Faculté des sciences exactes Département d'Informatique

Thèse pour l'obtention du diplôme de Doctorat en Sciences

(Spécialité Informatique)

L'apprentissage à base de cas pour la composition des Services Web

Option : Système d'Information et Technologies des Connaissances

Présenté par

Yamina HACHEMI

Devant le jury composé de :

Président :	**Pr Mimoun MALKI**	**ESI Sidi Bel Abbes**
Examinateurs :	**Pr Baghdad ATMANI**	**Université d'Oran**
	Pr Zakaria ElBERRICHI	**UDL Sidi Bel Abbes**
	Pr Amine ABDELMALEK	**Université de Saida**
	Dr Fatima DEBBAT	**Université de Mascara**
Directeur :	**Pr Sidi Mohammed BESLIMANE**	**ESI Sidi Bel Abbes**

Evolutionary Engineering and Distributed Information Systems Laboratory

Remerciements

Un travail de recherche ne se conduit jamais seul, et il m'est agréable de remercier ici tous ceux qui m'ont permis de le mener à bien.

Je remercie très chaleureusement mon directeur de thèse, Monsieur le Professeur BENSLIMANE Sidi Mohammed, qui, malgré ses nombreuses occupations, a accepté de prendre la direction de cette thèse en cours de route, transformant ainsi les difficultés rencontrées en une expérience enrichissante. Je lui suis également reconnaissante de m'avoir assuré un encadrement rigoureux tout au long de ces années. Monsieur BENSLIMANE a su diriger mes travaux avec beaucoup de disponibilité, de tact et d'intérêt. Il m'a toujours accordé généreusement le temps nécessaire pour partager avec moi ses idées et sa grande expérience. Pour tout ce que vous m'avez donné, je vous remercie très sincèrement.

Je suis grandement reconnaissante au Professeur, Monsieur MALKI Mimoun, d'avoir accepté de juger ce travail et d'en présider le jury de soutenance. Que vous soyez assuré de mon entière reconnaissance.

Les examinateurs qui ont accepté de siéger sur le jury de cette thèse doivent aussi trouver ici l'expression de ma reconnaissance, soit Messieurs ELBERRICHI Zakaria, ATMANI Baghdad, AMINE Abdelmalek et DEBBAT Fatima. Merci d'avoir accepté d'évaluer cette thèse et d'avoir contribué aux discussions lors de la soutenance.

Je ne saurais terminer sans souligner le soutien amical et chaleureux de mes collègues du laboratoire EEDIS pour l'appui scientifique et personnel, dont ils ont fait preuve tout au long de ces années de thèse.

Je dois beaucoup à ma famille et mes amis. Mes remerciements vont particulièrement à mon Grand-père, qui m'a toujours encouragé dans mes études. J'adresse des remerciements de même ordre à ma mère et mon père, qui m'ont constamment encouragée et soutenue tout au long de ces années. Je ne saurais passer sous silence l'apport inestimable des autres membres de ma famille (grand-mère, frères, sœur, oncles, etc.) qui m'ont soutenue, de près ou de loin durant ces études doctorales.Je ne saurais terminer sans souligner le soutien amical et chaleureux de ma sœur Ilhem de tous les jours qui m'a soutenue durant ce parcours doctoral.

Et enfin que tous ceux qui, de près ou de loin, ont apporté leur contribution à cette étude, trouvent ici l'expression de ma profonde gratitude.

Dédicaces

A ma mère et mon père.

A mes grands-parents.

A mes frères et sœurs et leurs familles.

A mon fiancé.

A mes amis.

A mes collègues.

Résumé

Les applications distribuées convergent rapidement vers l'adoption d'un paradigme basé sur les architectures orientées services (SOA), selon lequel une application est obtenue par la composition d'un ensemble de services. La sélection de services web est un processus indispensable dans la composition de service web. Mais cette tâche devient difficile vu le nombre important de services présents sur le web et qui offre des fonctionnalités similaires. Donc quel est le meilleur service ? Les préférences des utilisateurs sont la clé pour sélectionner les meilleurs services pour la composition.

Dans ce travail de thèse, nous avons proposé un modèle de composition de service web basé sur les préférences de l'utilisateur. Pour améliorer le processus de composition de services web, nous proposons de combiner deux techniques d'intelligence artificielle qui sont la planification et le raisonnement par mémorisation. Nous utilisons la méthode de planification basée sur les cas avec les préférences de l'utilisateur qui utilise les expériences réussies dans le passé pour résoudre des problèmes similaires.

Dans notre approche, nous intégrons les préférences des utilisateurs dans la phase de sélection, d'adaptation et de planification. Nos principales contributions sont une nouvelle méthode de remémoration de cas, un algorithme étendu d'adaptation et de planification avec les préférences des utilisateurs. Les résultats obtenus offrent plus d'une solution à l'utilisateur et considèrent les propriétés fonctionnelles et non-fonctionnelles des utilisateurs.

Mots Clés : Composition de service Web ; planification basée sur les cas ; Préférences de l'utilisateur, planification avec préférences.

Abstract

Distributed applications converge quickly towards the adoption of a paradigm based on service-oriented architectures (SOA), in which an application is obtained by the composition of a set of services. Web service selection is an indispensable process for web service composition. However, it became a difficult task as many web services are increased on the web and mostly they offer similar functionalities, which service will be the best. User preferences are the key to select and retain only the best services for the composition.

In this work, we have proposed a web service composition model based on user preferences. To improve the process of web service composition we propose to combine two techniques of artificial intelligence that are planning and reasoning by memorization. In this work, we use the case-based planning approach with user preferences, which uses successful experiences of the past to solve similar problems.

In our approach we integrate user preferences in the phase of selection, adaptation and planning. Our main contributions are a new method of case retrieval, an extended algorithm of adaptation and planning with user preferences. Results obtained offer more than a solution to the user and taking both functional and non-functional requirements.

Keywords : Web service composition ; case-based planning ; User Preferences ; planning with preferences.

Liste des tableaux

Table des figures

Table des matières

Chapitre 1

Introduction générale

Sommaire

1.1 Introduction

Les architectures orientées services et particulièrement, les services web permettent l'accès, la découverte et l'utilisation des applications présentes sur le web en utilisant les standards tels que XML[XML, 1998], SOAP[Mitra et al., 2007], WSDL[Chinnici et al., 2007], UDDI[Bellwood et al, 2002] et les protocoles de transports d'Internet.

La réutilisation et l'interopérabilité sont au cœur du paradigme des services web. Cette technologie permet une réutilisation et une interopérabilité transparente des composants web qui facilite le développement et l'intégration rapide des applications. Néanmoins, les technologies de services web comme WSDL et UDDI n'ont pas de sémantique nécessaire pour les machines pour qu'elles puissent utiliser d'une façon autonome cette technologie. Une solution pour ce manque de sémantique est le but du web sémantique. L'objectif est d'apporter une structure au web existant à travers des annotations sémantiques en utilisant des langages

comme RDF[Klyne et al., 2004] et OWL[McGuiness et al., 2004]. Ainsi, il sera possible aux machines et aux utilisateurs de collaborer d'une manière plus efficace. Plusieurs langages de services web sémantiques sont développés comme OWL-S[Martin et al., 2007] et WSMO[Fensel et al., 2002] de telle sorte que des agents logiciels peuvent utiliser les annotations sémantiques pour faire face de manière plus efficace à la découverte, la composition, l'invocation et l'exécution de service Web.

Si un seul service ne peut satisfaire les fonctionnalités attendues par l'utilisateur, une composition de plusieurs services existant ensemble s'avère nécessaire dans le but de répondre à la requête. L'objectif de la composition est de combiner les fonctionnalités de plusieurs services web dans un processus métier afin de répondre aux applications complexes qu'un seul service ne peut le faire. Toutefois, la composition de services est loin d'être une tâche triviale.

Avec la prolifération d'Internet et la large acceptation du commerce électronique, un nombre croissant de services Web est offert sur le net. En même temps, la découverte et la composition de services Web a gagné beaucoup de terrain comme un moyen de soutenir l'automatisation des processus d'affaires. Notre recherche est motivée par la nécessité de faciliter et personnaliser les processus de découverte et de composition à travers des compositions de services Web réutilisables.

La composition de services web est une tâche difficile et complexe, et dépasse la capacité humaine à faire face à tous l'ensemble du processus de la composition manuellement. Un des facteurs de la complexité est de trouver des services appropriés pour la composition. Néanmoins, la composition de services web est généralement basée sur les propriétés fonctionnelles seulement négligeant un point très important qui est les préférences des utilisateurs. Il existe dans la littérature un nombre de techniques de composition classé sous diverse forme. Il existe des solutions manuelles, semi-automatiques et automatiques utilisant les techniques de l'intelligence artificielle.

Le problème avec la plupart des techniques de composition est que :

1. Les approches essayent de résoudre le problème de la composition par faire une composition à partir de zéro tout en négligeant la réutilisation ou l'adaptation de compositions ou des parties de compositions déjà existantes.
2. Les études considèrent que l'utilisateur connait exactement ce qu'il veut et comment l'obtenir.
3. Composer des services Web par l'intermédiaire d'interfaces de services concrets conduit à des compositions fortement couplés dans lesquelles chaque service impliqué dans la chaîne est lié à une instance de service Web.

Cependant, la prolifération des services web présents sur le net durant ces dernières années a engendré un champ de recherche de services web fort étendu. Les préférences des utilisateurs sont utilisées pour la sélection de services appropriés. Par exemple dans la planification d'un voyage, il existe plusieurs compositions de services web qui permettent à l'utilisateur de voyager d'une source à une destination, mais la planification d'un tel voyage doit prendre en considération les préférences des utilisateurs comme le coût du voyage, des préférences des compagnies aériennes ou des hôtels, le temps et la date du voyage.

En se basant sur l'hypothèse que le monde est régulier, donc des problèmes similaires ont des solutions similaires, il serait aisé d'adapter des solutions à des problèmes similaires qui deviendraient un point de départ pour la résolution de nouveaux problèmes. La planification à partir de cas est une méthode de résolution de problème qui utilise une bibliothèque de cas, ou un cas est associé à un problème et une description du but avec un plan qui résout le problème. Dans des situations similaires, RàPC (Raisonnement à Partir de Cas)[Schank, 1982] permet de bénéficier des expériences de planification déjà résolues en réutilisant les plans trouvés.

Dans notre travail, on définit une nouvelle approche pour trouver les meilleures compositions de services web en utilisant la planification à partir de cas et en se basant sur les préférences des utilisateurs. L'approche proposée diffère des autres travaux dans plusieurs points. L'approche inclut l'aspect non-fonctionnel représenté

par les préférences des utilisateurs tout au long du processus de composition et du RàPC. Pour cela, un algorithme qui permet de déterminer les meilleurs services web qui peuvent être utilisés dans la composition est donné. On génère le plan de composition pour une nouvelle requête en cherchant tous les plans possibles dans la base de cas. Le nouveau plan généré et le problème peuvent être ajoutés comme un cas dans la base des cas pour une utilisation ultérieure. On propose de sélectionner plusieurs plans en utilisant un algorithme de remémoration basé sur deux phases et utilisant les préférences des utilisateurs.

1.2 Problématique

Si la réutilisation de service est basée sur des services concrets, cela mènera à effectuer des changements sur le flux de données qui vont de quelques modifications des liaisons, à refaire toute la conception des parties de la description du flux de travail. Par conséquent, les services devraient être interprétés à un niveau abstrait pour faciliter leurs compositions et permettre à plus de généralisation et à un niveau élevé de réutilisation.

La réutilisation des compositions apporte avec elle un ensemble d'autres questions, qui comprennent la façon dont ces compositions sont générées, stockées et récupérées. Nous devons donc fournir des réponses aux questions suivantes :

- Comment représenter les données de composition ?
- Comment le système va-t-il effectuer ses recherches ?
- Quelles sont les meilleures solutions qui vont être présentées à l'utilisateur ?
- Comment le système adaptera-t-il les solutions trouvées ?
- Comment offrir une meilleure composition qui répond aux attentes de l'utilisateur ?

Dans la section suivante, nous allons présenter un aperçu de la solution que nous introduisons pour fournir des réponses à ces questions. Notre solution sera basée sur les points de motivations suivantes :

1. La complexité de la tâche de composition nous conduit à la réutilisation des compositions déjà faites. Cette réutilisation a l'avantage de ne pas commencer à partir de zéro à chaque fois qu'une nouvelle fonctionnalité est nécessaire.
2. Pour la réutilisation efficace, un niveau d'abstraction plus élevé doit être considéré. Cela permettrait de généraliser les concepts de service et de ne pas faire de liaisons avec les instances de service spécifiques.
3. La personnalisation des compositions peut être atteinte en identifiant d'abord plus clairement les besoins et exigences de l'utilisateur, puis en utilisant la réutilisation et l'adaptation de ces compositions passées en fonction de ces besoins.
4. Les compositions peuvent être exécutées avec des services concrets.

1.3 Contributions

Dans notre approche, nous avons abordé les problèmes soulignés ci-dessus et donc notre objectif principal est de mettre l'utilisateur (développeur ou autre) dans une situation où il peut utiliser les expériences passées.

Cette approche que nous avons appelée CBP-P4WSC (Case-Based Planning with Preferences for Web Service Composition) est similaire à celle adoptée dans Feng [Feng , 2009],Lee [Lee et al., 2010], Henni [Henni et al., 2013] et Torres [Torres et al., 2014], qui bénéficient de la réutilisation des compositions passées dans leurs travaux dans le cadre de la composition. Cependant, dans ces méthodes, il est supposé que les solutions prennent en considération uniquement l'aspect fonctionnel et négligent l'aspect non-fonctionnel.

Pour rendre ce processus de réutilisation efficace, nous avons d'abord envisagé le raisonnement à partir de cas RàPC, qui est prêt pour le stockage, la réutilisation et l'adaptation des expériences passées en vue de répondre à des problèmes actuels. Pour la composition, nous avons opté pour la planification qui est une technique

de l'intelligence artificielle donnant des compositions automatiques. Cependant, un autre point très important est omis dans ces compositions, il s'agit de l'aspect non-fonctionnel où les exigences de l'utilisateur ont été exprimées.

Pour résumer, le but de ce travail est de présenter une structure qui permet la personnalisation de la sélection et de la composition de services à travers la réutilisation des connaissances passées de composition. Ainsi, nous essayons d'encoder et de stocker des expériences de compositions qui pourraient ensuite être récupérées, réutilisées et adaptées grâce à des techniques de recherche, de sélection, d'adaptation et de composition.

Pour la première phase appelée transformation, nous avons transformé la requête de l'utilisateur en un problème de composition. Dans notre cas, le problème de composition est traduit en un problème de planification basé sur les propriétés fonctionnelles et des préférences de l'utilisateur. Dans la deuxième phase qui est la remémoration ou la recherche de solutions similaires, étant donné un nouveau problème, le système sélectionne les meilleures solutions pour répondre au problème de l'utilisateur. La troisième phase adapte les solutions trouvées selon les exigences de l'utilisateur. Pour être en mesure d'utiliser les cas dans notre système, nous avons choisi d'utiliser une représentation composée de plusieurs parties selon les besoins du système.

Le processus de remémoration de cas est divisé en deux parties : la recherche et la sélection et il est dirigé par différentes mesures de similarité. Plusieurs phases peuvent être déclenchées selon le résultat obtenu. Si la solution trouvée correspond exactement à la demande de l'utilisateur, le problème est donc résolu. Mais ce résultat est rare, dans la plupart des situations une adaptation est exigée. Il existe un autre cas où le système ne trouve pas de solution, dans ce cas une planification avec préférence est utilisée.

Nos propositions s'organisent autour des objectifs permettant d'atteindre des

compositions qui répondent aux attentes des utilisateurs. Notre travail est concrétisé par une méthode de sélection et de composition de services Web sémantiques basée sur la planification à partir de cas. Son originalité réside dans son contour de contributions qui couvre un ensemble d'aspects visant à remédier aux limites des approches existantes. Dans ce contexte, nos contributions s'articulent autour de ces trois points :

- Modèle de cas : notre approche utilise un modèle de description de cas qui couvre en plus des aspects fonctionnels des services web, les aspects non-fonctionnels des services. Ce qui nous permet de trouver non seulement les services Web répondant à une requête, mais aussi d'en sélectionner les meilleurs en prenant en considération des préférences de l'utilisateur.
- La remémoration : notre approche retrouve tous les cas similaires au problème en suivant un processus de recherche qui se base sur deux étapes. D'abord, la recherche qui extrait toutes les solutions dont le problème (une partie du problème) est similaire aux problèmes déjà résolus. Ensuite, la sélection qui se base sur les préférences des utilisateurs pour choisir les meilleures solutions.
- L'adaptation des solutions trouvées en utilisant une méthode de fusion de solutions.

1.4 Organisation du document

Cette thèse est organisée comme suit :

Chapitre 2 Présente un ensemble des concepts, techniques et outils relatifs à nos travaux. Nous introduisons les concepts de services web et de web sémantique, et nous présentons la manière par laquelle le web sémantique a été intégré dans la communauté des services web. Nous présenterons l'utilité des services web, ainsi que le problème de leur composition.

Chapitre 3 Ce chapitre introduit les principes du paradigme de Raisonnement à

Partir de Cas. Il présente le carré d'analogie dirigeant le RàPC et décrit son cycle de référence. Quelques techniques de recherche de similarité y sont également exposées et on abordera par la suite la planification et son application dans le domaine du RàPC.

Chapitre 4 Dans ce chapitre, on expose les approches fondées sur les techniques de raisonnement à partir de cas pour la résolution du problème de composition. Nous analysons des approches de composition de services Web en vue d'identifier les mécanismes qui font leurs points forts, et qui par la suite, pourront servir de guide pour déterminer les fondements d'une approche de composition optimisée. Une étude comparative entre les différentes approches est établie pour positionner notre travail.

Chapitre 5 Dans ce chapitre, nous décrivons les fondements de notre approche de composition de services Web. Nous commençons par élucider l'ensemble des critères retenus qui délimitent le contour de notre contribution. Nous y exposons ensuite les mécanismes et les fondements de notre approche CBP-P4WSC, tout en suivant le cycle du RàPC.

Chapitre 6 Est consacré à l'expérimentation et à l'évaluation de notre système. Nous y présentons les résultats d'une expérimentation visant à valider l'approche adoptée et à tester les algorithmes conçus.

Chapitre 7 Ce dernier chapitre conclut notre rapport et présente les perspectives de nos travaux.

Chapitre 2

La composition des Services Web

Sommaire

2.1 Introduction

A une époque où la mondialisation s'impose comme cadre de fonctionnement pour les entreprises de toute taille, l'intégration des systèmes d'information de diverses entreprises, en vue de la constitution de l'entreprise étendue, semble être un enjeu technologique et économique majeur. Les services web se présentent donc comme des technologies essentielles dans ce nouveau modèle économique. Ils permettent d'intégrer la dimension d'Internet et la standardisation des échanges imposée par ce nouveau contexte. Ils contribuent à la mutualisation des moyens et des ressources lors de la mise en place des entreprises multi-canaux, multi-partenaires, multi-filiales et facilitent, grâce à l'utilisation des standards, l'intégration de services fournis par les divers systèmes d'information interconnectés via Internet. Les services web ont été conçus pour faciliter les échanges de données, mais aussi l'accès aux applications au sein des entreprises et surtout entre les entreprises. Dans ce chapitre, nous présentons les services web, en citant l'apparition de service, suivie de la définition de l'architecture orientée service et le web sémantique puis nous traitons la composition des services web.

2.2 Objet, Composant et Service

L'évolution des langages de programmation a amené de nouveaux outils aidant à la conceptualisation des problèmes en informatique. En effet, l'avènement de l'orientée objet a facilité l'abstraction du problème à résoudre en fonction des données du problème lui-même (classes et objets). L'apparition de la programmation orientée objet a donné lieu à de nouvelles technologies de distribution des applications [Wolfgang et al., 2001] telles que RMI et CORBA. L'objectif principal des modèles objet est d'améliorer la modélisation d'une application, et d'optimiser la réutilisation du code produit. Cependant, l'intégration d'entités logicielles existantes peut s'avérer difficile si leur modèle d'exécution est incompatible avec le modèle imposé

par le langage objet choisi pour le développement de nouvelles entités. Par ailleurs, les modèles et les langages objets ne sont pas, en général, adaptés à la description des schémas de coordination et de communication complexes [Marvie et al, 2000].

Pour palier aux défauts de l'approche objet, l'approche composant est apparue. Cette approche est fondée sur des techniques et des langages de construction des applications qui intègrent, d'une manière homogène, des entités logicielles provenant de diverses sources. Un composant est une boite noire, communiquant avec l'extérieur à travers une interface dédiée, permettant la gestion du déploiement, de la persistance, etc [Michael, 2002]. En plus du concept d'objet, le composant se caractérise par la notion de déploiement, qui gère son cycle de vie de l'installation à l'instanciation. Cette notion permet aux développeurs de se focaliser sur la logique métier du composant, et délègue la gestion des propriétés non-fonctionnelles à l'environnement d'exécution l'hébergeant. L'enjeu est de faciliter la production de logiciels fiables, maintenables, évolutifs et toujours plus complexes. L'un des maîtres mots est la réutilisation par la disponibilité d'ingrédients logiciels, facilement composables et adaptables. La distribution de composants fait naître de nouvelles difficultés qu'il convient de gérer efficacement afin de préserver la souplesse et de garantir l'évolutivité du système d'information.

D'une part, l'interdépendance de composants distribués diminue la maintenabilité et l'évolutivité du système. Pour préserver son efficacité, une architecture distribuée doit minimiser l'interdépendance entre chacun de ces composants qui risque de provoquer des dysfonctionnements en cascade dont il est souvent complexe de détecter la cause et de déterminer précisément l'origine. D'autre part, la préservation de la qualité de service du système est une lourde tâche. Ces difficultés imposent la nécessité d'une architecture plus flexible ou les composants sont réellement indépendants et autonomes, le tout permettant de déployer plus rapidement de nouvelles applications. D'où l'apparition de l'architecture orientée service [Michael, 2002]. De façon similaire aux approches par objet ou par composant, l'approche service cherche à

fournir un niveau d'abstraction encore supérieur, en encapsulant des fonctionnalités et en permettant la réutilisation de services déjà existants.

2.3 Architecture orientée services SOA

SOA est un paradigme largement adopté fournissant un ensemble de méthodes pour le développement et l'intégration de systèmes dont les fonctionnalités sont développées sous forme de services interopérables et indépendants. Généralement, le paradigme de l'architecture orientée service est basé sur un ensemble de principes qu'il faut respecter [Huhns et al., 2005], [Erl, 2007] :

Couplage faible Le couplage est le niveau d'interaction entre deux ou plusieurs composants logiciels, deux composants sont couplés s'ils échangent de l'information. Ce couplage est fort s'ils échangent beaucoup d'information et il est faible dans le cas contraire. Dans une architecture SOA, le couplage entre les applications clientes et les services doit être faible. Vu que la communication avec les services Web est réalisée via des messages décrits par le standard XML caractérisé par sa généricité et son haut niveau d'abstraction, les services Web permettent la coopération d'applications tout en garantissant un faible taux de couplage. Par conséquent, il est possible de modifier un service sans briser sa compatibilité avec les autres services composant l'application.

Interopérabilité L'interopérabilité permet à des applications écrites dans des langages de programmation différents et s'exécutant sur des plateformes différentes de communiquer entre elles. En manipulant différents standards que ce soit XML ou les protocoles d'Internet, les services Web garantissent un haut niveau d'interopérabilité des applications et ceci indépendamment des plateformes sur lesquelles elles sont déployées et des langages de programmation dans lesquels elles sont écrites. Ainsi, en s'appuyant sur un format d'échange

de messages standard et sur l'ubiquité de l'infrastructure d'Internet, l'interopérabilité est donc une caractéristique intrinsèque aux services Web.

Réutilisabilité L'avantage de la réutilisation est qu'elle permet de réduire les coûts de développement en réutilisant des composants déjà existants. Dans le cas de l'approche service Web, l'objectif de la séparation des opérations en services autonomes est en effet pour promouvoir leur réutilisation. Ainsi, lorsqu'un client définit ses exigences, il est généralement possible de réutiliser des services déjà existant pour satisfaire une partie des exigences. Ceci facilite la maintenance de l'application et permet un gain de temps considérable.

Découverte Est une étape importante qui permet la réutilisation des services. En effet, il faudra être en mesure de trouver un service afin de pouvoir en faire usage. L'approche services Web tend à diminuer autant que possible l'intervention humaine en vue de permettre une découverte des services automatique. En effet, pour réaliser son application, un développeur peut simplement interroger un moteur de recherche de services afin de trouver le service adéquat.

Composition Des collections de services peuvent être coordonnées et assemblées afin de former une composition de services. Cette possibilité de construire de nouveaux systèmes à partir de services existants constitue un des avantages de l'AOS. La composition permet la réutilisation des services web, après la découverte, il faut être en mesure de composer ces derniers en exploitant les technologies offertes par Internet et en utilisant un ensemble de standards pour la composition.

L'approche la plus populaire pour mettre en œuvre une architecture SOA est l'utilisation de services Web.

2.4 Service web

Un service Web est défini [W3C, 2004] comme *un système logiciel conçu pour soutenir l'interaction interopérable machine-to-machine sur un réseau.* Il dispose d'une interface décrite dans un format exploitable par des machines. Autres systèmes interagissent avec les services Web de la manière prescrite par sa description en utilisant des messages SOAP [Mitra et al., 2007], généralement transmis via le protocole HTTP avec une sérialisation XML en conjonction avec d'autres normes liées au Web. Cette définition met en évidence les principales technologies utilisées au sein de l'Architecture Service Web, tels que WSDL [Chinnici et al., 2007], protocole simple Object Access (SOAP), HTTP et Extensible Markup Language (XML). En outre, elle implique UDDI [Bellwood et al, 2002].

2.4.1 Architecture des services Web

L'architecture SOA permet aux fournisseurs de services de publier leurs services et aux consommateurs de services de les découvrir. Elle facilite le développement d'applications en combinant les services faiblement couplés à travers des entités indépendantes (ou organisations). L'architecture générale de la SOA est, comme indiqué dans la figure 2.1 .

Comme l'illustre la figure 2.1, l'architecture d'un service web est composée de trois rôles : le fournisseur de service, l'annuaire de services et le client ou utilisateur du service. Le fournisseur de services crée un service et fournit sa description fonctionnelle dans un annuaire commun. Le fournisseur est chargé de publier son service en fournissant la description au format WSDL. Les consommateurs de services peuvent effectuer des recherches dans cet annuaire pour retrouver leurs services appropriés et récupère la description du service. Ils peuvent ensuite invoquer ces services via un message SOAP.

La figure 2.2 présente l'utilisation des technologies de web services par les acteurs

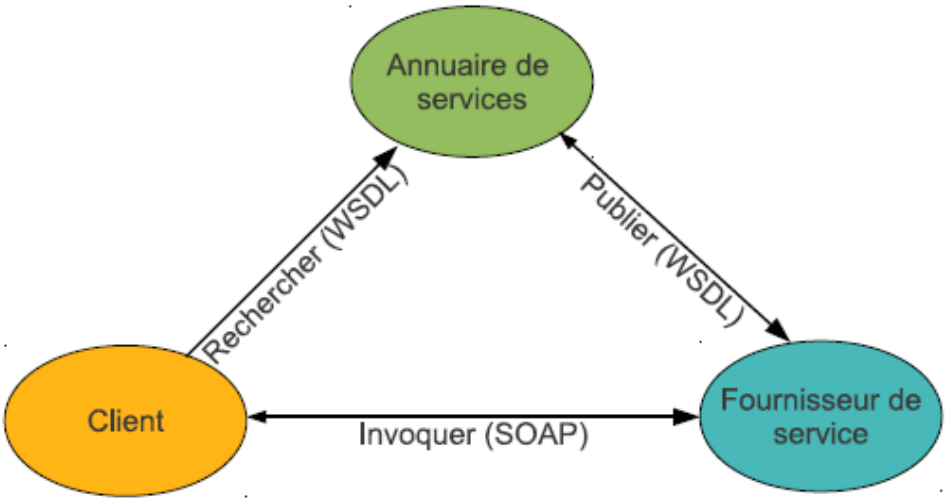

FIGURE 2.1: L'architecture d'un service Web.

fournisseur et demandeur :

- Le fournisseur de service publie ses web services sur l'annuaire en utilisant UDDI.
- Le demandeur recherche un web services avec les caractéristiques X, Z et Y.
- L'annuaire trouve un service avec les caractéristiques X, Z et Y, et envoie les informations sur le fournisseur du service et sur le service.
- Le demandeur demande le contrat du service du fournisseur.
- Le fournisseur envoie le contrat du service (WSDL).
- Le demandeur de service appelle le web services selon son contrat (appel en SOAP).
- Le web services retourne le résultat de l'appel (SOAP).

L'originalité de l'infrastructure des services web consiste à les mettre en place en se basant exclusivement sur les protocoles d'Internet tels que http et les formats standards d'échange tels que XML. L'infrastructure des services web s'est concrétisée autour de trois spécifications considérées comme des standards, à savoir SOAP, UDDI et WSDL [Chinnici et al., 2007]. Dans la suite de cette partie, nous allons présenter UDDI, WSDL et SOAP qui constituent les langages de base des services

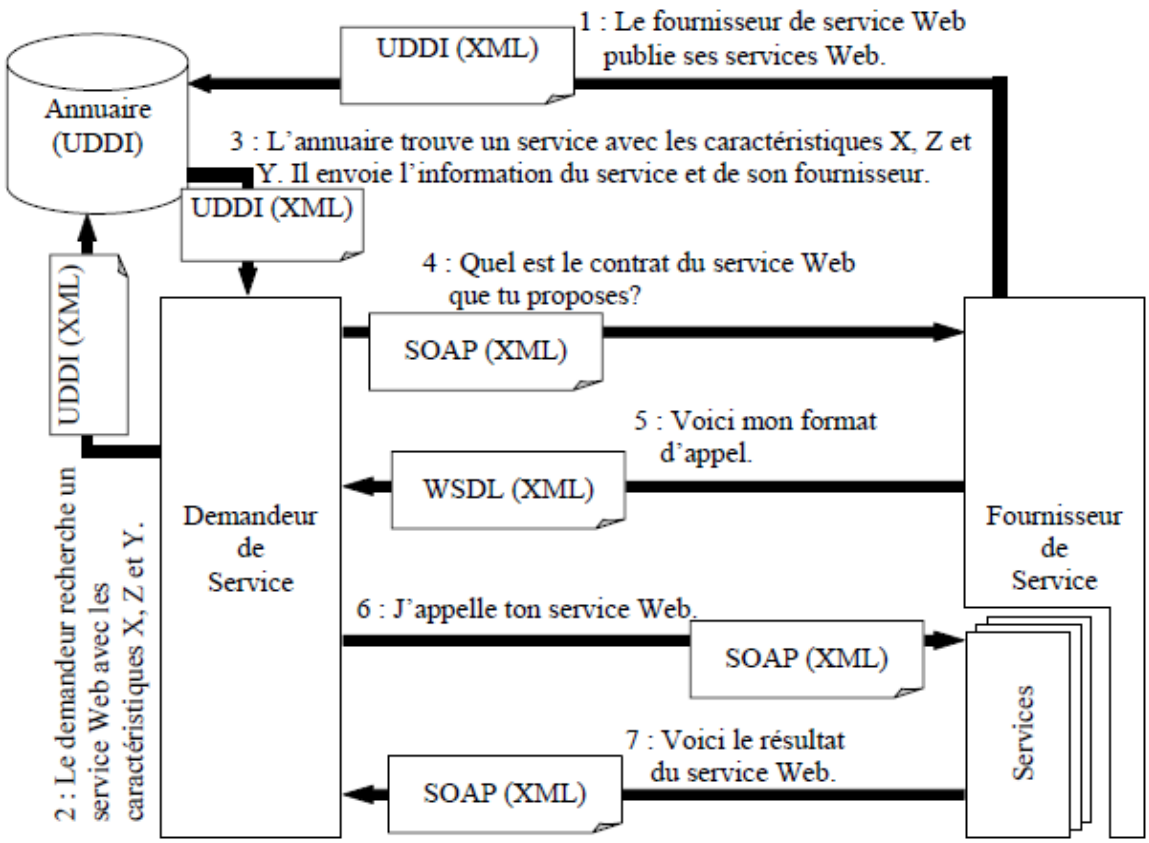

FIGURE 2.2: L'utilisation des technologies du web services par les acteurs fournisseur et demandeur.

