfiques, c'est-à-dire celles ayant au moins une proposition nouvellement générée ou améliorée dans leurs préconditions (inputs). Une proposition améliorée est, en fait, une ancienne proposition (c'est-à-dire existant dans un état antérieur) avec une nouvelle meilleure qualité $bestQ$, et qui a donc un nouveau meilleur chemin $bestPath$ (c.-à-d.les informations de suivies déjà enregistrées ont été mis à jour par les meilleures valeurs calculées nouvellement). Pour chaque état, nous avons trois types de propositions, à savoir proposition nouvelle, améliorée et ancienne. La définition formelle de *ITPG* est la suivante :

Définition 15. *graphe de planification de suivi amélioré (Improved Tracking Planning Graph)*
Le graphe de planification de suivi amélioré (ITPG) est un graphe de planification de suivi (TPG) dans lequel chaque couche i :
P_i est divisé en sous-ensembles NGP_i, IGP_i et OGP_i. Ces derniers désignent respectivement les nouvelles propositions, les propositions améliorées et les anciennes propositions, c'est-à-dire $Pi = NGP_i \cup IGP_i \cup OGP_i$ et $NGP_i \cap IGP_i \cap OGP_i = \emptyset$. où :

- *OGP_i : Propositions qui sont déjà générées et ont la même valeur de la meilleure qualité, best, et en conséquence le même meilleur chemin (c'est-à-dire leur bestPath n'a pas été amélioré).*

$$OGP_i = \{p \in P_i \,|\, \forall op \in P_{i-1}, (op = p) \rightarrow (bestQ(p) = bestQ(op))\} \tag{4.18}$$

- *IGP_i : Propositions qui sont déjà générées, mais qui ont une nouvelle valeur de la meilleure qualité (c'est-à-dire leur bestPath a été amélioré).*

$$IGP_i = \{p \in P_i \,|\, \exists op \in P_{i-1}, (op = p) \wedge (bestQ(p) < bestQ(op))\} \tag{4.19}$$

- *NGP_i : Propositions nouvellement générées.*

$$NGP_i = \{p \in P_i \,|\, p \notin P_{i-1}\} \tag{4.20}$$

De plus, pour un ensemble d'actions applicables $actions(A_i)$

$$\begin{aligned} A_i = ACTIONS(P_i - 1) = \{a \in A \,|\, & PRECOND(a) \subseteq P_i - 1 \\ & \wedge \exists p \in PRECOND(a) \cap (NGP_{i-1} \cup IGP_{i-1}) \,\} \end{aligned} \tag{4.21}$$

De plus, pour l'état P_i généré :

$$P_i = RESULT(P_{i-1}, A_i) = \{NGP_i \cup IGP_i \cup OGP_i\} \tag{4.22}$$

Exemple 7. *La figure 4.3 présente le graphe de planification de suivi amélioré (ITPG) de l'exemple 3. Dans chaque couche, l'ensemble de propositions est divisé en trois (03) parties, à savoir : proposition ancienne (OGP_i), proposition améliorée (IGP_i) et nouvelle proposition (NGP_i). L'ensemble de propositions de la première couche (couche 0) ne contient que les nouvelles propositions. Ces dernières sont considérées comme anciennes dans la couche suivante. Dans la couche 2, la proposition* E *est considérée comme améliorée, car son chemin est mis à jour : de (w2) à (w3, w7). Les meilleurs chemins et les meilleures valeurs de la qualité des propositions sont calculés tels que présentés dans la définition du graphe de planification de suivi (TPG) (équations 4.13 et 4.15).*

comme dans l'exemple, si nous utilisons le graphe de planification pour résoudre un problème de composition de service Web sensible à la qualité de service (WSC QoS-aware), il est recommandé de poursuivre la construction du graphe de planification jusqu'à ce que le point fixe soit atteint, même si les propositions du but avaient déjà été obtenues. Cela permet d'avoir une nouvelle solution ayant une valeur de qualité de service qui est meilleure que celle de l'ancienne solution trouvée.

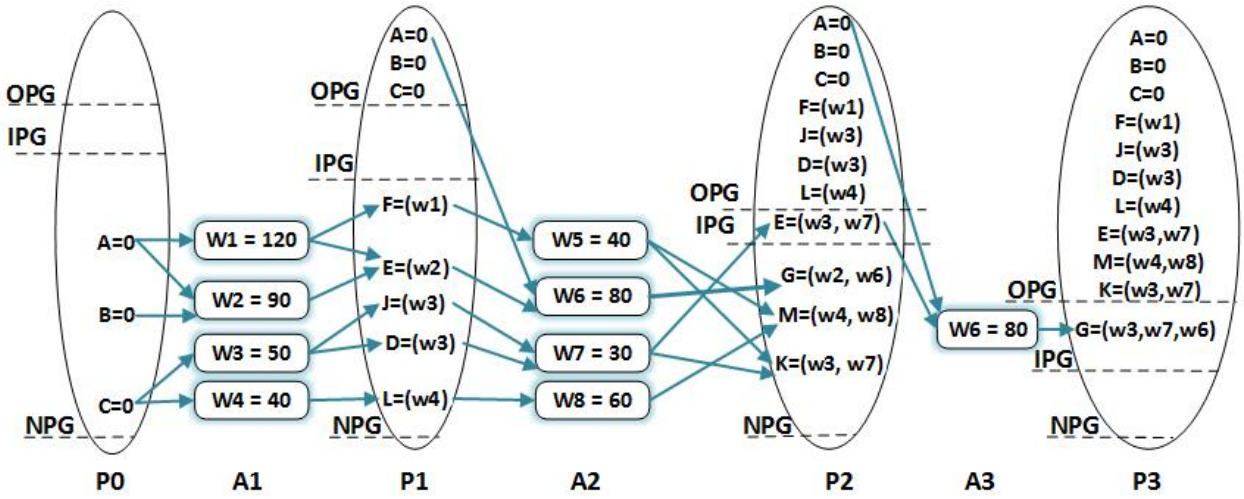


Figure 4.3 exemple de graphe de planification de suivi amélioré (ITPG)

Cependant, cette addition de couches peut être inutile s'il n'y a pas de nouvelle solution, c'est-à-dire qu'aucune des propositions du but n'est affectée par les améliorations causées par les couches ajoutées. Par conséquent, cela engendre une perte de temps pendant l'extraction de la solution.
Pour cette raison, nous introduisons le concept de niveau utile (Useful level), qui est défini comme suit :

Définition 16. *Useful level (Niveau utile)*
Le niveau utile (Useful level) est une couche du graphe de planification, et que son ensemble de propositions contient toutes les propositions du but, et il y en a au moins une qui est nouvellement générée ou améliorée.

$$UsefulLevel(i) \Leftrightarrow (g \subseteq P_i) \wedge (\exists p \in (g \cap (NGP_i \cup IGP_i))) \tag{4.23}$$

Nous utilisons cette équation (Équation 4.23) dans nos algorithmes expliqués dans la section suivante.

4.4 Les algorithmes de notre approche

4.4.1 Description des algorithmes

Nous utilisons la structure du graphe de planification de suivi amélioré(*ITPG*) et les équations ci-dessus dans nos algorithmes :

L'algorithme principal est le GraphPlan (Algorithme 1), qui est responsable de la construction de l'espace du problème à travers le *graphe de planification de suivi amélioré (ITPG)*, et assure également l'extraction de la solution si elle existe.

Les lignes 1 à 5, initialisent le graphe de planification de suivi amélioré(ITPG), la meilleure qualité de service de la solution ($+\infty$ signifie la plus mauvaise valeur de la qualité de service)et son chemin. De la ligne 6 jusqu'à 16, l'algorithme construit l'ITPG couche par couche. Pour chaque couche, il calcule les actions applicables (la ligne 8) selon l'équation 4.21.

Par la suite, (la ligne 9) fait appel à l'algorithme RESULTwithTracking qui se base sur l'équation 4.22. Cet algorithme crée un nouvel état divisé en trois sous-ensembles de propositions, à savoir *NGPi, IGPi et OGPi* (nouvelle, améliorée et ancienne proposition générée), comme indiqué dans la définition 15. La ligne 10, ajoute la nouvelle couche enrichie avec les informations de suivi au graphe ITPG. La ligne 11 à 14, l'algorithme principal appelle *ExtractSolution* afin d'extraire le chemin de la solution s'il en existe une nouvelle.

Notre algorithme (Algorithme 1) peut être vu comme un algorithme à tout moment (timing algorithm). C'est-à-dire il peut retourner des solutions (ligne 12) dès qu'elles deviennent disponibles avant que l'algorithme s'arrête. Pour éviter les extractions de solutions inutiles, nous utilisons le concept de niveau utile useful level (Definition 16 et l'équation 4.23) comme condition (ligne 11). C'est à dire les propositions de l'ensemble du but g sont atteintes, et qu'au moins une de ces propositions est nouvellement générée ou améliorée.

Par la suite, l'algorithme extrait la nouvelle solution en appelant l'Algorithme ExtractSolution (la ligne 12) et évalue sa qualité à partir des valeurs de qualités des propositions du but(la ligne 13) en utilisant l'équation 4.16.

Le chemin actuel de la solution et sa valeur de qualité sont enregistrés pour les comparer à une autre solution possible.

Notre algorithme *GraphPlan* (Algorithme 1) s'arrête quand il devient stable. Il est basé sur le théorème 1 de la stabilisation ITPG, c'est-à-dire lorsqu'il n'y a pas de proposition nouvelle ou améliorée (ligne 16).

L'algorithme 2, *RESULTwithTracking*, assure la création de l'état de proposition P_i à chaque couche. En se basant sur les équations 4.13 et 4.15, il calcule la valeur de la meilleure qualité et le chemin de chaque proposition, respectivement (lignes 4 et 3). Ensuite, si la proposition est nouvelle ou a été améliorée, l'algorithme enregistre les nouvelles informations de suivi. Si la proposition est ancienne, l'algorithme reste inchangé (lignes 2 à 16). Ligne 18 fournit les propositions divisées en *NGP, IGP et OGP*, comme le montre la définition 15.

L'algorithme 3, *ExtractSolution*, calcule la solution actuelle (la ligne 2) en multipliant les chemins des propositions du but en se basant sur l'équation 4.17. L'algorithme s'appuie également sur l'équation 4.16 pour évaluer la qualité de la solution actuelle (la ligne 1). Pour retourner la nouvelle solution, en plus de la condition de l'amélioration de la qualité de la solution (la ligne 3), nous prenons aussi en considération comme condition la minimisation du nombre de services de la solution ($|actualSol|$ et $|lastSol|$) dans la ligne 3.

Notant que dans notre structure de graphe de planification (IMPG) proposée, si la qualité de la solution n'est pas améliorée, sa qualité est conservée. Cependant, pour la lisibilité de notre algorithme, nous ajoutons, dans la ligne 3, la condition de l'égalité des qualités lorsque nous comparons les tailles de l'ancienne et de la nouvelle solution.

