Faculté de technologie **Département d'Informatique**

Composition Personnalisée des Services Web

THÈSE

présentée et soutenue publiquement le : **19 Novembre 2014**

pour l'obtention du

Doctorat de l'université Djilali Liabes – Sidi Bel Abbes

(spécialité Informatique)

par

MEKOUR Mansour

Composition du jury

Président :	MALKI Mimoun, Professeur à l'Université de Sidi Bel Abbes
Examinateurs :	CHIKH Mohamed Amine, Professeur à l'Université de Tlemcen
	BOUCHIHA Djelloul, Maître de Conférences au Centre Universitaire de Naama
Directeur :	BENSLIMANE Sidi Mohammed, Maître de Conférences à l'Université de Sidi Bel Abbes

Evolutionary engineering and distributed information systems Laboratory (EEDIS)

Résumé

Le paradigme AOS (architecture orientée service) est devenu un standard pour la conception et le développement des applications distribuées à base de services web. La composition de services implique la construction de services à valeur ajoutée très souvent par la découverte, l'intégration et l'exécution des services préexistants. Cela peut être effectué de manière à ce que des services préexistants soient orchestrés en un ou plusieurs nouveaux services qui répondent mieux à une application composite. Malgré tous les avantages qu'elles apportent en termes d'interopérabilité et de réutilisation, les solutions de développement associées au paradigme AOS sont destinées aux programmeurs et restent difficiles à comprendre par le monde de l'entreprise. Pour être en phase avec le monde de l'entreprise, les applications à base de services web doivent être décrites en termes de propriétés extra-fonctionnelles qu'elles permettent de satisfaire et non seulement en termes de fonctionnalités qu'elles permettent de réaliser. Ceci permet de minimiser la discordance conceptuelle entre les services logiciels et l'énoncé des exigences des utilisateurs.

L'étude présentée dans ce document nous a permis d'identifier les problèmes liés d'une part, à la composition de services web, et d'autre part à l'association des propriétés extra fonctionnelles à cette composition. Nous nous sommes intéressés à ces deux problématiques qui nous ont conduits à la conception du framework $Ws-BeC$ (*Web service-Behavioral composition*) pour la composition de services web en tenant compte des propriétés comportementales des services composites. Le modèle permet aux concepteurs de composer des services et de prendre en compte des contraintes liées à la portée du comportement, par le biais de restrictions et de préférences, ainsi que des QoS des services composants. Au moment de l'exécution − par le biais de sélection, d'intégration voire de chevauchement de processus de services − le framework choisit d'une manière très approprié et beaucoup plus efficace, parmi les services accessibles, ceux qui répondent mieux aux besoins de la composition tout en respectant les caractéristiques comportementales.

Mots-clés: service web, composition dynamique de services, QoS, comportement de service, représentation et réécriture de processus de service, matching de processus, sélection, intégration, chevalement de processus de service.

Abstract

The SOA (service-oriented architecture) paradigm has become the standard for the design and development of distributed Web service-based-applications. Services composition involves the development of customized services often by discovering, integrating, and executing existing services. This can be done in such a way that already existing services are orchestrated into one or more new services that fit better to the composite application. Despite all the advantages they offer in terms of interoperability and reuse, the development solutions associated with the SOA paradigm is intended for programmers and are difficult to understand by the business world. To be in tune with the business world, service-based applications should be described in terms of requirements they can meet and not only in terms of features can they achieve. This minimizes the discrepancy between the conceptual software services and the statement of user requirements.

The work presented in this document led us to identify two main issues : (i) web service composition and (ii) behavioral properties associated with this composition. As we are interested in both issues we have designed a framework called $Ws-BeC$ (*Web service-Behavioral composition*) which covers the design of compositions of Web services as well as their executions. The framework relies on a model for the composition of Web services associated with behavioral properties that takes into account the expression of the behavioral properties that composite services are required to fulfil. In addition, compositions can be parameterized so as to allow the end user to impose that certain quality of services constraints and preferences are satisfied by the executions of the composite service. At execution time − based on service selection, integration and even interleaving − the framework selects, in a very appropriate way and much more efficient, among accessible services, those which best meet the composition's needs. The framework automatically ensures that beahvioral properties associated with the composition are fulfilled by exploiting the behavioral properties of the underlying selected services.

Keywords: web service, dynamic composition of services, QoS, service behavior, representation and rewriting of service processes, processes matching, selection, integration, interleaving of service processes.

Remerciements

Ces travaux de thèse se sont déroulés au sein du laboratoire EEDIS (Evolutionary engineering and distributed information systems Laboratory, à l'Université Djilali Liabes − Sidi Bel Abbes) dans le cadre du projet de recherche IISIE (Intégration et Interopérabilité des Systèmes d'Information d'Entreprise à base de service sémantique) du programme PNR (Programme National de Recherche).

Mes sincères remerciements aux respectueuses personnes que je tiens à reconnaître.

Je tenais en premier lieu à remercier vivement Monsieur le Docteur BENSLIMANE Sidi Mohammed mon directeur de thèse, pour sa collaboration, sa disponibilité, ses grandes compétences et ses précieux conseils qui m'ont permis d'enrichir mon travail. Je le remercie encore pour son contact constructif, si généreux sur le plan humain comme scientifique. Je le remercie également pour son soutien tout au long du déroulement de ma thèse.

J'adresse mes profondes reconnaissances aux membres du jury de m'avoir fait l'honneur d'avoir accepté de consacrer un temps précieux à l'examen, l'appréciation, et le jugement de ce travail.