Web.

2.4.1.1 Découverte UDDI

UDDI [Bellwood et al, 2002](Universal Description, Discovery and Integration) est une spécification technique sponsorisée par OASIS [OASIS], et il est le résultat d'un accord interindustriel entre plusieurs organisations telles que Microsoft, Ariba, IBM, etc. Il offre un mécanisme de registre distribué de services permettant leur publication et leur découverte. Le registre se compose de trois parties : les pages blanches, pages jaunes et pages vertes. Les données stockées dans l'UDDI sont structurées en XML :

Pages blanches : Comprennent des descriptions générales sur les fournisseurs de

Services. Nous y retrouvons donc des informations comme le nom de l'entreprise, ses coordonnées, la description de l'entreprise mais également l'ensemble de ses identifiants.

Pages jaunes : Comportent des descriptions détaillées sur les fournisseurs de services catalogués dans les pages blanches de façon à classer les entreprises et les services par secteurs d'activités. De plus, elles recensent les services Web de chacune des entreprises sous le standard WSDL.

Pages vertes : Fournissent des informations techniques précises sur les services fournis. Typiquement, on indiquera dans ces informations les adresses Web des services et les moyens d'y accéder. Ces informations incluent la description du service, du processus de son utilisation et des protocoles utilisés pour son invocation.

2.4.1.2 Description WSDL

WSDL (Web Service Description Language) est un langage de description de services Web basé sur XML proposé par W3C [Chinnici et al., 2007].

Le W3C a défini notamment les catégories d'informations à prendre en compte dans la description d'un service Web. Les éléments décrits dans WSDL sont principalement les opérations proposées par le service Web, les données et messages échangés lors de l'appel d'une opération, le protocole de communication et les ports d'accès au service. Le standard WSDL offre une description sur deux niveaux, abstrait et concret. Le niveau abstrait est utilisé principalement lors du processus de sélection tandis que le niveau concret est plutôt utilisé lors de l'invocation des opérations du service Web.

La partie abstraite : Décrit les messages et les opérations disponibles via les éléments suivants :

Les types de données : Informe sur les structures de données des paramètres

utilisés par le service. Il est optionnel et un seul autorisé. La balise utilisée est <types>).

Les messages : Encapsule les données véhiculées pour l'opération invoquée. Pour chaque opération du service, il existe deux éléments le premier correspond à la requête tandis que le second correspond à la réponse. La balise est <message> et plusieurs messages sont autorisés.

Les opérations : composé d'une ou plusieurs opérations décrites par des éléments <operation>. Chaque opération possède un nom,0 ou plusieurs messages en entrée et 0 ou plusieurs messages de sortie ou d'erreur.<portType> (plusieurs autorisés) .

La partie concrète :

Décrit le protocole et le type d'encodage à utiliser pour les messages nécessaire pour invoquer un service Web particulier. Les principales informations décrites au niveau concret sont le protocole de communication et le service.

Le protocole de communication : définit les protocoles de communication utilisés pour l'invocation du service Web. Cette définition permet d'établir le lien, d'une part, entre le document et les messages SOAP et d'autre part, entre les messages SOAP et les opérations invoquées. Plusieurs protocole de communication sont autorisés et la balise utilisée est <binding>.

Service : L'accès au service est défini par une collection de ports d'accès regroupés dans un élément XML noté "service". Ces ports représentent les adresses URI (Uniform Resource Identifier) du service. Ainsi, un même service Web peut être accessible depuis plusieurs ports

2.4.1.3 Transport SOAP

SOAP (Simple Object Access Protocol) [Mitra et al., 2007] est un protocole pour l'échange d'informations structurées avec les services Web, recommandé par le W3C

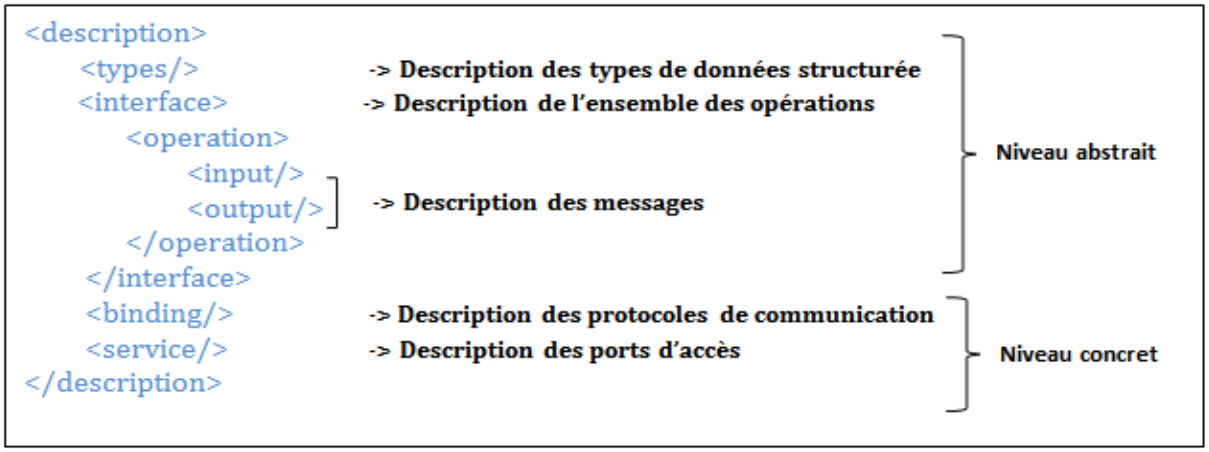

FIGURE 2.3: Structure générale d'un document WSDL 2.0.

[w3c].Il utilise principalement les protocoles HTTP (Hyper-Text Tranfer Protocol) et SMTP (Simple Mail Transfer Protocol) pour le transport de messages. SOAP se base sur le standard XML pour encoder les données. Par conséquent, il profite des avantages de généricité, d'abstraction et de portabilité qu'offre ce standard pour la normalisation et la structuration des données. SOAP forme la couche inférieure de la pile des protocoles des services Web, fournissant un framework pour l'échange de messages sur lequel les services Web peuvent se baser.

Un message SOAP est structuré en trois parties : un entête et un corps à l'intérieur d'une enveloppe. L'entête est facultatif et il apporte des données supplémentaires au message SOAP. Le corps renferme, du côté client, l'opération du service invoquée ainsi que des valeurs des paramètres nécessaires cette invocation, et du côté service, le résultat de l'exécution de l'opération invoquée.

FIGURE 2.4: Structure d'un message SOAP.

2.4.2 Propriétés des services Web

Les propriétés que possède un service Web sont diverses, mais on peut les grouper en propriétés décrivant l'aspect fonctionnel du service web et d'autres définissant l'aspect non-fonctionnel [Sheth, 2003]. Chacun de ces aspects fournit un type très spécifique d'information pour la description de service.

2.4.2.1 Propriétés fonctionnelles

L'aspect fonctionnel fournit des informations sur ce que le service peut fournir à ses clients en termes de fonctionnalités, lors de son invocation. Tandis que l'aspect non-fonctionnel, fournit des informations sur tous les paramètres décrivant le comportement ou l'utilisation du service tels que la qualité de service, la fiabilité, la performance, la sécurité, ou encore le prix [Chung et al., 1999].

Les services sont associés à quatre propriétés (les entrées, les sorties, les préconditions et les effets), appelées propriétés fonctionnelles des services.

2.4.2.2 Propriétés non-fonctionnelles

Les services sont également associés à leurs propriétés non-fonctionnelles. Ces propriétés sont souvent utilisées pour décrire les caractéristiques du service de manière à faciliter leur découverte et sélection. Ce qui suit est une liste possible des propriétés non fonctionnelles : nom de service, l'auteur du service, le langage du service, la confiance du service, la fiabilité du service, et le coût du service. Un langage de préférences doit permettre d'exprimer les préférences sur les deux propriétés fonctionnelles et non-fonctionnelles des services. Autrement dit, le langage de préférence devrait soutenir l'expression des préférences sur le choix des services.

2.5 Service web sémantique

Dans cette section nous allons définir les services Web sémantiques, puis étudier les langages permettant la description de ce type de services Web. Le Web sémantique est un type de service web, son objectif principal est de définir et de lier les ressources du Web afin de simplifier leur utilisation, leur découverte, leur intégration et leur réutilisation dans le plus grand nombre d'applications [Berners-Lee et al., 2001].

Le Web sémantique doit fournir l'accès à ces ressources par l'intermédiaire de descriptions sémantiques exploitables et compréhensibles par des machines. Cette description repose sur des ontologies. Selon [Gruber, 1993], une ontologie est une spécification explicite d'une conceptualisation. Une conceptualisation est un modèle abstrait qui représente la manière dont les personnes conçoivent les choses réelles dans le monde et une spécification explicite signifie que les concepts et les relations d'un modèle abstrait reçoivent des noms et des définitions explicites. Donc, les services Web sémantiques sont la combinaison de deux technologies : des services Web et du Web sémantique [McIlraith et al., 2003].

2.5.1 Langages de description de services Web sémantiques

Les services Web sémantique devraient aussi être décrits sémantiquement afin de pouvoir automatiser leur découverte, sélection, composition, etc. Les langages de description de service web tel que WSDL permet de décrire les services mais d'une manière syntaxique. Il existe des approches à base de langage syntaxique comme SAWSDL, WS*-spécifications d'une part et des approches à base de modèles sémantiques d'autres part. Par la suite, nous définirons le langage OWL-S et WSMO.

2.5.2 OWL-S

OWL-S (Ontology Web Language for Service) [Martin et al., 2007] est une ontologie Web [Horrocks et al., 2003] pour les services Web. Il a été développé en vue de

soutenir la découverte automatique, la sélection et la composition de services Web. OWL-S permet de décrire les services Web de façon non-ambiguë et interprétable par des programmes [Claro et al., 2005].

L'ontologie OWL-S structure notamment la description d'un service Web en trois composants : le profil du service (ServiceProfile), son modèle de processus (ServiceModel) et ses liaisons (ServiceGrounding). La figure 2.5 résume la structure de l'ontologie supérieure d'OWL-S.

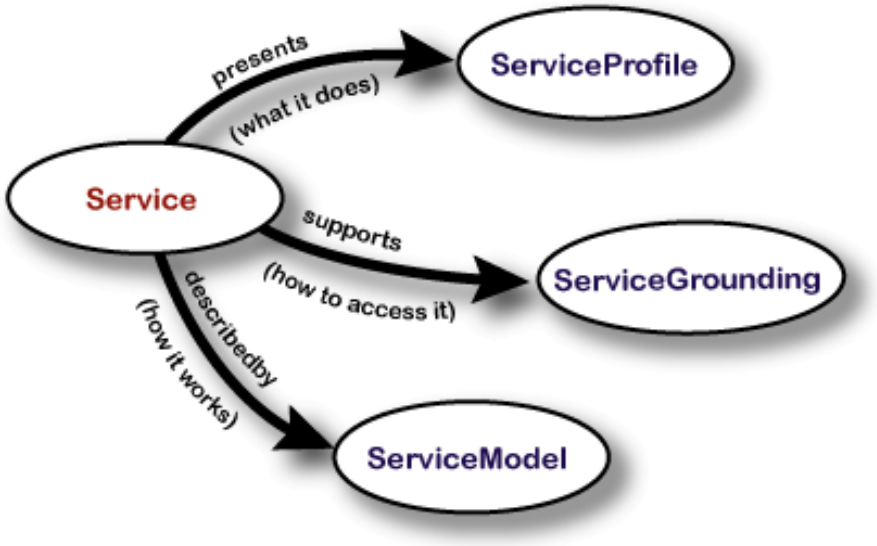

FIGURE 2.5: Structure de l'ontologie de services OWL-S.

ServiceProfile Le profil du service indique ce que le service fait. Cette section est utilisée à la fois par les fournisseurs pour publier leurs services et par les clients pour spécifier leurs besoins pour déterminer si le service répond aux besoins nécessaires. ServiceProfile est utilisé pour publier le service en décrivant ses propriétés fonctionnelles (par exemple, entrée, sortie, pré- condition, et les effets) et les propriétés non-fonctionnelles (par exemple, la confiance de service, la fiabilité, l'objet, le coût, etc.).

ServiceModel Le modèle de processus décrit comment le service fonctionne et comment l'utiliser. Il est utilisé pour décrire essentiellement le fonctionnement (modèle) d'un service composite. Il décrit comment demander le service et, en

outre, il explique ce qui se passe lorsque le service est exécuté. OWL-S modélise les services en tant que processus défini par ses entrées/sorties.

Service Grounding Explique comment interagir avec le service. Souvent, le service grounding spécifie un protocole de communication, des détails spécifiques au service ou la façon de contacter le service. Ce type d'informations est particulièrement utile pour l'invocation automatique de services.

OWL-S définit trois types de processus : les processus atomiques (AtomicProcess), simples (SimpleProcess) et composites (CompositeProcess). Chaque processus possède des entrées, des sorties, des pré-conditions et des effets. Les processus atomiques n'ont pas de sous-processus et peuvent être exécutés en une seule étape. Les processus simples fournissent une vue abstraite d'un processus existant. Cependant, à la différence des processus atomiques, un processus simple n'est pas associé à un grounding. Un processus composite est composé d'autres processus via les structures de contrôle telles que Sequence, Split, Split-Join, Any-Order, Choice, If-Then-Else, Repeat-While, Repeat-Until, et Iterate.

2.5.3 WSMO

WSMO (Web Service Modeling Ontology) est une ontologie qui décrit les différents aspects de la composition dynamique de services Web, y compris la découverte dynamique, la sélection, la médiation et l'invocation. Il est basé sur WSMF (Web Service Modelling Framework) [Fensel et al., 2002] qui précise les principaux éléments de description sémantique des services Web. Par ailleurs, pour modéliser un service Web, WSMO utilise le langage WSML (Web Service Modeling Language) [Bruijn J. et al., 2005]. WSMO propose quatre éléments clés pour modéliser les différents aspects des services Web sémantiques : les ontologies, les médiateurs, les objectifs ou goals, et les services.

WSMO Ontologies Les ontologies fournissent une terminologie pour décrire sémantiquement des éléments appartenant à des domaines spécifiques en fournissant les concepts et les relations entre eux. Elles regroupent un ensemble d'atributs à savoir : concepts, relations, fonctions, instances et axiomes. Afin de décrire les propriétés sémantiques des relations et des concepts, une ontologie WSMO fournit aussi un ensemble d'axiomes, qui sont exprimés dans un langage logique.

WSMO Mediators WSMO utilise un ensemble de médiateurs pour résoudre les incompatibilités (structurelles, sémantiques ou conceptuelles) détectées au niveau données ou processus afin de connecter les ressources WSMO hétérogènes.

WSMO goals WSMO décrit les buts qu'un utilisateur de service Web cherche à satisfaire à travers le goal. L'utilisateur définit ses critères de recherche en décrivant l'interface et les fonctionnalités attendues à travers la description du but. Un médiateur spécifique est utilisé pour connecter le but demandé avec le service Web correspondant.

WSMO Web services Différents aspects sont considérés dans la description d'un service Web WSMO, à savoir l'aspect fonctionnel et l'aspect comportemental. Pour ce faire, WSMO utilise deux points de vue différents : la capacité (Capability) et l'interface (Interface). La capacité renseigne sur les propriétés fonctionnelles du service en décrivant les variables partagées (SharedVariables) entre les préconditions (Preconditions), postconditions (Postconditions), assomptions (Assumptions) et effets (Effects).

2.6 Processus de composition

La Composition est l'une des principes fondamentaux de l'architecture orientée service (SOA) [Erl, 2007]. Le processus de composition de services Web doit combiner les fonctionnalités d'un ensemble de services si l'objectif du concepteur

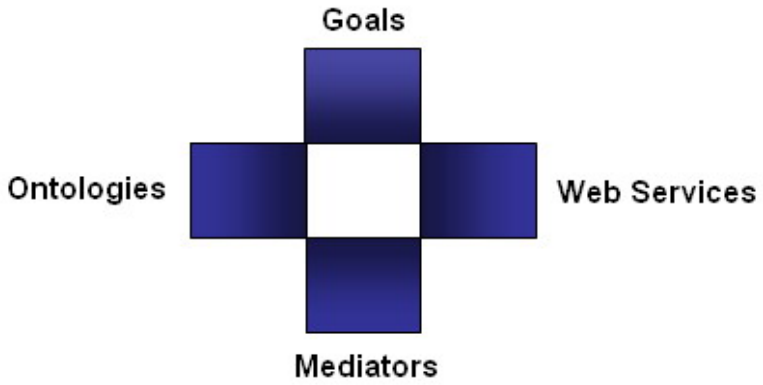

FIGURE 2.6: Structure de WSMO

d'une application n'est pas atteint par l'invocation d'un simple service Web élémentaire [Benatallah et al., 2005],[Alonso et al., 2004]. Ceci a pour effet de permettre une intégration des applications plus rapide, moins coûteuse et avec des perspectives prometteuses d'évolution et de réutilisation pour les entreprises. Les services web composants sont les services web invoqués lors d'une composition de services web. Il est composé des processus de découverte, de sélection et de coordination de services qui doivent coopérer pour répondre à un objectif complexe. Cependant, La mise en œuvre d'une composition de services Web s'avère un processus global complexe qui engendre des problèmes tels que la sélection des services web composants et l'implémentation des interactions entre ces services.

2.6.1 Définitions

Différents travaux ont défini la composition de services Web [Srivastava et al., 2003], [Alonso et al., 2004], [Yang et al., 2004], [Dustdar et al., 2005], [Claro et al., 2006], nous retenons les définitions de Gardarin [Gardarin, 2007] et Benatallah [Benatallah et al., 2005].

Selon Gardarin [Gardarin, 2007], la composition est une technique permettant d'assembler des services Web afin d'atteindre un objectif particulier, par l'intermédiaire de primitives de contrôle (boucle, test, traitement d'exception, etc.) et

d'échange (envoi et réception de messages). Les services composants existent au préalable et peuvent ne pas être fournis par la même organisation.

Dans [Benatallah et al., 2005], les auteurs ont défini la composition de services Web comme étant un moyen efficace pour créer, exécuter, et maintenir des services qui dépendent d'autres services.

2.6.2 Cycle de vie de la composition de services

Les auteurs ont défini le cycle de vie d'une composition de services Web reposant sur six activités [Benatallah et al., 2002].

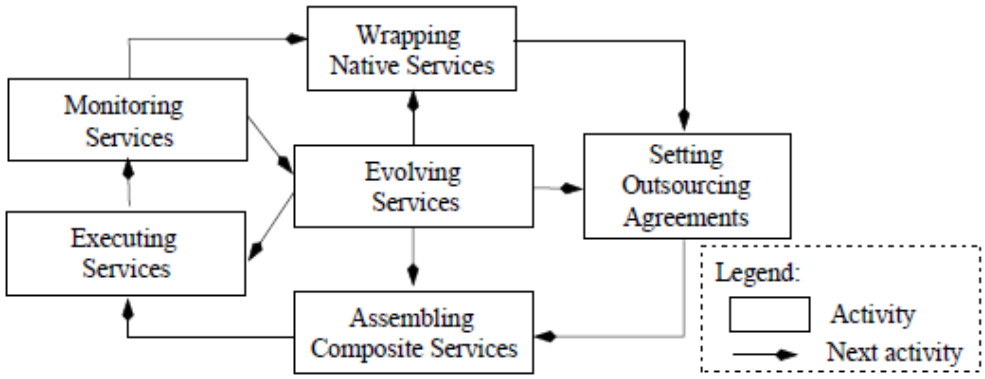

FIGURE 2.7: Cycle de vie d'une composition de services Web

L'encapsulation de services natifs (Wrapping services) : La première activité du cycle de composition de services web permet de s'assurer que lors d'une composition tout service peut être invoqué, indépendamment de son modèle de données, de son format de message, et de son protocole d'interaction.

L'établissement d'accord d'externalisation (Setting outsourcing agreements) : Cette seconde activité consiste à négocier, établir, et appliquer des obligations contractuelles entre les services.

L'assemblage de services composants (Assembling composite services) : Cette activité permet de spécifier, à un haut niveau d'abstraction, l'ensemble des services à composer afin d'atteindre l'objectif attendu. Cet assemblage

comporte une phase d'identification des services et de spécification de leurs interactions conformément aux descriptions et aux accords entre services.

L'exécution de services composants (Executing services) : Cette activité consiste en l'exécution des spécifications de la composition précédemment définies.

Le contrôle de l'exécution de services composites (Monitoring services) : La phase de contrôle permet de superviser l'exécution de la composition en vérifiant, par exemple, l'accès aux services, les changements de statut, les échanges de messages. Ce contrôle permet de détecter des violations de contrats, de mesurer les performances des services appelés et de prédire des exceptions.

L'évolutivité des services (Evolving services) : Cette dernière phase permet de faire évoluer la composition en modifiant les altérations de l'organisation de services,

2.6.3 Approches de composition de service web

Les types de composition de service web dépendent des propriétés comportementales. Elles décrivent la manière avec laquelle un service composite est constitué. L'aspect comportemental est traité essentiellement dans le cadre de la composition des services où l'information sur le comportement interne du service composite (la chorégraphie) et le comportement externe des services qui le composent (l'orchestration), est primordiale afin d'interagir avec le service composite. Selon [Benatallah et al., 2005] ces deux approches sont basées sur les workflows et leur objectif est la collaboration entre les processus métiers impliquant plusieurs services. Cependant, elles se différencient par leurs points de vue concernant la coordination entre les services intervenant dans une composition de services web.

2.6.3.1 Chorégraphie

D'après [Barros et al., 2005], la chorégraphie permet de décrire la composition comme un moyen d'atteindre un but commun en utilisant un ensemble de services Web. La collaboration entre chaque service Web de la collection (faisant partie de la composition) est décrite par des flots de contrôle. Le fait que la chorégraphie mette en oeuvre un ensemble de services Web afin d'accomplir un but commun apparaît aussi dans les travaux de [Benatallah et al., 2005]. Pour concevoir une chorégraphie, les interactions entre les différents services doivent être décrites. La logique de contrôle est supervisée par chacun des services intervenant dans la composition. L'exécution du processus est alors distribuée.

Elle est aussi appelée composition dynamique [Peltz, 2003]. En effet, l'exécution n'est pas régie de manière statique comme dans une composition de type orchestration. Dans une chorégraphie, à chaque pas de l'exécution (i.e. à chaque étape de la composition), un service Web choisit le service Web qui lui succède et implémente ainsi une partie de la chorégraphie.

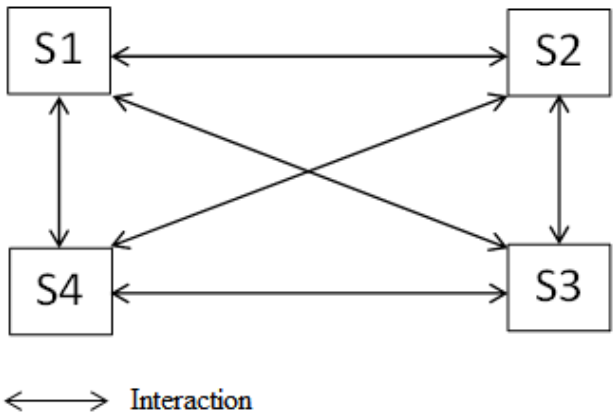

FIGURE 2.8: Vue générale de la chorègraphie.

2.6.3.2 Orchestration

[Benatallah et al., 2005] définit l'orchestration comme un ensemble d'actions à réaliser par l'intermédiaire de services Web. Un moteur d'exécution, un service Web jouant le rôle de chef d'orchestre, gère l'enchaînement des services Web par une logique de contrôle. Pour concevoir une orchestration de services Web, il faut décrire les interactions entre le moteur d'exécution et les services Web. Ces interactions correspondent aux appels, effectués par le moteur, d'action(s) proposée(s) par les services Web composants.

L'orchestration peut être vue comme une composition ascendante : les services Web utilisés dans la composition existent au préalable et sont appelés selon un enchaînement prédéfini afin de réaliser un processus précis. La Figure 2.9 illustre le principe de l'orchestration. La requête du client est transmise au moteur d'exécution (Moteur). Ce dernier, d'après le processus préalablement défini, appelle les services selon l'ordre d'exécution.

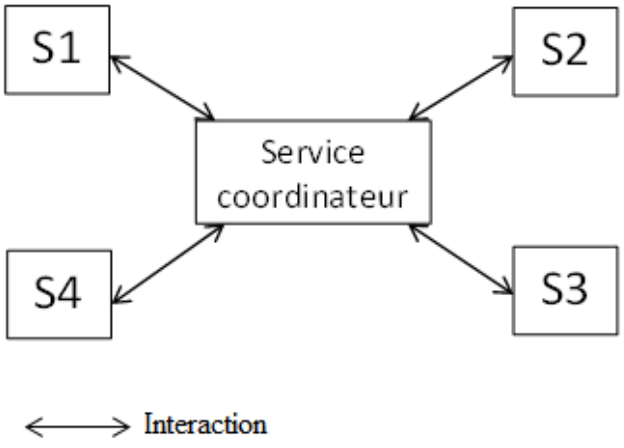

FIGURE 2.9: Vue générale de l'orchestration.

2.6.4 Les modèles de composition

De nombreux industriels et consortium (IBM, Microsoft, le W3C,...) travaillent afin de mettre en œuvre un langage de composition de services Web standard. Il existe plusieurs langages d'orchestration de services Web comme : BPEL [Andrews et al., 2003], WSCL [Banerji et al., 2002], XPDL [XPDL, 2008], BPML [Thiagarajan et al., 2002], etc. Mais le langage le plus utilisé parmi ces derniers est BPEL. Dans cette section, nous étudions quelques langages de composition comme BPEL4WS, BPML et WSCL.

2.6.4.1 BPEL

BPEL [Andrews et al., 2003] est un langage orienté processus permettant l'implémentation de services Web composites sous la forme de workflows exécutables par des moteurs d'orchestration BPEL. Un workflow BPEL est lui même constitué d'activités de contrôle, de gestion des données et d'interaction avec des partenaires. Ce langage permet de définir des partenaires (servant de canaux de communication avec les services Web et les clients) ainsi que des variables, et exhibe des activités simples et composites.

Activités simples – Invoke permet d'appeler le port d'un service Web partenaire.

- Receive permet de se mettre en attente de la réception d'un message.
- Reply correspond à la réponse adressée au client de la composition de services.
- Wait effectue une temporisation.
- Assign permet d'affecter une valeur à une variable.
- Throw rejette une exception dans le workflow.
- Terminate arrête l'instance du workflow.
- Empty correspond à une instruction sans effet.

Activités composites – Flow exécute un ensemble d'activités de manière concurrente et reliées entre elles par des liens.

- Sequence exécute séquentiellement un ensemble d'activités,
- Switch permet de sélectionner une branche d'activités parmi plusieurs, en fonction de conditions.
- While effectue des itérations jusqu'à satisfaction d'un critère.
- Pick bloque le workflow jusqu'à ce qu'un événement spécifique se produise (réception d'un message, alarme temporelle, etc.).
- Scope définit une zone restreinte du workflow permettant la définition de variables, de fautes et de gestionnaires d'exception, et compensate permet d'invoquer un processus de compensation d'exception.

2.6.4.2 BPML

C'est un méta-langage de modélisation [Thiagarajan et al., 2002] des processus métier dont les premières spécifications sont apparues en 2001 [Agarwal et al., 2001]. BPML fournit un modèle abstrait et une grammaire pour exprimer des processus métier abstraits et exécutables. En utilisant BPML, les processus d'entreprise, les services web complexes et les collaborations multi-partenaires peuvent être définis et dirigés par des compositions d'activités qui exécutent des fonctions spécifiques. Un processus BPML peut être une partie d'une composition. Chaque activité dans le processus possède un contexte qui définit les comportements communs et les activités s'exécutant dans tel contexte. Ainsi, un processus peut être défini comme un type d'activité complexe qui déclare son propre contexte d'exécution. La spécification BPML décrit dix-sept types d'activités et trois types de processus. La description WSDL d'un service web peut être importée ou référencée dans une spécification BPML. Une standardisation des documents BPML est proposée en utilisant RDF pour la sémantique des métadonnées, des métadonnées XHTML et Dublin Core pour améliorer la lisibilité et le traitement de l'application.

2.6.4.3 WSCL

WSCL [Banerji et al., 2002] permet de définir le comportement externe visible des services web en spécifiant les conversations du niveau métier ainsi que les processus métiers publiques supportés par un service web. Les conversations sont définies en utilisant la syntaxe de XML. Un document WSCL spécifie les documents XML échangés comme une partie de la conversation ainsi que l'ordre dans lequel ils sont échangés. WSCL fournit un ensemble minimal de concepts nécessaires pour spécifier les conversations.

2.6.5 Composition statique vs dynamique

La composition des services Web peut être soit une composition statique soit une composition dynamique [Dustdar et al., 2005] :

La composition statique est appelée aussi composition offline, précompilée ou encore proactive. C'est une composition qui utilise des services basiques qui sont préalablement définis d'une façon figée et qui ne peuvent pas changer en fonction du contexte du client. Ce type de composition engendre des applications peu flexibles, parfois inappropriées avec les exigences des clients. Ce type de composition est plus approprié aux environnements stables où les services préalablement identifiés par l'utilisateur ne sont pas susceptibles de changer à court terme. Néanmoins, la modification d'un service composant ou sa substitution par un autre suscite également des modifications au niveau de la conception du service composite.

La composition dynamique Appelée aussi composition on-line, post compilée ou encore réactive. Elle se réfère à la sélection des services basiques à la volée. Autrement dit, la sélection des services basiques ne peut pas être prédéfinie à l'avance mais elle le sera au moment de l'exécution en fonction des contraintes imposées par le client. Ceci permet d'élaborer différents scénarii de composition

qui offrent les mêmes fonctionnalités et qui tiennent compte de la dynamique de la situation du client.

Contrairement à la composition statique où le nombre de services fournis est limité et les services à composer sont spécifiés au préalable, la composition dynamique, elle, est initiée par une requête de l'utilisateur. Elle permet de découvrir, sélectionner et combiner dynamiquement les services à partir des annuaires de services Web, au moment de l'exécution de ladite requête, tout en tenant compte des contraintes de l'utilisateur. Toutefois, la réalisation de la composition dynamique dans un environnement où le nombre de services fournis est en constante évolution n'est pas une simple tâche.