Algorithm 1 GraphPlan

Input: A, s_0, g

Output: $SolPath$

1: $SolPath \leftarrow \emptyset$

2: $SolQuality \leftarrow +\infty$

3: $P_0 = \{(p, bestPath, bestQ) | p \in s_0, bestPath = \emptyset, bestQ = 0\}$ {Etat initial}

4: $NGP_i \leftarrow P_0$ {toutes les propositions de l'état initial sont "new"}

5: $ITPG.AddLayer(\emptyset, P_0)$; {ITPG is Improved Tracking PG}

6: $i \leftarrow 1$

7: **repeat**

8: $A_i \leftarrow ACTIONS(P_{i-1})$ {All the applicable actions}

9: $P_i \leftarrow ResultWithTracking(P_{i-1}, A_i)$ {allow us to find the propositions and its tracks}

10: $ITPG.AddLayer(A_i, P_i)$

11: **if** $UsefulLevel(i)$ **then**

12: $SolPath \leftarrow ExtractSolution(ITPG, SolPath, SolQuality)$

13: $SolQuality \leftarrow evaluateActualSol(g)$

14: **end if**

15: $i \leftarrow i + 1$

16: **until** $(NGP_i \cup IGP_i = \emptyset)$

Algorithm 2 RESULTwithTracking

Input: P_{i-1}, A_i

Output: P_i

1: $P_i \leftarrow \emptyset$;
2: **for** each $p \in Output(A_i)$ **do**
3: $NewU_Q \leftarrow U_{QAi}(p)$
4: $newPath \leftarrow Path_{Ai}(p)$
5: **if** $p \notin P_{i-1}$ **then**
6: $p.bestQ \leftarrow NewU_Q$
7: $p.bestPath \leftarrow newPath$
8: $NGP_i \leftarrow NGP_i \cup \{p\}$
9: **else**
10: **if** $(NewU_Q < bestQ(p))$ **then**
11: $p.bestPath \leftarrow newPath$
12: $p.bestQ \leftarrow NewU_Q$
13: $IGP_i \leftarrow IGP_i \cup \{p\}$
14: **end if**
15: **end if**
16: **end for**
17: $OGP_i \leftarrow P_{i-1} \setminus IGP_i$
18: $P_i \leftarrow NGP_i \cup IGP_i \cup OGP_i$

Algorithm 3 ExtractSolution

Input: $ITPG = (P_0, A_1, P_1, ..., A_i, P_i), lastSol, lastSolQuality$ {ITPG is an Improved Tracking PG}

Output: $solutionPath$

1: $actualSolQ \leftarrow evaluateSol(g)$

2: $actuelSol \leftarrow path(g)$

3: **if** $(actualSolQ < lastSolQ)$ OR
$((actualSolQ = lastSolQ) AND (|actualSol| < |lastSol|))$ **then**

4: $solutionPath \leftarrow actuelSol$

5: **else**

6: $solutionPath \leftarrow lastSol$

7: **end if**

4.4.2 Les propriétés des algorithmes

Nous présentons les propriétés de notre GraphPlan en citant les théorèmes suivants (nous avons reporté la présentation de leurs démonstrations à la section 5.3.1).

Théorème 1. *Chaque* ITPG *se stabilise à la couche k, qui est la première couche qui satisfait* $NGP_{k+1} \cup IGP_{k+1} = \emptyset$.

Notre algorithme GraphPlan (Algorithme 1) s'arrête si les propositions ne peuvent plus être générées ou améliorées. C'est la condition de la stabilisation de notre graphe de planification amélioré.

Théorème 2. *Le temps nécessaire pour ajouter une nouvelle couche avec ses informations de suivi au graphe ITPG est polynomial dans la taille du problème de planification.*

Par d'autres termes, l'expansion de notre graphe de planification amélioré $ITPG$ est polynomiale.

Théorème 3. *La complexité de l'extraction de la solution est asymptotiquement linéaire dans la taille de l'ensemble des actions.*

Théorème 4. *Notre GraphPlan a une complexité polynomiale.*

4.5 Minimisation de l'espace de recherche

Bien que notre algorithme *GraphPlan* (Algorithme 1) a un temps polynomial dans la taille du problème de planification, la minimisation de cet espace de recherche devient cruciale si l'espace du problème est trop large. Cet espace est présenté de façon heuristique par le graphe de planification.

Dans cette sous-section, nous proposons un algorithme, *PlanningGraphMinimizer*, qui est utilisé pour minimiser le graphe de planification construit par notre *GraphPlan*. La définition formelle du graphe de planification minimal est présentée ci-dessous :

Définition 17. *Graphe de planification minimal*
*Le graphe de planification minimal(*minimal planning graph(MPG)*) est un sous-graphe du graphe de planification* $PG = (V_p \cup V_a, E) = (P_0, A_1, P_1, ...A_n, P_n)$*, où nous ne retenons que les propositions intéressantes ainsi que ses actions connexes, c'est à dire* $MPG = (Vm_P \cup Vm_a, Em) = (Pm_0, Am_1, Pm_1, ...Am_n, Pm_n)$ *où :*
$Pm_n = g$ *(dans l'état* P_n*, nous éliminons chaque proposition n'appartenant pas au but.*

$$Am_i = \{a \in A_i \,|\exists p \in output(a) \wedge p \in Pm_i\} \tag{4.24}$$

$$Pm_{i-1} = \{p \in P_{i-1} | \exists a \in Am_i \text{ and } p \in input(a)\} = input(Am_i) = \bigcup_{i=1}^{|Am_i|} input(ai) \tag{4.25}$$

De plus, pour l'ensemble des arêtes Em :

$$Em = \left\{ e = (pf, pt) \in E \,\middle|\, (pf, pt) \in (V_{pm} \cup V_{am})^2 \right\} \tag{4.26}$$

C'est-à-dire chaque arête e *de* Em *doit relier deux sommets de* MPG *ayant différent types.*

Exemple 8. *La figure 4.4 montre la minimisation du graphe de planification présenté dans l'exemple 3 (Figure 4.1).*
Dans l'ensemble de propositions de la dernière couche, nous ne gardons que les propositions qui font partie de l'ensemble du but, à savoir G et K. Ensuite, dans l'ensemble d'actions nous gardons

seulement les actions qui peuvent générer au moins l'une des propositions de l'ensemble du but. Pour cette raison, nous supprimons les actions w_1, w_2, w_3, w_4 et w_8. Ensuite, nous ajoutons les entrées de cet ensemble d'actions. Pour cela, nous n'avons que les propositions A, F, E, J et D dans l'ensemble des propositions P1.

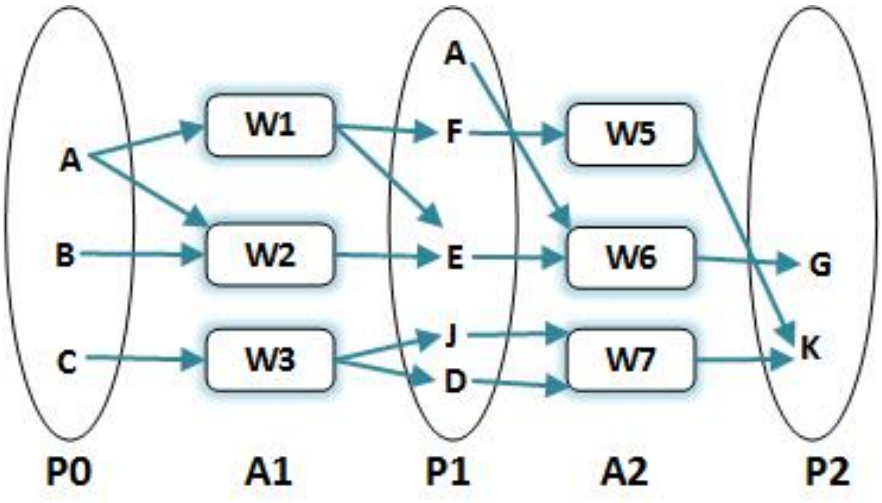


Figure 4.4 exemple de Minimal Planning Graph

Nous utilisons ces notions dans l'Algorithme 4, *PlanningGraphMinimizer*, qui élimine toute action ou proposition qui n'a aucune relation avec les propositions de l'ensemble du but.

L'algorithme reçoit le graphe ITPG, l'ensemble du but et le $usefulLevel$ (Définition 16). La valeur de $usefulLevel$ sert à limiter le nombre de niveaux dans la solution attendue.

Ainsi, par cette valeur, nous minimisons le nombre de couches dans le graphe (la ligne 2) et les itérations de la boucle (la ligne 4).

La ligne 2 crée un graphe de planification minimal (MPG) contenant la valeur de $usefulLevel$ de couches vides.

La ligne 3 permet de rendre l'ensemble de propositions de la dernière couche égal à l'ensemble de propositions du but. Ensuite, de la ligne 4 à 15, l'algorithme parcourt toutes les couches (la ligne 4). Dans chaque couche, il consulte toutes les couches de propositions (lignes 5 à 13). Pour chaque proposition, il extrait de la même couche du graphe envoyé ($ITPG$) les actions concernées CA (qui peuvent générer la proposition) dans la ligne 6. La ligne 7 met à jour l'ensemble des actions de MPG en ajoutant les actions concernées dans la couche courante i (basé sur l'Équation 4.24).

Également, la ligne 8 met à jour l'ensemble des propositions de la couche précédente en ajoutant toutes les entrées des actions concernées CA (basé sur Équation 4.25). S'il n'y a pas d'actions concernées (c'est-à-dire $CA = \emptyset$), la proposition ne peut pas être générée dans cette couche. Pour cette raison, il rapporte la consultation de cette proposition à la couche précédente (la ligne 10) et supprime la proposition de l'ensemble des propositions de la couche courante(la ligne 11).
L'intersection dans la ligne 14 signifie les propositions du but qui sont générées ou améliorées dans le graphe envoyé ($ITPG$). L'ajout de ces propositions sert à garder les chemins de toutes les solutions possibles, dans le graphe de planification minimal.

Algorithm 4 PlanningGraphMinimizer

Input: $ITPG(P_0, A_1, P_1, ...A_n, P_n), g, usefulLevel$

Output: $MPG(Pm_0, Am_1, Pm_1, ...Am_k, Pm_k)$ {is a minimal planning graph}

1: $k \leftarrow usefulLevel$
2: initializeGraph(MPG,k)
3: $Pm_k \leftarrow g$
4: **for** $i = k \rightarrow 1$ **do**
5: **for** each $p \in Pm_i$ **do**
6: **if** $(CA = \{a \in A_i \,|\, p \in output(a)\} \neq \emptyset)$ **then**
7: $Am_i \leftarrow Am_i \cup CA$
8: $Pm_{i-1} = Pm_{i-1} \cup input(CA)$
9: **else**
10: $Pm_{i-1} = Pm_{i-1} \cup \{p\}$
11: Remove p from Pm_i
12: **end if**
13: **end for**
14: $Pm_{i-1} = Pm_{i-1} \cup (P_{i-1} \cap g)$
15: **end for**

Théorème 5. *La minimisation du graphe de planification est polynomiale en la taille du problème de planification.*

Nous retardons sa démonstration à la section 5.3.1 du chapitre de la validation.

Notez que l'algorithme PlanningGraphMinimizer peut être applicable et utile pour tout type de graphe de planification, c'est-à-dire que notre algorithme de minimisation peut être utile pour d'autres approches basées sur le graphe de planification. Pour sa contribution globale, nous l'avons présentée dans une sous-section distincte (d'autres mises à jour de l'algorithme peuvent être nécessaires).

Dans notre cas et pour de meilleurs gains, nous proposons un scénario différent de celui présenté dans la sous-section précédente comme suit :

a. Génère le graphe de planification de suivi amélioré (improved Tracking planning graph (ITPG)), sans calculer les meilleurs chemins lors du calcul des informations de suivi. Nous ne calculons que la valeur de la meilleure qualité de propositions, ce qui nous permet d'obtenir le graphe de planification minimal(IMPG).

b. Appliquer l'algorithme PlanningGraphMinimizer sur le graphe de planification amélioré (ITPG) obtenu pour avoir le graphe de planification minimal (MPG).

c. Complétez le calcul des informations de suivi dans le graphe obtenu (MPG), en calculant les meilleurs chemins de ses propositions. Nous obtenons le graphe de planification de suivi amélioré complété (défini dans la définition 15) en considérant, dans chaque couche, uniquement les actions et les propositions présentées sur le graphe de planification minimisé (MPG). Ici, il y a l'avantage de limiter le calcul des informations de suivi au nombre de couches de MPG. Ce nombre de limites est déterminé par la valeur de niveau utile (useful level value) (Définition 16).

d. Au cours de la complétude du processus d'information de suivi, nous calculons chaque nouvelle solution en multipliant les meilleurs chemins des propositions du but en utilisant notre formule proposée (définition 13) comme décrit dans l'algorithme ExtractSolution (Algorithme 3).