Je suis vivement reconnaissant au Professeur MALKI Mimoun qui, malgré un emploi du temps chargé, a accepté de présider le jury de soutenance de ma thèse.

Je suis vivement reconnaissant au Professeur CHIKH Mohamed Amine qui malgré ses nombreuse fonctions s'est intéressé à ce travail en acceptant aussi d'en être examinateur.

Je souhaite exprimer toute ma reconnaissance au Docteur BOUCHIHA Djelloul pour le grand honneur qu'il me fait en acceptant d'être l'examinateur de cette thèse.

Ces années de doctorat ont été l'occasion de nombreuses collaborations et rencontres avec des chercheurs, des doctorants de tous horizons qui ont enrichi mon travail d'une dimension humaine essentielle. J'ai une pensée particulière pour l'ensemble des membres du laboratoire EEDIS pour l'appui scientifique, mais aussi personnel, dont ils ont fait preuve tout au long de ces années de thèse. Je tiens aussi à remercier les enseignants des universités Djilali Liabes, Moulay Tahar ainsi que l'université Stambouli.

Je dois un grand merci de tout coeur, à toute ma famille, sans qui je ne serai rien. Plus particulièrement : ma mère, mon père, mes soeurs et mes frères pour leur soutien moral, leurs conseils et surtout leur remarquable compréhension tout au long de ces années de thèse.

Enfin, je tiens à remercier toutes les personnes qui, de près ou de loin, directement ou indirectement, ont contribué à l'aboutissement de ce travail. J'espère qu'elles trouveront, dans ces quelques lignes, l'expression de ma profonde gratitude et ma reconnaissance légitime pour leur aide précieuse.

À mes parents ;

À mes soeurs et mes frères ;

À tous ceux qui me sont chers.

Table des matières

Table des figures

Liste des tableaux

Glossaire

A2A : Application-to-Application
ALPS : Applied Logic Programming-Languages and Systems
AOS : Architecture Orientée Service
API : Application Program Interfaces
B2B : Business-to-Business
BPMI : Business Process Management Initiative
BPEL : Business Process Execution Language
BPEL4WS : Business Process Execution Language for Web Services
CCM : CORBA Component Model
DOM : Document Object Model
CU : Conversation Unit
DE : DIANE Elements
DSD : DIANE Service Description
EAI : Enterprise Application Integration
EEDIS : Evolutionary Engineering and Distributed Information Systems laboratory
EJB : Enterprise JavaBeans
ICS : Invocable Composite Service
IISIE : Intégration et Interopérabilité des Systèmes d'Information d'Entreprise à base de service sémantique
GPR : Global Production Rule
KIF : Knowledge Interchange Format
O-QSC : QoS-Aware Service Composition
Orc : **Orc**hestration
OWL : Web Ontology Language
OWL-S : Web Services Ontology Language
PAS : Provided Atomic Service
PCS : Provided Composite Service
PSB : Provided Services Behavior

PNR : Programme National de Recherche
PSL : Process Specification Language
QoS : Quality of Service
RCS : Required Composite Service
RDF : Resource Definition Language
RDFS : RDF Schema
RDQL : Resource Definition Query Language
RSB : Required Services Behavior
SAX : Simple API for XML
SAWSDL Semantic Annotation for WSDL
SB : Service Behavior
SOA : Service Oriented Architecture
SOAP : Simple Object Access Protocol
SOC : Services Oriented Computing
SOG : Symbolic Observation Graphs
SPD : : Service Process Description
SPF : Service Process Formalism
UDDI : Universal Description, Discovery and Integration
UPML : Unified Problem-Solving Method Development Language
USDL : Universal Service-Semantics Description Language
VSM : Vector-Space Model
W3C : World Wide Web Consortium
Wf-nets : Workflow Nets
Ws-BeC : Web service-Behavioral Composition
WS-BPEL : Business Process Execution Language for Web Services
WS-CDL : Web Service Choreography Description Language
WSCI : Web Service Choreography Interface
WSCL : Web Services Conversation Language
WSDL : Web Services Description Language
WSFL : Web Services Flow Language
WSMF : Web Service Modeling Framework
WSML : Web service Modeling Language
WSMO : Web Service Modeling Ontology
WSMX : Web Service Modelling eXecution environment
XML : eXtensible Markup Language

Chapitre 1

Introduction Générale

Sommaire

1.1 Contexte

Depuis l'invention du World Wide Web par Tim Berners-Lee et Robert Cailliau[1] et le développement des technologies associées, le web et l'internet sont devenus bien plus qu'un simple instrument de partage d'information. La manière dont les systèmes d'information interagissent au travers des réseaux, et la façon dont les applications sont développées ont été complètement remises en cause. En effet, l'internet fournit un moyen universel aux organisations pour intégrer leurs applications (on parle alors de services), partager leurs ressources et savoir-faire, afin de minimiser leurs coûts, d'offrir de nouvelles applications à valeur ajoutée, sans pour autant perdre de leur

1. Au sein du CERN à Genève, en 1989

autonomie. Ainsi, l'évolution de ces systèmes d'information et le développement de processus métiers entre plusieurs entreprises ont fait du web le support idéal des interactions inter processus. Cependant, la mise en oeuvre de processus métiers interagissant sur le web reste une tâche complexe. Le concept de service web, basé sur les standards de l'internet, vise à faciliter le développement de ce type de processus. Cependant, chaque entreprise a ses propres règles de gestion et donc ses propres services. Ces derniers, qui vont devoir interagir avec des services provenant d'autres entreprises, doivent être traités comme des boîtes noires, où seulement les interfaces qu'ils fournissent sont connues (dans le chapitre 3 nous donnons les définitions relatives à ces notions).