2.6.6 Composition manuelle Vs automatique

Le degré d'automatisation peut aussi être pris en considération pour identifier la nature de la composition de services. Généralement on trouve trois types : manuelle, semi-automatique et automatique [Foster et al., 2003].

La composition manuelle Est l'approche traditionnelle où l'utilisateur des services Web programme manuellement toutes les étapes du processus de composition des services composites dont il a besoin. L'inconvénient d'une telle approche est qu'elle exige des connaissances approfondies et des efforts considérables de la part de l'utilisateur qui n'a pas toujours le profil développeur. Cette tâche reste d'une part, ardue et d'autre part, sensible aux erreurs dues au dynamisme et à la flexibilité du Web. Des changements apportés au niveau des services Web composés peuvent affecter le fonctionnement du service composite obtenu et susciter des changements au niveau du processus de composition qui doit être relancé.

La composition automatique Est une approche qui prend en charge tout le processus de composition et le réalise automatiquement, sans qu'aucune intervention de l'utilisateur ne soit requise. Le processus de composition dans ce

cas est un processus générique destiné à différents utilisateurs qui souhaitent chercher et composer des services qui répondent à leurs requêtes spécifiques.

La composition semi-automatique Est une approche où l'utilisateur maintient un certain contrôle et supervise le processus de composition envisagé sans avoir besoin de connaissances approfondies en programmation. Il peut, avec l'assistance des outils existants, modéliser et concevoir des services web composites, sélectionner les services appropriés sur la base de suggestions sémantiques proposées et intervenir en permanence durant le processus de composition.

2.7 Conclusion

La technologie des services Web permet à des applications de dialoguer à distance via Internet, indépendamment des plates-formes et des langages sur lesquels elles reposent, en s'appuyant sur des protocoles standards. Dans ce chapitre, nous avons mis en relief les différents langages proposés par le consortium W3C permettant aux fournisseurs de décrire leurs services Web. Cette étape de description est le premier processus indispensable dans le cycle de vie d'une application basée sur une Architecture Orientée Service SOA. Ensuite, nous avons introduit la notion sémantique dans les services web. Les services Web, de plus en plus utilisés dans le monde industriel, posent de nouveaux problèmes. Par exemple comment composer ces services ? Nous avons défini par la suite la composition de services web et cité les différents types de composition et les langages de composition.

Le chapitre suivant introduira les techniques utilisées, le raisonnement par mémoration qui est le raisonnement à partir de cas et la technique de planification.

Chapitre 3

Raisonnement à Partir de Cas

Sommaire

3.1 Introduction

La composition des services est loin d'être une tâche simple. Cela est dû au processus de recherche et de sélection. Ainsi, nous devons trouver les meilleures méthodes et moyens qui facilitent ce processus au demandeur de service. Parmi ces méthodes, nous nous intéressons à l'approche par raisonnement à partir de cas (RàPC). Ce choix est justifié par le fait que le RàPC est une approche qui permet de combiner la résolution de problèmes et l'apprentissage.

Dans ce chapitre, nous présentons le paradigme du Raisonnement à Partir des Cas (RàPC). Ensuite, nous exposons en détail les principes fondamentaux du RàPC pour nous familiariser avec les différents termes utilisés. Nous exposons également le cycle de raisonnement du RàPC et les phases qui en découlent. Nous présenterons aussi la notion de similarité et quelques techniques de recherche de similarité appliquées dans les systèmes RàPC. Nous en aborderons la planification à base de cas avant de conclure.

3.2 Le principe du Raisonnement A Partir de Cas

Le Raisonnement A Partir de Cas (en anglais : Case Based Reasoning ou CBR) est une méthode de raisonnement qui utilise les expériences précédentes pour résoudre de nouveaux problèmes. Cette méthode consiste à modéliser un aspect du raisonnement humain qui consiste à se baser sur des situations déjà vécues. L'origine du RàPC vient de Minsky et Schank de l'université de Yal [Schank, 1982] qui consiste à modéliser un aspect du raisonnement humain se basant sur des situations déjà vécues pour répondre à une nouvelle situation rencontrée. Par exemple, un médecin utilise les cas précédents pour déterminer la maladie d'un nouveau patient tout comme un développeur de service web travaillant sur une nouvelle solution en

utilisant des problèmes déjà résolus. Le RàPC est une méthodologie issue de l'intelligence artificielle et de la psychologie cognitive et orientée vers la résolution de problèmes. Il consiste à résoudre un les problèmes à partir d'expériences passées (cas) c'est-à-dire les problèmes résolus précédemment. Le but est d'utiliser ces cas pour résoudre de nouveaux problèmes.

3.2.1 Définition

Selon Michalski [Michalski, 1986] la résolution de problème par analogie consiste à transférer de la connaissance à partir des épisodes précédents de résolution de problèmes aux nouveaux problèmes qui partagent des aspects significatifs de l'expérience précédente correspondante, et à utiliser les connaissances transférées pour construire des résolutions pour les nouveaux problèmes. Mais la définition la plus utilisée est celle de Reisbeck et Schank [Reisbeck et Schank, 1989], le RàPC résout des problèmes en utilisant ou en adaptant des solutions de problèmes antérieurs.

Beaubouche r [Beaubouche r, 1994]précise que le RàPC consiste à retrouver des connaissances à partir d'épisodes passés dans un domaine particulier, connaissances qui partagent des aspects significatifs avec des expériences passées correspondantes, et à utiliser les connaissances transférées pour construire des solutions aux nouveaux problèmes ou pour justifier des solutions du domaine.

D'après Aamodt et Plaza, le raisonnement à partir de cas est une approche d'apprentissage et de résolution de problèmes basée sur les expériences passées [Aamodt et al., 1994]. Watson quant à lui, définit le RàPC comme une méthodologie de résolution des problèmes qui utilise différentes technologies pour ce faire [Watson, 1999]. Il considère que le RàPC n'est pas une technologie en soi.

En résumé, le RàPC consiste à utiliser les informations et les connaissances précédentes de cas déjà résolus pour résoudre un nouveau problème.

3.2.2 Le carré d'analogie

Le RàPC s'inspire également du raisonnement par analogie dont il est considéré comme un cas particulier. Les auteurs [Mille et al., 1996] ont proposé un modèle de carré d'analogie (illustré dans la figure 3.1) qui permet de faire le lien entre la description du cas et sa solution.

Le raisonnement par analogie vise à caractériser une situation en cours, appelée cible, en la mettant en correspondance avec une situation déjà rencontrée, appelée source. Si on trouve une ressemblance entre ces situations, des relations peuvent être établies entre le problème cible et le problème source.

Parmi les cas disponibles, un cas source similaire au problème cible est sélectionné en se référant aux valeurs de ses descripteurs et en utilisant une mesure de similarité Alpha. Puisque les descripteurs du problème cible peuvent être similaires et non pas identiques à ceux décrivant le problème source, les descripteurs solution peuvent être adaptés par la suite. Cette adaptation est effectuée conformément à des relations de dépendances Beta entre les descripteurs du problème et celles des descripteurs de la solution.

En fonction de ces dépendances et des écarts alpha constatés entre les problèmes cible et source, l'adaptation permet de proposer une solution cible candidate qui pourra être évaluée par la vérification de sa conformité aux dépendances particulières qui pourraient exister entre le problème et la solution cibles (figure 3.1) [Mille et al., 1996] .

3.2.3 Définition d'un cas

Un cas est une expérience qui constitue une leçon permettant au système de RàPC de résoudre des problèmes de différentes natures. D'après Fuchs [Fuchs et al., 2006], le cas est défini comme étant la description informatique d'un épisode de résolution de problème.

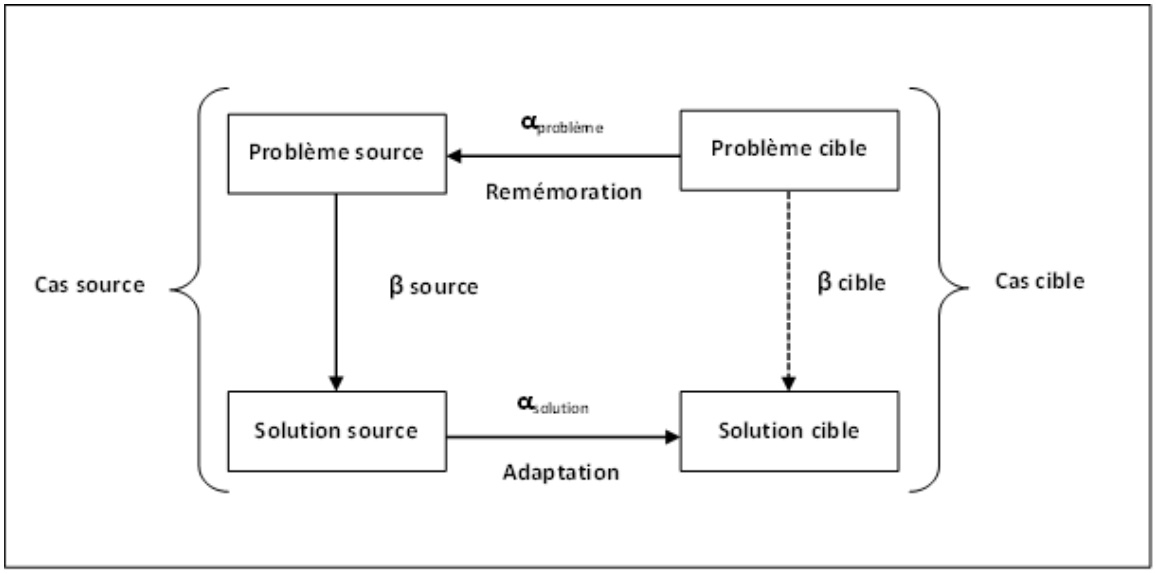

FIGURE 3.1: Carré d'analogie [Mille et al., 1996].

Selon Mille [Mille, 1995], la définition d'un cas passe par trois étapes : la première est la synthèse qui consiste à trouver une structure permettant de satisfaire des spécifications. La deuxième est l'analyse qui, à partir d'une structure particulière, consiste à trouver le comportement associé. La troisième concerne l'évaluation qui consiste à vérifier que le comportement est conforme à ce qui est attendu.

Les informations contenues dans un cas varient selon le domaine d'application et les objectifs à atteindre. La représentation de cas constitue une étape fondamentale dans un système de RàPC puisqu'un nouveau problème est résolu en utilisant une solution passée. Ainsi, la recherche de cas similaires doit donc être à la fois efficace et rapide afin de décider comment la mémoire de cas devrait être organisée et classée. Il est primordial de choisir les informations à stocker dans chaque cas et la forme correspondante.

3.2.3.1 Structure d'un cas

Un cas en RàPC est généralement composé de deux parties : le problème et la solution. Le problème noté Pb concerne la partie dans laquelle on trouve les objectifs à atteindre. Quant à la solution Sol(Pb), elle regroupe la description de la solution

apportée par le raisonnement, sa justification, son évaluation ainsi que les étapes qui ont mené à cette solution.

D'autres informations peuvent être ajoutées à la structure d'un cas selon le besoin comme le résultat de l'évaluation, l'explication des échecs, etc. On peut distinguer deux types de cas : *cas source et cas cible*. Le cas source est celui dans lequel le problème est résolu et la solution est renseignée. Quant au cas cible, c'est celui qui porte le problème et dont sa partie solution n'est pas encore trouvée.

Cas-source =(Pb-source, Sol(Pb-source))

Cascible =(Pb-cible, Sol(Pb-cible))

Chacune des deux parties est décrite par un ensemble de descripteurs. Un descripteur d englobe toutes les informations qui nous permettent de décrire le problème. Il est représenté par une paire (a,v), où a est un attribut défini par un nom et v est la valeur qui lui est associée. L'attribut représente une caractéristique du domaine applicatif qui peut être numérique ou symbolique [Gebhardt et al., 1997].

Un cas source est représenté par le tuple$(d_1^s, ..., d_n^s, D_1^s, ..., D_m^s)$où :

- d_i^s est un descripteur du problème source.
- D_j^s est un descripteur de la solution source.

Un cas cible est représenté par le tuple$(d_1^c, ..., d_n^c, D_1^c, ..., D_m^c)$où :

- d_i^c est un descripteur du problème cible.
- D_j^c est un descripteur de la solution cible.

3.2.3.2 Base de cas

Une base de cas ou librairie de cas est la mémoire qui contient tous les cas sources précédemment retenus. Dans un système RàPC, la recherche rapide et efficace des cas sources les plus similaires au cas cible est primordial d'où la nécessité d'organiser la base de cas. Cette organisation a pour but de retrouver facilement et efficacement

les cas similaires. D'après Fuchs, il existe deux méthodes pour l'organisation de la base de cas : l'organisation simple et l'organisation hiérarchisée [Fuchs et al., 2006].

Dans l'organisation simple, les cas sont stockés sans aucune structuration ou organisation particulière. Durant la remémoration, il est obligatoire de revoir l'ensemble des cas pour la recherche des cas similaires, afin d'en sélectionner le plus approprié. Toutefois, ce type d'organisation n'est pas recommandé lorsque le nombre de cas est considérable, en raison du processus d'appariement qui sera coûteux en termes de temps de recherche.

L'organisation hiérarchisée est adoptée pour des bases de cas complexes. La base de cas est structurée de telle façon à permettre de rechercher un cas source de manière sélective, moyennant un processus de classification éliminatoire. Ceci permet de rétrécir l'espace de recherche à un sous-ensemble réduit, regroupant un ensemble restreint de cas à considérer.

3.2.4 Cycles du Raisonnement à Partir de Cas

Le RàPC dispose d'un cycle composé de plusieurs phases dont le nombre varie selon les différentes sources bibliographiques. Le cycle classique de résolution de problèmes du RàPC a été proposé par Aamodt et Plaza [Aamodt et al., 1994]. Ces auteurs ont proposé les quatre phases suivantes : la remémoration ou la recherche, la Réutilisation ou l'adaptation, la révision et la mémorisation. Ce cycle a été amélioré par Mille [Mille, 1999], en ajoutant une phase appelée Elaboration avant la remémoration. Selon Fuchs, le cycle est composé de trois phases à savoir la remémoration, l'adaptation et la mémorisation [Fuchs et al., 2006].La Figure 3.2 montre le cycle de RàPC avec ces cinq phases.

Dans notre étude, nous exploitons le cycle de RàPC composé de cinq phases. Dans ce qui suit, nous allons expliquer chaque étape du cycle de vie du raisonnement à partir de cas.

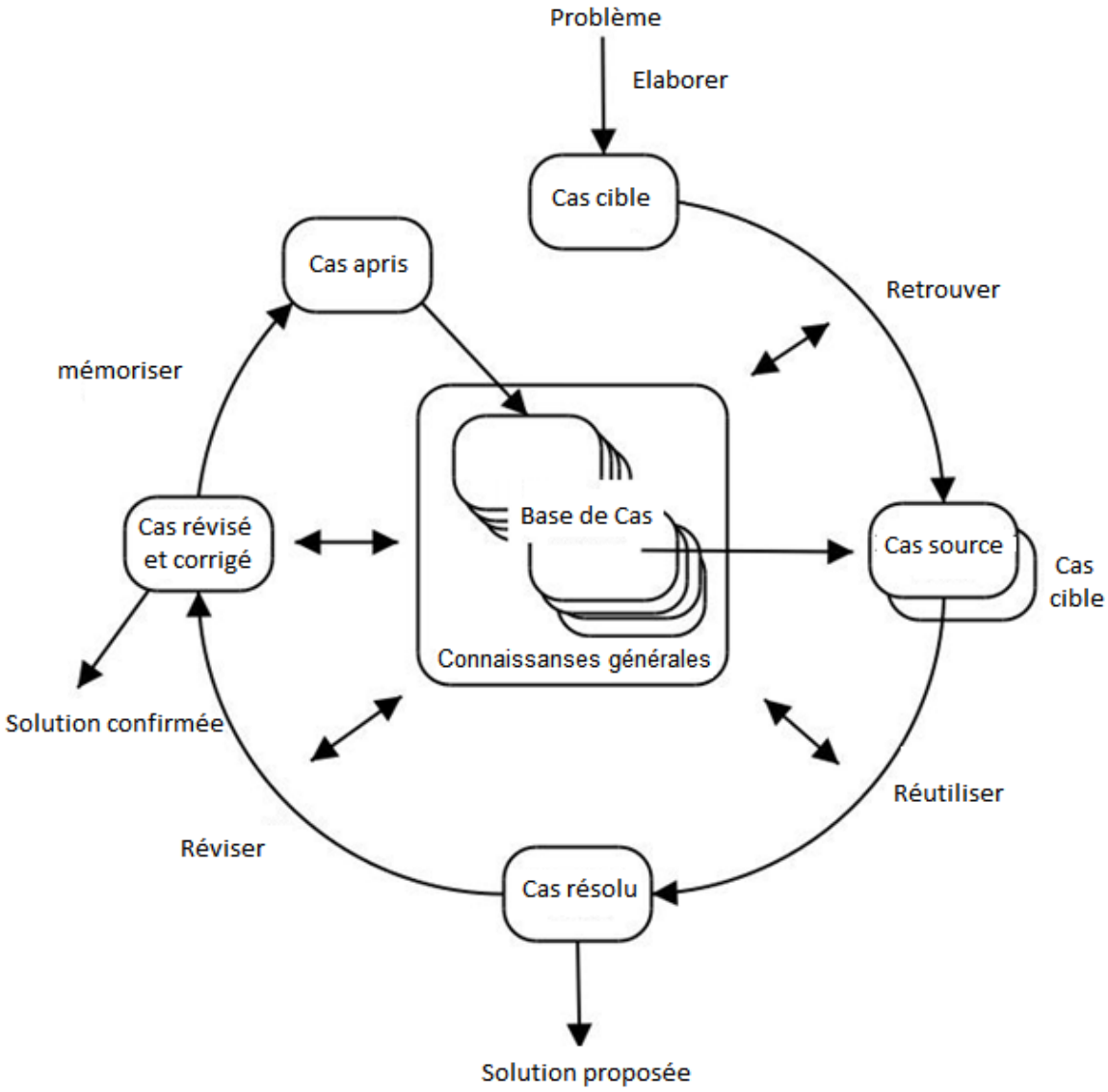

FIGURE 3.2: Le cycle de raisonnement à partir de cas selon [Mille, 1999].

3.2.4.1 Élaboration

Dans le cas général, lorsqu'un problème est posé dans un système du RàPC, les connaissances à son sujet peuvent être de natures très différentes. Certaines de ces connaissances peuvent être totalement étrangères à la résolution du problème. D'autres, en revanche, peuvent être absentes et pourtant indispensables pour permettre d'envisager une résolution.

La première phase du raisonnement à partir de cas a donc pour but de compléter et de formaliser la description du problème en se fondant sur des connaissances.

Cette première phase de formalisation de la description du problème correspond à la phase d'élaboration. Une première formalisation de la phase d'élaboration est récemment proposée dans [Fuchs et al., 2006] et donne la définition suivante :

une étape qui consiste, à partir de l'entrée du système de RàPC, à construire le problème cible .

Cette phase nécessite deux tâches principales qui sont la création et la préparation du cas. Ces tâches permettent d'affecter des descripteurs au nouveau cas cible possédant une sémantique liée au problème. Cette affectation de descripteurs doit être envisagée dans un but d'anticiper au maximum l'adaptabilité des cas qui seront remémorés. Elle peut être faite soit par l'utilisateur qui, à partir de ses données, va constituer un problème à résoudre, soit par le système qui sait lui-même élaborer un problème en fonction des données dont il dispose. Cette étape consiste alors, à partir de l'entrée du système du RàPC, à construire le problème cible.

3.2.4.2 Remémoration

La phase de remémoration ou de recherche est une étape très importante dans le cycle du RàPC. Cette importance vient du fait que l'idée fondamentale de ce raisonnement est de dire que des problèmes similaires peuvent avoir des solutions similaires et peuvent guider des futurs raisonnements. Donc, la remémoration consiste à retrouver des cas sources similaires au cas cible, c'est-à-dire de chercher et de retrouver, dans la base de cas, des cas dont les problèmes possèdent des descripteurs proches au problème posé. Les cas sources trouvés ont déjà des solutions enregistrées qui peuvent être utilisées dans la résolution du cas cible.

La recherche de cas proches nécessite une mesure de similarité entre cas. Cette mesure se réduit le plus souvent à une mesure de similarité entre problèmes. Elle dépend directement de la représentation des données et de leur complexité. Elle doit être adéquate au problème. En d'autres termes, la similarité est déterminée par le degré d'appariement entre le problème posé (cible) et les problèmes sources de la

base de cas. Pour cela, le choix d'une mesure de similarité est déterminant pour l'efficacité du RàPC.

3.2.4.3 Adaptation

On trouve plusieurs définitions dans la littérature concernant la phase d'adaptation ou de réutilisation. Parmi elles, celle de Lopez qui définit l'adaptation comme un processus proposant une solution à un nouveau problème à partir des solutions appartenant aux cas sources remémorés [Lopez de Mantaras et al., 2005]. Fuchs et ses collègues [Fuchs et al., 1999] considèrent l'adaptation comme un plan dont l'état initial est la solution de départ et l'état final est la solution adaptée.

La phase d'adaptation des cas est le processus qui permet de transformer les solutions retrouvées en une nouvelle solution pour le problème posé. Pour pouvoir adapter ses solutions, il faut savoir déterminer la différence entre le cas source et le cas cible, puis décider quelles sont les parties qui peuvent être transférées au nouveau cas.

L'adaptation a pour tâche de construire une solution Sol(cible) du problème cible en s'appuyant sur les solutions Sol(source) du cas remémoré, appelé cas source et noté (source, Sol(source)). Cette phase peut être semi-automatique via une intervention humaine soit d'une manière automatique à l'aide d'algorithmes, de méthodes, de formules, de règles, etc.

Il existe plusieurs façons pour adapter les cas remémorés selon Wilke et Bergmann [Wilke et al., 1998] résumé dans la figure 3.3. Parmi ces types nous citons :

Adaptation simple, réinstantiation ou par copie : dans ce cas, la solution du cas similaire sera directement utilisée comme solution du problème courant. Ainsi, nous considérons que les similarités sont suffisantes et que les différences entre le cas trouvé et le problème peuvent être négligeables.

Adaptation transformationnelle : ce type d'adaptation ne prend pas en

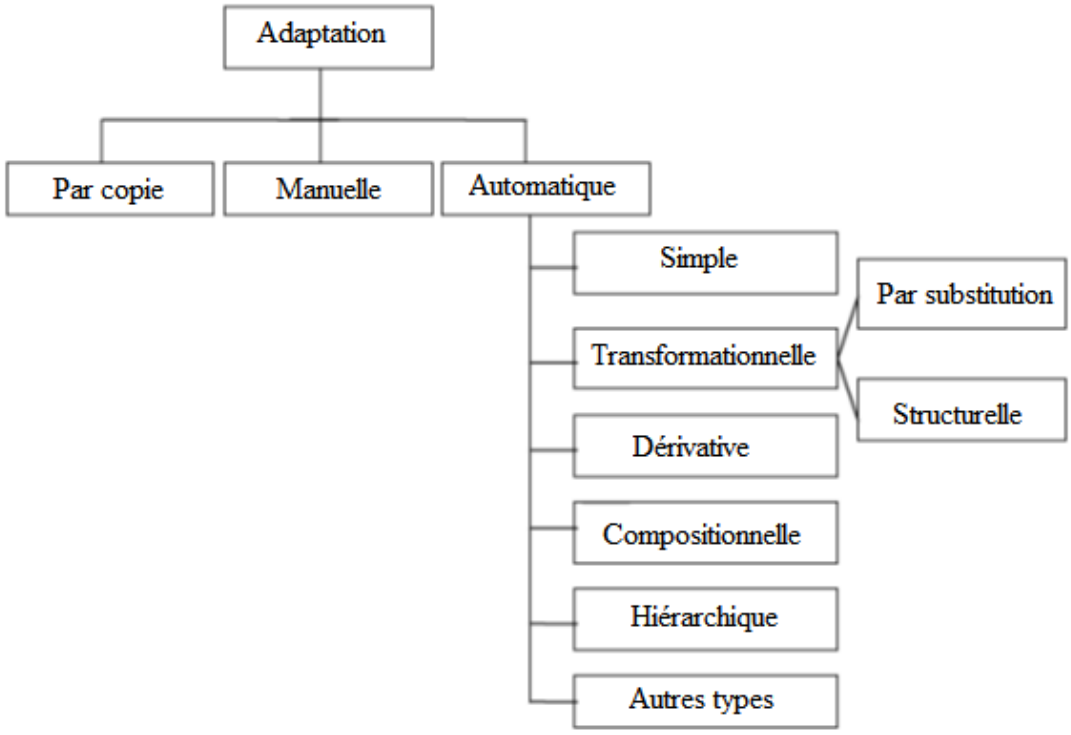

FIGURE 3.3: Types d'adaptation dans les systèmes de RàPC.

compte la manière avec laquelle les solutions des cas similaires ont été générées. C'est-à-dire qu'on ne dispose pas de toutes les connaissances pour résoudre le problème à partir de zéro. Il utilise un ensemble de règles d'adaptation pour transformer les solutions des cas candidats.

Adaptation compositionnelle : utilise deux ou plusieurs cas similaires remémorés pour effectuer l'adaptation en composant les différentes solutions proposées ;

Adaptation dérivative : ce type d'adaptation peut être utilisé lorsque l'on dispose du chemin qui mène à la solution pour chaque cas de la base. L'adaptation dérivative adopte les chemins menant aux solutions des cas sélectionnés pour le nouveau problème afin d'en construire sa nouvelle solution. Il existe d'autres types d'adaptation comme l'adaptation substitutionnelle, hiérarchique, etc.

3.2.4.4 Révision

Une fois, une adaptation trouvée, la solution issue doit être testée afin de vérifier si elle convient pour résoudre le problème posé. Ce test peut s'effectuer de plusieurs manières [Mille, 1999], la solution peut être essayée dans le monde réel ou bien selon le domaine. Nous pouvons faire appel à un logiciel de simulation ou à un expert.

La phase de révision consiste donc à continuer éventuellement l'élaboration de la solution cible si besoin. Par conséquent, si avec les précédentes actions la solution est jugée insatisfaisante alors elle va être corrigée. Cette étape de révision peut être considérée comme une autre forme d'adaptation à une seule différence est que nous partons d'une solution incorrecte mais adaptée au problème au lieu de solutions correctes inadaptées. D'où, la révision s'avère déterminante pour l'apprentissage du système et son évolution.

3.2.4.5 Mémorisation

La phase de mémorisation ou d'apprentissage est la dernière du cycle de RàPC. Cette phase consiste à ajouter le nouveau cas composé du nouveau problème et de sa solution dans la base de cas et permet de synthétiser les nouvelles connaissances qui vont être réutilisées ultérieurement.

Le stockage d'un nouveau cas permet donc d'enrichir la base de cas permettant l'augmentation de l'expérience du système ou l'apprentissage. Pour cela, il faut sélectionner les cas à mémoriser, mais certains cas peuvent être très proches et par conséquent la mémorisation peut conduire à une duplication de données dans la base de cas. Donc, il est nécessaire de prendre en compte quelques considérations. Selon Benard et ses collègues [Benard et al.,2008], pour pouvoir ajouter un cas, il faut vérifier :

1. Que le contexte ne peut pas être complètement identifié avec le cas source de la base,

2. L'objectif pour un contexte identique est différent que celui qu'on connaissaient dans la base,
3. L'action pour atteindre un objectif doit être nouvelle.

.

Si le nouveau cas est digne d'intérêt, il peut alors être enregistré pour enrichir la base de cas. Cette phase du raisonnement est généralement à la charge de l'expert du domaine responsable du système. Nous pouvons alors considérer que la fonction d'apprentissage qui consiste à ajouter des nouveaux cas ou modifier des connaissances pour résoudre des situations d'échec correspond à un apprentissage supervisé.

3.3 Notion de Similarité

La principale difficulté dans la phase de remémoration réside dans le choix de la méthode de sélection concernant les cas à remémorer. Tout comme la plupart des phases du Raisonnement A Partir de Cas, diverses méthodes ont été utilisées pour remémorer les cas. La plupart de ces méthodes consistent en une comparaison des cas, réalisée de façon globale.

La recherche de cas similaires dépend de la représentation de cas, de leur indexation et de leur organisation dans la base de cas. Elle s'appuie sur des mesures de similarité qui permettent de mesurer la similarité entre le problème posé et les cas candidats. L'objectif de ces mesures de similarité est de chercher dans la base de cas le cas le plus proche du problème actuel.

3.3.1 Mesure de similarité

Les mesures de similarité peuvent être locales ou globales. Elles sont locales lorsque les mesures sont relatives aux caractéristiques de cas. Elles s'inspirent généralement de la notion de distance et dépendent du type de descripteur qu'il soit

numérique, symbolique ou taxonomique.

Les mesures globales sont les mesures relatives aux cas. Il s'agit d'agréger les similarités locales afin de trouver une similarité globale. Plusieurs mesures peuvent être utilisées dans différents domaines, telles que :Weighted block-city, Mesure euclidienne et Mesure de Minkowski[Dhouib, 2009].

La recherche de cas similaires peut être effectuée par plusieurs algorithmes dont les plus importants sont : Algorithme du K plus proches voisins (KPPV) , Approches inductives ou Induction basée sur la connaissance.

3.4 Systèmes à base de RàPC

Le RàPC a été utilisé dans différents domaines comme la médecine Kasimir [Aquin et al., 2004], Discipline juridique Ashley [Ashley, 1990], l'aviation aérospatiale Drama [Wilson et al., 2000] , les recettes de cuisine Chef [Hammond, 1986], Systèmes industriels Creek [Aamodt, 2004], Déjà Vu [Smyth et al., 1996] et d'autres systèmes. Par la suite, nous citerons quelques systèmes à base de RàPC.

CHEF [Hammond, 1986], qui un système de planification de menus, utilise une méthode d'adaptation de recettes. Ce système commence par une adaptation substitutionnelle qui consiste à substituer les ingrédients dans le cas retrouvé comportant une recette afin de satisfaire les besoins du menu. Si la solution proposée n'est pas satisfaisante alors il procède à une adaptation transformationnelle. Cette adaptation est donc utilisée afin de modifier la solution proposée en ajoutant ou en supprimant des étapes dans la recette. Ces modifications sont issues du résultat des différentes substitutions d'ingrédients.

Déjà Vu [Aamodt, 2004] Est un planificateur pour le contrôle de déplacement de véhicules autonomes dans des environnements industriels. Dans déjà Vu un problème est défini comme une tâche de déplacement de véhicule, et la solution est représentée

par un programme informatique spécifiant les actions à effectuer le déplacement. La technique d'adaptation utilisée est hiérarchique.

PARIS [Smyth et al., 1996] PARIS (Plan Abstraction and Refinement in an Integrated System) est un système de planification basé sur les cas qui permet la réutilisation flexible des cas par l'abstraction et le raffinement. Ce système n'est pas créé pour un domaine d'application spécifique.

3.5 La Planification et le raisonnement à partir de cas

3.5.1 La planification

La planification est une technique d'intelligence artificielle qui désigne un domaine de recherche pour la conception des systèmes pouvant générer automatiquement des plans. Le planificateur est le système qui génère les plans.