Ce jumelage entre notre approche proposée basée sur l'enregistrement des informations de suivi et notre algorithme de minimisation a montré des avantages importants en termes de temps de

composition et en termes de nombre de services de la solution. Pour cette raison, nous considérons ce jumelage comme une stratégie pour minimiser le nombre de services tel que présenté dans la section ci-dessous.

4.6 Stratégies de minimisation de nombre de services de la composition

En plus du temps d'exécution, nous prenons en compte le nombre de services dans le processus de composition (QoS-Aware ASC process). Dans cette section, nous mettons l'accent sur les stratégies de la minimisation de nombre de services utilisées dans notre approche présentée dans les sections précédentes. Nos stratégies sont divisées en deux catégories comme suit :

a. Minimisation globale : Les stratégies de minimisation globale permettent de minimiser le nombre global de services dans le graphe de planification, ce qui augmente la possibilité de partager les mêmes services dans les chemins des propositions du but. Cela minimise le nombre de services dans le chemin de la solution, car il est obtenu en multipliant les chemins des propositions du but.

   Nous avons deux stratégies pour la minimisation globale des services.

   a. La structure améliorée d'ITPG (Définition 15) :

      L'ITPG, par rapport à PG classique, minimise le nombre global de services du graphe de planification. Pendant la phase de développement du PG (planning graph), nous ajoutons uniquement des actions qui semblent intéressantes pour la solution.

   b. La Minimisation d'ITPG (section 4.5) :

      En appliquant l'algorithme de minimisation (Algorithme 4), nous minimisons davantage le nombre global de services dans le graphe de planification.

b. Minimisation spécifique : Les stratégies de minimisation spécifiques utilisées dans notre approche influencent directement sur les chemins des propositions contenant les propositions du but ou sur le chemin de la solution. Ce sont les deux stratégies utilisées dans notre approche :

   a. Selection de services :

      Il s'agit du cas où il existe plus d'un service qui génère une proposition. Nous choisissons le service ayant le maximum de sorties dans la couche de l'ensemble de propositions. Cela augmente également la possibilité des chemins des propositions qui partagent

les mêmes services dans chaque niveau. Cette stratégie est utilisée par l'algorithme RESULTwithTracking (Algorithme 2), mais pour des raisons de simplicité, elle n'est pas présentée dans son pseudo-code.

b. Selection de chemin de la solution :
Il s'agit le cas où il existe plus d'une solution ayant la même qualité. Nous retenons celle ayant le minimum de nombre de services. Nous avons appliqué cette stratégie dans l'algorithme ExtractSolution (Algorithme 3).

4.7 Conclusion

Le travail présenté dans ce chapitre s'inscrit dans le cadre de l'adaptation de la composition de service. Notre objectif spécifique est d'améliorer le temps d'exécution des ASC sensible à la qualité de service tout en minimisant autant que possible le nombre de services dans la composition. Les techniques de planification, en particulier le graphe de planification, sont considérées comme les plus efficaces pour résoudre le problème ASC.

Pour minimiser le temps consacré à l'extraction de la solution, nous avons établi notre approche sur l'exploitation des informations de trace enregistrées pendant la construction du graphe de planification. De plus, pour récupérer le temps consacré à la garde de trace de la construction, nous avons proposé une amélioration de la structure du graphe de planification qui permet de minimiser le nombre d'éléments du graphe (c'est-à-dire les propositions et les actions) et donc de réduire le temps d'exécution nécessaire à sa construction. Pour plus de minimisation, nous employons notre algorithme de minimisation proposé. De plus, ces minimisations, couplées à nos autres stratégies proposées (services et choix de solutions) ont un effet positif sur le nombre de services de la solution.

Dans le chapitre de validation qui suit, nous allons présenter, la faisabilité et l'efficacité de l'application de notre approche proposée.

CHAPITRE 5

IMPLÉMENTATION ET VALIDATION

5.1 Introduction

Dans ce chapitre, nous nous intéressons à la validation de notre approche de l'adaptation des services Web. Notre approche se constitue, essentiellement, de deux contributions, à savoir la gestion sémantique basée agent du contexte et la composition de service Web basée sur une amélioration du graphe de planification.
Par conséquent, ce chapitre est divisé en deux sections principales qui montrent la validation de nos propositions :
Dans la section 5.2 qui est consacrée à la validation de notre approche de la gestion sémantique du contexte, nous allons expliquer l'environnement matériel et logiciel sur lequel notre système a été développé et expérimenté. Nous allons présenter et discuter les résultats de l'expérimentation que nous avons réalisée pour évaluer la performance de notre approche.

La section 5.3 est consacrée également à la validation de notre approche de la composition sensible à la qualité de service (QoS-aware ASC). Nous allons commencer par la présentation des démonstrations de nos théorèmes présentées dans le chapitre 4. Par la suite, une explication de l'environnement de l'implémentation et l'expérimentation sera présentée. Puis, nous allons montrer les résultats de l'évaluation que nous avons réalisée, les discuter et les comparer à certains travaux voisins qui utilisent les mêmes ensembles de données (data sets) que nous avons utilisés.

5.2 Implémentation et évaluation de notre approche de gestion sémantique du contexte

Dans le chapitre 3, nous avons présenté notre approche de la représentation sémantique du contexte, ses instances et leurs changements, et la façon dont ces notions s'intègrent dans une architecture globale basée sur différents agents acteurs afin d'assurer la gestion sémantique du contexte. Par la suite, nous allons présenter l'évaluation de l'implémentation de notre approche à travers les résultats des expérimentations réalisées.

5.2.1 Environnement d'implémentation et d'évaluation

L'intergiciel a été mis en œuvre en Java en utilisant JDK 1.8.0_2, tandis que les contextes, leurs instances et les journaux des changements ont été encodés à l'aide des langages sémantiques (RDF et RDF(S)) nous avons utilisé l'infrastructure Jena 2.12.1.
Actuellement, nous avons utilisé les threads pour implémenter les différents agents de notre intergiciel.
De plus, nous avons utilisé la bibliothèque Java jmathplot pour dessiner les diagrammes de résultats.

Nous avons effectué des tests préliminaires sur ordinateur portable Acer Aspire 5742 équipé de 6.0 Go de RAM, processeur Intel i5 M480 à 2.67 GHz, le système d'exploitation Windows 7 de Microsoft a été utilisé, la Machine Virtuelle Java version 1.8.0_25-b18.

5.2.2 Évaluation de performance

La performance de notre système a été mesurée en se basant sur notre implémentation actuelle. Les tests sont organisés en deux catégories :

- la quantité d'informations communiquées
  La première catégorie des tests s'est concentrée sur la taille d'informations communiquées lorsqu'il y a des changements du contexte. Dans cette catégorie, nous avons utilisé le même modèle contextuel de l'exemple d'illustration (voir la section 3.2 du chapitre 3) et nous avons créé des senseurs virtuels font périodiquement des changements aléatoires. Cela pour augmenter le nombre de changements afin de bien évaluer la performance.
  La figure 5.1 présente la mesure de quantités par l'utilisation du journal de changements (en couleur bleue) et l'autre par la communication de l'instance du contexte (en couleur rouge).
  Où l'axe X présente le temps en seconde et l'axe Y présente la taille en octets.
  Ce résultat montre le bénéfice en quantité d'information si on utilise la technique du journal de changements (il n'a pas dépassé 1.6 Ko) au lieu de communiquer la totalité de l'instance.

Figure 5.1 Résultats des tests sur la quantité d'informations

En notant, qu'après chaque envoi du journal on construit refaire un autre de nouveau.

- La durée d'exécution

  Dans cette catégorie de tests, on a utilisé un grand modèle de contexte qui contient 307 éléments entre concepts et relations. Cela pour nous permettre de faire un nombre de changements considérable à chaque création et envoi du journal.

  Donc, on s'intéresse ici au calcul du temps nécessaire pour la création et l'envoi du journal (couleur bleue) et de l'instance du modèle de contexte selon le nombre des changements réalisés.

  L'axe X de la figure 5.2 montre le nombre de changements réalisés et l'axe Y montre la durée

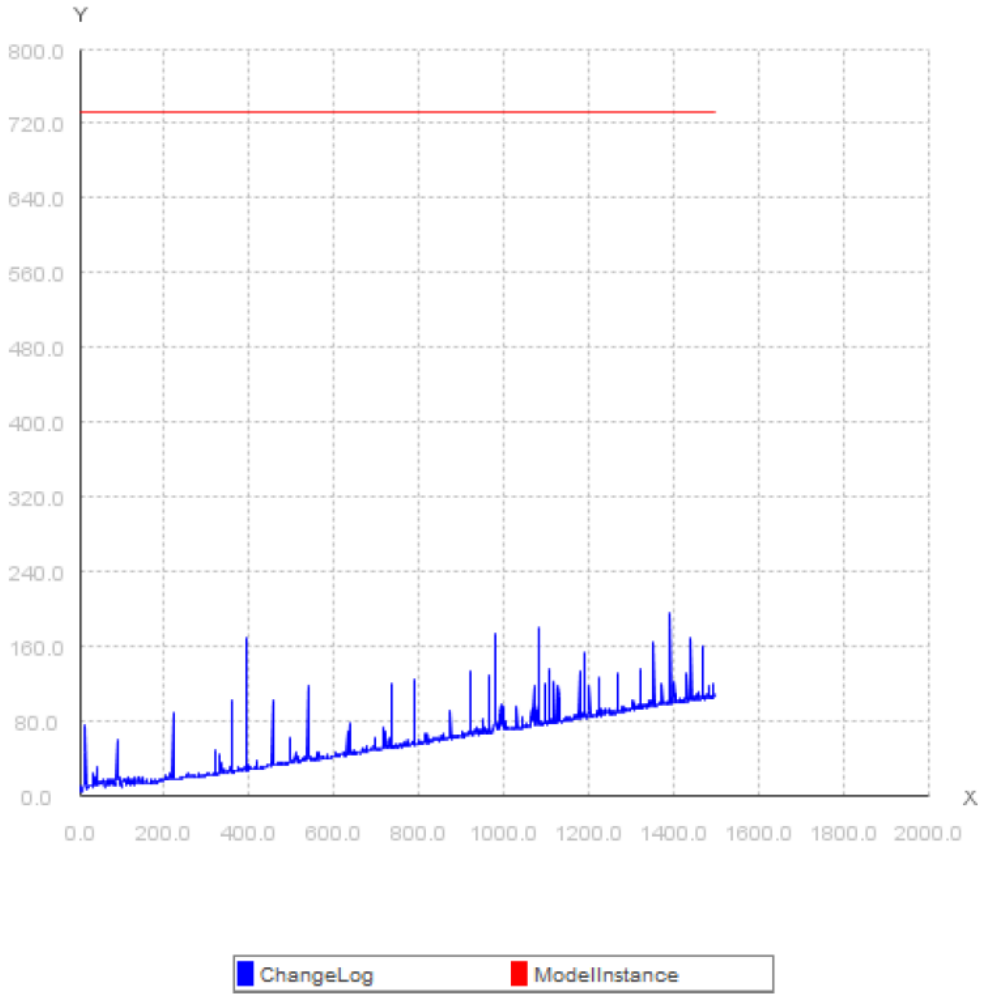


Figure 5.2 Résultats des tests sur le temps d'exécution

en milliseconde.

On peut constater la grande différence entre l'utilisation directe de l'instance du contexte et l'utilisation du journal qui n'a pas dépassé le 0,2 seconde pour un grand nombre de changements (1500 changements).

Ces résultats sont encourageants pour l'utilisation de notre approche sur de petites machines. D'autres expérimentations sont programmées à faire. Elles concernent notamment les tests sur les différents types des dispositifs et avec des senseurs physiques.

5.3 Validation de notre approche de composition de service Web

Dans le chapitre 4, nous avons présenté notre approche de la composition du service Web en montrant notre amélioration du graphe de planification et nos algorithmes. Nous avons montré notre approche d'une manière formelle afin qu'on puisse prouver la faisabilité de notre amélioration, en démontrant la stabilité de notre modèle amélioré du graphe de planification et que sa construction et l'extraction de la solution soient toujours en temps raisonnable.