Les services web ont été créés pour faciliter les interactions entre plusieurs partenaires dans le but de produire un service à valeur ajoutée. Mais, paradoxalement, le développement de services créés par chaque entreprise de manière autonome a donné lieu à une hétérogénéité qui pose divers problèmes au moment de l'exécution de la composition obtenue.

L'étude présentée dans ce mémoire nous a permis d'identifier les problèmes liés d'une part, à la composition de services web, et d'autre part à l'association de propriétés comportementales à cette composition. C'est en nous intéressant à ces deux problématiques que nous avons conçu une plate-forme de composition de services dont le principal objectif est de maximiser les chances pour une exécution réussite, tout en satisfaisant au mieux les besoins des clients de la composition.

Dans la perspective de réagir mieux et plus vite aux sollicitations des marchés, les entreprises doivent faire face à l'intégration et à l'automatisation de leurs différentes unités organisationnelles, ainsi qu'à la construction de partenariats. Il en découle une problématique d'intégration de systèmes d'information hétérogènes et répartis. Le problème a été posé dans un premier temps dans un contexte intra-entreprise. Des outils ont été proposés pour cela dans la classe des EAIs (Intégration

d'Applications d'Entreprise). Ces outils sont dédiés à un domaine d'application particulier, non extensible car basés sur des solutions propriétaires, et lourds à mettre en oeuvre. Les systèmes de gestion de flot de tâches (workflows) et les intergiciels (middleware) sont des technologies qui ont été développées, entre autre, afin de masquer l'hétérogénéité des systèmes à intégrer et de décrire de manière explicite la logique d'exécution d'un processus métier qui s'appuie sur plusieurs applications intra-organisation [Emerich., 2000]. Comme dans le contexte des EAIs, le processus métier est ici considéré selon une modalité d'interaction dite d'applications vers applications (Application-to-Application, A2A).

Cette modalité s'oppose à celle dite de métier à métier (Business-to-Business, B2B). L'intégration d'applications inter-organisation est alors abordée par le biais de la mise en place de "serveurs d'applications". Bien que les applications de type EAI et B2B partagent le même objectif, qui est de fournir des plates-formes pour l'intégration d'applications, les méthodes pour leur mise en place sont diffèrentes. Dans le cadre de l'intégration intra-entreprise (EAI), la gestion des applications à intégrer est centralisée, les schémas d'interactions entre ces applications sont statiques car toutes les applications appartiennent à la même organisation. En revanche, dans le cadre de l'intégration inter-entreprise (B2B), les applications à intégrer appartiennent à des organisations différentes, il s'en suit que la gestion ne peut plus être centralisée, et que les schémas d'interactions évoluent dans le temps.

Les systèmes d'information distribués ont évolué vers des architectures à base de services (Service Oriented Architecture - SOA [Yu et Zhang., 2004, Benatallah et al., 2005]). Selon ce modèle d'architecture, les processus métiers des entreprises sont encapsulés par des services qui interagissent les uns avec les autres par le biais d'échanges de messages. Ce schéma d'interactions faiblement couplées permet d'élargir le spectre des alliances possibles. Ces services sont appelés services web lorsqu'ils s'appuient sur les technologies du web pour interagir et communiquer les uns avec

les autres. Un service web est un logiciel, répondant à un ensemble de besoins fonctionnels, indépendant de la plate-forme d'exécution, auto-descriptif et qui peut être découvert et exécuté via l'internet.

Les efforts de standardisation effectués dans ce cadre ont permis de définir des protocoles standards qui permettent de masquer l'hétérogénéité des applications. Les technologies web et les standards qui en découlent permettent d'envisager l'émergence de solutions technologiques pour faciliter l'intégration d'applications accessibles par l'Internet [2]. Une définition détaillée et argumentée des notions mentionnées ici est donnée dans le chapitre 3. Ainsi, dans le domaine de l'intégration et de l'automatisation des interactions de processus métier inter-entreprise et intra-entreprise, les services web semblent constituer une solution prometteuse [Brambilla et al., 2002, Benatallah et al., 2002, Benatallah et al., 2003]. Mais la transition entre les solutions existantes et les nouvelles technologies est progressive de telle sorte que les plates-formes des intergiciels existants sont étendues avec des extensions appropriées pour développer, déployer et maintenir des services web. De plus, les standards proposés ne couvrent qu'une partie des aspects des services : la description, la coordination, la sécurité, ou la gestion des transactions, etc. En particulier, lorsqu'il s'agit de composer des services, les modèles visent à standardiser la description des services et à automatiser leur coordination. Cependant, ces modèles se limitent à la prise en compte des besoins fonctionnels laissant de côté les aspects extra-fonctionnels, en particulier ceux liés à la notion de qualité de services ou de propriétés compotementales. Le chapitre 3 discute de ces modèles de manière détaillée.

1.2 Problématique

Une application obtenue par composition de services web possède des caractéristiques spécifiques. D'une part, les interactions entre les services sont de durées va-

2. www.webservices.org/index.php/article/381/-1/1

riables et s'inscrivent le plus souvent dans un contexte inter-organisationnel. D'autre part, les services impliqués ont été conçus et implantés indépendamment les uns des autres, et non dans la perspective de participer à une composition. En conséquence, lorsqu'il s'agit de munir une composition de services de propriétées extra fonctionnelles et de propriétées comportementales, les principes habituellement utilisés dans les systèmes distribués ne sont plus adaptés.