Un planificateur a comme entré, un problème à résoudre qui consiste en une description de l'état initial et du but à atteindre et un domaine de planification décrit par un ensemble d'actions qui vont permettre les transitions entre les états. La solution fournie par le planificateur est un plan qui en partant de l'état initial permet d'atteindre le but. Comme solution, le planificateur peut fournir plusieurs plans au même problème. Une sélection doit être faite pour trouver le meilleur plan appelé plan-solution. Néanmoins, la planification classique possède certaines limites comme l'insensibilité aux évolutions du domaine, la supposition de la connaissance complète de l'environnement, etc.

3.5.1.1 Représentation d'un problème de planification

La définition 1 présente d'une manière formelle un problème de planification [Fikes et al., 1971]. En planification, un domaine D décrit le cadre dans lequel va

être résolu le problème (Définition 2). Il est indépendant de n'importe quel état initial ou but. Intuitivement, un domaine modélise toutes les actions que peut effectuer le système pour lequel on planifie. Dans l'exemple du monde de blocs, le système pour lequel on planifie est le bras robotique.

Formellement, le domaine est défini par l'ensemble des types de variables, l'ensemble des prédicats et A l'ensemble des opérateurs instanciés de O (voir la définition 2). Une solution à un problème de planification classique est appelée plan (voir la définition 3).

Définition 1. Un problème de planification pour un domaine de planification D est défini par le couple (s_0 ; s_g) où : s_0 est l'état initial et s_g est le but à atteindre.

Définition 2. Un domaine de planification D est un triplet (S ;A ; γ) tel que :

S : ensemble des états avec S = 2 à la puissance des atomes instanciés de L

A : l'ensemble des opérateurs instanciés de O,

$\gamma(s;a)$ = (s – $effets^{-}$(a)) $\bigcup$ $effets^{+}$(a) si $a \in S$ et a est applicable à $s \in S$. La fonction étant la fonction de transitions entre états.

Définition 3. Un plan solution est défini par une séquence d'actions (a_1 ; a_2 ; ... ; a_n) ou chaque action ai est une instance d'opérateur de O permettant d'aller de s_0 à s_g.

3.5.1.2 Langages de planification

Dans la littérature, trois principaux langages de planification permettent de représenter un problème de planification : STRIPS [Fikes et al., 1971], ADL [Pednault, 1994] et PDDL [Ghallab et al., 1998]. Récemment, différents formalismes de planification utilisés en intelligence artificielle ont été unifiés dans une syntaxe standard dite PDDL (Planning Domain Description Language). Ce langage permet

aux chercheurs de la communauté de planification classique d'échanger des problèmes de tests et de performance et de comparer les résultats. PDDL contient des sous langages pour STRIPS, ADL et les réseaux hiérarchiques de tâches (HTN). Les progrès réalisés ces dernières années permettent au langage PDDL de décrire des domaines complexes et de plus en plus proches des problèmes réels tels que la prise en compte de la dimension temporelle [Fox et al., 2003], ou la gestion des ressources [Maris, 2009].

Langage PDDL

La définition d'un problème de planification à l'aide de PDDL nécessite : la définition d'un domaine générique commun à tous les scénarios ; une description d'un problème. Dans ce qui suit, nous présentons la description en PDDL d'un domaine, d'un opérateur et d'un problème de planification. La figure 3.4 présente la description d'un domaine de planification. Nous notons :

- La déclaration des types qui passe par l'utilisation du mot clé : types ;
- La déclaration des prédicats qui passe par l'utilisation du mot clé :predicates ;
- La définition générique des opérateurs qui passe par l'utilisation du mot clé : action.
- Un opérateur est défini par son nom, ses paramètres, ses pré-conditions et ses effets.
- Les paramètres (:parameters) d'un opérateur sont des objets typés ou des instances des prédicats (ou atomes) spécifies par les pré-conditions (:precondition) d'application de l'action.
- Les pré-conditions contiennent aussi des conjonctions d'atomes. Les effets de l'opérateur sont des conjonctions d'ajouts d'atomes et des conjonctions de retraits d'atomes.

Une instance 3.5 est composée des objets autorisés à être manipulés, une description de l'état initial et une description de l'état final.

```
(define (domain DOMAIN_NAME)
 (:requirements :strips :equality ....)
 (:typing T1 T2 T3 ... TN - objects)
 (:predicates (PREDICATE_1_NAME [?A1_1 - T1_1 ... ?AN_1 - TN_1])
              (PREDICATE_2_NAME [?A1_2 - T1_2 ... ?AN_2 - TN_2])
              ...)
 (:action ACTION_1_NAME
   :parameters ([?P1 ?P2 ... ?PN])
   :precondition PRECOND_FORMULA
   :effect EFFECT_FORMULA)
...)
```

FIGURE 3.4: Définition d'un problème de planification.

```
(define (problem PROBLEM_NAME)
 (:domain DOMAIN_NAME)
 (:objects OBJ1 OBJ2 ... OBJ_N)
 (:init ATOM1 ATOM2 ... ATOM_N)
 (:goal CONDITION_FORMULA))
```

FIGURE 3.5: Définition d'un problème de planification.

Plusieurs techniques de planification ont été développées pour régler les problèmes de la planification classique comme la planification à partir de cas.

3.5.2 Planification à partir de cas

La Planification à partir de cas (Case-Based Planning), ou la planification par la réutilisation [Spalzzi, 2001]. La principale observation dans CBP est que dans la plupart des domaines du monde réel dans lequel la planification est appliquée, la typologie des problèmes qui devraient être résolus reste similaire. Par conséquent, il est prévu que les solutions des problèmes précédemment analysées peuvent être utiles lors de la résolution de nouveaux problèmes à l'intérieur du même domaine.

Dans ces cas, il peut être plus efficace d'adapter un plan existant, plutôt que de partir de zéro. La formalisation du cas cible de planification proposée par Liberatore (2005), est une paire $(\prod_0, \pi_0)$ ou $\prod_0$ est le problème de planification et π_0 est le plan solution, tant dit qu'une base de cas est un ensemble de cas $\{(\prod_0, \pi_0), 1 \leq i \leq m\}$

En général, les étapes suivantes sont exécutées lorsqu'un nouveau problème de planification est résolu par un système de RàPC. La remémoration du plan : récupérer les cas de la base qui sont analogues à la (cible) problème actuel. L'adaptation du plan : la réutilisation des plans récupérés pour produire un nouveau plan valide. La révision du plan : sert à tester le succès de réparation si une panne survient lors de l'exécution. L'apprentissage consiste à stocker le plan comme un nouveau cas dans la base de cas.

3.6 Conclusion

Dans ce chapitre, nous avons présenté le paradigme de Raisonnement à Partir des Cas à travers ses origines, ses principes et son cycle en explicitant chacune de ses phases. En effet, la remémoration de problèmes a pour but de rechercher une solution à un nouveau problème en exploitant une base de cas. Cette recherche est une phase cruciale dans le cycle RàPC. La remémoration s'appuie sur des mesures de similarité pour identifier les cas sources similaires relatifs à un problème posé. Les autres étapes du cycle RàPC représentent aussi des phases indispensables pour arriver à de meilleurs résultats.

Nous nous somme intéressé par la suite à la planification basée sur les cas. Cette technique se base sur la réutilisation des plans passés pour le développement de nouveaux plans.

Dans le chapitre suivant, nous présenterons un état de l'art sur les approches de composition de services web en se basant sur la méthode RàPC et la planification.

Chapitre 4

La composition des services web basée sur le RàPC

Sommaire

4.1 Introduction

Dans ce chapitre, nous allons présenter les différentes recherches effectuées dans le domaine de la composition de services web en se basant sur le raisonnement à partir de cas. Les travaux dans la littérature se sont intéressés à l'optimisation du temps de découverte et à la composition de services Web en exploitant le raisonnement à Partir des Cas (RàPC). Nous étudions leurs détails dans les sections qui suivent dans le souci de dégager leurs caractéristiques et pouvoir les comparer. Nous allons voir également, dans la section 4.1.1, les approches syntaxiques de composition de services web utilisant la technique RàPC. Ensuite, nous adressons dans la section 4.1.2 les approches de composition sémantiques de services. Dans la section 4.1.3, nous aborderons les approches qui ont utilisé la planification pour la composition de service web. Enfin, Nous discutons dans la section 4.2 les différents travaux présentés pour nous permettre de mieux positionner notre approche, et voir comment il serait possible d'adresser le problème de la composition de services web en tenant compte des aspects non-fonctionnels. Nous avons classé les différents travaux étudiés selon trois classes : les approches syntaxiques, sémantiques et à base de planification.

4.1.1 Approches syntaxiques

La première classe est composée des travaux syntaxiques [Limthanmaphon et al., 2003], [Cheng et al., 2006] et [Diaz et al., 2006] c'est-à-dire les travaux qui se basent sur une définition sans sémantique soit dans la représentation de cas ou dans le processus de la recherche ou d'adaptation.

4.1.1.1 Approche de Limtanmaphon et al, 2003

Le travail de Limtanmaphon et ses collègues [Limthanmaphon et al., 2003] est un des premiers travaux qui ont utilisé le RàPC pour la composition des services web. Les auteurs ont proposé un framework pour la composition des services web en utilisant la technique RàPC pour le processus de découverte de services. Ils présentent un modèle pour le processus de composition exécuté en deux modes, proactif et réactif comme expliqué dans la figure 4.1.

La phase proactive : dans cette phase un pré-assemblage de services pour la composition est proposé. Il inclut les services composants, les paramètres et les relations entre ses services.

La phase réactive : cette phase traite le processus de découverte et d'intégration des services composants existants. Le modèle de composition de services web est constitué de trois composants principaux : l'analyseur de requête (request analyst), les agents (outsource agents) et le composeur de services (services composer). Les relations entre les services sont définies et utilisées durant la phase proactive. La technique utilise le RàPC pour évaluer les requêtes des utilisateurs et planifier la composition de service.

Lors du traitement d'une nouvelle requête qui nécessite une composition de services, la mesure de similarité est utilisée pour trouver les cas les plus proches dans la base de cas. Les services sélectionnés sont les services avec la similarité la plus élevée et qui sont proposés aux utilisateurs.

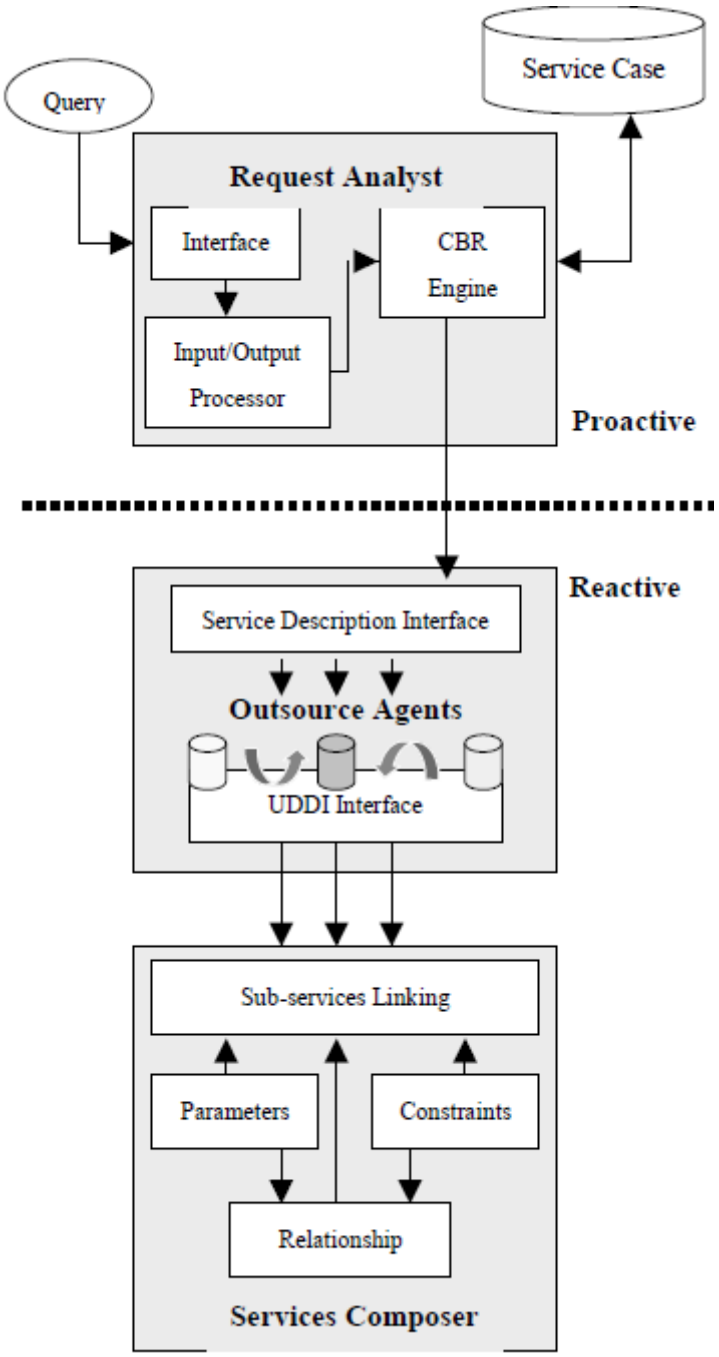

FIGURE 4.1: Architecture de l'approche proposée [Limthanmaphon et al., 2003].

4.1.1.2 Approche de Cheng et al, 2006

Cheng et ses collègues proposent une méthode de composition de service automatique [Cheng et al., 2006]. L'idée de base de cette méthode consiste à appliquer le Raisonnement à partir de cas à l'étape de découverte de la composition de service.

Dans le processus de composition de services, cette méthode peut selon les requêtes des utilisateurs rechercher, adapter des cas et procurer une composition de services.

La structure de la base des services est divisée en deux couches (figure 4.2). La première est la couche des services de base (basic service layer). Elle est liée à un domaine spécifique et les experts du domaine font la conception des cas de services. La deuxième couche est une couche d'application de service (application service layer). En fixant certaines contraintes individualisées, les services fournis par la première couche peuvent être utilisés directement par les utilisateurs dans la conception d'application sans se soucier de la façon dont les services sont combinés. Le cas dans la couche d'application est préconfiguré par des experts.

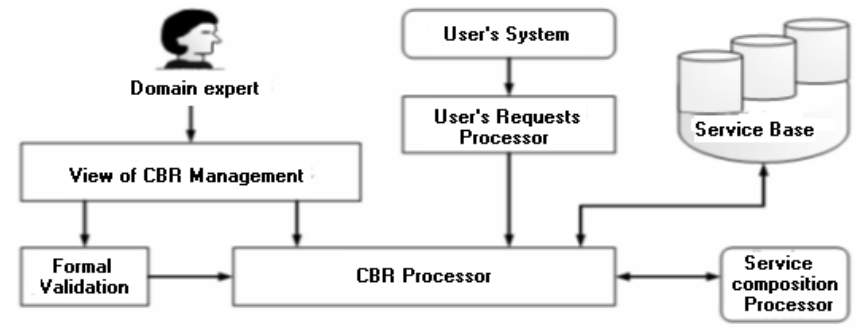

FIGURE 4.2: Approche proposée par [Cheng et al., 2006]

1. L'utilisateur pose sa requête dans le processeur de requête
2. Le processeur divise la requête à deux parties, les paramètres et les contraintes. Le résultat est fourni au processeur CBR.
3. Le processeur CBR va classer la requête de l'utilisateur selon les paramètres et les contraintes. Le résultat obtenu est utilisé dans la recherche des cas similaires.
4. Le processeur CBR interprète les descriptions au contrôleur logique de relations.
5. Le processeur de composition de service prend des décisions selon le contrôleur logique de relations.

6. Si l'utilisateur est satisfait du service composé, les relations logiques vont être enregistrées comme un cas dans la base des cas de services. Selon les réponses des utilisateurs, les experts du domaine vont compléter et modifier la base de cas.

4.1.1.3 Approche de Diaz et al, 2006

Le RàPC est utilisé comme un mécanisme de classification et de sélection pour approuver la précision des résultats trouvés dans la recherche de services web. La recherche est basée sur les caractéristiques fonctionnelles des services web. Le modèle proposé [Diaz et al., 2006] est composé des documents WSDL et UDDI et une base de cas qui classe les services web selon leurs fonctionnalités (voir figure 4.3).

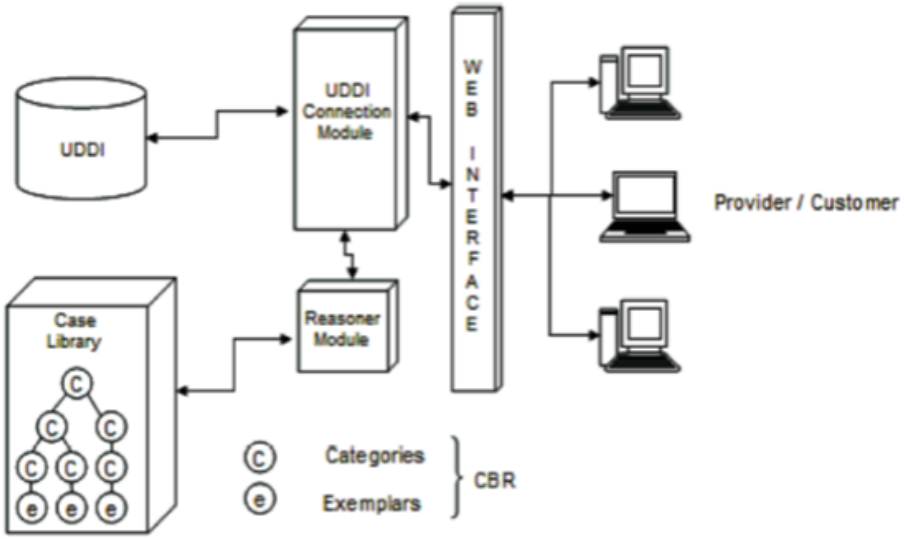

FIGURE 4.3: Approche proposée par [Diaz et al., 2006]

Quand un service web est enregistré, les étapes suivantes sont exécutées :

1. La première étape consiste à fournir les informations nécessaires par l'interface web (Web Interface). Ces informations sont le document WSDL et les caractéristiques fonctionnelles du service web ;
2. Le module de connexion (connexion module) génère un identificateur unique et enregistre les informations dans l'UDDI correspondant. Après l'enregistrement

du service dans l'annuaire UDDI, l'identificateur unique est envoyé au module de raisonnement (reasoner module) ;

3. Le module de raisonnement détermine la catégorie où le nouveau cas représenté par le web service peut être indexé en se basant sur les caractéristiques données dans la première étape et l'identificateur ;
4. A la fin de l'étape quatre, le service web est enregistré dans l'annuaire UDDI et la base de cas est mise à jour.

4.1.2 Approches sémantiques

La deuxième classe contient les approches qui ont introduit la sémantique dans les différentes étapes de la composition de service web [Lajmi et al., 2006], [Lajmi et al., 2009], [Thakker et al., 2007], [Liu et al., 2009] et [Sun et al., 2011].

4.1.2.1 Approche de Lajmi et al, 2006

Dans le but de réaliser une plateforme pour la composition de services web, Lajmi et ses co-auteurs [Lajmi et al., 2006] et [Lajmi et al., 2009] ont proposé le système WeSCo-CBR basé principalement sur les ontologies et le RàPC. Pour apporter un guidage semi-automatique à l'utilisateur, les auteurs ont défini une ontologie OWL-S qui décrit les différentes fonctionnalités des services web à base de OWL. Ainsi, dans le but de faciliter le traitement d'une requête utilisateur, ils procèdent par la transformation de cette requête sous une forme compréhensible et manipulable par la machine. Cette étape permet alors de représenter la requête par une formule ontologique sous la forme d'un ensemble de concepts de l'ontologie que les auteurs ont défini. Pour ce faire, ils divisent la requête en trois parties :

- Instances : Une requête peut contenir un ensemble de données. Ces données peuvent être considérées comme des valeurs d'attributs d'un ou plusieurs objets. La

partie des instances de la requête est représentée conformément aux classes de ses objets.

- Variables : Une requête peut contenir des variables. Ces variables peuvent être considérées comme des attributs d'une ou plusieurs classes qui représentent la partie des variables de la requête.

- Activités : C'est l'ensemble des activités abstraites (fonctionnalités) d'une requête. En effet, les activités d'une demande sont déduites des classes de variables et d'instance.

Après avoir défini les instances et les variables de la requête, la recherche d'activités est lancée afin d'obtenir toutes les activités requises. Dans le système WeSCo-CBR, un cas est composé des trois éléments suivants :

- Le problème qui regroupe quatre parties, à savoir le profil utilisateur, les activités, les variables et les instances ;
- La solution qui est l'ensemble des activités sous forme d'un schéma de composition ;
- L'évaluation qui est le taux de pertinence de la solution.

Le processus de réutilisation dans WeSCo-CBR consiste, pour une nouvelle requête, à récupérer un cas antérieur similaire mémorisé et éventuellement évaluer et mémoriser le nouveau cas. Une requête donnée peut exiger un ensemble d'activités abstraites reflétant des services Web de différents domaines métiers. C'est ainsi que les auteurs proposent d'organiser la base de cas par domaine métier afin de réduire l'espace de recherche, pour une activité abstraite donnée, au domaine métier correspondant. Pour chaque nouvelle requête, le système cherche les cas similaires et en sélectionne le cas pertinent, en fonction de sa ressemblance avec la requête.

4.1.2.2 Approche de Thakker et al, 2007

Les auteurs [Thakker et al., 2007] ont proposé une approche basée sur RàPC pour la composition de services web. Leur framework présenté dans la figure 4.4 se

base sur la sémantique pour la recherche et la sélection des services. L'utilisation de la sémantique permet l'extensibilité et la réutilisabilité. Ils ont implémenté un outil RàPC sémantique, qui capture les expériences d'exécution de services et les considère comme des cas. Ils utilisent ces cas dans la recherche de solution pour les nouveaux problèmes. Ensuite, les ontologies sont utilisées dans l'implémentation du framework. La sémantique est utilisée dans la description des paramètres du problème et dans l'implémentation des composants du système RàPC.

Dans l'architecture proposée, il existe deux rôles principaux qui sont l'administrateur de cas (case administrator) et le demandeur de cas (case requestor). L'administrateur de cas est responsable de la maintenance de la base de cas. Le demandeur de cas est chargé de rechercher dans la base de cas des solutions pour le problème. La figure 4.4 illustre le framework proposé par les auteurs.

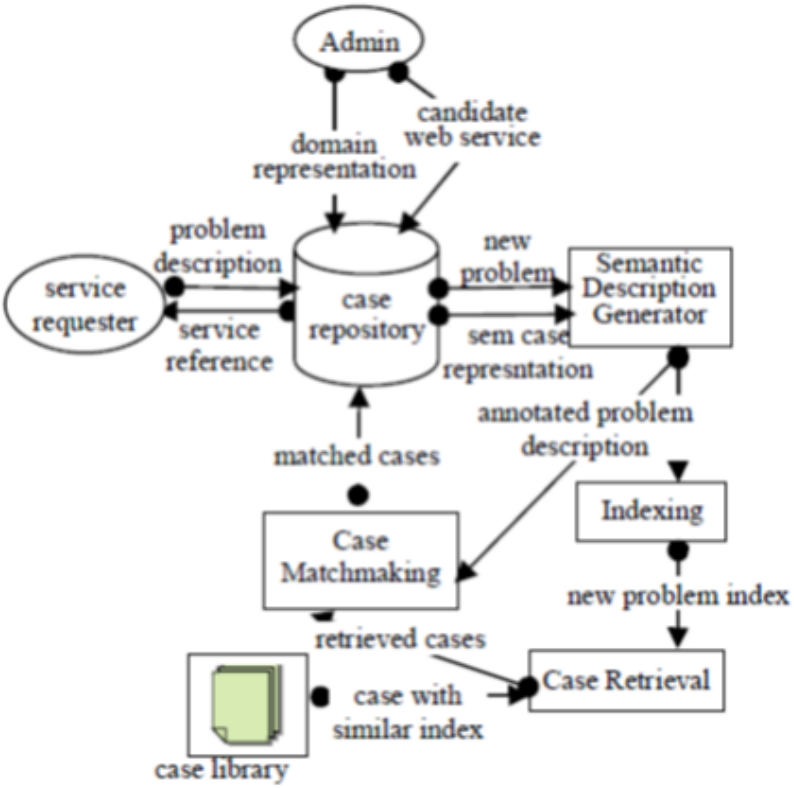

FIGURE 4.4: Approche proposée par [Thakker et al., 2007].

1. L'administrateur remplit le répertoire avec des cas enrichis de représentations

sémantiques pour la requête de l'utilisateur, les services et les expériences d'exécution de services web ;

2. L'utilisateur donne en entrée les exigences de service et reçoit les références des services web à travers l'interface ;
3. Le système recherche des cas similaires et le résultat avec le problème est renvoyé au module de génération de description sémantique pour annoter le problème ;
4. Le problème annoté est transmis au module d'indexation qui va calculer l'index correspondant au nouveau problème. L'index est transmis à la recherche de cas ;
5. Le module de recherche de cas essaie de trouver les cas avec des index similaires ;
6. Aprés que les cas ont été trouvés et le problème annoté le module donne le ou les services sélectionnés.

Afin de permettre la récupération rapide des cas appropriés lors de la procédure de recherche, les auteurs proposent d'organiser la base de cas en fonction de certains termes du vocabulaire représentant des caractéristiques (features) des services, ce qui permet de la subdiviser en un nombre de partitions égales au nombre d'instances du vocabulaire des features utilisées [Thakker et al., 2007]. Ainsi, sur la base du vocabulaire utilisé dans la description du nouveau problème, ce système détermine la partition de la base de cas qui concerne ce problème. Ensuite, un algorithme de matching est appliqué sur les cas de la partition sélectionnée. Pour ce faire, ils appliquent une fonction de similarité qui permet de calculer le degré global de similarité pour chaque cas remémoré.

4.1.2.3 Approche de Liu et al ,2009

Une méthode pour la composition de services web avec l'utilisation du raisonnement à partir de cas est proposée par les auteurs [Liu et al., 2009] où ils utilisent OWL-S pour la représentation des cas. Pour la phase de recherche, les auteurs présentent une méthode de similarité dirigée par la sémantique. Une stratégie d'adaptation est présentée ainsi qu'une méthode d'indexation pour la base de cas pour une meilleure réutilisation des cas dans le système RàPC. La représentation commune des cas doit toujours contenir au minimum deux parties :

- La partie description de service appelée problème composée des entrées et des sorties.
- La partie solution au problème qui représente le processus de composition.

La recherche des cas similaires au problème de la requête de l'utilisateur dans la base de cas est divisée en deux étapes : la recherche du cluster et la recherche du cas proche à la requête. Dans la première étape, la recherche est indexée. La base de cas est composée de plusieurs clusters. Après le choix du cluster, le problème est comparé à tous les cas appartenant à ce dernier pour trouver le plus similaire. Si la solution ne correspond pas exactement à la demande de l'utilisateur une adaptation est alors proposée.

1. Lorsque le système reçoit la requête d'un utilisateur, le système doit retrouver les cas dont la partie description de problème est similaire au problème posé ;
2. Le système calcule alors la distance entre la requête de l'utilisateur et les indices pour trouver le cluster le plus proche ;
3. Après avoir trouvé le cluster, une autre recherche est lancée pour calculer la similarité entre la requête et les cas qui se trouvent dans ce cluster. le système choisit le cas le plus similaire au problème posé par l'utilisateur ;
4. Dans la phase de recherche un seul service est obtenu dont la partie problème est similaire à celle de la requête de l'utilisateur mais nécessitant toutefois une

adaptation.

4.1.2.4 Approche de Sun et al,2011

Dans le but de proposer une approche pour faire la découverte, la composition et recommandée des services web en utilisant le RàPC, les auteurs proposent [Sun et al., 2011] un système appelé CWSR (Case Web web Service Reasoner). Ce système (voir figure 4.5)est proposé pour faire toutes les tâches précédentes.

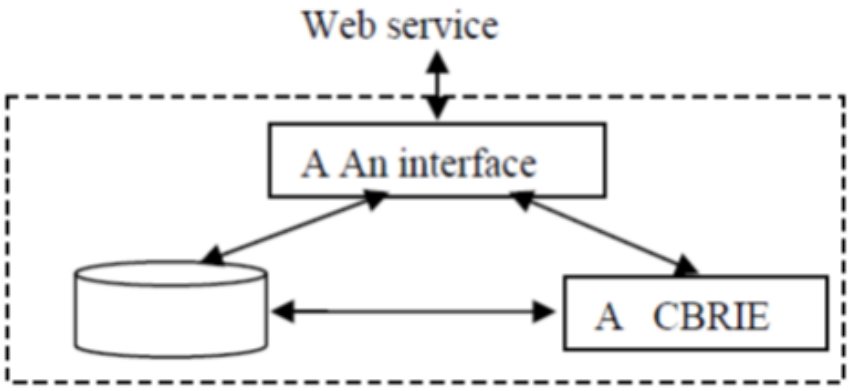

FIGURE 4.5: L'architecture du système CWSR [Sun et al., 2011]

L'architecture du système est composée de :

Un agent interface : L'agent interface se compose de certains types de systèmes de traitement de langage naturel qui permettent à l'utilisateur d'interagir avec les stratégies de service Web.

Une base globale de services web (GWB) : est constituée de tous les cas de services Web que le système recueille périodiquement et les nouveaux services Web découverts lorsque le système est en cours d'exécution.

Un outil d'inférence (CBRIE) : comprend le mécanisme de manipulation de la GWB pour inférer des services Web sur la base de RàPC. Quel que soit la demande de l'utilisateur : la découverte, la composition, la recommandation, etc.

4.1.3 Approches par planification

La dernière classe qui contient les approches [Feng , 2009], [Lee et al., 2010], [Henni et al., 2013] et [Torres et al., 2014] combinent la planification avec le raisonnement à partir de cas pour la composition de services web.

4.1.3.1 Approche de Feng, 2009

Dans ce travail, les auteurs [Feng , 2009] proposent une approche pour la composition fonctionnelle et automatique des services web sémantique basée sur la méthode de planification à partir de cas. Leur approche génère le plan de la nouvelle demande de l'utilisateur en adaptant automatiquement le plan existant du problème déjà résolu. Le plan nouvellement construit conjointement à la nouvelle demande peut maintenant être stocké comme un cas dans la base de cas pour une réutilisation future.

L'objectif de cette approche est d'adapter la solution trouvée. Un cas contient la solution à un problème résolu, il est constitué de même composants incluant le problème, la solution qui est un plan et une évaluation. Dans le contexte des services web, la requête de l'utilisateur est le problème et le plan qui répond à la requête représente la solution au problème.

Dans la phase de recherche, les auteurs ont proposé d'utiliser plusieurs mesures de similarités afin de trouver la meilleure solution. Une étape d'ajustement du plan est proposée avant de passer à l'adaptation.

4.1.3.2 Approche de Lee et al, 2010

Les auteurs [Lee et al., 2010] proposent une approche pour la construction d'un modèle de but et l'extraction des intentions à partir des requêtes tout en se basant sur la satisfaction de ses intentions et en combinant les services internes et externes

pour répondre aux besoins des utilisateurs. Le service interne représente les fonctions systèmes conçues pour répondre aux besoins des utilisateurs. Quant au service externe, il représente les services fournis par les fournisseurs de service externe.