5.3.1 Démonstration des théorèmes

Dans cette sous-section, nous montrons les preuves des différents théorèmes présentés dans le chapitre 4après la démonstration des théorèmes Dans cette section, après la démonstration des théorèmes, nous présenterons l'implémentation et l'évaluation de nos algorithmes et l'analyse des résultats des expérimentations que nous avons réalisées.

Théorème 1. *Chaque* ITPG *se stabilise à la couche k, qui est la première couche qui satisfait* $NGP_{k+1} \cup IGP_{k+1} = \emptyset$.

Cela signifie qu'il n'y a aucune action applicable dans $A_{k+1} = \emptyset$ qui peut générer au moins une nouvelle proposition ou améliorer au moins une proposition de l'ensemble de propositions précédentes. Par conséquent, $A_{k+2} = \emptyset$ (d'après notre définition des actions applicables de notre modèle ITPG présenté par l' équation 4.21) et $P_{k+2} = P_{k+1}$. Pour cette raison, nous pouvons dire que notre modèle de graphe de planification est stabilisé.
Donc, pour prouver que notre modèle de graphe de planification a toujours un point de stabilisation, il suffit de prouver que le cas de $NGP_i \cup IGP_i = \emptyset$ est toujours accessible.

Démonstration
Soit (A, so, g) un problème de planification, tel que : A est un ensemble d'actions, $s0$ est un état initial, g est un ensemble de propositions du but et P est un ensemble de propositions. Supposons que $|P| = n$ et $|A| = m$.

Nous allons prouver que $NGP_i = \emptyset \wedge IGP_i = \emptyset$ est toujours accessible.

- $NGP_i = \emptyset$

  Selon la définition de notre modèle, dans chaque couche i, les propositions NGP_{i-1} de la couche précédente seront incluses dans l'ensemble de propositions OGP_i, ou dans l'ensemble IGP_i s'il y a une amélioration en ce qui concerne le chemin, c'est-à-dire $\forall p \in NGP_{i-1} \rightarrow p \in OGP_i \vee p \in IGP_i$.

  De plus, et aussi par définition de notre modèle, pour une couche donnée, il n'est pas possible d'avoir une proposition incluse dans OGP ou IGP et qui sera sera incluse dans l'un des ensembles NPG des couches suivantes, c'est-à-dire $\forall p \in OGP_i \cup p \in IGP_i \rightarrow \neg\exists j \in \mathbb{N}\left(p \in NGP_j \wedge j \geq i\right)$. Par conséquent, il n'y a pas de répétition d'apparence dans l'ensemble NGP. Pour plus d'explication, si une proposition p apparaît dans l'ensemble NGP d'une couche donnée, elle ne peut pas réapparaître dans l'ensemble NGP des couches suivantes, c'est-à-dire $\forall p \in NGP_i \wedge p \in NGP_J \rightarrow i = j$.

  D'autre part, le nombre de propositions est borné (c'est-à-dire $|P| = n$).Ainsi, après certains moments, l'ensemble de propositions NGP doit être égal à l'ensemble vide $NGP = \emptyset$. Pour plus d'explication, dans le pire des cas, le $\bigcup_{i=0}^{k} NGP_i = P$, après NGP_{k+1} doit être vide, c'est-à-dire $NGP_{k+1} = \{\}$.

- $IGP_i = \emptyset$

  Tel que : IPG_i est un ensemble de propositions améliorées qui existaient dans l'ensemble précédent de propositions, c'est-à-dire $\forall p \in IGP_i \rightarrow p \in OGP_{i-1} \cup IGP_{i-1} \cup NGP_{i-1}$. Puisque l'ensemble d'actions est borné (c'est-à-dire $|A| = m$), le nombre d'actions pouvant générer une proposition p est également borné. Formellement, l'ensemble $gA(p) = \{w \in A \,|P \in Output(w)\}$ est borné par m, avec $m = |A|$. Par conséquent, le nombre d'actions qui peuvent améliorer le chemin de la proposition p est également borné.

  D'autre part, puisque la valeur de la meilleure qualité de toute proposition est bornée, le nombre d'améliorations de la proposition p par une action a est également limité. En d'autres termes, l'action a ne peut pas améliorer le meilleur chemin d'une proposition après que cette dernière ayant la valeur de la meilleure qualité. Peu importe, si cette action a été déjà exécutée ou non.

  Ainsi, puisque le nombre de propositions est borné, ainsi que ses éventuelles améliorations, il

est certain d'atteindre une situation où aucune proposition ne peut être améliorée, c'est-à-dire $IPG = \emptyset$.

Nous avons prouvé que le cas de $NGP_i = \emptyset$ et $IGP_i = \emptyset$ doit être atteint. Cela signifie que $NGP_i \cup IGP_i = \emptyset$ doit également être atteint pour notre modèle.

Par conséquent, notre algorithme GraphPlan (Algorithme 1) s'arrête si les propositions ne peuvent plus être générées ou améliorées.

Théorème 2. *Le temps nécessaire pour ajouter une nouvelle couche avec ses informations de suivi au graphe ITPG est polynomial dans la taille du problème de planification.* **Démonstration**
Soit A un ensemble d'actions, $s0$ est un état initial, g est un ensemble de propositions du but et P est un ensemble de propositions. Suppose que $|P| = n$ et $|A| = m$.
Ainsi, le nombre d'éléments contenus dans chaque ensemble de propositions et dans chaque ensemble d'actions ne peut pas être supérieur à n et m, respectivement, car une proposition et une action ne peuvent apparaître qu'une seule fois dans le même ensemble, c'est-à-dire $\forall i : |P_i| \leq n \wedge |A_i| \leq m$.

En outre, sachant que chaque proposition dans P_i a un meilleur chemin dont la taille dans le pire des cas est $k \times m$, où k est la couche actuelle, la taille de la couche k devient $k \times M \times n + m$ dans le pire des cas. Par conséquent, l'expansion de $ITPG$ est polynomiale.

Théorème 3. *La complexité de l'extraction de la solution est asymptotiquement linéaire dans la taille de l'ensemble des actions.*

Démonstration
Soit a un ensemble d'actions, $s0$ est un état initial, g est un ensemble de propositions du but et P est un ensemble de propositions. Supposons que $|P| = n$ et $|A| = m$.

Étant donné que k est le nombre de couches, l'algorithme d'extraction de la solution (Algorithme 3) fait la multiplication des chemins des propositions du but, la taille de chaque chemin de proposition est $k\ timesm$ dans le pire des cas. Par conséquent, la complexité de la multiplication est $(|g| - 1) \times$

$k \times m$. Parce que, dans la pratique [1], le $|g|$ et k sont beaucoup plus petits que m, il s'ensuit que la solution d'extraction est asymptotiquement bornée par m, donc sa complexité est $O(m)$, c'est-à-dire une complexité linéaire dans la taille de l'ensemble des actions m et la solution peut être obtenue en temps linéaire.

Théorème 4. *Notre GraphPlan a une complexité polynomiale.*

Démonstration

Selon le théorème 1, chaque graphe de planification $ITPG$ a des couches distinctes finies, et du théorème 2, la taille de chaque couche est polynomiale. Par conséquent, la complexité de la construction du graphe de planification ITPG est polynomiale. D'autre part, la complexité de l'extraction de la solution est linéaire, selon le théorème 3. En conséquence, il s'ensuit que notre GraphPlan est de complexité polynomiale dans la taille du problème de planification.

Théorème 5. *La minimisation du graphe de planification est polynomiale en la taille du problème de planification.*

Démonstration

Soit le tuple (A, so, g) un énoncé d'un problème de planification, où A est un ensemble d'actions, $s0$ est un état initial, g est un ensemble de propositions du but et P est un ensemble de propositions. Supposons que $|P| = n$, $|A| = m$, et k est le nombre de couches du MPG.

Tout d'abord, l'algorithme PlanningGraphMinimizer (Algorithme 4) a besoin de $k \times n$ itérations. Dans chaque itération, il vérifie une condition spécifique qui nécessite l'extraction des actions concernées. Ce dernier est délimité par $m \times n$, dans le pire des cas, en raison de $|output(a)| \leq n \wedge |A_{mi}| \leq m$.

Nous prenons la partie la plus complexe de la condition, c'est-à-dire lorsqu'il y a des actions concernées. Cette partie, dans le pire des cas, est limitée par $m + n \times m$.

1. Cette hypothèse est basée sur la taille de l'ensemble des actions m est très grande que le nombre de sorties de la requête ($|g|$), à la grande échelle de service. Ainsi$|g| << m$. De plus, en pratique, le nombre de couches du graphe de planification k est très petit par rapport à m, par exemple, dans notre expérimentation, on a le cas de $k = 7$ et $m = 15000$.

Deuxièmement, pour conserver toutes les solutions possibles (la ligne 14), l'algorithme Planning-GraphMinimizer nécessite $k \times |(g-1)| \times n$ opérations dans le pire des cas. Par conséquent, l'algorithme est délimité asymptotiquement par $k \times (m+1) \times n^2$. Ainsi, sa complexité est $O(n^2)$. Par conséquent, sa complexité est polynomiale en la taille du problème de planification.

5.3.2 Environnement d'implémentation et d'évaluation

Afin d'évaluer la faisabilité et l'utilité de notre approche proposée dans le contexte des services à grande échelle, nous avons utilisé les ensembles de données communs de la compétition sur le WSC[Bleul *et al.* (2009)], qui sont utilisés dans de nombreux travaux ASC, tels que ceux de [Yan *et al.* (2012); Chen & Yan (2014); Yan *et al.* (2008)]. Chaque ensemble de données contient des fichiers WSDL et WSLA pour la fonctionnalité des services Web et la description de la qualité et des documents OWL pour la description des concepts d'ontologie. Les ensembles de données tiennent compte de deux critères de QoS, à savoir le temps de réponse et le débit. Nous avons donc utilisé les dix (10) ensembles de données de la compétition Web Service Challenge WSC'10 [Blake *et al.* (2010)] (les cinq (05) ensembles de données WSC'09 [Kona *et al.* (2009)] avec d'autres ensembles de données de test à plus grande échelle [Weise *et al.* (2013)]) dans notre expérimentation. Afin d'évaluer la faisabilité et l'utilité de notre approche proposée dans le contexte des services à grande échelle, nous avons utilisé les ensembles de données communs de la compétition sur le WSC[Bleul *et al.* (2009)], qui sont utilisés dans de nombreux travaux ASC, tels que ceux de [Yan *et al.* (2012); Chen & Yan (2014); Yan *et al.* (2008)]. Chaque ensemble de données contient des fichiers WSDL et WSLA pour la fonctionnalité des services Web et la description de la qualité et des documents OWL pour la description des concepts d'ontologie. Les ensembles de données tiennent compte de deux critères de QoS, à savoir le temps de réponse et le débit. Nous avons donc utilisé les dix (10) ensembles de données de la compétition Web Service Challenge WSC'10 [Blake *et al.* (2010)] (les cinq (05) ensembles de données WSC'09 [Kona *et al.* (2009)] avec d'autres ensembles de données de test à plus grande échelle [Weise *et al.* (2013)]) dans notre expérimentation. Le nombre de services dans nos ensembles de données varie de 500 à 20.000, tandis que le nombre de concepts varie de 5.000 à 100.000 [Weise *et al.* (2013)].

De plus, pour vérifier l'exactitude des résultats obtenus par notre mise en œuvre, nous avons utilisé un outil de vérification proposé par la compétition WSC2009-2010 [Bleul *et al.* (2009)].