Dans le domaine des services web, des outils ont été développés pour concevoir la composition des applications, leur implantation et leur exécution. Dans ce cadre, il existe de nombreux langages pour décrire des processus métiers. Dans le chapitre 3 nous montrons les caractéristiques et les limites de ces spécifications. En général, ces langages permettent de modéliser les processus à très bas niveau par le biais d'appels à des opérations offertes par les services web partenaires. Concernant l'association de propriétés comportementales à une composition, les approches proposées sont moins nombreuses et les solutions qui en sont issues sont partielles. Les langages et outils disponibles permettant modéliser des coordinations, qui s'appuient sur des services web, ne fournissent pas de concepts de haut niveau pour :

- exprimer les propriétés comportementales flexibles désirées au niveau du service composé ;
- assurer ces propriétés de façon automatisée en exploitant les propriétés comportementales des services composants.

Il s'avère en effet que certains services web, en particulier dans le domaine du commerce électronique, ont des propriétés comportementales inhérentes. Ceci est le cas notamment des services associés à la gestion de ressources (au sens large), comme par exemple la réservation de chambres d'hôtel, de places de spectacle, de services professionnels, etc. Des standards ont été proposés par exemple : WS-CDL (Web Service Choreography Description Language), WSCI (Web Service Choreography Interface), WSCL (Web Services Conversation Language) et WS-BPEL (Business

Process Execution Language for Web Services). Cependant leur portée est limitée et les compositions obtenues sont peu flexibles. Dans le chapitre 3 nous détaillons ces approches ainsi que leurs apports et leurs limites.

Il est courant que de nombreux services répondent à un même ensemble de besoins fonctionnels. Ces services se distinguent les uns des autres par leurs propriétés extra-fonctionnelles. La plupart des propositions existantes pour la composition de services web ne tirent pas partie de cette propriété : au moment de la conception, les services web participant à la composition sont fixés. Si, au moment de l'exécution, un des composants n'est pas disponible le processus doit s'arrêter et attendre l'intervention du concepteur qui peut, soit relancer l'exécution de la même composition, soit remplacer le service défaillant dans la composition avant de la relancer.

En résumé, les questions soulevées par la discussion ci-dessus sont :

- Comment spécifier parmi un ensemble de services potentiels, ceux qui entrent dans la composition de manière à maximiser les chances pour que la composition s'exécute d'une manière approprié et beaucoup plus flexible ?
- Comment exprimer les contraintes d'interactions des services compasants ?
- Comment exprimer les préférences d'interactions des activités compasantes ?
- Comment exprimer la coordination flexible de services en garantant les propriétées comportementales spécifiée ?

1.3 Scénario illustratif

Nous choisissons d'illustrer la problématique introduite ci-dessus par un scénario dans lequel une application est conçue pour l'organisation de voyages (voir *Figure* 1.1). L'application contrôlant le flux d'information entre les modules suivants : la réservation de vol, la réservation de bateau, la réservation d'hôtel, la location de voiture, la réservation de moto et la consultation de Météo.

Habituellement, une organisation d'un voyage nécessite l'opération d'achat d'un

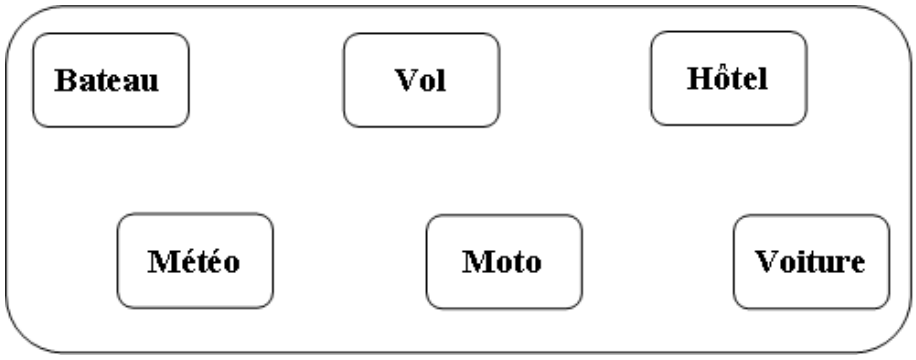


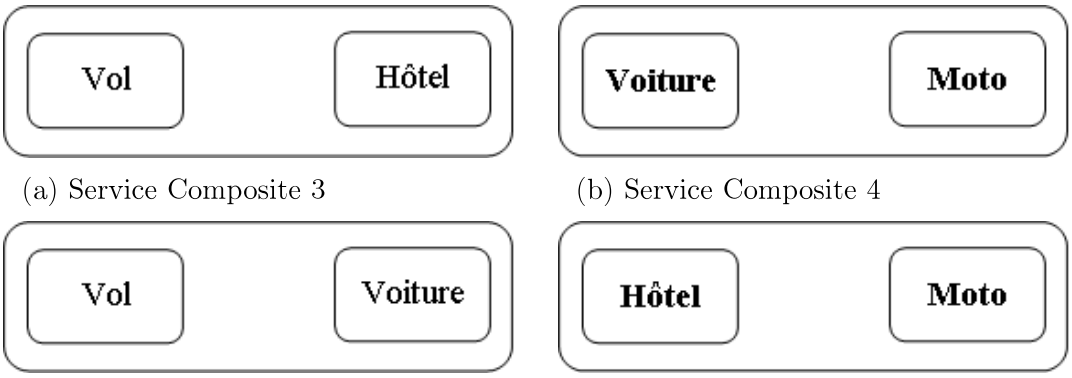
FIGURE 1.1 – Panoplie de services impliqués par l'organisation d'un voyage

billet d'avion ou de bateau, la réservation d'une chambre d'hôtel, le cas échéant la location d'une voiture ou d'une moto et la consultation de la météo pour le lieu de la destination. En se basant sur le principe de paradigme de traitement orienté services, ces opérations sont implémentées via une panoplie de services web. Par ailleurs :