Ces auteurs utilisent la planification pour combiner les deux types de services et utilisent le RàPC pour stocker la planification et les autres données et créer ainsi rapidement, de nouvelles planifications quand les utilisateurs présentent des besoins similaires.

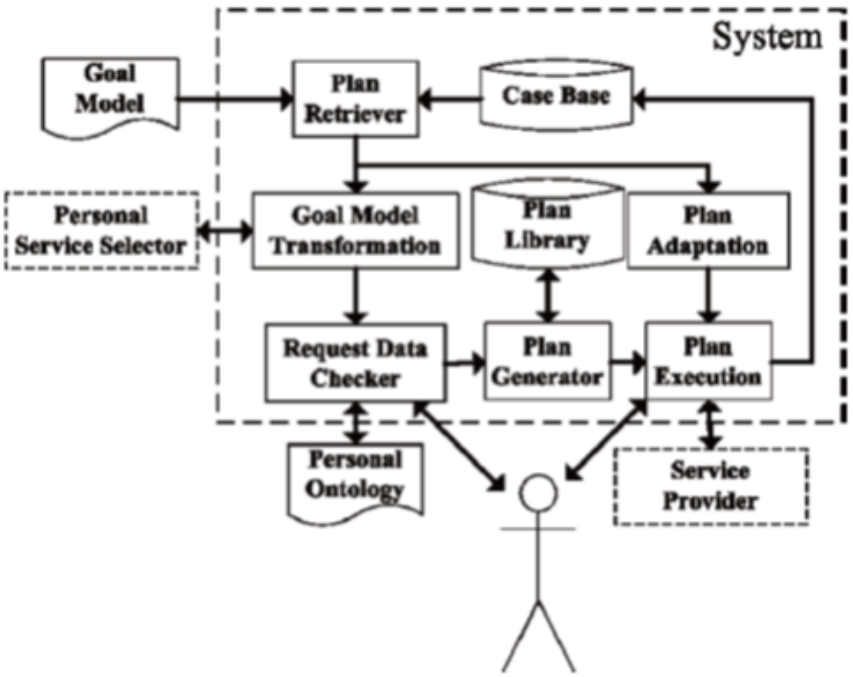

FIGURE 4.6: Architecture du système [Lee et al., 2010]

1. La recherche de plan se fait par le module retriever. Il cherche les cas similaires et envoie les modèles de but au vérificateur. S'il existe plusieurs solutions, celles-ci sont envoyées à l'utilisateur ;
2. Le transformateur utilise les actions et les champs objet dans un modèle but pour comprendre quel type de fournisseur cherche un utilisateur. Si l'utilisation de services externes est nécessaire, les paramètres relatifs à l'entrée et à la sortie sont donnés à la sélection de service personnel ;

3. Le vérificateur demande si l'information est complète pour exécuter un service . Sinon, alors l'ontologie personnelle est d'abord consultée. Si plus d'information s'avère nécessaire, l'utilisateur est invité à saisir des informations ;
4. Générateur de plan utilise JSHOP2 pour générer des plans basés sur le domaine du problème préalablement défini ;
5. L'adaptation du Plan vérifie les différences entre le modèle de but et le cas récupéré dans la base de cas, puis utilise des règles pour modifier la solution ;
6. L'exécution vérifie comment un plan peut être exécuté. Quand un plan est exécuté, une évaluation suit cette opération.

4.1.3.3 Approche de Henni et al, 2013

Dans leur travail, les auteurs [Henni et al., 2013] proposent d'utiliser la planification et le raisonnement à partir de cas pour la composition de service web dynamique. Ils présentent un scénario comme une application directe (le système national de suivi de la vaccination des enfants) de leur proposition. L'utilisation de CBR fournit un moyen de mémoriser les expériences passées afin de réutiliser les solutions précédentes. D'autre part, l'utilisation de la planification peut offrir des solutions en l'absence de cas similaires précédents existant ou si la solution proposée ne répond pas aux besoins de l'utilisateur.

Le traitement d'une nouvelle requête passe par plusieurs étapes :

1. La nouvelle requête est introduite via l'interface des utilisateurs. Cette requête est considérée comme un nouveau cas et est annotée sémantiquement à travers OWL-S.
2. Le module de recherche essaie de trouver une solution dans la base de cas.
3. Si la solution ne correspond pas exactement à la demande de l'utilisateur, alors une adaptation est nécessaire.

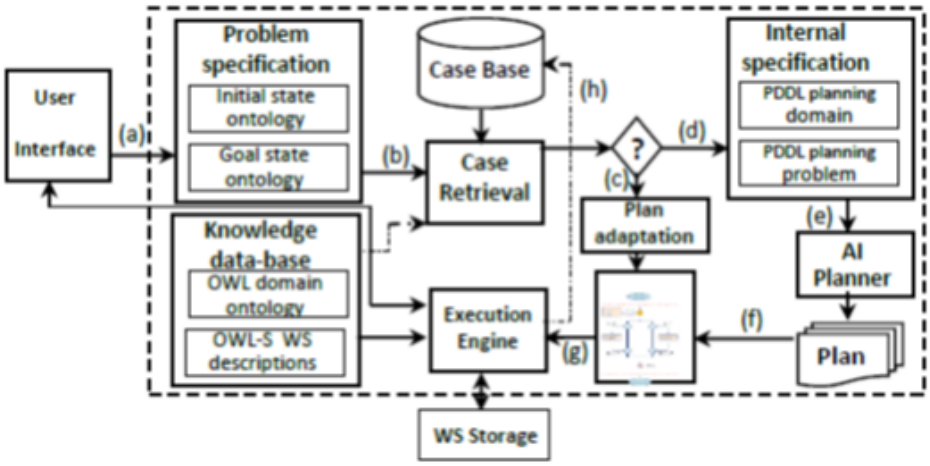

FIGURE 4.7: Architecture de la solution proposée [Henni et al., 2013]

4. Dans le cas où le système ne trouve pas de solution, le nouveau problème est traduit à un problème de planification. Un planificateur est utilisé pour trouver la nouvelle solution.
5. Le plan généré est transformé à OWL-S afin d'être exécuté.
6. L'outil d'exécution concrétise le nouveau service trouvé et retourne la solution à l'utilisateur.
7. Selon l'évaluation de la solution, le nouveau cas peut être enregistré dans la base des cas.

4.1.3.4 Approche de Torres et al, 2014

Les auteurs [Torres et al., 2014] proposent d'ajouter une étape d'apprentissage à un modèle de composition de services web intitulé INDYGO. Ils appliquent les techniques du web sémantique et la planification de l'intelligence artificielle dans les modèles de composition de services web. Leur méthode propose d'utiliser la méthode RàPC pour sauvegarder les compositions obtenues du modèle INDYGO et d'utiliser cette base de cas pour trouver des compositions rapidement.

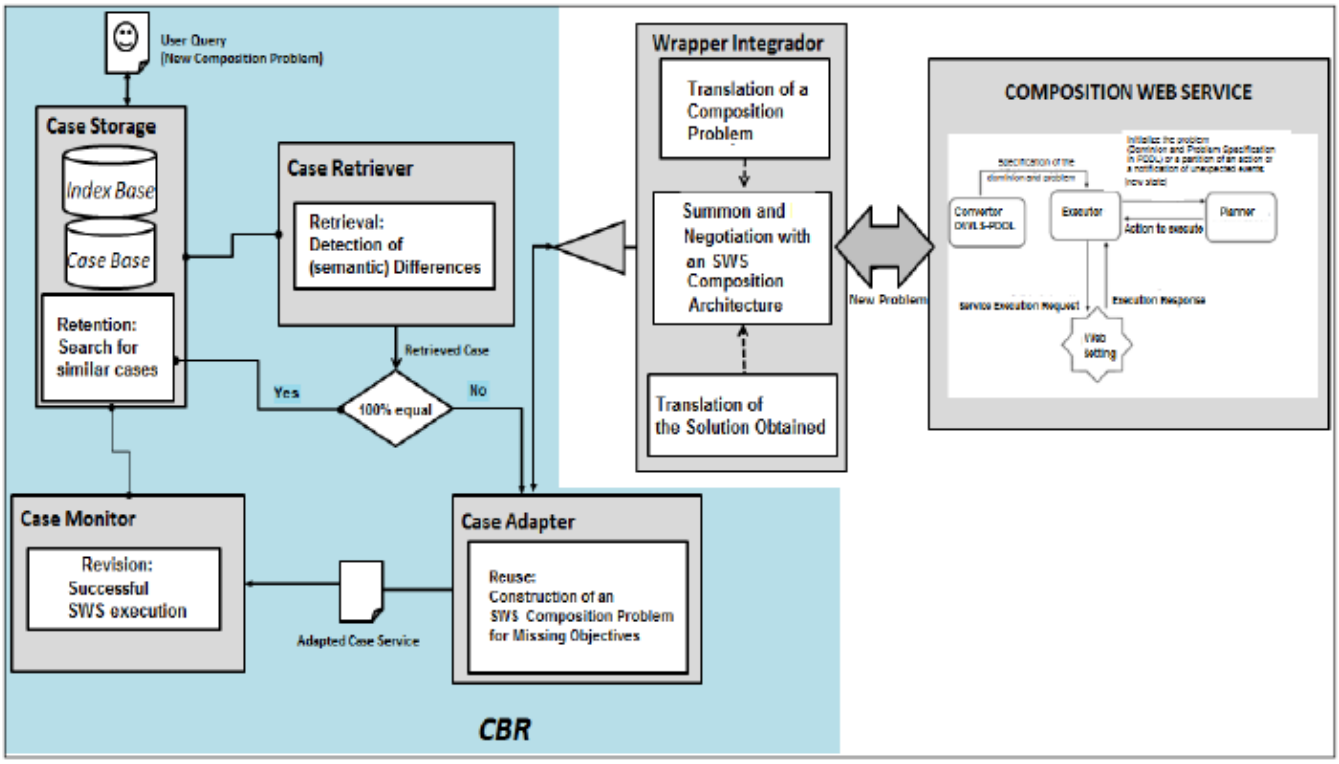

FIGURE 4.8: Modèle intégré CBR-INDYGO [Torres et al., 2014].

4.2 Etude comparative

Dans cette section, nous présentons une étude comparative entres les différentes approches étudiées. Nous identifions six critères pour comparer les divers travaux à base de RàPC présentés auparavant. Nous mettons le point sur des critères spécifiques pour l'évaluation des approches fondées sur ce type de raisonnement. Ces mécanismes constituent notamment des piliers de toute approche de composition de services Web à base de RàPC. Les critères de comparaison que nous avons identifiés sont les suivants : la représentation du cas, l'annotation sémantique, le type d'appariement des cas et mesure de similarité, le type d'adaptation, l'organisation de la base de cas, l'utilisation de l'aspect non-fonctionnel, la nature et le nombre de cas extraits.

Les tableaux 4.1 à 4.6 présentent une synthèse des résultats de notre étude comparative des travaux à base de RàPC que nous avons présentés. Ces tableaux comparatifs permettent en fait de montrer les caractéristiques de chacun de ces travaux.

4.2.1 La représentation du cas

Ce critère correspond à la formalisation du cas de service Web. Autrement dit, il fait référence aux données ou connaissances représentées dans un cas. Nous pensons que la formalisation d'un cas doit de préférence être alignée avec les standards des services Web. Le tableau 4.1 montre les différentes représentations d'un cas dans les approches étudiées.

Classe	Approche	La représentation du cas
Syntaxiques	Limthanmaphon et al, 2003	Le cas est un ensemble de services pré-assemblés S : (N, D)
	Cheng et al, 2006	Paramètres et contraintes
	Diaz et al, 2006	Caractéristiques fonctionnelles des services
Sémantiques	Lajmi et al, 2006	(profile utilisateur, Activities, Variables, Instances, ensemble d'activités, évaluation)
	Thakker et al, 2007	(Entrée, Sortie, Preference, contraintes, Solution)
	Liu et al, 2009	(Problème, Solution)
	Sun et al, 2011	(C=(p,q) p est la description structurée du service et q est la description du service solution)
Planification	Feng, 2009	(Problème, Solution, evaluation, count)
	Lee et al, 2010	(action, object, et constraintes)
	Henni et al, 2013	(description du problème, solution, résultat et justification)
	Torres et al, 2014	(description du cas, caractéristiques, problème, solution, historique)
	Hachemi et al, 2015	(Problème, Solution, Qos et compteur)

TABLE 4.1: Représentation des cas.

Nous résumons les différentes représentations des cas manipulés durant le cycle RàPC dans le tableau 4.1. A partir de ce tableau, nous pouvons constater que les auteurs définissent des représentations selon les besoins de leurs systèmes de composition. Nous remarquons aussi que la représentation du cas s'est améliorée au fur et à mesure du temps et des outils de manipulation des services. Au début, les représentations ne contenaient pas beaucoup d'informations qui sont selon notre

avis indispensables. Le cas ne contenait au début que le nom de service et sa description [Limthanmaphon et al., 2003]. Les travaux qui l'ont suivi utilisaient les paramètres fonctionnels des services web [Diaz et al., 2006], [Cheng et al., 2006]. Par la suite les auteurs [Lajmi et al., 2006], [Liu et al., 2009], [Thakker et al., 2007] et [Sun et al., 2011] ont introduit d'autres éléments comme une évaluation ou d'autres contraintes. Les travaux de [Feng , 2009], [Lee et al., 2010], [Henni et al., 2013] et [Torres et al., 2014]ont adaptés leurs cas selon la reformulation du problème de composition en un problème de planification. Ce que nous pouvons conclure de cette comparaison est que dans la plupart des travaux, les auteurs ont utilisés l'aspect fonctionnel des services web dans leurs représentations sauf pour [Thakker et al., 2007] qui a introduit les préférences. Malgré l'influence de l'aspect non-fonctionnel sur le résultat de la composition et sur ce que l'utilisateur attend du système.

4.2.2 Le type d'appariement des cas

La remémoration de cas de services Web est la phase fondamentale du processus de découverte à base de RàPC. Elle est fondée sur la recherche des cas qui peut être syntaxique ou sémantique. La pertinence des résultats et du processus de découverte dépend de la pertinence des mesures de similarités utilisées. Dans ce tableau 4.2, nous avons montré les différentes mesures de similarités utilisées.

Dans la phase de remémoration, les chercheurs utilisent différentes mesures de similarité selon leurs besoins. Les mesures de similarités diffèrent selon les représentations des cas élaborés par les approches proposées. Nous remarquons que dans les approches dites syntaxiques que les mesures de similarités utilisées omettent l'aspect sémantique dans la phase de recherche ce qui rend la recherche purement syntaxique ce qui va influer sur la qualité de la composition par la suite. Les approches sémantiques utilisent des mesures de similarités dirigées par la sémantique. Lajmi [Lajmi et al., 2006] utilise une mesure de similarité qui prend seulement une partie du problème (entrées et sorties) et néglige (les prés conditions et les effets). Tout

Classe	Approche	Similarité
Syntaxiques	Limthanmaphon et al, 2003	Nom de service
	Cheng et al, 2006	La distance euclidienne pondérée
	Diaz et al, 2006	Algorithme se basant sur les caractéristiques
Sémantiques	Lajmi et al, 2006	Mesure de Manhattan adaptée
	Thakker et al, 2007	Plus proche voisin
	Liu et al, 2009	Distances sémantiques
	Sun et al, 2011	The inference engine
Planification	Feng, 2009	L'algorithme de Kuhn-Munkres et Init-Goal matching
	Lee et al, 2010	Méthode de matching
	Henni et al, 2013	Mesure de similarité basée sur les poids des attributs de la description et de la solution
	Torres et al, 2014	Mesure de similarité sémantique
	Hachemi et al, 2015	Mesure de similarité pondérée et syntaxique

TABLE 4.2: Mesure de similarité.

comme Thakker [Thakker et al., 2007] qui aborde la mesure en général. Quant à Lee [Lee et al., 2010], il ne traite pas la mesure de similarité en détail. Alors que Feng [Feng , 2009] traite une grande partie du problème, mais toujours sans l'utilisation des préférences de l'utilisateur.

4.2.3 Technique d'adaptation

L'adaptation des cas sélectionnés de la base des cas dans le cycle RàPC est une étape cruciale. Puisque les cas trouvés ne correspondent pas tout le temps aux exigences du demandeur de services, un arrangement est nécessaire pour rendre la solution plus adéquate. Le tableau 4.3 montre les méthodes d'adaptation utilisées dans chaque approche.

En ce qui concerne la phase d'adaptation, qui consiste à reconstruire de nouvelles solutions pour les problèmes rencontrés, elle se révèle être une phase sensible. Car, soit les systèmes ne présentent pas d'adaptation, soit les systèmes proposent de simples règles d'adaptation ou des graphes d'expériences ou encore utilisent une

Classe	Approche	Technique d'adaptation
Syntaxiques	Limthanmaphon et al, 2003	/
	Cheng et al, 2006	/
	Diaz et al, 2006	/
Sémantiques	Lajmi et al, 2006	/
	Thakker et al, 2007	Substitution
	Liu et al, 2009	Différentes stratégies d'adaptation
	Sun et al, 2011	adaptation du problème et de la solution
Planification	Feng, 2009	ESPA
	Lee et al, 2010	Substitution
	Henni et al, 2013	Adaptation transformationnelle.
	Torres et al, 2014	Technique d'ajustement
	Hachemi et al, 2015	E-MPA

TABLE 4.3: Méthodes d'adaptation.

structure hiérarchique de la base de cas afin de réaliser une adaptation substitutionnelle au niveau des descripteurs de cas. Donc ce que nous pouvons dire est que peu de travaux traitent le problème de l'adaptation qui demeure au cœur du raisonnement à partir de cas.

4.2.4 L'organisation de la base de cas

Les cas sont classés dans une mémoire appelée base de cas. Afin de faciliter la recherche du cas le plus approprié au problème posé, il faut organiser la base de cas. Le choix de la méthode d'organisation est important pour la remémoration rapide des cas. En effet, l'organisation de la base doit permettre d'accéder plus rapidement aux cas adéquats en fonction des éléments du contexte du problème cible. Notamment, lors de la recherche des cas, c'est la partie problème qui est sollicitée. Dans le tableau 4.4, nous avons montré les méthodes utilisées pour l'organisation de la base de cas.

La base de cas est l'endroit où les cas sont enregistrés pour une réutilisation ultérieure. Pour pouvoir gérer cette masse de données, les travaux ont suggéré un

Classe	Approche	La méthode utilisée pour la base de cas
Syntaxiques	Limthanmaphon et al, 2003	/
	Cheng et al, 2006	clustering
	Diaz et al, 2006	Réseaux de discrimination
Sémantiques	Lajmi et al, 2006	Partitionnement par domaine métier
	Thakker et al, 2007	Partitionnement
	Liu et al, 2009	Algorithme PAM (Partitioning Around Medoids)
	Sun et al, 2011	/
Planification	Feng et al, 2009	/
	Lee et al, 2010	/
	Henni et al, 2013	/
	Torres et al, 2014	Indexation
	Hachemi et al, 2015	/

TABLE 4.4: Organisation de la base de cas.

partitionnement en utilisant différentes méthodes. Ce partitionnement selon les approches facilite la tâche de recherche en termes de temps. Pour pouvoir gérer la base de cas, nous constatons que la plupart des systèmes ont proposé des méthodes comme le clustering, l'utilisation de graphe ou d'autres approches de partitionnement. Dans les travaux de planification, les auteurs n'ont pas utilisé ou proposé des méthodes pour organiser la base de cas.

4.2.5 Cycle du RàPC

Puisque notre système est à base de RàPC, nous avons établi un tableau récapitulatif qui représente les différentes étapes du cycle RàPC qui sont : l'élaboration, la remémoration, l'adaptation, la révision et l'apprentissage dans les différents travaux cités au paravent. Le tableau 4.5 montre si l'étape du cycle de RàPC a été ou non traitée dans les approches proposées pour la composition de services web.

Nous remarquons que peu de travaux traitent l'ensembles des différentes étapes du raisonnement à partir de cas. Pour la phase d'élaboration, la majorité des travaux

Approche	Élaboration	Remémoration	Adaptation	Révision	Apprentissage
Limthanmaphon et al, 2003	Oui	Oui	Non	Non	Non
Cheng et al, 2006	Oui	Oui	Non	Non	Oui
Diaz et al, 2006	Non	Oui	Non	Non	Oui
Lajmi et al, 2006	Oui	Oui	Non	Non	Oui
Thakker et al, 2007	Oui	Oui	Oui	Oui	Oui
Feng et al, 2009	Non	Oui	Oui	Oui	Oui
Sun et al, 2009	Non	Oui	Oui	Non	Oui
Liu et al, 2009	Non	Oui	Oui	Non	Non
Lee et al, 2010	Oui	Oui	Oui	Non	Oui
Henni et al, 2013	Non	Oui	Oui	Non	Oui
Torres et al, 2014	Non	Oui	Oui	Non	Oui
Hachemi et al, 2015	Oui	Oui	Oui	Oui	Non

TABLE 4.5: Cycle de RàPC

proposent de transformer la requête en une manière plus adéquate pour le système. Nous pouvons remarquer que tous les travaux utilisent la phase de remémoration pour la recherche des services web. Pour l'adaptation, il est aisé de constater que les auteurs au début ne s'intéressent pas à cette phase contrairement aux travaux élaborés ces dernières années. Néanmoins, peu de travaux ont abordé la phase de révision, en effet, elle nécessite une recomposition si la solution ne satisfait pas les exigences de l'utilisateur. La dernière phase est l'apprentissage, c'est la phase qui permet au système d'apprendre continuellement et est abordée par presque tous les travaux.

4.2.6 Autres critères de comparaison

Pour que notre synthèse soit complète, nous avons fais appel à d'autres critères de comparaison pour mieux comprendre les travaux établis dans le contexte de la

composition des services web en utilisant le RàPC.

L'annotation sémantique L'annotation sémantique désigne le langage utilisé pour décrire sémantiquement les services Web et les cas. L'étude présentée au chapitre 3 dresse les différents langages de description sémantique des services web.

Nature de composition Représente la nature de la composition : automatique, semi-automatique ou manuelle.

Nombre de cas extrait Ce critère représente le nombre de cas extraits dans la phase de remémoration dans chaque approche pour être réutilisé par la suite.

Aspect non-fonctionnel La plupart des approches de compositions de services web se basent sur l'aspect fonctionnel des services c'est-à-dire le comportement de ces derniers et en négligeant l'aspect non fonctionnel. Ce critère montre les aspects pris dans la composition.

Pour commencer, nous dirons que les approches syntaxiques n'utilisent aucun langage de description de services web sémantique sauf pour [Diaz et al., 2006] qui utilise OWL-S pour représenter la base de cas seulement. Les autres approches utilisent le langage OWL-S pour la description des services web. La nature de la composition diffère d'une approche à une autre mais nous remarquons que les approches sont presque toutes automatiques ce qui exclut l'utilisateur dans tout le processus de la composition. Un autre point très important qui est l'utilisation ou non de l'aspect non-fonctionnel des services web dans la composition des services web. Nous remarquons qu'il n'est pas pris en considération dans la plupart des travaux sauf pour [Thakker et al., 2007] et [Lee et al., 2010] qui considèrent les préférences de l'utilisateur, mais pas durant tout le processus de la composition.

Classe	Approche	Sémantique	Nature	Aspect non-fonctionnel	Nbr de cas extrait
Syntaxiques	Limthanmaphon et al, 2003	Non	Automatique	Non	Un cas
	Cheng et al, 2006	Non	Automatique	Non	Un cas
	Diaz et al, 2006	Oui	Automatique	Non	Plusieurs cas
Sémantiques	Lajmi et al, 2006	OWL-S	Semi-automatique	Non	Un cas
	Thakker et al, 2007	OWL-S	Automatique	Oui	Plusieurs cas
	Liu et al, 2009	OWL-S	Automatique	Non	Un cas
	Sun et al, 2011	/	Automatique	Non	Plusieurs cas
Planification	Feng et al, 2009	OWL-S	Automatique	Non	Un cas
	Lee et al, 2010	OWL-S	Semi-automatique	Oui	Un cas
	Henni et al, 2013	OWL-S	Automatique	Non	Un cas
	Torres et al, 2014	OWL-S	Automatique	Non	Un cas
	Hachemi et al, 2015	OWL-S	Automatique	Oui	Plusieurs cas

TABLE 4.6: Tableau comparatif.

4.3 Synthèse

Cette étude comparative nous a permis de situer notre travail et de faire des choix quant à la création d'un système de composition de service web en utilisant le raisonnement par mémoration RàPC. Nous proposons une approche de composition de service web sémantique en utilisant la planification à partir des cas.

La représentation du cas est élaborée en prenant en considération l'aspect fonctionnel des services web et les préférences des utilisateurs. En outre, la partie problème du cas est structurée d'une manière à ressembler un problème de planification avec l'ajout des préférences des utilisateurs. De plus, la partie solution ressemble à un plan de composition de service web. Enfin, d'autres champs ont été ajoutés au

cas comme la qualité et le compteur.

Dans la phase de remémoration, nous exploiterons la base de cas pour trouver les meilleures solutions. Dans notre travail, cette phase se divise en deux étapes : recherche et sélection. Dans ces deux étapes, le système essaie de trouver les meilleurs cas qui sont similaires au problème posé. Ainsi, notre méthode de remémoration présente plusieurs avantages :

- En cas d'absence de correspondance exacte à notre problème, le système procède par décomposer le problème d'une manière à trouver tous les cas qui répondent au problème (une partie du problème). L'implication des préférences des utilisateurs pour sélectionner les meilleures solutions pour la composition est présente et l'utilisateur pourra déterminer parmi ces résultats si la solution ne lui correspond pas.
- Pour l'étape de l'adaptation nous avons precédé à l'amélioration d'un algorithme d'adaptation qui utilise plusieurs solutions ou plan dans l'adaptation du résultat selon (et toujours) les préférences de l'utilisateur, grace à cet algorithme une nouvelle solution et produite.
- Notre système propose toujours des solutions aux problèmes posés même s'il ne trouve pas de cas qui sont similaires dans la base de cas. Ainsi, il procède à une nouvelle composition de service web en utilisant un des planificateurs qui prend en considération l'aspect fonctionnel du service composé et les préférences des utilisateurs. Par conséquent le système apprend un nouveau cas.

4.4 Conclusion

Cet état de l'art de la composition des services Web utilisant la technique du RàPC, nous a permis de tirer des conclusions relatives aux aspects et approches de composition des services Web. Il est à noter d'abord qu'une description exhaustive de ces services s'avère nécessaire pour favoriser, en particulier, les processus de leur

sélection et composition. Pour être la plus complète possible, elle doit tenir compte des classes de propriétés de services : fonctionnelles et non-fonctionnelles. L'utilisation du RàPC permet une réutilisation de ces compositions au lieu de refaire de nouvelles compositions à partir de zéro.

Cette étude comparative et les différentes limitations soulevées nous ont amené à proposer une nouvelle approche de composition de services Web. Notre proposition s'intégrera dans le contexte d'une approche fondée sur les principes du RàPC pour la composition des services Web. Nous souhaitons exploiter ce paradigme en alignement avec la planification de la composition des services Web, tout en visant, d'une part, à optimiser le temps de leur sélection et composition, et d'autre part, à améliorer la qualité des résultats, en permettant la recherche et la sélection de services appropriés répondant aux besoins fonctionnels de l'utilisateur, mais aussi à ses préférences et ses exigences particulières. Ainsi, nous envisageons de définir une architecture globale de cette plateforme, d'identifier et de cadrer ses composants. En particulier, l'idée est de définir des mécanismes d'exploitation de la base de cas de cette plateforme, en veillant à optimiser le processus de recherche des cas similaires, et de proposer des techniques de sélection des cas les plus appropriés et d'adapter les résultats trouvés.

Chapitre 5

Approche de composition de services web basée sur la PàPC et les préférences des utilisateurs

Sommaire

5.1 Introduction

Dans ce chapitre, nous présentons notre approche de composition de services Web. Dans la première partie, le problème de composition est défini comme un problème de planification. Même si plusieurs approches précédant ce travail ont déjà traité le problème de la composition comme un problème de planification, notre principale contribution consiste à étendre la définition du problème de composition de services et à réutiliser les solutions trouvées dans le passé en incluant les préférences des utilisateurs.

Dans notre modèle, la définition du problème de composition de services seb (actions en planification) est étendue pour gérer les aspects fonctionnels et non-fonctionnels des services web. Cette extension nous permet de répondre à de nouvelles requêtes dans lesquelles les buts peuvent être générés par le plan.

Pour l'implémentation de notre approche, nous avons étendu deux algorithmes pour la remémoration et l'adaptation. En utilisant ces algorithmes, le système construit le plan en explorant les services disponibles. Il retourne un plan qui est un ensemble de services permettant d'atteindre le but.

Ce chapitre est consacré à la description détaillée de notre approche. On commence par un exemple introductif suivi d'un ensemble de contributions. Après une présentation générale de l'objectif fondamental de notre contribution et de la méthodologie adoptée, l'accent est mis sur les principales techniques et étapes de l'approche que nous avons proposée, à savoir : la formalisation et la description des différentes parties d'un cas, la recherche et la sélection fonctionnelle et non-fonctionnelle des services, l'adaptation appropriée de ses services, et enfin la révision et l'apprentissage de la base des cas.

5.1.1 Exemple de motivation

Dans cette section, nous illustrons à travers un exemple le principe de la méthode de recherche proposée. L'exemple consiste à planifier un voyage, dans lequel un utilisateur cherche à trouver un service qui prend comme entrée une requête de voyage et comme résultat produit une réservation de vol, une réservation dans un hôtel, l'allocation d'une voiture et l'enregistrement dans une conférence. De plus, l'utilisateur a exprimé le voeu de voyager avec la compagnie aérienne Air Algérie.

Notre approche utilise la méthode de RàPC, donc les compositions se trouvent dans une base de cas. Supposons que le système a déjà résolu des problèmes similaires dans le passé et aprés une recherche dans la base de cas, nous remarquons que le système n'a pas trouvé une solution qui comporte tous les services, mais des parties complémentaires. La phase de remémoration a donné comme résultat trois cas : Case1, Case2, Case3.

Case1 contient les services BookFlight pour la résrvation de vol, BookHotel pour l'hotêl et RentCar pour l'allocation de voiture.

Case2 produit trois services qui sont BookFlight, BookHotel et ConferenceRegister pour l'enregistrement dans la conférence.

Case3 contient les services BookFlight, BookHotel et RentCar.

On remarque que Case1 produit les mêmes services que fournit Case3, mais en plus, il prend en charge les préférences exigées par l'utilisateur. Le case2 fournit le service d'enregistrement à la conférence. Le planificateur va décider de l'ordre d'exécution de ces tâches. La table 5.1 montre le nouveau problème et les cas trouvés après l'étape de recherche.

Lors du calcul de la similarité entre la nouvelle demande de l'utilisateur (la planification de voyage pour participer à une conférence) et les cas présents dans la base de cas, le système n'a pas obtenu une solution directe. Cependant, en décomposant le but, il a trouvé plusieurs solutions. Le premier cas et le troisième contiennent

Request	Input	Output	Goal	User preferences
User query	Travel request()	Travel planning for conference attendance	BookFlight - BookHotel - RentCar - ConferenceRegister	Air Algérie
Request 1	Travel request()	Travel planning	BookFlight - BookHotel - RentCar	
Request 2	Travel request()	Conference attendance	BookFlight - BookHotel - ConferenceRegister	
Request 3	Travel request()	Travel planning	BookFlight - BookHotel - RentCar	Air Algérie

TABLE 5.1: Nouvelle requête et cas résolus.

la planification de voyage mais, dans notre cas le système prend le cas avec préférences qui correspond aux demandes de l'utilisateur. Le deuxième cas contient la participation à la conférence. Le système fusionne les solutions des Case2 et Case3 pour obtenir une solution au problème. Enfin, le plan et la demande vont former un nouveau cas avec lequel le système va décider de mettre à jour ou non la base de cas.