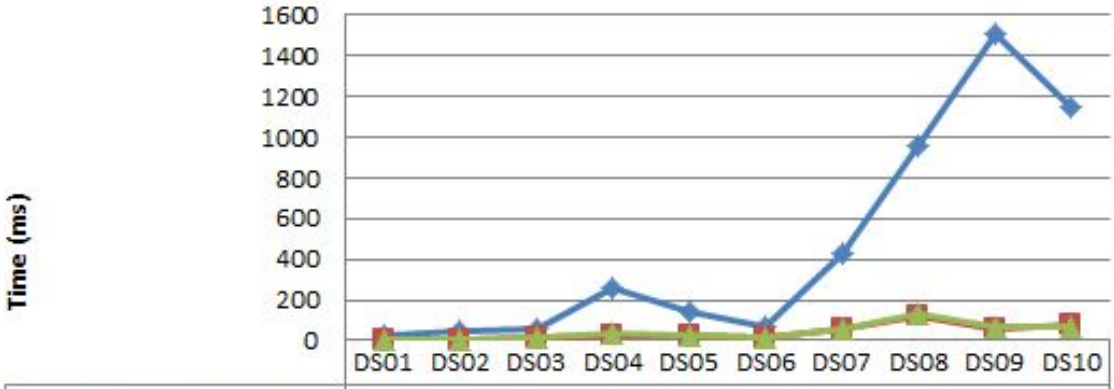
Nous avons implémenté les algorithmes de notre approche en Java (JDK 1.8.025). Nous avons également utilisé JDOM API [Jason Hunter (2015)] pour la gestion des fichiers XML. Pour accélérer l'exécution des algorithmes, nous avons utilisé des techniques comme les tables d'index (index tables) et les index inversés (inverted index), qui ont stocké en utilisant la technique des tables de hachage comme dans [Yan *et al.* (2008, 2012)].
Nos expérimentations ont été effectuées en utilisant un HP Compaq $dc5850$ desktop équipé avec 8 GB de RAM, et 2.3 GHz AMD Phenom(tm) $9600B$ Quad-Core processeur avec 2.3 GHz, Microsoft Windows 7 (64 bit), et Java Virtual Machine $1.8.0\_25 - b18$.

5.3.3 Évaluation et discutions des résultats

Le but de cette expérimentation était de démontrer la faisabilité et l'utilité de notre approche basée, entre autres, sur l'amélioration de la structure du graphe de planification (ITPG) et l'algorithme de minimisation proposé. Notre expérimentation a été réalisée sur la base des trois éléments de comparaison suivants :

a. Le temps de construction du graphe de planification classique est comparé au temps de construction du graphe de planification amélioré. La planification améliorée utilisée ici est un ITPG (Graphe de planification de suivi amélioré), mais qui n'implique pas le suivi des chemins, c'est-à-dire que l'on obtient le graphe de planification amélioré en calculant la meilleure valeur de qualité plutôt que le meilleur chemin pour chaque proposition.

   La figure 5.3 montre la différence dans la construction du temps entre notre graphe de planification amélioré et le graphe de planification classique. Le PG classique est implémenté dans notre machine en utilisant les mêmes techniques que les tables d'index et l'index inversé. Étant donné que ses valeurs de temps de construction sont différentes d'une QoS (par exemple, temps de réponse) à une autre (par exemple débit) pour le graphe PG amélioré qui est basée sur la génération de propositions nouvelles ou améliorées,

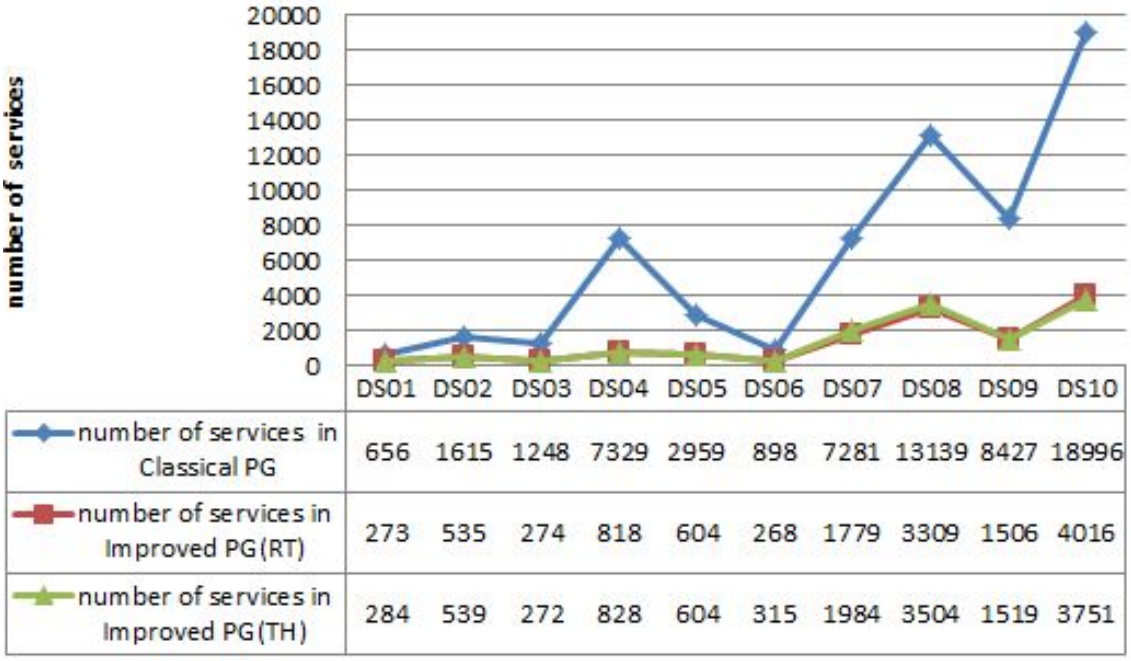


	DS01	DS02	DS03	DS04	DS05	DS06	DS07	DS08	DS09	DS10
RunTime of Dev. of PG	23	46	57	260	138	70	430	955	1503	1152
RunTime of Dev. of Improved PG (RT)	6	8	13	27	24	18	56	126	59	77
RunTime of Dev. of Improved PG (TH)	8	10	12	33	22	14	63	137	66	70

Figure 5.3 Comparaison du temps de développement du PG classique et du PG amélioré

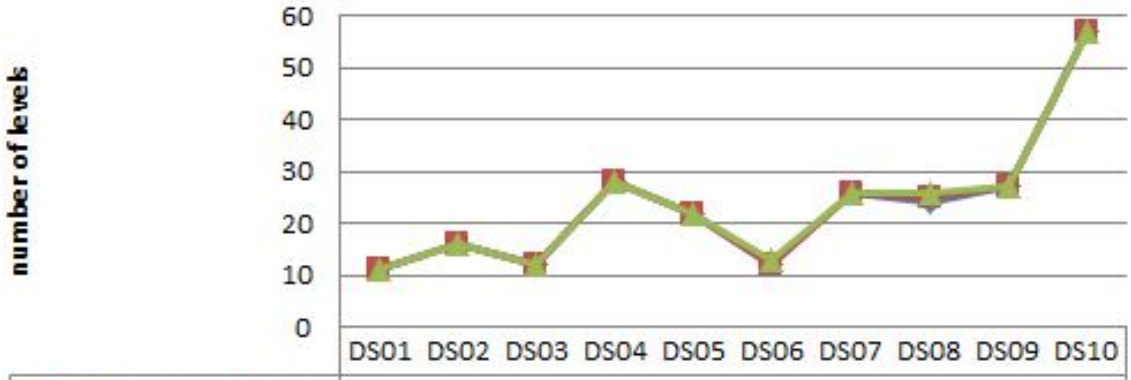
Le gain de temps du développement du graphe de planification est plus important lorsque la taille de l'ensemble de données est plus grande. Ce gain peut être expliqué par la minimisation du nombre de services dans le cas du PG amélioré par rapport au PG classique, tel que présenté dans la figure 5.4. La figure, qui montre une énorme différence dans le nombre de services dans certains grands ensembles de données (par exemple, les ensembles de données 08, 09 et 10). Logiquement, cette différence dans le nombre de services nécessitera un traitement plus de la machine, et donc des temps de calcul plus longs.

Enfin, nous parlons de la stabilisation de notre graphe de planification, car nous avons atteint notre condition spécifique de stabilisation, à savoir nous arrêtons quand il n'y a pas de proposition nouvelle ou améliorée présente. La figure 5.5 montre le nombre de couches dans la PG classique ainsi que notre PG améliorée pour chaque critère de QoS. Dans la plupart des cas, notre PG améliorée a la même valeur que le nombre de couches dans la PG classique. La différence dans certains cas (par exemple, DataSet 06 et DataSet 08) s'explique par le fait qu'il reste encore des améliorations à apporter dans les couches d'ajout de notre PG améliorée. Notez que ces améliorations peuvent avoir une influence positive sur la qualité du résultat si certaines des propositions améliorées appartiennent au but.



	DS01	DS02	DS03	DS04	DS05	DS06	DS07	DS08	DS09	DS10
number of services in Classical PG	656	1615	1248	7329	2959	898	7281	13139	8427	18996
number of services in Improved PG(RT)	273	535	274	818	604	268	1779	3309	1506	4016
number of services in Improved PG(TH)	284	539	272	828	604	315	1984	3504	1519	3751

Figure 5.4 Comparaison du nombre de services pour le PG classique et PG amélioré



	DS01	DS02	DS03	DS04	DS05	DS06	DS07	DS08	DS09	DS10
number of levels in Classical PG	11	16	12	28	22	12	26	24	27	57
number of levels in Improved PG (RT)	11	16	12	28	22	12	26	25	27	57
number of levels in Improved PG (TH)	11	16	12	28	22	13	26	26	27	57

Figure 5.5 Comparaison de la stabilisation du PG classique et du PG amélioré

Ainsi, bien que l'objectif principal de l'amélioration de la structure du graphe de planification soit de récupérer le temps perdu lors de l'enregistrement des chemins (meilleure qualité),

l'ITPG a présenté un avantage sur le temps et le nombre de services par rapport au graphe de planification classique.

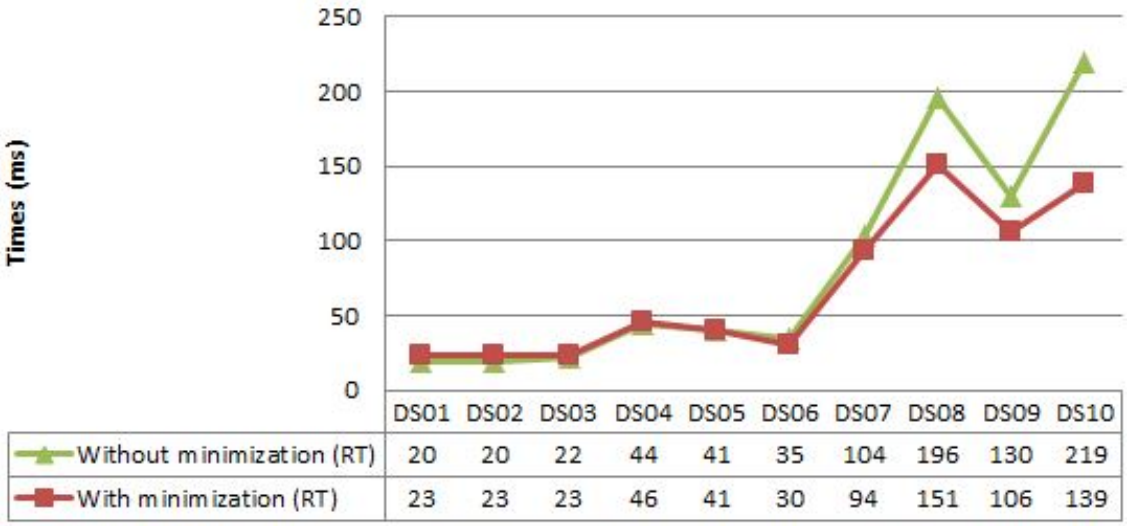
b. Le temps d'exécution de la composition est calculé en utilisant la structure de graphes de planification améliorée (ITPG), avec et sans minimisation, c'est-à-dire avec ou sans l'exécution de l'algorithme de minimisation). Dans le cas sans minimisation du graphe de planification, nous avons calculé les meilleurs chemins au moment du développement du PG amélioré. Dans l'autre cas, nous avons d'abord développé le graphe de planification amélioré. Ensuite, on obtient la PG minimale, par l'algorithme de minimisation. Enfin, nous avons calculé les meilleurs chemins sur le graphe de planification minimisé.