- Un seul service individuel peut eu être considéré comme une repense à un besoin clientèle donné.
- Des services peuvent eux être assemblés pour fournir des services composites à valeur ajoutée pour rependre à un besoin clientèle qu'un simple service parmi eux ne peut le résoudre (voir Figure 1.2).
- Des services composites, par la récursivité de la composition de service, peuvent eux aussi être combinés de façon à donner de nouveaux services composites pour rependre de plus en plus aux besoins beaucoup plus complexes. De même, nous pouvons envisager de composer encore des services que se soit simples ou composites pour rependre aux besoins des utilisateurs et également d'augmenter leur chance de réalisation (voir Figure 1.3).

Par consequent :

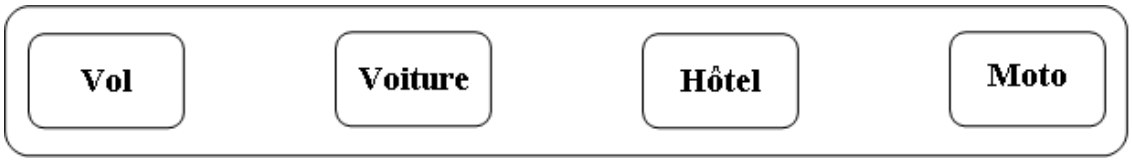
- Les services doivent être consommés selon des contraintes de leur fournisseurs. Par exemple, l'agence de voyage exige que pour utiliser son service d'hôtellerie, il faut aussi utiliser son service de vol. Ainsi, l'agence peut ne pas accepte



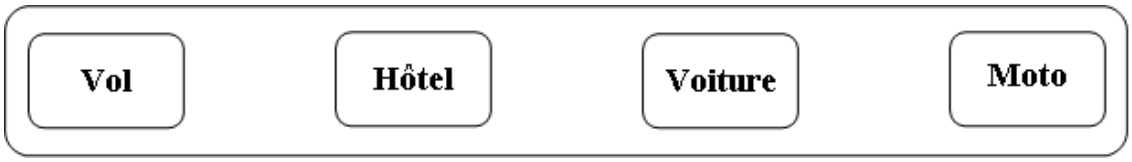
(a) Service Composite 3 (b) Service Composite 4

(c) Service Composite 1 (d) Service Composite 2

FIGURE 1.2 – Composition par assemblage des services individuels



(a) Service Composite 5



(b) Service Composite 6

FIGURE 1.3 – Récursivité de la composition de services

d'utiliser son service météo indépendamment.

- Les services doivent être utilisés selon les préférences clientèles. Par exemple, un client de l'agence de voyage préfère utiliser le service de vol le moins chère et parfois avec des options de remboursement dans le cas d'annulation de son voyage. Aussi, il peut exiger des services de même fournisseur (par exemple service d'hôtellerie et service de véhicule).
- Les services doivent être combinés d'une manière flexible et non figée, basé non seulement sur la signature de services, mais aussi sur leur comportement pour augmenter la chance de réalisation des besoins du client. Par exemple, pour une organisation d'un voyage, la configuration : vol et hôtel avec voiture

et moto peut être réalisée soit :

1. Par l'intégration des services individuels (par exemple des services comme ceux représentés par la figure 1.1),
2. Par la sélection du service composite (par exemple un seul service comme celui représenté par la figure 1.3b),
3. Par l'intégration des services composites (par exemple des services comme ceux représentés par les figures 1.2a et 1.2b),
4. Par le chevauchement des services composites (par exemple des services comme ceux représentés par les 1.2c et 1.2d),
5. Ou bien par une combinaison hybride par le biais de sélection, d'intégration et/ou de chevauchement des services disponibles.

En conclusion, pour garantir la consommation appropriée de services et augmenter les chances de réussite de leur composition, nous prônons :

- l'assouplissement des préférences utilisateurs par la restriction de sa portée au minimum requis,
- le paramétrage de la composition pour guider la sélection des services web potentiels,
- la composition appropriée de services web à la fois au niveau fournisseur et au niveau utilisateur par le biais des propriétés comportementales.

1.4 Objectifs

Ce travail de recherche a pour principal objectif de définir un modèle de composition personnalisée de services web en favorisant la flexibilité de la composition. La notion de personnalisation nous a mené d'adopter les propriétés non fonctionnelles permettant la sélection parmi un ensemble de services potentiels ceux jugés les plus pertinents. La recherche de flexibilité nous a poussé à prendre en compte

deux aspects fondamentaux qui sont : (i) l'orientation comportement et (ii) la prise en compte de la sémantique. L'orientation comportement de services permet d'offrir le meilleur élément qui détermine la manière d'interaction avec le service en adoptant la composition, récursive, de services. Le modèle permet la prise en compte non seulement des propriétés comportementales de la composition requise, des besoins d'utilisateur représentés par un service composite abstrait, mais aussi des services composants impliqués par cette composition. Tirant partie de l'existence d'un grand nombre de services répondant aux mêmes besoins fonctionnels et afin d'augmenter les chances pour les compositions munies de propriétés comportementales de s'exécuter d'une manière approprié et beaucoup plus flexible, l'étude s'appuie sur les principes de sélection, d'intégration et de chevauchement de services à l'exécution. La sémantique permet d'offrir le meilleur moyen pour rendre les échanges interservices plus flexibles et plus intelligibles à travers l'utilisation des technologies de web sémantique (RDF, OWL, OWL-S, etc.).