5.1.2 Contributions

Ce travail a pour but de favoriser l'intégration des besoins et des préférences des utilisateurs dans le processus de la composition de services web. Notre solution assure l'intégration des préférences de l'utilisateur dans la sélection des meilleures solutions et l'utilisation convenable des cas trouvés par le biais des améliorations apportées [Hachemi et al., 2015]. Les orientations de notre approche sont concrétisées par plusieurs mécanismes :

- Modèle de représentation de problème : l'expressivité du problème à résoudre influe sur la qualité des résultats trouvés. Notre approche utilise un modèle de représentation du problème qui prend les aspects fonctionnels et non-fonctionnels des services web pour assurer une description expressive des exigences de l'utilisateur de services Web.
- Exploiter le paradigme RàPC : notre approche opte pour la planification à partir de cas pour la rationalisation du traitement et pour gagner un temps considérable perdu à refaire des compositions qui existent déjà. Donc, est-il nécessaire d'élaborer des mécanismes pour optimiser les temps de traitement et tenir compte des expériences réussies ? L'introduction des préférences des utilisateurs dans le cycle du RàPC permet une meilleure réutilisation de cas.
- La recherche des compositions : la robustesse d'un mécanisme de raisonnement à partie de cas réside dans sa recherche. Pour une meilleure réutilisation, nous proposons la décomposition du problème qui permet de trouver des fragments de solutions qui vont par la suite être réutilisés pour former une solution au problème de composition posé.
- Mécanisme de remémoration : notre mécanisme de recherche trouve toutes les solutions possibles pour un problème. L'extraction de tous les cas dans l'étape de la recherche permet de choisir par la suite les meilleures solutions en se basant sur les préférences de l'utilisateur.
- L'adaptation : les solutions trouvées ne correspondent pas toujours aux demandes des utilisateurs mais nécessitent une adaptation. Notre méthode d'adaptation des solutions trouvées fusionne les solutions trouvées pour former une ou plusieurs solutions pour mieux répondre au problème posé.

5.1.3 Approche proposée

Dans cette section, nous donnons un aperçu de l'approche de composition proposée. L'idée principale de cette approche est d'exploiter les techniques offertes par

le RàPC et la planification présentées dans le chapitre 3 dans le but de définir une méthodologie de composition automatique de services web.

L'approche proposée appelée la planification à base de cas et des préférences des utilisateurs pour la composition de services web (case-based planning with preferences CBP-P4WSC), traite les problèmes de la composition d'une manière à la fois simple et efficace. Par ailleurs, pour réaliser les différentes phases du processus de composition, cette approche s'appuie sur l'apprentissage et la réutilisation qui sont les concepts fondamentaux du RàPC tout en s'appuyant sur les préférences des utilisateurs dans la composition pour des résultats meilleurs. CBP-P4WSC est une méthodologie qui traite les principaux problèmes qui apparaissent dans le domaine de la composition de services web. Elle divise le processus de la composition en plusieurs phases en bénéficiant des étapes du RàPC et en ajoutant d'autres étapes offrant ainsi des solutions pour la spécification formelle, la recherche et la composition.

Notre approche se base sur deux techniques d'intelligence artificielle qui sont le raisonnement à partir de cas et la planification. Le raisonnement à partir de cas est utilisé pour minimiser le temps de recherche si la solution existe et le système apprend à partir des nouveaux cas résolus. La planification qui prend en considération les préférences des utilisateurs si le système ne trouve pas de solution dans la base de cas. Le processus illustré par la figure 5.1, permet d'exploiter et de réutiliser les compositions de services web disponibles. Ces derniers agissent comme des services réutilisables pour la réalisation de nouvelles compositions de services web. La méthodologie CBP-P4WSC prend place à travers les phases suivantes :

Phase de transformation (génération du problème) Le but de la phase de transformation consiste à générer un nouveau problème de composition partant d'une requête de l'utilisateur. Le but de cette étape est de reformuler la requête de l'utilisateur en un problème dont la forme correspond exactement aux problèmes déjà résolus et qui sont stockés dans la base des cas. Ce nouveau

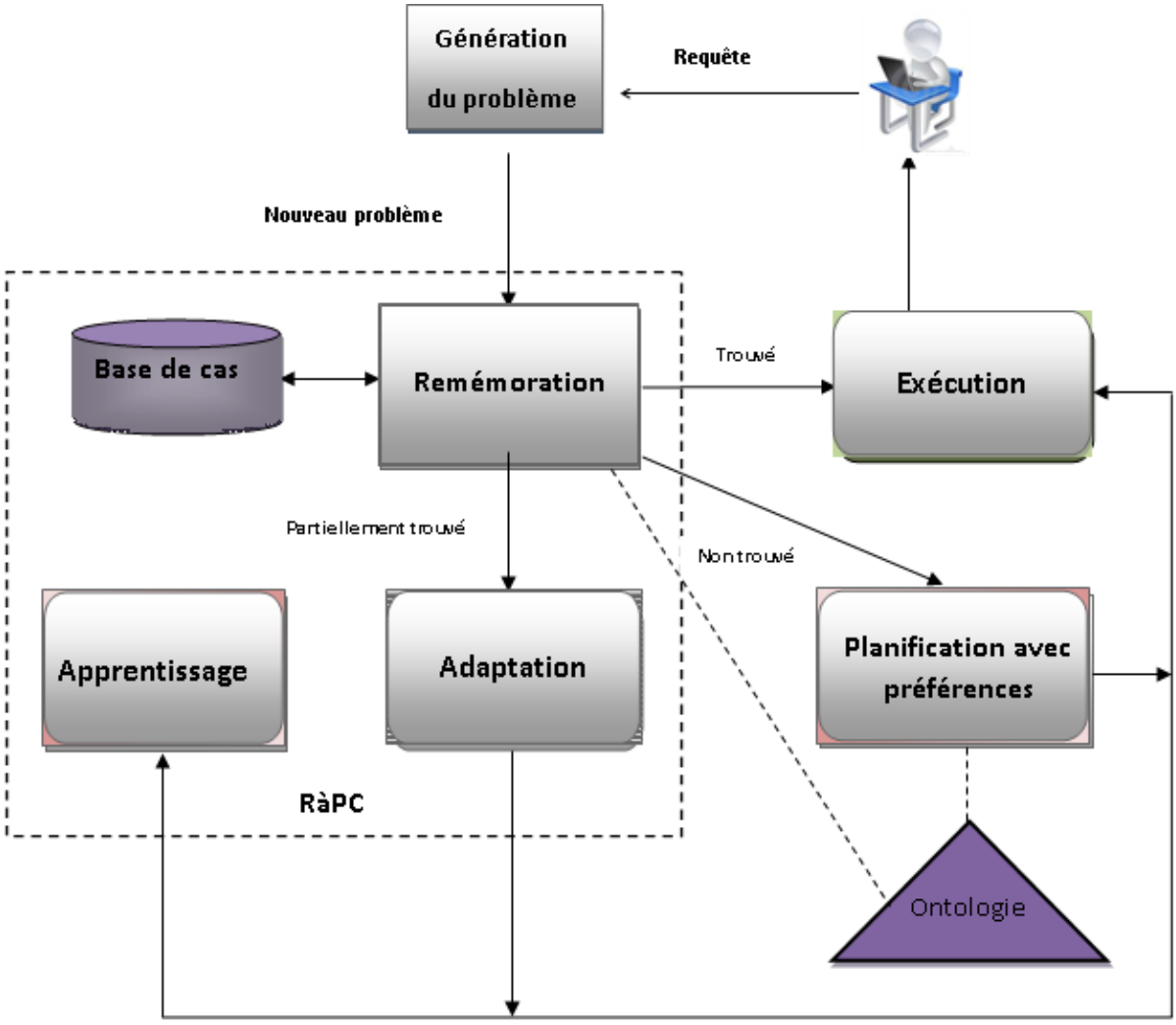

FIGURE 5.1: CBP-P pour la composition des services web.

problème va être utilisé ensuite par le système pour lui trouver des solutions. De nouveaux problèmes sont obtenus de cette transformation.

Phase de remémoration (recherche) L'objectif de la phase de remémoration est de trouver et sélectionner les services web requis pour la composition et de fournir les informations permettant de les inclure dans un scénario de composition. L'idée est qu'un certain nombre de services web peut répondre aux exigences fonctionnelles spécifiées par un utilisateur. Ensuite, il faut choisir parmi un ensemble de services trouvés lequel répond le mieux aux exigences de l'utilisateur. Pour cela, la sélection est faite en se basant sur les préférences de l'utilisateur fournis dans la requête. Dans cette phase, toutes les solutions

possibles pour les nouveaux problèmes vont être extraites de la base de cas en utilisant les mesures de similarités et en se basant sur les préférences des utilisateurs.

Phase de planification Si dans la phase de remémoration nous ne trouvons pas de cas similaires au problème posé ou si la solution fournie ne répond pas aux exigences de l'utilisateur, une composition est déclenchée en utilisant un planificateur. Ce planificateur essaie de trouver une composition qui répond aux besoins fonctionnels de l'utilisateur et en respectant ces préférences en même temps. Pour cela, un planificateur est utilisé pour trouver une nouvelle composition de services web en utilisant la requête de l'utilisateur et ses préférences.

Phase d'adaptation La phase de remémoration peut retourner des cas dont la partie problème est similaire à la demande de l'utilisateur, cependant dans certains cas la partie solution peut ne pas satisfaire exactement la requête de l'utilisateur donc une adaptation de la solution est nécessaire. Une fois les plans solution sélectionnés, la phase d'adaptation utilise ces plans pour construire la solution finale qui répond aux exigences de l'utilisateur.

Phase d'exécution Après avoir trouvé une solution au problème de composition posé par l'utilisateur soit directement par la recherche, l'adaptation ou la planification, la phase d'exécution consiste à exécuter la solution trouvée pour l'utilisateur.

Phase de révision et d'apprentissage L'étape de révision est lancée selon les remarques de l'utilisateur sur la solution trouvée. Après cette étape, le nouveau problème et la solution trouvée vont composer un nouveau cas. Le système va décider si ce nouveau cas va être ajouté ou non à la base de cas, c'est ce qui constitue l'étape d'apprentissage.

5.2 Le formalisme RàPC

Cette section se focalise sur les différentes phases du mécanisme de notre proposition ainsi que l'ensemble des modélisations nécessaires pour formaliser les cas dans notre système de composition de service. Nous aborderons dans les prochaines sections la formalisation de la description du problème à résoudre et sa mise en forme pour la phase de remémoration, la description de la partie solution pour une meilleure réutilisation et la formalisation des autres parties du cas pour nous aider dans les différentes phases de notre approche CBP-P4WSC.

5.2.1 Modèle de représentation de cas

L'utilisation de la technique du RàPC nécessite l'identification du cas qui doit être représenté par un modèle adapté à chaque problématique. Cette modélisation nous permet de décrire chaque composant du cas. Par conséquent, elle facilite la recherche des cas similaires et la sélection des meilleurs cas.

Un cas est utilisé pour stocker l'expérience de problèmes déjà résolus. Ce dernier est représenté par les mêmes composants : un problème, une solution et d'autres informations Cas= (pb, sol (pb)). Le cas enregistré dans la base sert d'inspiration pour résoudre de nouveaux problèmes appelés cas cibles.

Dans le contexte de la composition de service web, les requêtes des utilisateurs peuvent être considérées comme des problèmes et les plans de composition sont les solutions à ces problèmes. La figure 5.2 montre la représentation d'un cas dans le RàPC et dans notre système CBP-P4WSC.

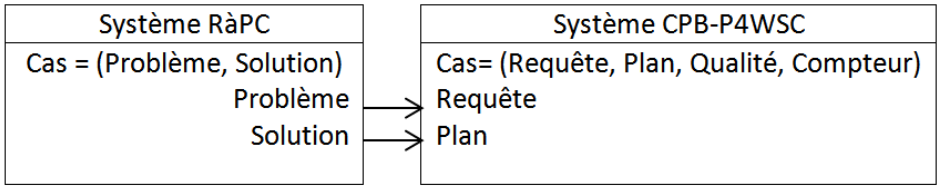

FIGURE 5.2: Composants d'un cas dans notre système CPB-P4WSC.

Par la suite, nous utilisons ces définitions pour représenter les cas dans notre système CBP-P pour la composition de services web.

Définition 4. Un cas est définie comme un quintuple C=(PB,S,Q,N) où :

PB Le problème posé par l'utilisateur.

S La solution au problème.

Q La qualité de la solution proposée.

N Un compteur.

CBP-P4WSC est dédié à la composition automatique des services Web en réponse à la requête d'un utilisateur demandant un service avec des besoins spécifiques. Ainsi, la partie problème du cas reflète la requête de l'utilisateur où il doit pouvoir décrire ses particularités en termes de besoins fonctionnels et non-fonctionnels. De même, la partie solution doit contenir un plan de composition de services web qui répond aux demandes prédéterminées par l'utilisateur. Dans notre approche, nous avons ajouté d'autres champs à la représentation d'un cas. Le premier est une évaluation de la composition enregistrée en terme de qualité de service et le deuxième est un compteur pour nous indiquer combien de fois ce cas a été réutilisé et pour nous permettre de maintenir la base des cas par la suite.

5.2.1.1 Représentation de la partie problème

Pour former un cas, nous commençons par représenter la partie problème PB qui reflète la requête de l'utilisateur cherchant une composition bien spécifique d'un ensemble de services. La représentation de cette partie au niveau de notre système CPB-P4WSC s'appuie sur notre problématique de composition de service Web décrivant les critères fonctionnels et non-fonctionnels d'un service.

Ainsi, dans la partie problème, nous distinguons les propriétés fonctionnelles des propriétés non-fonctionnelles. Ces propriétés sont tirées du modèle de description de services Web OWL-S. Notre cas se compose alors d'une partie non-fonctionnelle

représentée par les préférences des utilisateurs et d'autre part, notre problème de composition est un problème de planification donc la partie fonctionnelle est composée des entrées/sorties de services, de l'état initial et de l'ensemble de buts à atteindre d'où la notation :

$$\text{PB} = (\text{I,O,T,SG,A})$$

Composants	Descriptions
Entrées I=(I_1,I_2,...,I_n)	Liste des paramètres d'entrée disponibles au début du service requis.
Sorties O=(O_1,O_2,...,O_m)	Liste des paramètres indiquant les sorties nécessaires après l'exécution du service requis.
Etat initial (T)	Représente l'état initial.
Les buts (SG)	L'ensemble de buts à atteindre.
Préférences (A)	Représente les préférences de l'utilisateur quant aux services demandés

TABLE 5.2: Descriptions des différents composants d'un PB.

Dans l'étape de translation, il y'a des éléments qui sont absolument obligatoires et aucune recherche ne sera lancée sans leur remplissage. Ces éléments sont les entrées, les sorties et l'ensemble de buts. Par contre, les éléments qui sont optionnels et l'absence de leur valeur ne bloque pas la découverte mais elle peut conduire à des résultats faux sont les préférences. Toutefois, l'existence d'information leur donne automatiquement un aspect obligatoire à considérer lors de la sélection.

5.2.1.2 Représentation de la partie solution

Dans notre approche de composition de service web, on utilise la planification à base de cas. Puisqu'on a reformulé le problème de composition en un problème de planification, la solution est une séquence d'opérations à exécuter.

Nous nous intéressons à la formalisation de la partie solution du cas. Nous détaillons par la suite les informations que nous pensons suffisantes à fournir aux utilisateurs souhaitant exploiter la solution obtenue.

Dans notre approche la solution est un plan de composition. Comme tout plan, notre solution est constituée des éléments suivants :

S=O_{p1},O_{p2},. . .,O_{pm}

Le plan solution est une séquence d'opérations instanciées qui permettront d'atteindre l'ensemble de buts partant d'un état initial et en respectant les aspects non-fonctionnels.

5.2.1.3 Représentation des autres parties

Pour la représentation d'un cas, il existe deux champs obligatoires qui sont le problème et la solution mais, nous pouvons ajouter d'autres informations qui nous semblent utiles. Dans notre cas, nous avons ajouté deux champs à la représentation de notre cas : une qualité de la composition et un compteur.

Qualité La qualité de la composition est un ensemble d'attributs qui représentent la qualité de service obtenu selon le temps de réponse, le côut, la disponibilité, etc.

Compteur Permet au système de savoir combien de fois ce cas a été réutilisé pour résoudre un nouveau problème. Ce compteur nous permet de mettre à jour la base de cas en supprimant les cas non utilisés.

5.2.2 Remplissage de la base de cas

Tout système de RàPC est construit autour d'une base de cas qui est un ensemble de cas de composition de services déjà trouvés. Ces cas appelés cas sources sont représentés conformément à la structure décrite dans les sections passées.

Une base de cas possède plusieurs fonctions, elle sert à stocker les cas résolus dans le passé. C'est une source pour la recherche des cas suite à la réception d'une requête et elle apprend continuellement à partir de nouveaux cas suite à des ajouts de nouvelles compositions.

5.3 Méthodologie CBP-P4WSC

L'approche CBP-P4WSC que nous proposons est fondée sur cinq phases de traitement qui, à l'aide d'ontologie, prennent en compte l'aspect sémantique des services Web, pour pouvoir repérer parmi les cas disponibles, des cas similaires à un nouveau problème, en proposer une meilleure solution, et enrichir la base de cas par le ou les nouveaux cas identifiés. Les cinq phases principales de traitement consistent en la transformation de la requête de l'utilisateur, la remémoration des services pouvant y répondre et la sélection du ou des services les plus adéquats, l'adaptation des solutions trouvées, la planification de nouvelles solutions si nécessaire et la mémorisation des nouveaux cas identifiés et validés pour l'enrichissement de la base de cas. Nous allons détailler par la suite, chaque phase de notre approche.

5.3.1 Génération du problème

Dans cette étape, le système génère un nouveau problème à résoudre. Ce problème est obtenu en transformant la requête de l'utilisateur en un problème de planification. L'interface utilisateur est utilisée par un nouvel utilisateur qui n'a pas de connaissances sur les technologies de services web. Une des interfaces les plus familiarisées et utilisées pour invoquer les services est l'interface en langage naturel. Plusieurs méthodes concernant ce type de langage pour les services web ont été proposées [Bosca et al., 2006], [Englmeier et al., 2006] et [Jang et al., 2007] . L'exemple suivant illustre la transformation d'une requête décrite en langage naturel en un problème de planification. Cette partie n'est pas prise en compte dans notre travail.

Exemple

Soit la requête complexe d'un utilisateur " *Pour assister à une conférence, je veux réserver un vol d'Alger à Paris le 23 Novembre, de louer une voiture à Paris et réserver une suite dans un hôtel du 23 Novembre jusqu'au 25 du même mois. Je préfère voyager avec Air Algérie".*

Cette requête en langage naturel est transformée en un problème de planification PB = (I, O, S, SG, A) où :

I = Travel request

O = Travel planning for conference attendance

S= $\emptyset$

SG= BookFlight, BookHotel, RentCar, ConferenceRegister

A= preference p1 (sometime (initiate (book-flight Air Algérie)))

La transformation de la requête de l'utilisateur en un problème à résoudre est le résultat de la première phase qui est la translation. Ainsi, l'entrée de la prochaine phase qui est la remémoration est un nouveau problème. Ce problème est composé des informations nécessaires à la recherche et la sélection.

5.3.2 Remémoration de cas

Pour tolérer une base de cas large et en croissance continue, le système a besoin d'une technique de recherche scalable. La remémoration est le processus de recherche et d'extraction des cas appropriés présents dans la base des cas. Le but de la remémoration est de trouver des cas qui ont le potentiel d'être les plus utilisables. Cette phase exige une combinaison de recherche et de sélection. Pour cela, une mesure de similarité est utilisée comme un outil pour trouver les cas les plus proches et qui correspondent à la recherche lancée.

Chaque fois qu'un utilisateur a une demande, la base de cas est consultée pour trouver les cas dont la description du problème est semblable à la demande de l'utilisateur. Le module principal dans ce processus est le calcul de similarité entre chaque cas et le nouveau problème.

Dans notre système CBP-P4WSC, à l'arrivée d'une nouvelle requête et après la reformulation en un nouveau problème, deux processus sont utilisés dans la phase de remémoration. Le premier est le processus de recherche des cas qui correspondent

au problème posé. Le deuxième processus est celui de la sélection qui se charge de choisir parmi les cas trouvés ceux qui correspondent le mieux à la requête en se basant sur les préférences de l'utilisateur dans le but de sélectionner les meilleures solutions. Les plans sources obtenus de la phase de remémoration sont adaptés pour répondre à la requête de l'utilisateur.

5.3.2.1 Calcul de similarité

Pour calculer la similarité fonctionnelle entre un cas de l'ensemble CB des cas candidats et le problème (PB), nous procédons par agrégation des similarités entre les entrées/sorties et l'état initial/buts. Pour la similarité entrées/sorties, nous utilisons la distance sémantique entre le problème et les cas présents dans la base CB. En se basant sur les calculs effectués auparavant, nous calculons la similarité entre l'état initial et l'ensemble des buts. Etant donné un problème PB= (I, O, S, SG, A) et un cas de la base de cas C=(I′, O′, S′, SG′, A′).

Ainsi, la formule que nous retenons pour le calcul de la similarité fonctionnelle entre les cas, est la suivante :

$$Sim(PB, C) = W_1 * Sim(Input, Output) + W_2 * Sim(Initial, Goals) \quad (5.1)$$

Où :

- $W_1 + W_2 = 1$

$$Sim(input, output) = Sim_{input} + Sim_{output} \quad (5.2)$$

Où :

- Sim_{input}= Sim (Input, Input ′)
- Sim_{output}= Sim (Output,Output′)

$$Sim(initial, goals) = Sim_{initial} + Sim_{goal} \quad (5.3)$$

Où :

- $\text{Sim}_{initial}$= |IM|/|initial|. IM est l'ensemble des correspondances dans l'état initial du problème.
- Sim_{goal}= |GM|/|goal|. GM est l'ensemble de correspondances dans le but du cas.

Calcul de similarité des préférences : Dans l'étape de remémoration l'étape de sélection se charge de calculer la similarité des préférences des utilisateurs pour choisir les services qui répondent aux mieux aux exigences des utilisateurs. Le calcul se limitera aux cas trouvés dans la phase de recherche et non la base de cas. Nous notons que le calcul de similarité dans cette étape se limite à une similarité syntaxique qui compare les préférences des utilisateurs avec les cas trouvés et choisit laquelle est la plus proche.

5.3.2.2 Phase de recherche

Le problème généré de la phase d'élaboration est l'entrée de l'algorithme de recherche de plans. Nous avons l'ensemble des entrées (I), des sorties (O), l'état initial (T), l'ensemble de buts (SG) et les préférences des utilisateurs (A). Dans cette étape, nous recherchons des correspondances pour le problème de composition posé.

Si le résultat de la recherche est une correspondance exacte avec le problème, le système copie la solution et la recherche est arrêtée. Sinon nous devons rechercher toutes les solutions possibles pour l'ensemble des buts dans la base de cas. Si l'ensemble des cas trouvés est vide alors le système doit décomposer les buts à des sous-buts. Quand nous trouvons une solution qui répond à un but, nous ajoutons le but avec la solution. Nous répètons ce processus jusqu'à ce que l'ensemble des sous buts soit vide.

L'algorithme 1 donne comme résultat un ensemble de plans candidats pour la

deuxième étape. Si l'ensemble des plans est vide, cela signifie que le système n'a pas trouvé de solutions adéquates au problème dans la base de cas. Le système doit procéder à une nouvelle composition en utilisant un planificateur. Dans cette étape, une planification avec préférence est utilisée pour obtenir la composition.

Algorithme 1 : Algorithme de recherche de solutions

Input : $PB = (I, O, T, SG, A)$ /* New problem*/

Output : SP

```
1  CP ← φ
2  repeat
3      Search (P)
4      if CP = φ then
5          Decompose (G)
6      else
7          Select (P)
8          Add (P, Ap) to SP
9          U = G − Ap
10 until U = φ;
11 Return (SP)
12 if SP = φ then
13     Planning (NQ)
14 else
15     Plan selection (SP)
```

L'étape de recherche de la phase de remémoration est une étape très importante dans notre méthode de composition. Elle nous permet de déclencher la prochaine phase de notre approche CBP-P4WSC selon le résultat obtenu.

1. Si nous trouvons la solution exacte, le système exécute la solution directement.
2. Si nous trouvons plusieurs solutions complémentaires, nous passons à l'étape de sélection qui consiste à choisir les meilleures solutions pour un problème donné, et le résultat est transféré à la phase d'adaptation pour construire la solution.
3. Si le système ne trouve pas de solution, il déclenche la phase de planification

avec les préférences.

5.3.2.3 Phase de sélection

Après l'étape de la recherche, une étape de sélection est exécutée. Elle est utilisée pour obtenir les meilleurs plans pour la prochaine étape. Les entrées de cette étape sont l'ensemble de buts (SG), l'ensemble des plans candidats (SP) et les préférences des utilisateurs (A). Les solutions sont sélectionnées selon les préférences des utilisateurs. La sortie de l'algorithme 2 de sélection est un ensemble de plans qu'on appellera les plans préférés.

Les préférences représentent les propriétés de l'utilisateur : lorsque plusieurs services sont disponibles pour effectuer la même activité, leurs propriétés, telles que le coût total, le choix des compagnies aériennes ou des hôtels, des dates et du temps deviennent importantes dans le processus de sélection. Pour pouvoir utiliser les préférences dans les services web, un modèle est nécessaire pour capturer les descriptions de ces propriétés à partir de la requête de l'utilisateur. Les préférences sont exprimées en utilisant le langage PDDL. Plusieurs systèmes de planification à base de cas ne peuvent fournir qu'un seul plan. Ce plan peut répondre à un sous-ensemble de sous buts et donc nécessite une adaptation pour atteindre les buts restants. Dans notre système, la phase de récupération peut donner plusieurs plans qui sont à la base du nouveau plan.

Algorithme 2 : Algorithme de sélection de solutions

Input : SG, SP, A

Output : SPP

```
1 for (G) from (SG) do
2     Find plan (P) that match preferences (A)
3     if found matched (P) then
4         SP = SP − G
5 SPP ← SP
6 Return (SPP)
```

Dans la plupart des cas, les solutions trouvées (plans) ne correspondent pas exactement aux attentes de l'utilisateur. Ainsi, une adaptation est nécessaire. L'ensemble des plans préférés résultant de la phase de sélection vont former l'entrée de la phase d'adaptation.

5.3.3 L'adaptation multi-plans

L'utilisation de plusieurs plans durant la phase d'adaptation est un des aspects intéressants du paradigme de la planification à partir de cas. Un seul cas source qui soit complètement similaire à la nouvelle situation peut ne pas être trouvé dans des situations complexes de planification avec des buts multiples. Pour cela, la planification peut être vue comme un processus de fusion et d'adaptation de ces cas complémentaires si plusieurs cas ont été trouvés pour des parties indépendantes du nouveau problème.

Les approches d'adaptation connues et référenciées dans la littérature pour la réutilisation des expériences passées sont : l'adaptation substitutionnelle, transformationnelle et générative [Lopez de Mantaras et al., 2005].

Dans notre travail, nous allons nous focaliser sur les techniques d'adaptation compositionnelle qui composent des nouvelles solutions en utilisant de multiples plans. Il existe plusieurs méthodes d'adaptation. L'algorithme MPA (Multi Plan Adaptation) [Ram et al., 1996] qui est une extension de l'algorithme SPA (Systematic Plan Adaptor) [Hanks et al., 1995] et qui permet la réutilisation de plusieurs plans ou on a modifié les entrées de l'algorithme par l'ensembles des plans préférés au lieu de la base de cas.

L'algorithme E-MPA divise les différents plans en petits morceaux, qui sont ensuite recombinés ensemble.

Notre algorithme prend comme entrée l'ensemble des plans choisis dans la phase

de remémoration et un plan partiel parmi les plans trouvés.

Algorithme 3 : Algorithme d'adaptation multi plans

Input : P, SPP

Output : PP

$PP \leftarrow Copy - Plan(P)$
$Igs \leftarrow GetIntermediateGoalStatement(PP)$
$plan \leftarrow RetrieveBestPlan(SPP, igs)$
$clipping, mapping \leftarrow FitPlan(plan, igs)$
for *cgpinmapping* **do**
    if $Producer - Exists(oc - gl - pair, PP)$ **then**
        $Splice - Link(oc - gl - pair, PP, clipping)$
    else
        $Splice - Step(oc - gl - pair, PP, clipping)$

    $AddNewOpenCond - GoalPairs(mapping, PP)$
return (PP)

La phase d'adaptation fournit comme résultat un ou un ensemble de plans qu'on appelle des plans préférés. Le résultat représente une ou plusieurs compositions possibles au problème posé par l'utilisateur. Donc la solution prend en considération les caractéristiques fonctionnelles et non-fonctionnelles fournies par l'utilisateur.

5.3.4 La planification avec préférences

Si le système ne trouve pas de correspondances du problème dans la base des cas, il procède à une composition de services web à partie de la requête de l'utilisateur. Dans cette section, nous décrivons comment aboutir à une composition de services web tout en ajoutant les préférences des utilisateurs dans le processus. Si au bout de la phase de remémoration le système ne peut pas trouver des cas similaires dans la base de cas, le système procèdera à une nouvelle planification à partir du problème de planification.

Les techniques de planification de l'intelligence artificielle peuvent contribuer à

la résolution du problème de composition. Ce dernier peut être défini comme un problème de planification. En effet, les services sont modélisés comme des actions et la composition comme un plan de connexions des Service web. Dans un premier temps, le planificateur construit le plan solution en explorant les Web services disponibles. Ensuite, il développe les différents états intermédiaires à partir de l'état initial, en appliquant les services, jusqu'à atteindre l'état but.

Pour la spécification des préférences des utilisateurs, nous utilisons le langage PDDL 3.0 (Preference language Domain Definition Language) [Gerevini et al., 2006]. La syntaxe du langage est une amélioration de la version PDDL2.2 en ajoutant les préférences et les contraintes difficiles. Il inclut aussi une façon de définir une fonction qui mesure la qualité du plan.

La modélisation des préférences des utilisateurs dans PDDL3 :

Dans le langage PDDL3, les préférences sont décrites :

Syntaxe : (preference [name] <GD>)

Exemple :

- A= p1, p2,p3, ...
- (and (preference p1 (always (clean truck1)))
- (preference p2 (and (at end (at package2 paris))
- (sometime (clean track1))))
- (preference p3 (...))
- ...)

Il existe plusieurs approches dans le domaine de la composition de services avec les préférences des utilisateurs HPLAN-P [Baier et al., 2007], SGPLAN [Hsu et al., 2007], SCUP [Lin et al., 2008] et HTNPlan-P [Sohrabi et al., 2009].