   En plus de ces procédures principales, le temps d'exécution mesuré a également couvert le traitement de demande de contestation et la génération du fichier BPEL de la solution.

   Comme dans la compétition WSC'10 (WSC'10 challenge), pour chaque jeu de données, nous avons pris le minimum des cinq temps d'exécution mesurés. Ce minimum est considéré comme étant le plus réaliste (puisque tous les algorithmes sont déterministes et que le temps d'exécution ne peut qu'accroître à cause des facteurs environnementaux, par exemple, l'ordonnancement et la communication). Les figures 5.6 et 5.7 montrent la différence d'exécution entre ces deux cas (c'est-à-dire avec et sans minimisation) pour le temps de réponse et le débit, respectivement. Le gain de temps obtenu par le processus de minimisation est plus important avec les ensembles de données les plus grands (les ensembles de données 08, 09 et 10). Pour les ensembles de données restants, les valeurs mesurées sont plus proches. Nous pouvons expliquer cela par le temps d'exécution du processus de minimisation est presque équivalent au temps coulé au calcul des meilleurs chemins pour les propositions non intéressantes. Ainsi, dans la prochaine évaluation et comparaison avec le travail connexe, nous prenons les valeurs de notre approche de composition avec un processus de minimisation.

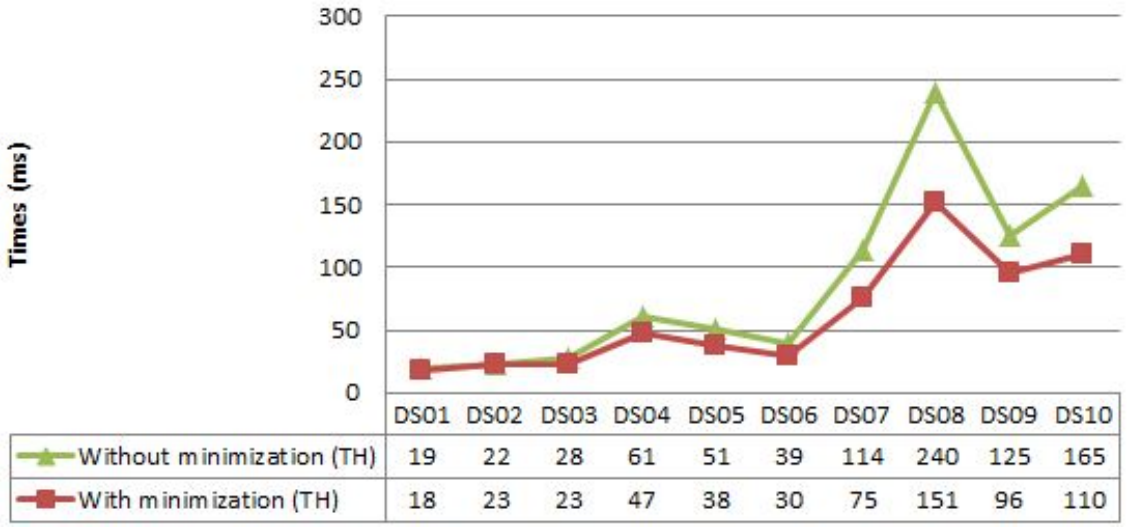
   Notez que malgré l'amélioration significative observée dans le temps d'exécution, surtout si l'ensemble de données est très grand, le processus de minimisation prend peu ou pas de temps, par rapport au temps global de la composition. Il n'a pas dépassé 3% au pire des cas.

c. Comparaison de notre approche ASC (notre algorithme GraphPlan avec minimisation) avec le travail utilisant les mêmes ensembles de données. Ces travaux sont les trois premiers gagnants (top-3) de la compétition sur la composition de service Web (WSC'10 challenge).



	DS01	DS02	DS03	DS04	DS05	DS06	DS07	DS08	DS09	DS10
Without minimization (RT)	20	20	22	44	41	35	104	196	130	219
With minimization (RT)	23	23	23	46	41	30	94	151	106	139

Figure 5.6 Composition exécution de notre mise en œuvre avec et sans minimisation pour le critère RT



	DS01	DS02	DS03	DS04	DS05	DS06	DS07	DS08	DS09	DS10
Without minimization (TH)	19	22	28	61	51	39	114	240	125	165
With minimization (TH)	18	23	23	47	38	30	75	151	96	110

Figure 5.7 Durée de la composition de notre implémentation avec et sans minimisation pour le critère de débit

Comme le montre le tableau 5.1, les critères de comparaison sont :

- Les valeurs QoS, soit le temps de réponse ou le débit.
- Le nombre de niveaux dans la composition. Comme dans la WSC'10, nous prenons la plus basse des deux solutions (l'une pour la qualité RT et l'autre pour la qualité TH) de chaque jeu de données, c'est-à-dire la composition la plus courte.

- Le nombre de services dans la composition. De même que le nombre de niveaux, nous choisissons la solution ayant le plus faible nombre de services.
- Le temps de composition (Composer Runtime dans la WSC'10) : désigne le temps nécessaire pour effectuer la composition à partir du moment où la requête est envoyée à quand le résultat est obtenu (fichier BPEL).

  Comme mentionné ci-dessus, et comme dans la WSC'10, pour chaque jeu de données, nous prenons un minimum de cinq temps d'exécution mesurés.

Du tableau 5.1, on peut déduire les points suivants :

a. Pour les résultats de QoS, que ce soit pour RT ou TH, notre approche donne toujours des résultats ayant la meilleure qualité, ce qui n'est pas le cas avec d'autres approches. Par exemple, le système CAS pour la QoS RT dans le cas de l'ensemble de données 7 (6780 au lieu de 3710). Les approches comparées peuvent ne renvoyer aucun résultat ; par exemple, les systèmes RUG et Tsinghua dans le cas de l'ensemble de données 10.

b. Pour le nombre de niveaux, notre approche donne toujours aussi une composition ayant le minimum de niveaux (respectant la condition d'avoir la meilleure qualité) par rapport à d'autres approches ; par exemple, les cas DS07, DS09 et DS10 pour les approches CAS, RUG et Tsinghua respectivement.

   Cet avantage est obtenu par le concept de niveau utile, qui enregistre le dernier niveau dans lequel une amélioration est observée dans le chemin (valeur de la qualité, spécifiquement) de l'une des propositions du but.

c. Le temps d'exécution de notre compositeur est toujours inférieur à celui des autres approches. Cela montre l'efficacité de notre approche en termes de temps d'exécution, qui est l'un des objectifs de notre contribution.

d. Pour le nombre de services, dans la plupart des cas (8 sur 10), les résultats de notre approche sont inférieurs à ceux des autres approches. Cela montre l'efficacité de notre approche en ce qui concerne le nombre de services dans la composition. Notre contribution a été assurée par l'application de nos stratégies de réduction du nombre de services (présentées dans la sous-section 4.6).

   D'autre part, il y a deux cas (DS01 et DS02) où les résultats de notre approche sont de

second ordre. Cela signifie que malgré l'amélioration du nombre de services, la composition obtenue par notre approche peut encore contenir des services redondants.

5.4 Conclustion

Dans ce dernier chapitre, nous avons montré l'évaluation de notre approche globale d'adaptation de services Web au contexte d'utilisation.
Les résultats de l'implémentation de notre intergiciel de la gestion de contexte sont encourageants pour les dispositifs à faible capacité et les réseaux à faible bande passante.

Concernant notre approche de composition de SW, les résultats de l'expérimentation que nous avons réalisée montrent que notre approche offre une solution qui répond toujours aux meilleures QoS avec une amélioration significative en termes de temps d'exécution ASC et de nombre de services du résultat.

La comparaison de nos travaux avec ceux de la littérature permet de valider nos résultats. Notre approche offre toujours une solution qui répond aux meilleures QoS avec une amélioration significative en termes de temps d'exécution de la composition et de nombre de services.

De plus, les preuves des théorèmes que nous avons présentées montrent la solidité et l'exactitude de notre contribution en termes d'amélioration au domaine de graphe de planification.

Tableau 5.1 Résultats de notre approche par rapport aux trois premiers gagnants de la compétition WSC2010.

		Qos RT	QoS TH	Min. Lev.	Min Serv.	Comp. Time
DS01	CAS	500	15000	3	5	78
	RUG	500	15000	3	10	188
	Tsinghua	500	15000	3	9	109
	Our approach	**500**	**15000**	**3**	7	**20**
DS02	CAS	1690	6000	6	20	94
	RUG	1690	6000	6	40	234
	Tsinghua	1690	6000	6	36	140
	Our approach	**1690**	**6000**	**6**	23	**23**
DS03	CAS	760	4000	3	10	78
	RUG	760	4000	3	11	234
	Tsinghua	760	4000	3	18	125
	Our approach	**760**	**4000**	**3**	**10**	**23**
DS04	CAS	1470	4000	5	73	156
	RUG	1470	4000	5	133	390
	Tsinghua	1470	4000	5	133	188
	Our approach	**1470**	**4000**	**5**	**42**	**56**
DS05	CAS	4070	4000	19	32	63
	RUG	4070	4000	19	4772	907
	Tsinghua	4070	4000	19	4772	531
	Our approach	**4070**	**4000**	**19**	**32**	**40**
DS06	CAS	1660	4000	9	30	93
	RUG	1660	4000	9	427	735
	Tsinghua	1660	4000	9	570	256
	Our approach	**1660**	**4000**	**9**	**22**	**30**
DS07	CAS	6780	4000	22	40	141
	RUG	-	-	-	-	-
	Tsinghua	3710	4000	16	7357	969
	Our approach	**3710**	**4000**	**16**	**31**	**93**
DS08	CAS	1800	3000	6	55	187
	RUG	-	-	-	-	-
	Tsinghua	1800	3000	5	248	531
	Our approach	**1800**	**3000**	**5**	**32**	**151**
DS09	CAS	3340	2000	15	49	125
	RUG	3340	1000	18	5103	2344
	Tsinghua	3340	2000	15	20239	1906
	Our approach	**3340**	**2000**	**15**	**37**	**106**
DS10	CAS	5500	1000	35	92	156
	RUG	-	-	-	-	-
	Tsinghua	-	-	-	-	-
	Our approach	**5500**	**1000**	**34**	**62**	**121**

CONCLUSION ET RECOMMANDATIONS

L'objectif de ce travail de doctorat était de traiter l'adaptation des services Web au contexte d'utilisation. Pour cela, nous avons proposé une approche qui se base sur des techniques clés comme les ontologies, agents et les techniques de planification. Notre approche comporte deux sous approches, à savoir la gestion sémantique du contexte et l'adaptation de la composition de service Web. Dans ce que suit, nous allons les résumer brièvement et présenter nos perspectives.

Le premier travail, qui rentre dans le cadre général des applications sensibles au contexte, vise la gestion du contexte dans un environnement ubiquitaire caractérisé par la dynamique et la distribution. Notre proposition est basée sur la notion d'ontologie qui apporte une représentation formelle et sémantique. Chose qui a contribué à la résolution du problème de l'hétérogénéité dans ce genre d'environnement. Aussi, nous avons proposé l'utilisation du journal de changements du contexte, qui garde les opérations de changements appliqués et minimise le coût de la communication. Grâce à la capacité des agents dans ce genre d'environnement, nous avons proposé également de manipuler ces différents éléments par des agents. Chose qui a apporté plus d'efficacité à notre intergiciel que les résultats des expérimentations de son implémentation sont clairement en faveur de l'utilisation de notre approche dans le cas des dispositifs et des réseaux à faible capacité.

Nous avons comme perspective de ce travail, l'étude d'effets des changements invoqués par le développeur sur le modèle du contexte c.-à-d. les changements de modélisation. De plus, des autres expérimentations seront effectuées afin de tester cette approche sur les différents types de dispositifs et avec des capteurs physiques.