1.5 Contributions

Les principales contributions de ce travail, rassemblées dans le framework $Ws-BeC$ (*Web service-Behavioral composition*), sont :

- La proposition d'un modèle dans lequel la composition s'exprime en termes de opérations (fonctionnalités) de services et non pas en termes de services web individuels, afin de permettre la perception de services composites comme étant des services composites et non pas comme étant des services atomiques. La fonctionnalité de services est associée à un ensemble de besoins fonctionnels souhaités (par exemple la réservation d'une chambre d'hôtel).
- La proposition d'un modèle de composition avec des propriétés comportementales afin de permettre la prise en compte des équivalences comportementes pour favoriser la flexibilité d'assemblage de services,

- La proposition d'un modèle dans lequel il est possible de paramétrer la composition par des contraintes et des préférences spécifiées respectivement par les fournisseurs/utilisateurs de services afin de guider la sélection, l'intégration et le chevauchement des services au moment de la composition,
- La réalisation d'un prototype pour l'évaluation et la validation expérimentale des modèles proposés.

Le framework proposé est caractérisé par :

- La flexibilité : il est possible de déterminer pour une composition donnée, au moment de la conception, quelles portions de services sont obligatoires et quelles sont optionnelles. Le framework considère que toutes les contraintes des fournisseurs sont traitées comme obligatoires, alors que les préférences des utilisateurs sont traitées comme optionnelles sauf celles qui sont spécifiées explicitement comme obligatoires par leurs utilisateurs. Cette caractéristique permet d'obtenir des compositions différentes selon la disponibilité des composants au moment d'exécution,
- L'adaptabilité : les services web sont choisis au moment de l'exécution. Cette sélection, s'appuyant sur un mécanisme de combinaison flexible, permet de choisir les services les mieux adaptés au contexte de l'exécution en termes de propriétés non fonctionnelles, en particulier les caractéristiques comportementales.

1.6 Plan de la thèse

Le reste de ce mémoire est structuré de la manière suivante :

Le chapitre 2 présente les notions de base sur l'architecture orientée service, la composition de services, et l'architecture étendue pour les services web. Il présente ainsi des études comparatives de différents modèles, langages et formalismes pour la composition et la représentation de services web.

Le chapitre 3 présente les travaux, les plus connues, d'appariement et de composition dynamique de services web.

Le chapitre 4 est consacré à la description détaillée de notre approche. Il s'agit d'introduire le cadre méthodologique associé et de présenter nos principales contribution. L'accent est mis sur les principales étapes de notre approche. D'abord une première étape de modélisation et de description comportementale réalisée durant la phase de conception. Une deuxième étape de stockage de ces descriptions dans un annuaire de description comportementale. Une troisième étape d'appariement des descriptions stockées est réalisée durant la phase d'exécution pour la sélection, l'intégration ainsi que le chevauchement des processus métiers.

Dans le chapitre 5, nous présentons le cadre applicatif de nos travaux. Nous illustrons la mise en oeuvre des différents éléments de l'architecture, et présentons le prototype que nous avons développé comme support à notre approche. En fin nous discutons les expérimentations mises en place pour évaluer nos propositions et montrer la faisabilité de notre approche.

Le dernier chapitre 6 dresse le bilan de notre travail et donne les perspectives qui restent ouvertes. Il effectue aussi la liaison avec d'autres travaux de recherche en cours.

Chapitre 2

Composition de Services web : Vision Générale

Sommaire

2.1 Introduction

Le paradigme de service web depuis sa naissance, en 2000, a été considérer comme une révolution pour le web. En effet, avec les trois technologies de base : SOAP (Simple Object Access Protocol) [1], WSDL (Web Services Description Language) [2] et UDDI (Universal Description Directory and Integration) [Kourtesis et Paraskakis., 2008], les services web proposent une architecture par composants qui permet à une application de faire l'usage d'une fonctionnalité située dans une autre application. Les services web ont connu un grand succès grâce au haut degré d'interopérabilité fourni par les standards. Les services web sont présentés comme le meilleur moyen pour concrétiser la SOA (Service-Oriented Architecture) et pour mettre en place le paradigme SOC (Services Oriented Computing).

Cependant, les services web possèdent aussi certaines lacunes. En effet, il se peut qu'un client ait des besoins spécifiques qui ne sont rendus par aucun service web. A cause de l'élargissement d'internet, il est rare pour un consommateur de découvrir un service web qui réponde à tous ses besoins. Pour cette raison, plusieurs services peuvent interagir et échanger des informations. L'objectif de cette interaction est l'accomplissement d'un but complexe, non concevable par un seul service web. Cette agrégation des compétences pour réaliser un but commun est réalisée par le biais d'un processus de composition de services web.

Dans ce chapitre nous allons présenter les différents travaux de recherches effectués dans le domaine de services web. Pour ce faire, nous débuterons, par une

1. www.w3.org/TR/soap/
2. www.w3.org/TR/wsdl

présentation des différentes terminologies et technologies de services web et de composition de services. Nous allons voir également, dans la section 2.2 la notion et les objectifs de service ainsi que l'architecture de référence pour les services web, dans la section 2.3 la terminologie de composition de services suivie d'une présentation de l'architecture étendue de services web. Nous étudions les différents travaux liés à notre contribution de composition de services web. La section 2.4 présentes les modèles de composition de services. La section 2.5 traitera quant à elle des langages de composition de services. Enfin, La section 2.6 détaille les formalismes de représentation de services.