Pour cette étape, nous utilisons un planificateur de composition de services web en se basant sur les préférences des utilisateurs. Le planificateur SGPlan a été élu meilleur planificateur lors du 5ème Concours international de planification

[Gerevini et al., 2006]. SGPlan partitionne un problème de planification en sous-problèmes, chacun avec un état but, et trouve un plan réalisable pour chacun. Sa contribution se concentre sur l'optimisation de préférences de but pour satisfaire la planification. Comme d'autres planificateurs, SGPLAN5 utilise une stratégie pour réitérer la sélection du meilleur plan après l'obtention du premier plan. La figure 5.3 montre une partie d'un plan trouvé par SGPLAN5.

```
0.001: (DRIVE TRUCK1 L4 L3) [355.600]
355.604: (LOAD PACKAGE4 TRUCK1 A2 L3) [1.000]
356.607: (LOAD PACKAGE5 TRUCK1 A1 L3) [1.000]
357.610: (DRIVE TRUCK1 L3 L1) [554.300]
911.913: (UNLOAD PACKAGE5 TRUCK1 A1 L1) [1.000]
912.916: (DELIVER PACKAGE5 L1) [1.000]
912.916: (UNLOAD PACKAGE4 TRUCK1 A2 L1) [1.000]
913.919: (DELIVER PACKAGE4 L1) [1.000]
913.919: (DRIVE TRUCK1 L1 L4) [880.500]
1794.422: (LOAD PACKAGE7 TRUCK1 A2 L4) [1.000]
1795.425: (LOAD PACKAGE8 TRUCK1 A1 L4) [1.000]
1796.428: (DRIVE TRUCK1 L4 L2) [1034.100]
2830.531: (UNLOAD PACKAGE8 TRUCK1 A1 L2) [1.000]
2831.534: (DELIVER PACKAGE8 L2) [1.000]
2831.534: (UNLOAD PACKAGE7 TRUCK1 A2 L2) [1.000]
2832.537: (DELIVER PACKAGE7 L2) [1.000]
```

FIGURE 5.3: Partie d'un plan trouvé par SGPLAN5.

5.3.5 Révision et apprentissage

Phase de révision : La phase de révision consiste à proposer, modifier et confirmer une solution. Le but de cette phase est de tester la solution adaptée afin de vérifier si elle convient à résoudre le problème posé, avant de décider si la solution va être mémorisée ou pas. La révision est prise en considération dans notre approche. Elle est confiée à l'utilisateur qui peut exprimer sa satisfaction ou non vis-à-vis du service composé. Si l'utilisateur lui convient la solution, le système passe à la phase d'apprentissage. Sinon une modification ou une réadaptation doit être réalisée pour subvenir aux attentes de l'utilisateur.

Phase d'apprentissage : Selon le degré de satisfaction de l'utilisateur, qu'il soit supérieur ou inférieur à un seuil défini par l'administrateur du système, le nouveau cas identifié sera mémorisé ou non dans la base de cas. Cette phase contribue également au processus de mise à jour et de nettoyage de la base de cas. L'utilisateur ayant effectué le test, peut notamment détecter que le service proposé a été désactivé ou a subi des changements par son fournisseur, et en informer l'administrateur du système. Ce dernier devra par la suite procéder à la mise à jour ou carrément à la suppression des cas concernés.

5.4 conclusion

Nous avons donné une description globale de notre approche de composition de services web et introduit ses composants tout au long de ce chapitre. Nous avons présenté les différentes représentations des données manipulées au sein de notre système. Nous avons expliqué les différentes phases du cycle de RàPC dans notre approche, et comment utiliser les expériences passées dans la composition. Nous avons aussi montré que notre approche permet également, à travers l'intégration des aspects fonctionnels et non-fonctionnels, de sélectionner non seulement les services Web pouvant répondre à la requête des utilisateurs mais, d'en sélectionner les meilleures solutions.

Le chapitre suivant sera consacré à l'implémentation de notre approche de composition de service web ainsi qu'à l'évaluation des résultats trouvés. Une étude de cas dans le domaine du transport sera utilisée.

Chapitre 6

Exprérimentation

Sommaire

6.1 Introduction

Ce chapitre est consacré à la présentation des détails d'implémentation de notre approche. La mise en œuvre du prototype démontre comment le raisonnement par cas et la réutilisation des modèles de service peuvent être intégrés pour augmenter l'efficacité de la méthode de composition. Ces cas peuvent ensuite être utilisés comme

point de départ vers un processus de composition plus efficace.
Nous y présentons l'architecture logicielle du système aussi bien que les principaux composants de notre modèle de composition de services web, et les différentes parties descriptives de notre système de raisonnement. Nous y exposons aussi l'interface que nous avons réalisée. Par la suite, nous discuterons les résultats obtenus du modèle implémenté pour évaluer notre approche.

6.2 Déroulement du processus de composition

La méthodologie du modèle proposé est présentée comme suit :

1. L'utilisateur pose sa requête pour un service composé.
2. La requête de l'utilisateur est traduite en un nouveau problème.
3. Le nouveau problème est comparé avec les cas présents dans la base en utilisant un mécanisme de recherche composé de deux étapes : la recherche et la sélection en se basant sur les préférences des utilisateurs. Il existe trois possibilités : la solution est complètement trouvée, partiellement trouvée ou non trouvée.
4. Si le système trouve une solution qui correspond exactement aux attentes de l'utilisateur donc elle est exécutée.
5. Si le système ne trouve pas de correspondance entre le nouveau problème et les cas présents dans la base, une planification est lancée en utilisant un planificateur qui prend en charge les préférences de l'utilisateur dans la composition.
6. Si le système trouve des solutions qui nécessitent des modifications, une adaptation est utilisée pour composer le nouveau service.
7. Le cas est révisé si l'utilisateur exige une modification.
8. L'apprentissage de la base de cas est lancé dans le cas où une nouvelle solution est trouvée à travers la planification ou l'adaptation.

6.3 Architecture proposée pour la composition

Pour pouvoir remédier à la complexité de la tâche de composition d'une part et à l'hétérogénéité et l'évolution des services web d'autre part, nous avons proposé de bénéficier des avantages des méthodes de l'intelligence artificielle qui sont la planification et le raisonnement à partir des cas. Pour cela, nous avons proposé une démarche composée de plusieurs modules comme le montre la figure 6.1. Le prototype intègre un certain nombre d'outils qui fournissent une interface pour les principaux processus de notre approche CBP-P4WSC, la translation de cas, la recherche, l'adaptation et la maintenance.

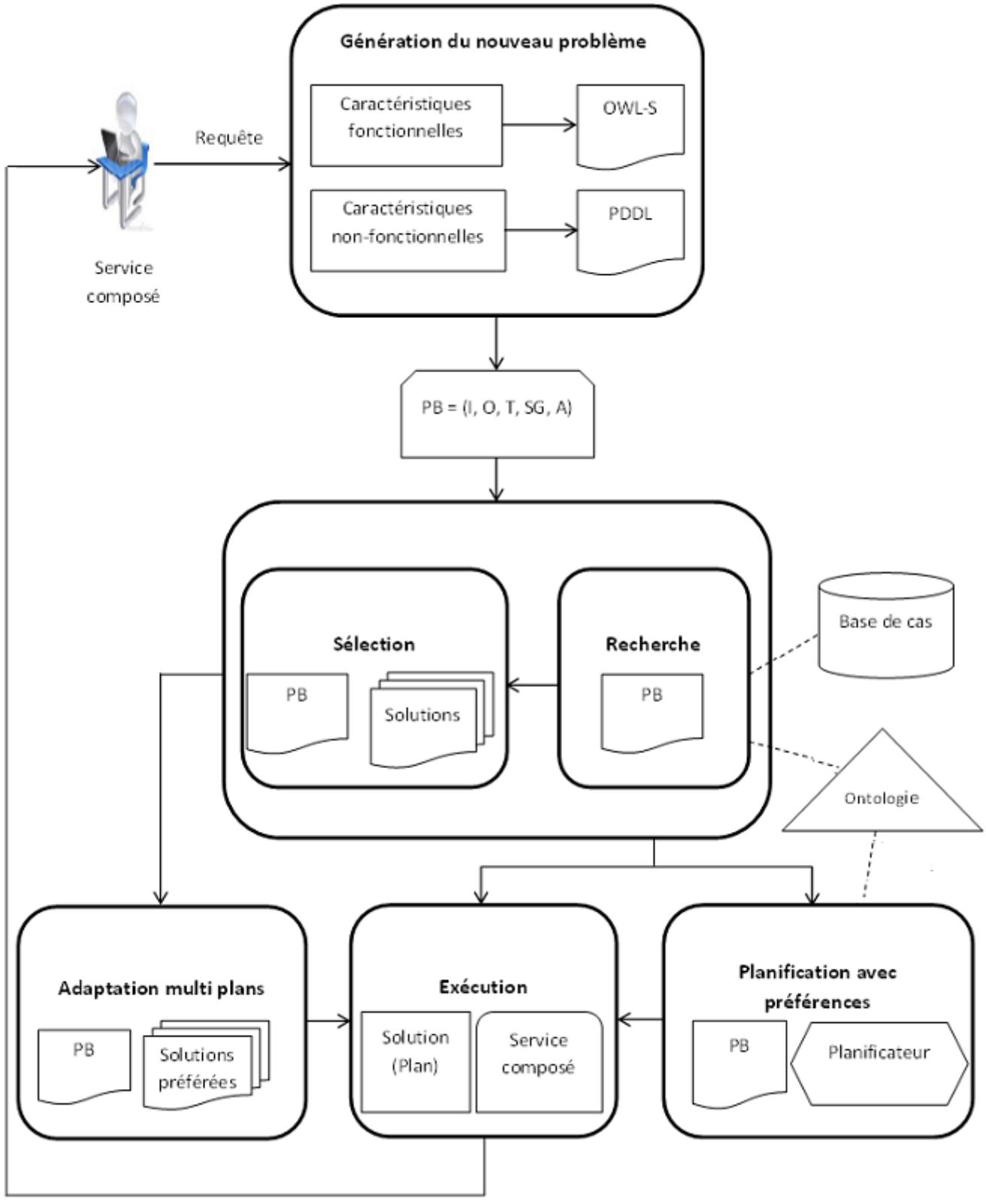

FIGURE 6.1: Architecture de l'approche proposée.

L'architecture de notre approche CBP-P4WSC de composition de service web est constituée de plusieurs modules qui permettent d'assurer les différentes phases

proposées dans le processus global. Ainsi, les modules sont comme suit :

Module de transformation Ce module prend la requête de l'utilisateur pour la transformer en un nouveau problème de composition. La requête doit contenir en plus des caractéristiques fonctionnelles du service demandé, les caractéristiques non-fonctionnelles et les préférences de l'utilisateur. Le module prend en entrée ces caractéristiques et produit en sortie un nouveau problème qui a la même structure que les problèmes des cas enregistrés dans la base de cas.

Module de remémoration Ce module a pour tâche de trouver les cas similaires dans la base de cas. Il prend en entrée le nouveau problème à résoudre ainsi que la base de cas pour chercher les cas les plus similaires. Ce module utilise un processus de remémoration composé de deux étapes : recherche et sélection. Dans la phase de recherche, le module compare le nouveau problème avec les parties problèmes des solutions qui se trouvent dans la base de cas. L'étape de sélection n'est déclenchée que si la recherche est concluante et réussit à trouver plusieurs solutions possibles. Les résultats de ce module sont différents et représentent l'entrée de différents modules par conséquent.

Module d'adaptation Nous avons recours à ce module lorsque le résultat est composé d'un ensemble de solutions qui doivent être adaptées selon les exigences de l'utilisateur. Durant ce processus, les solutions trouvées vont être modifiées et fusionnées pour obtenir une meilleure solution au problème posé par l'utilisateur.

Module de planification Ce module est responsable de la création de nouvelle composition. Il utilisera un planificateur qui prend les exigences fonctionnelles et les préférences de l'utilisateur comme entrée et qui produit un service composé. Nous faisons appel à ce module lorsque la recherche est infructueuse, c'est-à-dire que le module n'a pas trouvé de cas similaire dans la base de cas.

Module d'apprentissage La mise à jour de la base de cas consiste à ajouter ou supprimer des cas. Si le système arrive à résoudre un nouveau problème par

planification ou adaptation, la base de cas doit être mise à jour en ajoutant un nouveau cas composé du nouveau problème, la solution, la qualité et le compteur est incrémenté. Ce module est utilisé chaque fois qu'un nouveau cas est composé.

Module d'exécution La nouvelle solution doit être exécutée pour l'utilisateur. Ce module a comme entrée la nouvelle solution ou le plan et il doit le concrétiser avec des services réels qui vont être invoqués.

6.4 Expérimentation

Basé sur la spécification et la conception élaborée préalablement, un prototype a été développé.

6.4.1 Présentation de l'étude de cas

Nous utilisons le domaine Trucks qui est un domaine logistique de transport de paquets entre des locations en utilisant des camions et sous certaines contraintes. L'espace de chargement de chaque camion est organisé par zones : un paquet peut être (dé) chargé sur une zone d'un camion que si les zones situées entre la zone considérée et la porte du camion sont libres. En outre, certains paquets doivent être livrés dans un certain délai. Dans ce domaine, il est important de trouver des plans de bonne qualité. L'ensemble des problèmes dans Trucks comprennent plusieurs problèmes. Le nombre de préférences dans ces ensembles de problèmes varient en taille, avec plusieurs ayant plus de 100 préférences par problème.

Goals :

(delivered package1 l1)

(delivered package2 l2)

(delivered package3 l2)

Preferences :

(preference p1B (sometime-before (delivered

package2 l2) (delivered package1 l1)))

(preference p4A (within 919.7 (delivered package1 l1)))

(preference p4B (within 919.7 (delivered package2 l2)))

(preference p4C (within 1813.7 (delivered package3 l2)))

6.4.2 Présentation des outils technologiques utilisés

Pour mettre en œuvre le prototype, nous avons utilisé le NetBeens qui est un environnement de développement intégré (IDE) pour Java, placé en open source par Sun en juin 2000 sous licence CDDL (Common Development and Distribution License). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, XML et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages web). NetBeans est lui-même développé en Java, ce qui peut le rendre assez lent et gourmand en ressources mémoires. Nous avons choisi le langage java car il support des packages externes à la réutilisation parmi ces packages ceux qui aident à la manipulation des fichiers OWL et OWL-S ce qui répond à nos besoins. Le prototype a été développé sur un PC Windows équipé d'un processeur Intel Core I3-3110M 2.40 GHz CPU et d'une capacité mémoire de 4 GB.

Nous avons utilisé PDDL3.0 [23] pour représenter les préférences des utilisateurs. Pour la planification, nous avons choisi un planificateur indépendant qui est le SGPlan [27]. Enfin, OWL-S API a été utilisé pour la manipulation des fichiers OWL-S (la description sémantique du service correspondant à la requête).

6.5 Implémentation de l'outil

6.5.1 Construction de la Base de cas

Au début, la base de cas est vide. Pour pouvoir remplir cette base, nous avons collecté une base de cas formée de 20 problèmes. Ensuite, nous avons utilisé un planificateur pour trouver les différentes compositions aux problèmes. Chaque cas est alors décrit par plusieurs critères qui représentent tous les caractéristiques fonctionnelles et non-fonctionnelles du service composé. Il est aussi décrit par la solution qui représente le plan d'exécution de la composition, ainsi qu'une qualité de la composition et du compteur. La figure 6.2 montre l'interface de notre framework. Enfin, la construction de la base de cas se fait en utilisant cette interface.

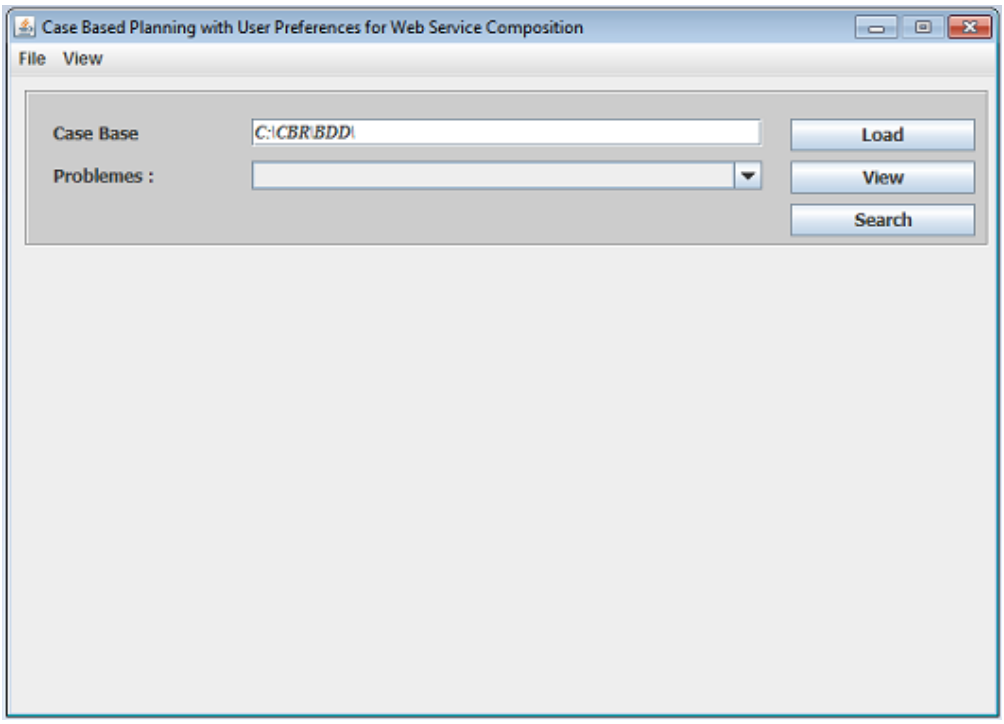

FIGURE 6.2: Interface de notre Framework.

6.5.2 Remémoration

Pour calculer la similarité entre un nouveau cas et la base de cas, nous commençons par l'étape de la recherche. Cette étape consiste à comparer le nouveau problème avec les problèmes résolu et en procédant par décomposition si nous ne trouvons pas de correspondance avec le problème entier. La figure 6.3 montre le résultat de la phase de recherche.

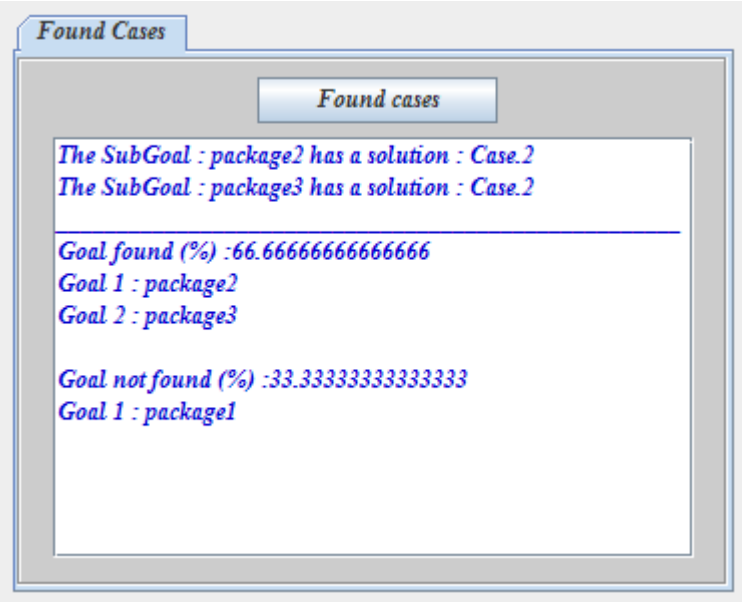

FIGURE 6.3: Résultats de l'étape de recherche.

Après la phase de la recherche, une phase de sélection est exécutée. Le résultat de la phase de la recherche est un ensemble de solutions trouvées. Ces solutions peuvent répondre au but ou un sous but et un but (sous but) peut avoir une ou plusieurs solutions. Pour cela, le système choisira la solution qui correspond le plus aux préférences de l'utilisateur. La figure 6.4 montre le résultat de cette phase.

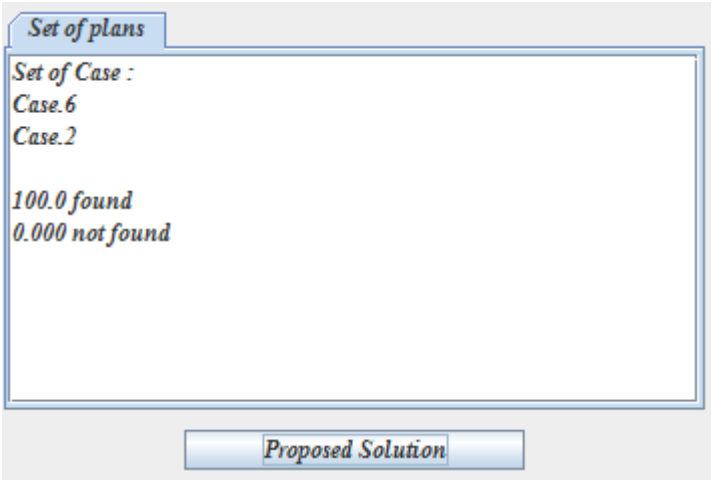

FIGURE 6.4: Résultats de l'étape de sélection.

6.6 Evaluation

Dans le cadre de l'évaluation du Framework proposé, nous avons effectué plusieurs expériences pour illustrer sa validité et son efficacité, et pour évaluer également ses performances et son évolutivité.

Le premier but de notre travail est de trouver des solutions à notre problème. Nous avons proposé de décomposer le but en un ensemble de sous but pour élargir nos chances à trouver des cas qui répondent à ces problèmes. Notre méthode de recherche de cas similaires extrait tous les cas qui sont similaires à notre problème.

Exemple Soit le problème présenté dans l'étude de cas, composé de trois buts et cinq préférences à respecter. L'exécution de la phase de recherche pour ce problème renvoie les résultats suivants :

Buts	Cas 2	Cas 3	Cas 6	Cas 7	Cas 13	Cas 15	Cas 16	Cas 19
But 1	0	0	1	1	0	0	0	0
But 2	1	1	0	0	1	1	0	0
But 3	1	0	0	1	1	0	1	1

TABLE 6.1: Nombre de buts trouvés pour chaque cas

Nous remarquons dans ces résultats que si le but de la recherche est de trouver une solution dont la partie problème correspond au problème posé par l'utilisateur, le système ne trouvera pas de solution, mais en le décomposant nous avons trouvé les résultats précédents.
Le tableau 6.1 représente les buts à atteindre et les cas correspondants. Chaque cas peut répondre à un ou plusieurs buts. La figure 6.5 montre la correspondance entre les buts et les cas.

La figure 6.5 montre que deux cas (Cas2 et Cas13) peuvent répondre aux mêmes buts, de même pour le but 1 où nous trouvons les cas 6 et 7.

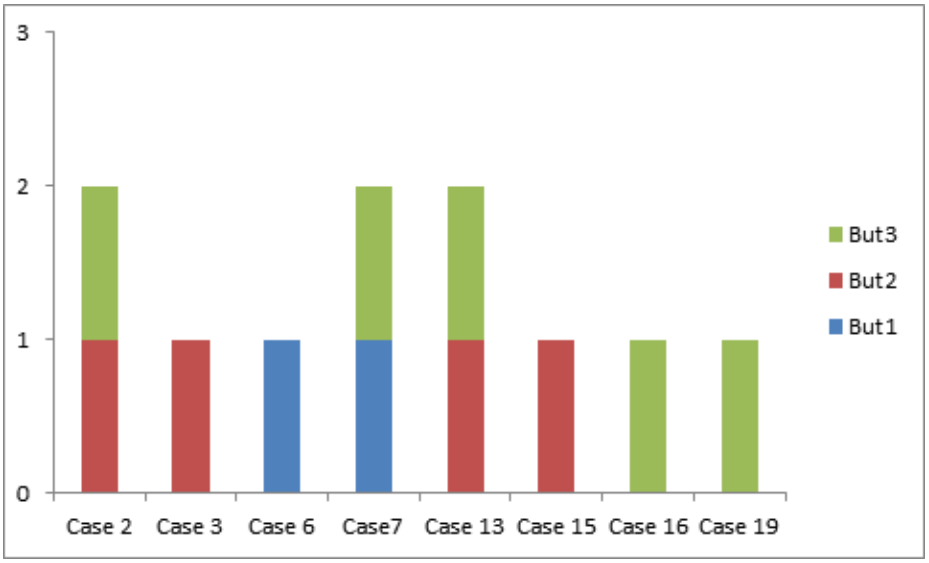

FIGURE 6.5: Cas trouvés par rapport aux buts recherchés.

Après cette phase de recherche, le système passe à la sélection, où il doit choisir parmi ces cas celui ou ceux qui peuvent répondre au problème posé. Pour cela, il utilise les préférences de l'utilisateur pour pouvoir choisir entre ces cas. Le résultat de cette étape est représenté dans le tableau 6.2.

Pour ce problème, le système a trouvé comme résultats de la phase de remémoration deux cas qui sont Cas 2 et Cas 6. Donc ces cas vont être utilisés dans la phase d'adaptation pour former la nouvelle solution en tenant compte des propriétés

Buts	Cas 2	Cas 6
But1	0	1
But2	1	0
But3	1	0

TABLE 6.2: Cas sélectionnés.

fonctionnelles et non-fonctionnelles de l'utilisateur.

La figure 6.6 montre le pourcentage moyen de succès de notre méthode d'extraction de cas en décomposant l'objectif principal en sous objectifs. Il est clair que lorsque le nombre de buts augmente, le pourcentage de récupération réussie augmente également. Cela est dû au fait qu'on est dans le domaine de la composition de services.

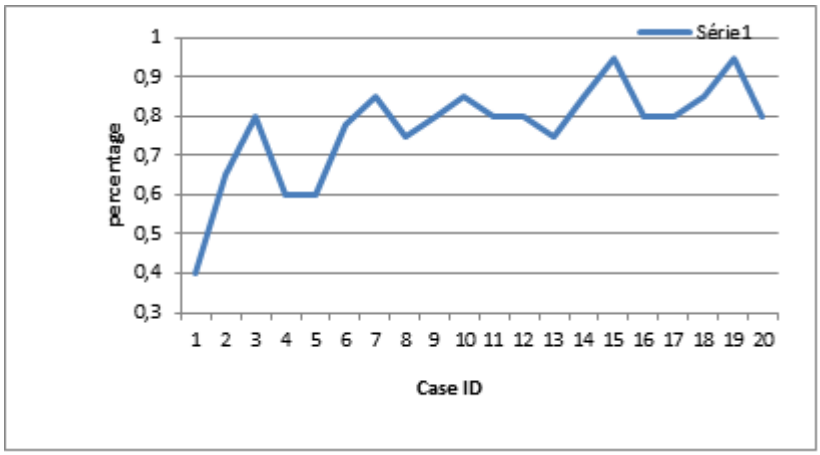

FIGURE 6.6: Pourcentage de récupération de service par rapport au nombre de buts.

La Figure 6.7 , montre la relation entre le nombre de cas trouvés par la décomposition de l'objectif (but). Par conséquent le résultat montre que l'utilisation de l'algorithme de remémoration des cas pour la composition de services Web par décomposition du problème est plus efficace que celui des méthodes basées sur la recherche du problème entier.

Après la phase de recherche, nous trouvons la phase de sélection qui consiste à sélectionner les meilleurs cas selon les préférences des utilisateurs. La figure 6.8,

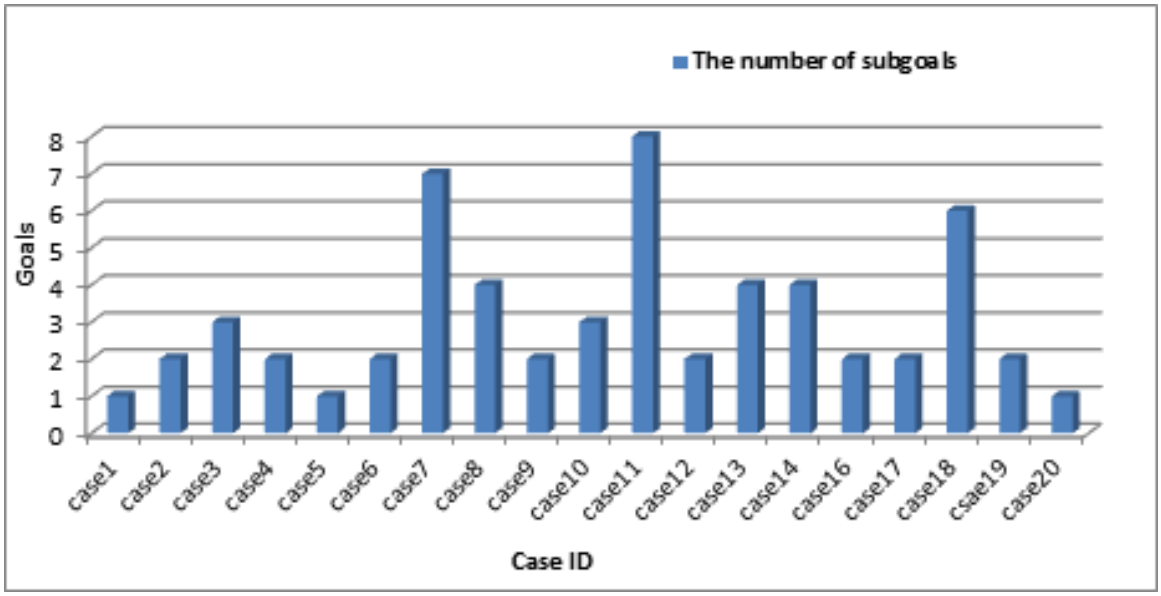

FIGURE 6.7: Résultats expérimentaux sur des cas trouvés.

montre les cas sélectionnés selon les préférences de l'utilisateur. Nous pouvons trouver plusieurs solutions qui répondent à l'objectif, mais pas les préférences de l'utilisateur. Nous pouvons montrer que le procédé de sélection peut donner de meilleures solutions.

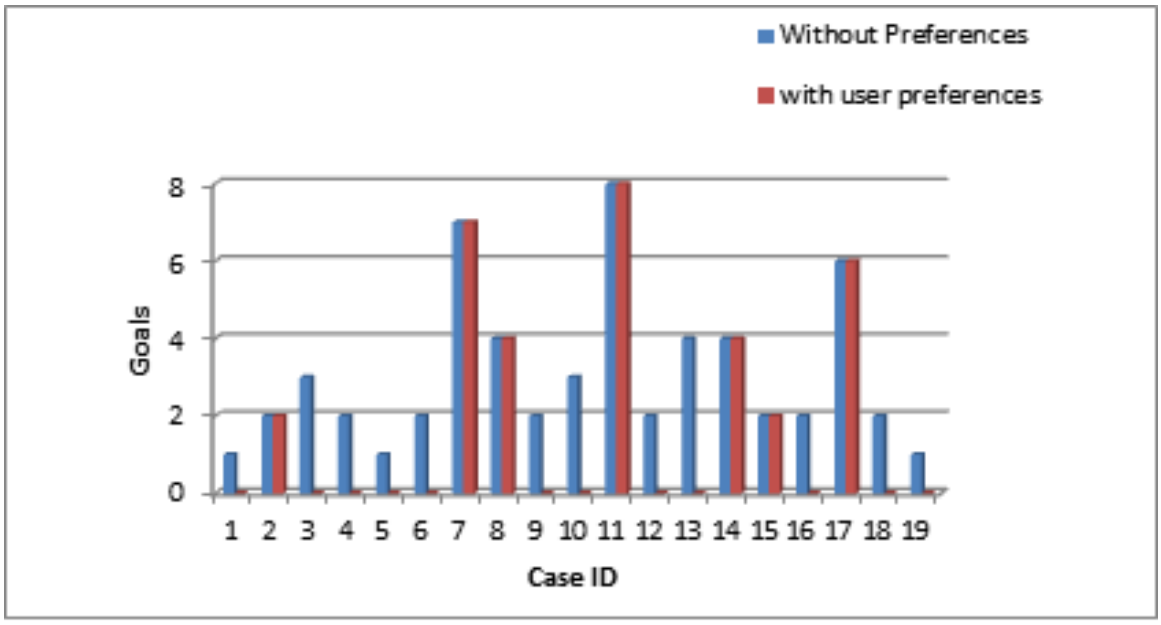

FIGURE 6.8: Résultats expérimentaux sur des cas sélectionnés.