Le deuxième travail s'inscrit dans le cadre de la composition automatique des services Web. L'objectif est d'améliorer le temps d'exécution de la composition dirigée par la qualité de services (QoS-aware service composition) tout en minimisant autant que possible le nombre de services dans la composition. Les techniques de planification AI, en particulier le graphe de planification, sont considérées comme les plus efficaces pour résoudre le problème ASC.

Pour minimiser le temps consacré à l'extraction de la solution, nous avons établi notre approche sur l'exploitation des informations de trace conservées pendant la construction du graphe de planification. De plus, pour récupérer le temps consacré à la garde des traces, nous avons proposé une amélioration de la structure de graphe de planification qui permet de minimiser le nombre d'éléments du graph (c'est-à-dire les propositions et les actions) et par conséquent de réduire le temps d'exécution nécessaire à sa construction. Pour plus de minimisation, nous employons notre algorithme de minimisation proposé. De plus, ces minimisations couplées à nos autres stratégies proposées (services et choix de solutions) pour la minimisation de nombre de services ont un effet positif sur le nombre de services de la solution. Par conséquent, cela minimise, voire élimine, tous les services redondants.

Les résultats de l'expérimentation que nous avons réalisée montrent que notre approche offre toujours une solution qui répond aux meilleures QoS avec une amélioration significative en termes de temps d'exécution de la composition et de nombre de services.

Ces résultats nous ont amenés à considérer que notre amélioration du graphe de planification pourrait être utile pour résoudre d'autres problèmes (autres que le problème de la composition de service Web) dans lesquels le graphe de planification peut être considéré comme une technique de solution.

Nous avons mené dans la description de notre approche toutes les définitions des concepts essentiels, théorèmes et les algorithmes nécessaires pour leur mise en œuvre.

Notre travail de l'adaptation de la composition des services a comme perspectives :
Premièrement, nous voulons étudier et évaluer notre approche sur tous les autres types de QoS, tout en notant que notre approche est proposée de manière qu'elle soit extensible à d'autres critères de QoS.
Deuxièmement, nous voulons améliorer notre approche pour qu'elle puisse éliminer plus de services redondants, tout en utilisant d'autres techniques d'optimisation.

Finalement, l'intégration de notre implémentation de notre approche comme un module (agent d'adaptation) dans notre intergiciel.

BIBLIOGRAPHIE

Abowd, G., Dey, A., Brown, P., Davies, N., Smith, M. & Steggles, P. (1999). Towards a better understanding of context and context-awareness. *International symposium on handheld and ubiquitous computing*, pp. 304–307.

Ahmed, B., Abdelouahed, G. & Kazar, O. (2017a). Semantic-based approach to context management in ubiquitous environment. *the 8th international conference on ambient systems, networks and technologies (ant 2017)*.

Ahmed, B., Okba, K. & Abdelouahed, G. (2017b). Automated web service composition algorithm based on improved planning graph. *Ijseia-international journal of software engineering and its applications*, 11(2), 65–96.

Amor, M., Fuentes, L. & Troya, J. M. (2003). Putting together web services and compositional software agents. *International conference on web engineering*, pp. 44–53.

Bachimont, B. (2000). Engagement sémantique et engagement ontologique : conception et réalisation d'ontologies en ingénierie des connaissances. *Ingénierie des connaissances : évolutions récentes et nouveaux défis*, 305–323.

Baron, M. (2011, Mars). Orchester les services web : Bpel [Document]. Repéré à http://www.w3.org/2004/Talks/0318-hh-2rws/slide5-0.html.

Belhanafi, N., Taconet, C. & Bernard, G. (2005). Camido, a context-aware middleware based on ontology meta-model. *Workshop on context awareness for proactive systems*, pp. 93–103.

Berbner, R., Spahn, M., Repp, N., Heckmann, O. & Steinmetz, R. (2006). Heuristics for qos-aware web service composition. *Web services, 2006. icws'06. international conference on*, pp. 72–82.

Bhiri, S., Godart, C. & Perrin, O. (2005). Reliable web services composition using a transactional approach. *e-technology, e-commerce and e-service, 2005. eee'05. proceedings. the 2005 ieee international conference on*, pp. 15–21.

Blake, M. B., Weise, T. & Bleul, S. (2010, 13–15,). Wsc-2010 : Web services composition and evaluation. *Proceedings of the ieee international conference on service-oriented computing and applications (soca'10)*, pp. 1–4. doi : 10.1109/SOCA.2010.5707190.

Bleul, S., Weise, T. & Geihs, K. (2009). The web service challenge-a review on semantic web service composition. *Electronic communications of the easst*, 17.

Blum, A. L. & Furst, M. L. (1997). Fast planning through planning graph analysis. *Artificial intelligence*, 90(1), 281–300.

Boudali, F. (2007). *Publication et découverte des web services pour le domaine du e-learning*. (Mémoire de maîtrise, INI, Algria).

Bouguettaya, A., Yu, Q., Liu, X. & Malik, Z. (2011). Service-centric framework for a digital government application. *Services computing, ieee transactions on*, 4(1), 3–16.

Boukhadra, A. (2011). *La composition dynamique des services web sémantiques a base d'alignement des ontologies owl-s»*. (Mémoire de maîtrise, Ecole national Supérieur d'Informatique, Algria).

Bouron, M. T. (1992). *Structures de communication et d'organisation pour la coopération dans un univers multi-agents*. (Thèse de doctorat, UNIVERSITE PARIS 6).

Broekstra, J., Klein, M., Decker, S., Fensel, D., Van Harmelen, F. & Horrocks, I. (2002). Enabling knowledge representation on the web by extending rdf schema. *Computer networks*, 39(5), 609–634.

Brusilovsky, P. (1998). Methods and techniques of adaptive hypermedia. Dans *Adaptive hypertext and hypermedia* (pp. 1–43). Springer.

Canfora, G., Di Penta, M., Esposito, R. & Villani, M. L. (2005). An approach for qos-aware service composition based on genetic algorithms. *Proceedings of the 7th annual conference on genetic and evolutionary computation*, pp. 1069–1075.

Cerami, E. (2002). *Web services essentials : distributed applications with xml-rpc, soap, uddi & wsdl*. " O'Reilly Media, Inc.".

CHAMPAGNE, R. (2014). cours mgl844 - architecture logicielle, "chapitre 4 comprendre les attributs de qualité". École de Technologie Supérieure (ÉTS).

Chen, H., Perich, F., Finin, T. & Joshi, A. (2004). Soupa : Standard ontology for ubiquitous and pervasive applications. *Mobile and ubiquitous systems : Networking and services, 2004. mobiquitous 2004. the first annual international conference on*, pp. 258–267.

Chen, M. (2015). *Qos-aware service composition and redundant service removal*. (Thèse de doctorat, Concordia University Montréal, Québec, Canada).

Chen, M. & Yan, Y. (2012). Redundant service removal in qos-aware service composition. *Web services (icws), 2012 ieee 19th international conference on*, pp. 431–439.

Chen, M. & Yan, Y. (2014). Qos-aware service composition over graphplan through graph reachability. *Services computing (scc), 2014 ieee international conference on*, pp. 544–551.

Chouchani, I. (2010). *Utilisation d'un algorithme génétique pour la composition de services web*. (Mémoire de maîtrise, Université du Québec à Montréal, Canada).

Conlan, O., Lewis, D., Higel, S., O'Sullivan, D. & Wade, V. (2003). Applying adaptive hypermedia techniques to semantic web service composition. *International workshop on adaptive hypermedia and adaptive web-based systems (ah 2003)*, pp. 700–709.

Costa, P. D., Almeida, J. P. A., Pires, L. F., Guizzardi, G. & van Sinderen, M. (2006). Towards conceptual foundations for context-aware applications. *Proc 3rd int ws model retriev context*.

Cui, L., Kumara, S. & Lee, D. (2011). Scenario analysis of web service composition based on multi-criteria mathematical goal programming. *Service science*, 3(4), 280–303.

David, B., Hugo, H., Francis, M., Eric, N., Michael, C., Chris, F. & David, O. (2004, February, 11). Web services architecture [Document]. Repéré à https://www.w3.org/TR/ws-arch/.

David Martin, e. a. (2004, November, 22). Owl-s : Semantic markup for web services [Document]. Repéré à https://www.w3.org/Submission/OWL-S/.

Devaraju, A. & Hoh, S. (2008). Ontology-based context modeling for user-centered context-aware services platform. *Information technology, 2008. itsim 2008. international symposium on*, 2, 1–7.

Dey, A. K. (2000). Enabling the use of context in interactive applications. *Chi'00 extended abstracts on human factors in computing systems*, pp. 79–80.

Dodani, M. H. (2004). From objects to services : A journey in search of component reuse nirvana. *Journal of object technology*, 3(8), 49–54.

Ejigu, D., Scuturici, M. & Brunie, L. (2007). An ontology-based approach to context modeling and reasoning in pervasive computing. *Pervasive computing and communications workshops, 2007. percom workshops' 07. fifth annual ieee international conference on*, pp. 14–19.

Erl, T. (2004). *Service-oriented architecture : a field guide to integrating xml and web services*. Prentice Hall PTR.

Erl, T. (2005). *Service-oriented architecture : concepts, technology, and design*. Pearson Education India.

Fensel, D. (2003). Ontologies : A Silver Bullet for Knowledge Management and Electronic Commerce. Secaucus. NJ, USA : Springer-Verlag New York, Inc.

Fensel, D. & Groenboom, R. (1997). Specifying knowledge-based systems with reusable components. *in proceedings of the 9th international conference on software engineering & knowledge engineering (seke-97*.

Ferber, J. (1995). *Les systèmes multi-agents : vers une intelligence collective*. InterEditions.

Ferry, N., Lavirotte, S., Rey, G. & Tigli, J.-Y. (2008). *Adaptation dynamique d'applications au contexte en informatique ambiante* (Rapport n°I3S/RR-2008-20-FR). Sophia Antipolis, France : Laboratoire I3S (Universite de Nice Sophia Antipolis/CNRS).

Finin, T., Fritzson, R., McKay, D. & McEntire, R. (1994). Kqml as an agent communication language. *Proceedings of the third international conference on information and knowledge management*, pp. 456–463.

Fouial, O. (2004). *Découverte et fourniture de services adaptatifs dans les environnements mobiles*. (Thèse de doctorat, Télécom ParisTech).

FÜRST, F. (2004). Opérationnalisation des ontologies : une méthode et un outil. *15e journées francophones d'ingénierie des connaissances (ic2004)*, pp. 199–210.

Fürst, F. (2004). *Contribution à l'ingénierie des ontologies : une méthode et un outil d'opérationalisation*. (Thèse de doctorat, Nantes).

Gaëlle, L. (2002). État de l'art ontologies et intégration/fusion d'ontologies [Document]. Repéré à http://casanuestra.free.fr/ontologie_fusion_lortal.pdf.

Ghallab, M., Nau, D. & Traverso, P. (2004). *Automated planning : theory & practice*. Elsevier.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2), 199–220.

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing ? *International journal of human-computer studies*, 43(5-6), 907–928.

Gu, T., Pung, H. K. & Zhang, D. Q. (2004). A middleware for building context-aware mobile services. *Vehicular technology conference, 2004. vtc 2004-spring. 2004 ieee 59th*, 5, 2656–2660.

Guarino, N. et al. (1998). Formal ontology and information systems. *Proceedings of fois*, 98(1998), 81–97.

Guermah, H., Fissaa, T., Hafiddi, H., Nassar, M. & Kriouile, A. (2014). An ontology oriented architecture for context aware services adaptation. *arxiv preprint arxiv :1404.3280*.

han, K. M., Bishop, J. & Baresi, L. (2007). *Survey and comparison of planning techniques for web services composition*.

Hernandez, N. (2005). *Ontologies de domaine pour la modélisation du contexte en recherche d'information*. (Thèse de doctorat, These de doctorat, Institut de Recherche en Informatique de Toulouse).