2.2 Notion de service

L'une des tendances historiques qui ont conduit à l'apparition des services web est l'utilisation de l'architecture par composants. Ce type d'architecture a engendré un développement rapide et évolutif d'applications distribuées et complexes. Au cours de ce développement, nous avons assisté à la mise en place de trois technologies [Emmerich et Kaveh., 2001] : CCM (CORBA Component Model), EJB (Enterprise JavaBeans) et .Net . La mise en ouvre de ces trois technologies a soulevé de nombreuses difficultés [Stal., 2002] :

- L'interdépendance des composants. Pour préserver son évolutivité, une application distribuée doit minimiser l'interdépendance entre ses composants. Cependant, cette action peut entraîner un dysfonctionnement où il est souvent difficile de déterminer précisément l'origine.
- Hétérogénéité des formats des composants. Pour préserver sa qualité de service, une application distribuée doit garantir une interopérabilité entre ses divers composants et une maîtrise de leur hétérogénéité. Cependant, la distribution de composant est un enjeu industriel où chaque technologie veut imposer son format, bien évidemment, incompatible avec les autres formats.

Ces difficultés ont contribué à l'apparition d'une nouvelle architecture de distribution où les composants sont indépendants, autonomes et décrits sous un seul format standardisé. Cette architecture est l'architecture orientée services [Stal., 2002].

2.2.1 Définitions

Les services web constituent une technologie pour mettre en ouvre le paradigme *SOA*. Les services web ont été proposés initialement par IBM et Microsoft, puis en partie standardisés par le consortium du W3C (World Wide Web Consortium).

Selon W3C[3] Un service web est un système conçu pour permettre l'interopérabilité des applications à travers un réseau. Il est caractérisé par un format de description interprétable/compréhensible automatiquement par la machine (en particulier WSDL). D'autres systèmes peuvent interagir avec le service web selon la manière prescrite dans sa description et en utilisant des messages SOAP, généralement transmis via le protocole HTTP et sérialisées en XML et en d'autres standards du web. Selon M. P. Papazoglou [Papazoglou., 2003] : Les services web sont des éléments auto-descriptifs et indépendants des plateformes qui permettent la composition à faible coût d'applications distribuées. Les services web effectuent des fonctions allant de simples requêtes à des processus métiers complexes. Les services web permettent aux organisations d'exposer leurs programmes résultats sur Internet (ou sur un intranet) en utilisant des langages (basés sur XML) et des protocoles standardisés et de les mettre en ouvre via une interface auto-descriptive basée sur des formats standardisés et ouverts. Techniquement, un service web est donc une entité logicielle offrant une ou plusieurs fonctionnalités allant des plus simples aux plus complexes. Ces entités sont publiées, découvertes et invoquées à travers le web grâce à l'utilisation d'Internet comme infrastructure de communication ainsi que de

3. www.w3.org/TR/2004/NOTE-ws-gloss-20040211/

formats de données standardisés comme XML.

2.2.2 Objectifs

Les services web visent essentiellement quatre objectifs fondamentaux expliquant son grand succès [Huhns et Singhl., 2005] :

- L'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 avec 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.
- Le couplage faible : Le couplage est une métrique indiquant le niveau d'interaction entre deux ou plusieurs composants logiciels. Deux composants sont dits couplés s'ils échangent de l'information. Nous parlons de couplage fort si les composants échangent beaucoup d'information et de couplage faible dans le cas contraire. 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.
- La réutilisation : 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 besoins, il est généralement possible de réutiliser des services déjà existants pour satisfaire une partie ses exigences. Ceci facilite la maintenance de l'application et permet un gain de temps considérable.

– La découverte et la composition automatiques : La découverte et la composition sont des étapes importantes qui permettent la réutilisation des services. En effet, il faudra être en mesure de trouver et de composer un service afin de pouvoir en faire usage. En exploitant les technologies offertes par Internet et en utilisant un ensemble de standards pour la publication, la recherche et la composition, l'approche services web tend à diminuer autant que possible l'intervention humaine en vue de permettre une découverte et une composition automatique des services les plus complexes. En effet, pour réaliser une application, un développeur peut simplement interroger un moteur de recherche de services afin de trouver le service adéquat et à l'aide de langages de coordination appropriés il peut l'intégrer avec le reste des services de l'application en question.

2.2.3 Architecture de référence

L'objectif de la mise en place d'une architecture pour la technologie service web est de proposer un schéma directif et une vision sur le fonctionnement et la dynamique de cette nouvelle technologie. Pour cela, une architecture de référence a été mise en place par le W3C. Cette architecture a été proposée afin de promouvoir l'interopérabilité et l'extensibilité des services web et de préserver ces deux objectifs lors des évolutions technologiques successives. Cette architecture est connue sous le nom SOA (Service-Oriented Architecture)[4]. Elle propose une perspective globale pour le développement, la gestion et le fonctionnement des services web et elle s'articule

4. www.w3.org/TR/ws-arch/

autour des trois rôles suivants :

- Le fournisseur : correspond au propriétaire du service. Il fournit une plateforme d'accueil du service. Il a pour rôle de créer un service, de l'héberger et de le publier pour le mettre à la disposition des clients ou des partenaires.
- Le client : correspond au demandeur du service. Il est constitué par l'application qui va rechercher et invoquer un service. L'application cliente peut être elle même un service web.
- L'annuaire : correspond à un registre de descriptions de services. Il offre des facilités de publication de services à l'intention des fournisseurs ainsi que des facilités de recherche de services à l'intention des clients.

Les interactions de base entre ces trois rôles incluent les opérations de transport, de description et de découverte qui sont assurées respectivement par les standards SOAP, WSDL et UDDI.