En conséquence, les résultats de l'expérience montrent que l'utilisation de l'algorithme de recherche donne de meilleurs résultats pour la recherche des cas similaires

dans la base de cas en subdivisant le but. L'algorithme de la sélection, à son tour, donne des résultats encourageants concernant les préférences des utilisateurs. Au lieu de donner une quelconque composition à l'utilisateur, le système lui donne les meilleurs services selon ses choix.

6.6.1 Les indicateurs d'évaluation

Pour pouvoir évaluer les performances d'un système en termes de qualité des résultats, il faut mesurer la différence entre les résultats attendus et les résultats obtenus. Pour cela, un ensemble d'indicateurs de mesure de performances [Nakache et Métais, 2005] est utilisé. Le tableau 6.3 montre les différents indicateurs et leurs descriptions.

Indicateurs	Descriptions
Rappel et Silence	Le rappel est égal au nombre de cas sources pertinents retrouvés par rapport au nombre de cas sources pertinents contenus dans la base de cas. Le silence est le nombre de cas source pertinents non extraits sur le nombre total de cas sources pertinents.
Précision et Bruit	La précision est le taux du nombre de cas sources pertinents retrouvés par rapport au nombre de cas sources total retrouvé par le système. Le bruit est égal au nombre de cas sources non pertinents extraits sur le nombre total de cas sources extraits.
Erreur	L'erreur est le taux de la somme des cas pertinents non retrouvés et des cas non pertinents retrouvés, par rapport au nombre total de cas sources extraits.

TABLE 6.3: Indicateurs d'évaluation

Les indicateurs Rappel et Silence sont utilisés pour savoir si tous les bons cas sont sélectionnés ou non. Le taux de rappel doit être élevé, sinon cela veut dire que plusieurs cas sources pertinents ne sont pas fournis et le taux de silence sera élevé. Le silence s'oppose alors au rappel (rappel + silence = 1).

Pour savoir si les cas sélectionnés sont tous bons, les indicateurs Précision et Bruit sont utilisés. La précision s'oppose alors au bruit (précision + bruit = 1). Si elle est élevée, cela signifie que peu de cas sources non pertinents sont proposés par le système et que ce dernier peut être considéré comme précis.

6.6.1.1 Evaluation des résultats à l'exemple

En examinant les différents cas de l'exemple considéré, nous avons identifié les cas pertinents répondant à la requête posée. Le tableau 6.4 présente un récapitulatif des résultats obtenus selon leur pertinence. La figure 6.9 expose ensuite les indicateurs de performance calculés sur la base de ces résultats.

	Cas extraits	Cas non extraits	Total
Pertinent	8	0	8
Non pertinent	2	10	12
Total	8	10	20

TABLE 6.4: Récapitulatif des résultats obtenus selon leur pertinence.

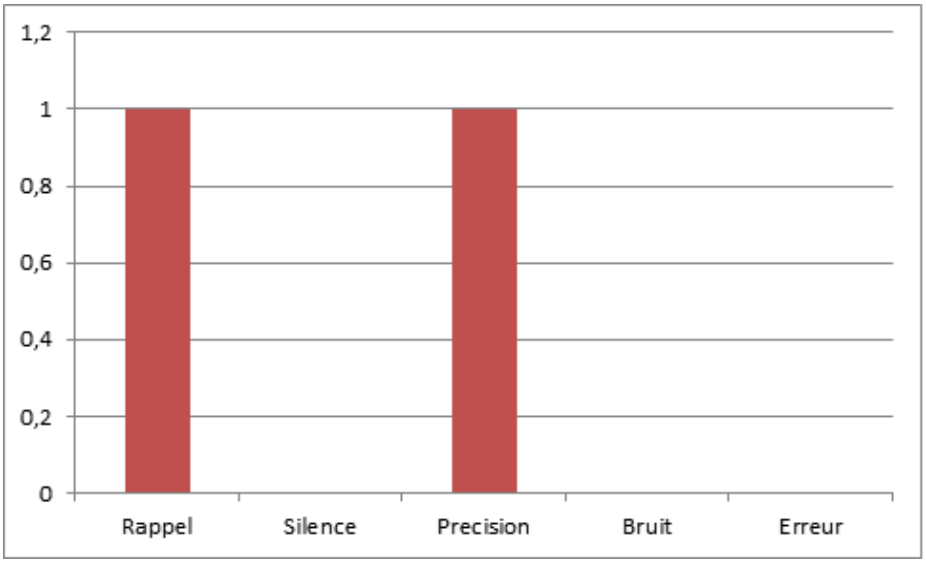

FIGURE 6.9: Graphe des indicateurs de performance de la qualité des résultats.

Ce qu'on peut dire des résultats obtenus est que notre système a pu trouver tous les cas possibles et pertinents et avec un taux d'erreur égale à zéro.

6.7 Conclusion

Dans ce chapitre, nous avons présenté les détails d'expérimentation de notre approche. Nous y avons exposé les détails de cette approche qui a permis de tester les algorithmes présentés. Nous avons décrit la mise en œuvre des différents modules de l'architecture proposée, en précisant le rôle de chaque élément utilisé. Ensuite, une présentation a été faite des outils utilisés et du prototype que nous avons développé comme support à notre approche. Nous avons présenté et discuté quelques expérimentations mises en place. Le but principal de cette implémentation est d'évaluer nos propositions, de tester la faisabilité de notre approche, et de démontrer que l'approche proposée peut montrer que les préférences non-fonctionnelles des utilisateurs peuvent influencer sur le résultat final de la composition. Enfin, nous pouvons constater que ces tests ont permis généralement de valider notre approche de composition des services Web dans sa globalité. Dans le prochain chapitre, nous présenterons nos conclusions et perspectives.

Chapitre 7

Conclusion et perspectives

Sommaire

7.1 Conclusion

Dans le présent rapport de thèse, nous avons proposé une approche fondée sur le raisonnement à partir de cas et la planification pour la sélection et la composition de services Web sémantiques. Les objectifs de cette approche varient de la capitalisation de l'expérience et le traitement sémantique jusqu'à la prise en considération des besoins fonctionnels aussi bien que non-fonctionnels et l'optimisation du temps de composition. Cette section résume les contributions majeures du travail rapporté dans ce manuscrit.

Utilisation du RàPC L'opération de la composition automatique de services Web est un mécanisme qui s'avère énormément couteux en termes de traitement vu le nombre élevé de services Web disponibles sur Internet qui rend la phase de recherche et de sélection très difficile, d'un côté et les problèmes liés à la composition d'un autre côté. Aussi, toute approche de composition de services Web devrait-elle être rationnelle en termes de traitement. Autrement, l'approche

CBP-P4WSC que nous avons proposée est fondée sur le retour d'expérience, notamment sur les mécanismes du RàPC (Raisonnement à Partir de Cas). Cela lui permet justement de capitaliser l'expérience pour délivrer des solutions inspirées des cas similaires déjà traités, et aussi d'en sélectionner les meilleurs sur la base de l'expérience de leur exécution.

La planification pour la composition : Le problème de la composition est vraisemblable à un problème de planification. Ainsi, nous avons utilisé la planification pour trouver de nouvelles solutions pour la composition soit directement en utilisant un planificateur qui utilise un langage qui permet de définir les préférences des utilisateurs ou par l'adaptation qui utilise les différents plans extraits pour former un nouveau plan qui répond à la requête de l'utilisateur.

Préférences des utilisateurs : Par ailleurs, outre le fait que notre approche prend en considération les besoins fonctionnels aussi bien que non-fonctionnels, cela permettra de trouver des composition de services web qui répondent exactement aux exigences de l'utilisateur sans avoir à perdre du temps à faire des compositions qui ne vont pas être utilisées par la suite si l'utilisateur n'est pas satisfait de la solution. Nous avons pris cet aspect tout au long du processus de composition.

Processus de remémoration : Il est à noter également que notre approche prévoit deux processus pour la remémoration des cas afin de sélectionner les meilleurs plans de composition. Un processus qui couvre la phase de recherche des solutions qui sont similaires au problème posé. Ce processus focalise essentiellement sur l'aspect fonctionnel puisque l'objectif de tout utilisateur serait naturellement de trouver une solution qui doit obligatoirement satisfaire son besoin fonctionnel. Notre algorithme de recherche procède par décomposition du problème en un ensemble de buts pour trouver tous les cas possibles. Un deuxième processus, la phase de sélection qui prend l'aspect non-fonctionnel

et qui répond dans la mesure du possible à ses exigences ou préférences non-fonctionnelles. Ce processus sélectionne les meilleures solutions en se basant sur les préférences de l'utilisateur. Notamment, cela permettrait de minimiser considérablement le temps de la composition.

7.2 Prespectives

Par ailleurs, nous pensons que notre travail ouvre d'autres voies de recherche. Plusieurs extensions peuvent être proposées. Ces perspectives répondent à un objectif principal qui est l'amélioration de la composition des services web. Les perspectives que nous exposons représentent des améliorations de l'approche de composition proposée.

Propriétés non-fonctionnelles complexes : La représentation des propriétés non-fonctionnelles que nous avons adoptée est une représentation simple qui couvre un large éventail des propriétés non-fonctionnelles pouvant être exprimées sous forme de contraintes simples. Nous envisageons de travailler sur des représentations complexes.

Méthode d'appariement des propriétés non-fonctionnelles : Repenser la manière d'apparier les propriétés non-fonctionnelles en intégrant les techniques d'appariement sémantiques dans la phase de sélection.

Organisation de la base de cas : pour une meilleure réutilisation des informations présentes dans la base de cas, une organisation de la base de cas peut faciliter la phase de recherche.

Bibliographie

[Aamodt et al., 1994] Aamodt A. et Plaza E. Case-Based Reasoning : Foundational Issues, Methodological Variations, and System Approaches. AI Communications, 7(i) : pp 39 59, 1994.

[Aamodt, 2004] Aamodt, A. Knowledge-Intensive Case-Based Reasoning and Sustained Learning. Proc. of the 9th European Conference on Artificial Intelligence, ECCBR'04, Lecture Notes in Artificial Intelligence, 1-15, Springer, 2004.

[Agarwal et al., 2001] Agarwal, A. et Arkin, A. The BPML specification. BPML Working Draft 0.4, http ://wwww.bpmi.org, March 2001.

[Alonso et al., 2004] Alonso, G., Casati, F., Kuno, H., Machiraju, V. Web Services : Concepts, Architectures and Applications. Springer, 2004, 354p.

[Alves et al., 2007] Web Services Business Process Execution Language Version 2.0. OASIS Standard, April 2007. Available at http ://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[Andrews et al., 2003] Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I.,Weerawarana, S. Business Process Execution Language for Web Services, Version 1.1. IBM Specification. Technical report, IBM, May 2003.

[Aquin et al., 2004] . D'Aquin, M., Lieber, J. et Napoli, A. Représentation de points de vue pour le RàPC. In Langages et Modèles à Objets - LMO'04. (Lille, France). Revue des Sciences et Technologies de l'Information, RSTI - série L'Objet, 10, 2-3, 245-258.

[Ashley, 1990] Ashley, K. Modelling Legal Argument : Reasoning with Cases and Hypotheticals. Mit Press, Cambridge, MA.

[Baier et al., 2007] Baier, A. and Bacchus, F. and Sheila, A. McIlraith. A Heuristic Search Approach to Planning with Temporally Extended Preferences Jorge Department of Computer Science University of Toronto Toronto, Canada (2007).

[Banerji et al., 2002] Banerji, A., Bartolini, C., Beringer, D. , Chopella, V., Govindarajan, K., Karp, A., Kuno, H. , Lemon, M. , Pogossiants, G. , Sharma, S. and Williams, S. Web Services Conversation Langage (WSCL) 1.0. W3C http ://www.ebxml.org/specs/ebBPSS.pef, March 2002.

[Barros et al., 2005] Barros, A., Dumas, M., Oaks, P. Standards for Web Service Choreography and Orchestration : Status and Perspectives. In : Procs of the 3rd International Conference the Business Process Management (BPM 2005), 1st International Workshop on Web Service Choreography and Orchestration for Business Process Management, Nancy, France, Sept. 2005, pp.1-15.

[Barros et al., 2006] Barros, A., Dumas, M. and Oaks, P. Standards for Web Service Choreography and Orchestration : Status and Perspectives. In Proceedings Business Process Management Workshops, pages pp. 61-74, Nancy, France, 2006.

[Bartolini et al., 2002] Banerji, A., Bartolini, C., Beringer, D. , Chopella, V. and Et. Web services conversation language (wscl) 1.0. Technical report, March 2002.

[Beaubouche, 1994] Beaubouche, N. Anais : raisonnement à partir de cas en résolution de problèmes, Thèse de doctorat, Université Paris 6, Paris, France.

[Bellwood et al, 2002] Bellwood T. et al. Universal Description, Discovery and Integration specification (UDDI) 3.0.2002 Online : http ://www.uddi.org/pubs/uddi-v3.00-published-20020719.htm

[Benard et al.,2008] Bénard, R., De Loor, P. La révision et l'apprentissage de cas pour les simulations temps-réel en réalité virtuelle, In 16 atelier du raisonnement à Partir de Cas, avril 2008, Nancy, France. pp 24-35.

[Benatallah et al., 2002] Benatallah, B., Dumas, M., Fauvet, M. C., Rabhi, F.A., Sheng, Q.Z. Overview of Some Patterns for Architecting and Managing Composite Web Services. ACM SIGecom Exchanges, 2002, vol.3, n°3, pp.9-16.

[Benatallah et al., 2005] Benatallah, B., Dijkman, R., Dumas, M., Maamar, Z. Service Composition : Concepts, Techniques, Tools and Trends. In : Stojanovic Z., Dahanayake A., Eds. Service- Oriented Software Engineering : Challenges and Practices. Idea Group Inc (IGI), 2005, pp.48- 66.

[Bergmann et al., 1996] Bergmann, R., Wlike, W. : PARIS flexible Plan adaptation by abstracting and Refinement , ECAI 1996 Worshop on Adaptation in Case-Based Reasoning.

[Berners-Lee et al., 2001] Berners-Lee, T. , Hendler, J. and Lassila, O. The Semantic Web. In the Scientific American Magazine, 284(5) :34–43, 2001.

[Bosca et al., 2006] Bosca, A., Corno, F., Valetto, G., Maglione,R.On-the-fly construction of web services compositions from natural language requests, in : Journal of Software 1.pp. 40–50, (2006) .

[Bruijn J. et al., 2005] Bruijn ,J. et al., Web Service Modeling Ontology (WSMO). W3C Member Submission, 2005. http ://www.w3.org/Submission/WSMO/.

[Cerami, 2002] Cerami, E. Web Services Essentials. O'Reilly and Associates, Inc. USA, February 2002.

[Cheng et al., 2006] Cheng, R., Su, S., Yang, F. and Li, Y. Using Case-Based Reasoning to Support Web Service Composition. Springer-Verlag Berlin Heidelberg, 2006.

[Chinnici et al., 2007] Chinnici, R., Moreau, J.-J., Ryman, A., Weerawarana, S. Web Services Description Language (WSDL) Version 2.0 Part 1 : Core Language. Disponible sur : <http ://www.w3.org/TR/wsdl20>.

[Chung et al., 1999] Chung, L., Nixon, B.A., Yu ,E. and Mylopoulos, J. Non-Functional Requirements in Software Engineering. KluwerAcademic Publishers, Boston Hardbound, October 1999.

[Claro et al., 2005] Claro, D. B. , Albers, P. and Hao, J.-K. Approaches of web services composition – comparison between BPEL4WS and OWL-S. In International Conference on Enterprise Information Systems (ICEIS 4), pages 208–213, 2005.

[Claro et al., 2006] Claro, D., Albers, P., Hao, J. Web Services Composition. In : Cardoso, J., Sheth, A., Eds. Semantic Web Services, Processes and Applications. Springer, 2006, pp.195-225.

[Dhouib, 2009] Dhouib, D. Aide multicritère au pilotage d'un processus basée sur le raisonnement à partir de cas. Thèse de doctorat de l'université de Paris 8 Vincennes – Saint-Denis, 2009.

[Diaz et al., 2006] Diaz, O. G. F., Salgado, R. S. , Moreno, I. S. and Ortiz, G. R. Searching and Selecting Web Services Using Case Based Reasoning. Springer-Verlag Berlin Heidelberg 2006.

[Dustdar et al., 2005] Dustdar, S. and Schreiner, W. A survey on web services composition, International Journal of Web and Grid Services 1 (2005), no. 1, 1-30.

[Englmeier et al., 2006] Englmeier,K., Pereira, J., Mothe, J. Choreography of web services based on natural language storybooks. In : Proceedings of the 8th International Conference on Electronic Commerce, pp. 132–138, 2006.

[Erl, 2007] Erl, T. Soa principles of service design, Prentice Hall, 2007.

[Feng , 2009] Feng, J. H. Z. Automated Composition of Semantic Web Services Using Case-Based Planning .International Forum on Information Technology and Applications, 2009.

[Fensel et al., 2002] Fensel, D., Bussler, C. The Web Service Modeling Framework WSMF. Electronic Commerce Research and Applications 1, 2002. Pages : 113-137.

[Fikes et al., 1971] Fikes, R. et Nilsson, N. STRIPS : A new approach to the application of theorem proving to problem solving. Artificial intelligence, 2(3-4) :189-208, 1971.

[Foster et al., 2003] Foster, H., Uchitel, S., Magee, J. et Kramer, J., Model-based Verification of Web Service Composition, October 2003.

[Fox et al., 2003] Fox, M. et Long, D. (2003). PDDL2. 1 : An extension to PDDL for expressing temporal planning domains. Journal of Articial Intelligence Research, 20(1) :61-124.

[Fuchs et al., 1999] Fuchs, B., Lieber, J., Mille, A., et Napoli, A. Towards a unified theory of adaptation in case-based reasoning. In Case Based Reasoning Research and Development, The 3rd International Conference on Case-Based Reasoning (ICCBR'99), volume 1650 de LNAI, pages 104–117, Seeon Monastery, Germany, August 25-27 1999. Springer Verlag.

[Fuchs et al., 2006] Fuchs, B., Lieber, J., Mille, A. et Napoli, A. Une première formalisation de la phase d'élaboration du raisonnement à partir de cas. Actes du 14ième atelier du raisonnement à partir de cas, Besançon, mars, 2006.

[Gardarin, 2007] Gardarin, G. Xml, des bases de données aux services web, Dunod, 2007.

[Gebhardt et al., 1997] Gebhardt, F., Voß, A., Gräther, W., Schmidt-Belz, B. Reasoning With Complex Cases. Kluwer academic publishers, 1997.

[Gerevini et al., 2006] Gerevini, A., Long, D. Plan constraints and preferences in pddl3, ICAPS Workshop on Soft Constraints and Preferences in Planning, (2006).

[Gerevini et al., 2006] Gerevini, A., Dimopoulos, Y., Haslum, P. and Saetti, A., Deterministic part, 5th International Planning Competition. http ://eracle.ing.unibs.it/ipc-5/, 2006.

[Ghallab et al., 1998] Ghallab, M. Ecole Nationale, Constructions Aeronautiques, Craig Knoblock Isi, Keith Golden, Scott Penberthy, David E Smith, Ying Sun, Daniel Weld, and Contact DrewMcdermott. Pddl - the planning domain definition language, version 1.2. Technical report, 1998.

[Gruber, 1993] Gruber, T. R. Toward principles for the design of ontologies used for knowledge sharing. Presented at the Padua workshop on Formal Ontology, March 1993, later published in International Journal of Human-Computer Studies, Vol. 43, Issues 4-5, November 1995, pp. 907-928.

[Hachemi et al., 2015] Hachemi, Y. et Benslimane, S.M. Preference-Based Web Service Composition : Case-Based Planning Approach. I.J. Information Technology and Computer Science, 2015, 06, 74-82.

[Hammond, 1986] Hammond, K.J. CHEF : a model of case-based planning. Proceedings of AAAI'86, Morgan Kaufman, 267-271, 1986.

[Hanks et al., 1995] Hanks, S., et Weld, D. A Domain-Independent Algorithm for Plan Adaptation. Journal of Artificial Intelligence Research 2 , pp. 319-360, (1995).

[Henni et al., 2013] Henni, F. et Atmani, B. Applying CBR Over an AI Planner for Dynamic Web Service Composition. International Journal of Information Technology and Web Engineering, 8/4, Octobre 2013, pages 36/47.

[Horrocks et al., 2003] Horrocks, I., Patel-Schneider, P., and van Harmelen, F. (2003). From SHIQ and RDF to OWL : The making of a Web ontology language. Journal of Web Semantics, 1(1) :7–26.

[Hsu et al., 2007] Hsu, C.-W., Wah, B., Huang, R., and Chen, Y. Constraint partitioning for solving planning problems with trajectory constraints and goal preferences. In Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI), pages 1924–1929, Hyderabad, India, (2007).

[Huhns et al., 2005] Huhns, M. N. and Singhl, M. P. Service-oriented computing : Key concepts and principles, IEEE Internet Computing 9 (2005), no. 1, 75-81.

[Jang et al., 2007] Jang, M., Sohn,J.C., Cho,H.K., Automated question answering using semanticweb services. In : Proceedings of the 2nd IEEE Asia-Pacific Service Computing Conference, pp. 344–348, (2007).

[Kavantzas et al., 2005] Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., Lafon, Y. and Barreto, C. Web services choreography description language version 1.0. Web Services Choreography Description Language Version 1.0.html, november 2005.

[Klyne et al., 2004] G. Klyne, J.J.Carroll, Resource Description Framework (RDF) : Concepts and Abstract Syntax. W3C Recommendation,Disponible sur : https ://www.w3.org/TR/rdf-concepts/, 2004.

[Lajmi et al., 2006] Lajmi, S., Ghedira, C., Ghedira, K. WeSCo-CBR : How to Compose Web services via Case Based Reasoning. Proceeding of the IEEE. International Conference on e-Business Engineering, Shanghai, pp.618-622, October 2006.

[Lajmi et al., 2009] Lajmi, S., Ghedira, C., Ghedira, K. CBR Method for Web Service Composition. Advanced Internet Based Systems and Applications, Volume 4879 of the series Lecture Notes in Computer Science pp 314-326.

[Lee et al., 2010] Lee, C-H. L., Liu, A. and Huang, H-H. Using Planning and Case-Based Reasoning for Service Composition. Journal of Advanced Computational Intelligence and Intelligent Informatics, 2010.

[Liberatore, 2005] Liberatore, P. On the complexity of case-based planning. Journal of Experimental and Theoretical Artificial Intelligence 17(3) :283–295, 2005.

[Limthanmaphon et al., 2003] Limthanmaphon,B. , Zhang, Y. Web Service Composition with Case-Based Reasoning. Proceedings of the Fourteenth Australasian Database Conference, Adelaide, pp.201– 208, 2003.

[Lin et al., 2008] Lin, N., Kuter, U., Sirin, E. : Web service composition with user preferences. In : Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 629–643. Springer, Heidelberg (2008).

[Liu et al., 2009] Liu, Y-Z., Qiu, S., Tao, H-J., Tian-Yizang, Wang, Y-D. A case-based reasoning approach to support web service composition. Proceedings of the Eighth International Conference on Machine Learning and Cybernetics, Baoding, 12-15 July 2009.

[Lopez de Mantaras et al., 2005] Lopez de Mantaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M.L., Cox, M., Forbus, K., Keane, M., Aamodt, A. et Watson, I. Retrieval, Reuse, Revise, and Retention in CBR. Knowledge Engineering Review, pp : 215-240, 2005.

[Maris, 2009] Maris, F. Planification SAT et Planification Temporellement Expressive. Les Systemes TSP et TLP-GP, 2009.

[Martin et al., 2007] Martin, D., Burstein, M., McDermott, D., McIlraith, S., Paolucci, M., Sycara, K., McGuinness, D., Sirin, E., and Srinivasan, N. (2007). Bringing semantics to Web services with OWL-S. World Wide Web Journal, 10(3) :243–277.

[Marvie et al, 2000] Marvie, R. et Pellegrini, MC. Etat de l'art des modèles de composants. Technical Report 2, CESURE RNRT Projet, May 2000.

[McIlraith et al., 2003] McIlraith, S., Martin D. Bringing Semantics to Web Services. IEEE Intelligent Systems. 2003, vol.18, n°1, pp.90-93.

[McGuiness et al., 2004] D.L.McGuiness, F. van Harmelen, OWL Web Ontology Language Overview. W3C Recommendation. Disponible sur : http ://www.w3.org/TR/owl-features/, 2004.

[Michael, 2002] S. Michael. Web services : beyond component-based computing. ACM, 45(10) : 71-76, 2002.

[Michalski, 1986] Michalski, R., Carbonell ,J. et Mitchell, T. (Eds.).Machine Learning, an Artificial Intelligence Approach, Morgan Kaufmann, Vol 2, pp. 271-391, 1986.

[Mille et al., 1996] Mille, A., Fuchs, B. et Herbeaux, O. A unifying framework for Adaptation in Case-Based Reasoning. In A. Voss, Ed., Proceedings of the ECAI'96 Workshop : Adaptation in Case-Based Reasoning, p. 22-28, 1996.

[Mille, 1995] Mille, A. Raisonnement basé sur l'expérience pour coopérer à la prise de décision, un nouveau paradigme en supervision industrielle. Thèse de doctorat, Université de Saint Etienne, 1995.

[Mille, 1999] Mille, A., Tutorial CBR : Etat de l'art de raisonnement à partir de cas.Plateforme AFIA'99, Palaiseau, 1999.

[Mille, 2006] Mille, A. Traces based reasoning (TBR) definition, illustration and echoes with storytelling. Rapport Technique RR-LIRIS-2006-002, LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/Ecole Centrale de Lyon, january 2006.

[Mitra et al., 2007] Mitra, N., Lafon, Y. SOAP Version 1.2 Part 0 : Primer (Second Edition). W3C Recommendation. Online : http ://www.w3.org/TR/SOAP/.

[Nakache et Métais, 2005] Nakache, D. et Métais, E. Evaluation : nouvelle approche avec juges. In INFORSID, p. 555–570, Grenoble, 2005.

[OASIS] OASIS. Organization for the Advancement of Structured Information Standards. Disponible sur : http ://www.oasis-open.org/home/index.php.

[Pednault, 1994] Pednault, E. ADL and the state-transition model of action. Journal of Logic and Computation, 4(5) :467, 1994.

[Peltz, 2003] Peltz, C. Web Services Orchestration and Choreography. IEEE Computer, 2003, vol.36, n°10, pp.46-52.

[Ram et al., 1996] Ram, A., Francis, A. : Multi-plan retrieval and adaptation in an experience-based agent. In : Leake, D.B. (ed.) Case-Based Reasoning : Experiences, Lessons, and Future Directions. AAAI Press, (1996).

[Reisbeck et Schank, 1989] Reisbeck, C.K, et Schank, R.C. Inside Case-Based Reasoning. Lawrence Erlbaum Associates, Hillsdale, NJ, US, 1989.

[Schank, 1982] Schank, RC. Dynamic memory : a theory of reminding and learning in computers and people. Cambridge UniversityPress, Cambridge, UK, 1982.

[Sheth, 2003] Sheth , A. P. Semantic Web Process Lifecycle : Role of Semantics in Annotation, Discovery, Composition and Orchestration. Invited Talk,Workshop on E-Services and the Semantic Web, at WWW 2003. Available at http ://lsdis.cs.uga.edu/lib/presentations/WWW2003-ESSW-invitedTalk-Sheth.pdf.

[Smyth et al., 1996] Smyth, B. et Keane, M.T. Using adaptation knowledge to retrieve and adapt cases. Knowledge-based systems, 9, 2, 127-135, 1996.

[Sohrabi et al., 2009] Sohrabi, S., Baier, J.A., McIlraith, S.A. : HTN planning with preferences. In :Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI), pp. 1790–1797, (2009).

[Spalzzi, 2001] Spalzzi, L. A survey on case-based planning. Artificial Intelligence Review 16(1) :3–36, 2001.

[Srivastava et al., 2003] Srivastava, B., Koehler, J. Web Service Composition – Current Solutions and Open Problems. In : Procs of the 13th International Conference on Automated Planning and Scheduling (ICAPS 2003), Workshop on Planning for Web Services, June 2003, Trento, Italy.

[Sun et al., 2011] Sun , Z ., Han , J., Ma, D. A Unified CDR Approach for Web Services Discovery, Composition and recommendation. In IPCSIT vol.3, IACSIT Press, Singapore, Pages 85-89 2011.

[Thakker et al., 2007] Thakker, D., Osman, T., Al-Dabass, D. Semantic-Driven Matchmaking and Composition of Web Services Using Case-Based Reasoning. Proceedings of the Fifth European Conference on Web Services, Washington , pp.67-76, November 2007.

[Thiagarajan et al., 2002] THiagarajan, R. , Srivastava, A. , Pujari, A. and Bulusu V. Bpml : A process modeling language for dynamic business models. In WECWIS, 2002.

[Torres et al., 2014] Torres, I.D and Guzmán-Luna, J. Applying Case-Based Learning to Improve the Efficiency in the Web Service Compositions. IACSIT International Journal of Engineering and Technology, Vol. 6, No. 3, June 2014.

[Watson, 1999] Watson, I. Case-based reasoning is a methodology not a technology. Knowledge - Based Systems, vol.12, n 5, Elsevier, UK, 1999.

[Waqar et al.„ 2004] Waqar, S. et Racca, F., Business services orchestration : The hypertier of information technology, Cambridge University Press, 2004.

[Wilke et al., 1998] Wilke, W. et Bergmann, R. Techniques and Knowledge Used for Adaptation During Case-Based Problem Solving. Proc. of IEA-98-AIE, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1998.

[Wilson et al., 2000] Wilson, D. C., Bradshaw, S. CBR Textuality. Expert Update, 3, 1, 28-37, 2000.

[Wolfgang et al., 2001] Wolfgang, E. et Nima, K. Cmponent technologies : Java beans, com, coeba, rmi, ejb and the coeba component model. SIGSOFT Softw. Eng. Notes, 26(5) : 311-312, 2001.

[w3c] Le World Wide Web Consortium, 1994 .http ://www.w3.org/

[XML, 1998] XML W3C. Extensible Markup Language (XML), https ://www.w3.org/XML/

[XPDL, 2008] Workflow Management Coalition, Xpdl support and resources, 2008. http ://www.wfmc.org/xpdl.html.

[Yang et al., 2004] Yang, J., Papazoglou, M.P. Service Component for Managing Service Composition Life-Cycle. Information Systems, vol.29, n°2, pp.97-125, 2004.