Hugo Haas, A. B. (2017, Mars, 03). Web services glossary [Document]. Repéré à https://www.w3.org/TR/ws-gloss/.

Jason Hunter, B. M. (2015). JDOM v2.0.6 API Specification. [Online ; accessed 26-August-2016].

Jeannette, F. N. C. (2006). Vers une integration des differentes approches de modelisation a base ontologique : application aux modèles plib et owl.

Jones, K. (2008). Building a context-aware service architecture. *Ibm developerworks*.

Kapitsaki, G. M., Kateros, D., Venieris, I. S. et al. (2008). Architecture for provision of context-aware web applications based on web services. *Personal, indoor and mobile radio communications, 2008. pimrc 2008. ieee 19th international symposium on*, pp. 1–5.

KARA, N. (2014). cours mgl825 - télématique et réseaux, "chapitre 08 : Applications". École de Technologie Supérieure (ÉTS).

Kara, N., El Barachi, M., El Bardai, A. & Alfandi, O. (2014). A new business model and architecture for context-aware applications provisioning in the cloud. *New technologies, mobility and security (ntms), 2014 6th international conference on*, pp. 1–5.

Keller, A. & Ludwig, H. (2003). The wsla framework : Specifying and monitoring service level agreements for web services. *Journal of network and systems management*, 11(1), 57–81.

Kil, H. (2010). *Efficient web service composition : From signature-level to behavioral description-level*. (Thèse de doctorat, The Pennsylvania State University The Graduate School).

Kil, H. & Nam, W. (2011). Anytime algorithm for qos web service composition. *Proceedings of the 20th international conference companion on world wide web*, pp. 71–72.

Kona, S., Bansal, A., Blake, M. B., Bleul, S. & Weise, T. (2009). Wsc-2009 : a quality of service-oriented web services challenge. *2009 ieee conference on commerce and enterprise computing*, pp. 487–490.

Kreger, H. et al. (2001). Web services conceptual architecture (wsca 1.0). *Ibm software group*, 5, 6–7.

la Vallée, U. M. (2017, Mars, 03). Le protocole soap [Document]. Repéré à http://igm.univ-mlv.fr/~dr/XPOSE2003/axis_seng/soap.html.

Labidi, S. & Lejouad, W. (2004). De l'intelligence artificielle aux systèmes multi-agents. Rapport de recherche Unité de recherche INRIA Sophia-Antoplois.

Lassila, O. & McGuinness, D. (2001). The role of frame-based representation on the semantic web. *Linköping electronic articles in computer and information science*, 6(5), 2001.

LaValle, S. M. (2006). *Planning algorithms*. Cambridge university press.

Lécué, F. (2008). *Composition de services web : Une approche basée liens sémantiques*. (Thèse de doctorat, Ecole Nationale Supérieure des Mines de Saint-Etienne).

Lemlouma, T. (2004). *Architecture de négociation et d'adaptation de services multimédia dans des environnements hétérogènes*. (Thèse de doctorat, Institut National Polytechnique de Grenoble-INPG).

Li, J., Yan, Y. & Lemire, D. (2014). Full solution indexing using database for qos-aware web service composition. *Services computing (scc), 2014 ieee international conference on*, pp. 99–106.

Lopez-Velasco, C. (2008). *Sélection et composition de services web pour la génération d'applications adaptées au contexte d'utilisation. français*. (Thèse de doctorat, THESE. Université Joseph-Fourier-Grenoble I).

Ludwig, H., Keller, A., Dan, A., King, R. P. & Franck, R. (2003). Web service level agreement (wsla) language specification. *Ibm corporation*, 815–824.

Luo, N., Yan, J., Liu, M. & Yang, S. (2006). Towards context-aware composition of web services. *Grid and cooperative computing, 2006. gcc 2006. fifth international conference*, pp. 494–499.

MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F., Metz, R. & Hamilton, B. A. (2006). Reference model for service oriented architecture 1.0. *Oasis standard*, 12, 18.

Mahfoud, S. (2011). *Interopérabilité sémantique des données échangées entre les services web, engagés dans une composition*. (Thèse de doctorat, Université de Boumerdes).

Mathieu, P. & Taquet, A. (2000). Une forme de négociation pour les systèmes multi-agents. *Jfiadsma*, pp. 133–147.

Mbao, M. (2007). *Distance sémantique et carte conceptuelle*. (Mémoire de maîtrise, Ecole des Mines d'Alès, Université Montpellier II, France).

Nijhuis, K.-H., Pires, L. F. & van Sinderen, M. (2007). Context-aware services for constrained mobile devices. *University of twente*.

Nwana, H. S. (1996). Software agents : An overview. *The knowledge engineering review*, 11(03), 205–244.

OASIS. (2002). Uddi version 2.04 api specification. [Online ; accessed 16-July-2016].

OASIS. (2007). Web Services Business Process Execution Language Version 2.0. [Online ; accessed 16-July-2016].

openclassrooms. (2017, Mars, 03). Les services web [Document]. Repéré à https://openclassrooms.com/courses/les-services-web.

Papazoglou, M. P., Traverso, P., Dustdar, S. & Leymann, F. (2008). Service-oriented computing : a research roadmap. *International journal of cooperative information systems*, 17(02), 223–255.

Papazoglou, M. P. (2003). Service-oriented computing : Concepts, characteristics and directions. *Web information systems engineering, 2003. wise 2003. proceedings of the fourth international conference on*, pp. 3–12.

Peer, J. (2005). *Web service composition as ai planning-a survey*.

PILLOU, J.-F. (2017, Mars, 03). Xml - introduction à xml [Document]. Repéré à http://www.commentcamarche.net/contents/1332-xml-introduction-a-xml.

Pistore, M., Traverso, P. & Bertoli, P. (2005). Automated composition of web services by planning in asynchronous domains. *Icaps*, 5, 2–11.

Poizat, P. & Yan, Y. (2010). Adaptive composition of conversational services through graph planning encoding. Dans *Leveraging Applications of Formal Methods, Verification, and Validation* (pp. 35–50). Springer.

Poslad, S. (2007). Specifying protocols for multi-agent systems interaction. *Acm transactions on autonomous and adaptive systems (taas)*, 2(4), 15.

Rada, R., Mili, H., Bicknell, E. & Blettner, M. (1989). Development and application of a metric on semantic nets. *Ieee transactions on systems, man, and cybernetics*, 19(1), 17–30.

Rampacek, S. (2006). *Sémantique, interactions et langages de description des services web complexes*. (Thèse de doctorat, Université de Savoie).

Razika, D. (2007). *Une architecture d'intégration des applications d'entreprise basée sur l'interopérabilité sémantique de l'ebxml et la mobilité des agents*. (Thèse de doctorat, Université de constantine, Algerie).

Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. *arxiv preprint cmp-lg/9511007*.

Rodriguez-Mier, P., Mucientes, M. & Lama, M. (2015). Hybrid optimization algorithm for large-scale qos-aware service composition.

Roman, D., Keller, U., Lausen, H., De Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C. & Fensel, D. (2005). Web service modeling ontology. *Applied ontology*, 1(1), 77–106.

Russell, S. J. (2010). *Artificial intelligence : a modern approach*.

Soukkarieh, B. (2010). *Technique de l'internet et ses langages : vers un système d'information web restituant des services web sensibles au contexte*. (Thèse de doctorat, Université de Toulouse, Université Toulouse III-Paul Sabatier).

Stojanovic, L. (2004). *Methods and tools for ontology evolution.* (Thèse de doctorat, Karlsruhe Institute of Technology).

Studer, R., Eriksson, H., Gennari, J., Tu, S., Fensel, D. & Musen, M. (1996). *Ontologies and the configuration of problem-solving methods*. AIFB, Univ.

Szyperski C.,Gruntz D., M. S. (2002). Component software : Beyond object -oriented programming.

Tamura, G., Villegas, N. M., Müller, H. A., Duchien, L. & Seinturier, L. (2013). Improving context-awareness in self-adaptation using the dynamico reference model. *Proceedings of the 8th international symposium on software engineering for adaptive and self-managing systems*, pp. 153–162.

TREMBLAY, G. (2007a). Une introduction aux services Web. LIRMM, Montpellier, France.

TREMBLAY, G. (2007b). Services Web : De l'orchestration à la chorégraphie. LIRMM, Montpellier, France.

Van Nguyen, T., Lim, W., Nguyen, H., Choi, D. & Lee, C. (2010). Context ontology implementation for smart home. *arxiv preprint arxiv :1007.1273*.

W3C. (2007a). Soap version 1.2 part 1 : Messaging framework (second edition). [Online ; accessed 16-July-2016].

W3C. (2007b). Web services description language (wsdl) version 2.0. [Online ; accessed 19-August-2016].

W3C. (2012). Web Ontology Language (OWL). [Online ; accessed 19-August-2016].

Wang, X. H., Zhang, D. Q., Gu, T. & Pung, H. K. (2004). Ontology based context modeling and reasoning using owl. *Pervasive computing and communications workshops, 2004. proceedings of the second ieee annual conference on*, pp. 18–22.

Weise, T., Blake, M. B. & Bleul, S. (2013). Semantic Web Service Composition : The Web Service Challenge Perspective. Dans Bouguettaya, A. & Sheng, Q. Z. (Éds.), *Web Services Foundations – Part 1 : Foundations of Web Services* (ch. 7, pp. 161–187). Secaucus, NJ, USA : Springer-Verlag New York, Inc. doi : 10.1007/978-1-4614-7518-7_7.

Welty, C. & Guarino, N. (2001). Supporting ontological analysis of taxonomic relationships. *Data & knowledge engineering*, 39(1), 51–74.

wikipedia. (2016a, juin, 22). Calcul des propositions, définition d'une proposition. Repéré à https://fr.wikipedia.org/wiki/Calcul_des_propositions.

wikipedia. (2016b, septembre, 17). Throughput. Repéré à https://fr.wikipedia.org/wiki/Throughput.

wikipedia. (2017, mars, 5). Feuilles de style en cascade. Repéré à https://fr.wikipedia.org/wiki/Feuilles_de_style_en_cascade.

Wooldridge, M. (2009). *An introduction to multiagent systems*. John Wiley & Sons.

Xiao, H., Zou, Y., Ng, J. & Nigul, L. (2010). An approach for context-aware service discovery and recommendation. *Web services (icws), 2010 ieee international conference on*, pp. 163–170.

Yan, Y., Xu, B. & Gu, Z. (2008). Automatic service composition using and/or graph. *E-commerce technology and the fifth ieee conference on enterprise computing, e-commerce and e-services, 2008 10th ieee conference on*, pp. 335–338.

Yan, Y. & Chen, M. (2015). Anytime qos-aware service composition over the graphplan. *Service oriented computing and applications*, 9(1), 1–19.

Yan, Y., Poizat, P. & Zhao, L. (2010). Repairing service compositions in a changing world. Dans *Software Engineering Research, Management and Applications 2010* (pp. 17–36). Springer.

Yan, Y., Chen, M. & Yang, Y. (2012). Anytime qos optimization over the plangraph for web service composition. *Proceedings of the 27th annual acm symposium on applied computing*, pp. 1968–1975.

Yu, T., Zhang, Y. & Lin, K.-J. (2007). Efficient algorithms for web services selection with end-to-end qos constraints. *Acm transactions on the web (tweb)*, 1(1), 6.

Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J. & Sheng, Q. Z. (2003). Quality driven web services composition. *Proceedings of the 12th international conference on world wide web*, pp. 411–421.

Zeng, L., Benatallah, B., Ngu, A. H., Dumas, M., Kalagnanam, J. & Chang, H. (2004). Qos-aware middleware for web services composition. *Ieee transactions on software engineering*, 30(5), 311–327.

Zheng, X. & Yan, Y. (2008). An efficient syntactic web service composition algorithm based on the planning graph model. *Web services, 2008. icws'08. ieee international conference on*, pp. 691–699.