Nous introduisons, par la Figure 2.1, un scénario type d'utilisation de cette architecture.

Ce scénario se déroule en plusieurs étapes qui sont les suivantes :

1. Le fournisseur définit la description de son service dans un document WSDL et la publie dans l'annuaire UDDI.
2. Le client, désirant trouver un service, interroge l'annuaire UDDI via un message SOAP.
3. L'annuaire retourne, via un message SOAP, une liste de services qui répondent à la requête du client. Le client n'a qu'à choisir un parmi la liste.
4. Le client récupère le document WSDL du service choisi. Ensuite, il examine ce document afin de récupérer les informations nécessaires lui permettant de se connecter au fournisseur et d'interagir avec le service considéré. Enfin, il invoque l'opération désirée par le biais d'une requête SOAP renfermant les paramètres d'entrée de l'opération.

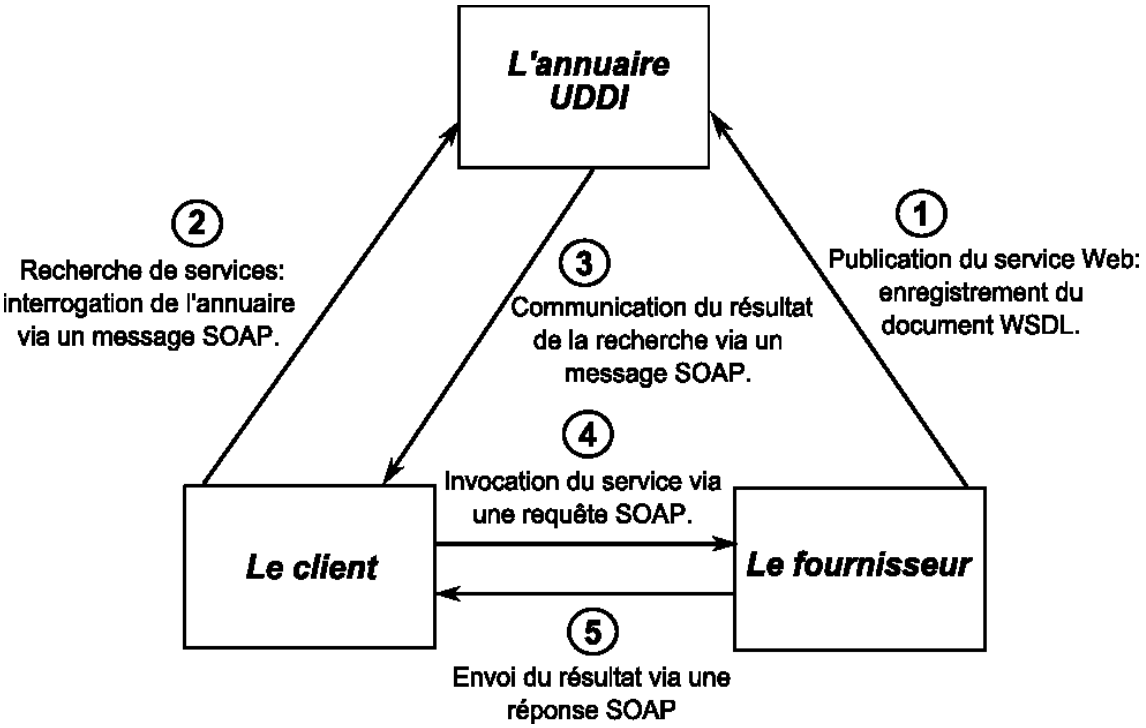


FIGURE 2.1 – Architecture de référence des services web : SOA

5. Le service, du côté du fournisseur, reçoit la requête, la traite, formule la réponse SOAP et l'envoie au client.

2.3 Notion de composition de services

2.3.1 Définition

Selon Gardarin [Dustdar et Schreiner., 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".

Selon S. Dustdar et W. Schreiner [Dustdar et Schreiner., 2005] : "L'infrastructure de base des services web suffit pour la mise en ouvre d'interactions simples entre un client et un service web". Si la mise en ouvre d'une application métier implique

l'invocation d'autres services web, il est nécessaire donc de combiner les fonctionnalités de plusieurs services web. Dans ce cas, nous parlons d'une composition de services web.

La composition ou l'agrégation de services web est une opération qui consiste à construire de nouvelles applications ou services appelés services composites ou agrégats par assemblage de services déjà existants nommés services basiques ou élémentaires (atomiques).

La composition spécifie quels services doivent être invoqués, dans quel ordre et sous quelles pré-conditions. Les services basiques peuvent être soient des services atomiques soient des services composites.

La composition de services web vise essentiellement quatre objectifs :

- Créer de nouvelles fonctionnalités en combinant des services déjà existants.
- Résoudre des problèmes complexes auxquels aucune solution n'a été trouvée.
- Faire collaborer plusieurs entreprises.
- Optimiser et améliorer une fonctionnalité existante.

2.3.2 Architecture étendue de services web

La composition de services a donné naissance à une extension de SOA (Service-Oriented Architecture) : l'architecture étendue [ESOA., 2005, Papazoglou et Van., 2007]. Cette architecture est constituée de plusieurs couches se superposant les unes sur les autres. De ce fait, elle est également appelée pile des services web. Cette pile ne trouve pas encore de consensus quand à sa définition standardisée. La Figure 2.2 est une proposition d'une telle pile [Papazoglou et Van., 2007].

Les couches de cette architecture sont les suivantes :

- Couche d'infrastructure de base : renferme les fondements théoriques et technologiques établis par l'architecture de référence.
- Couche de processus métier : s'appuie sur la couche d'infrastructure de base

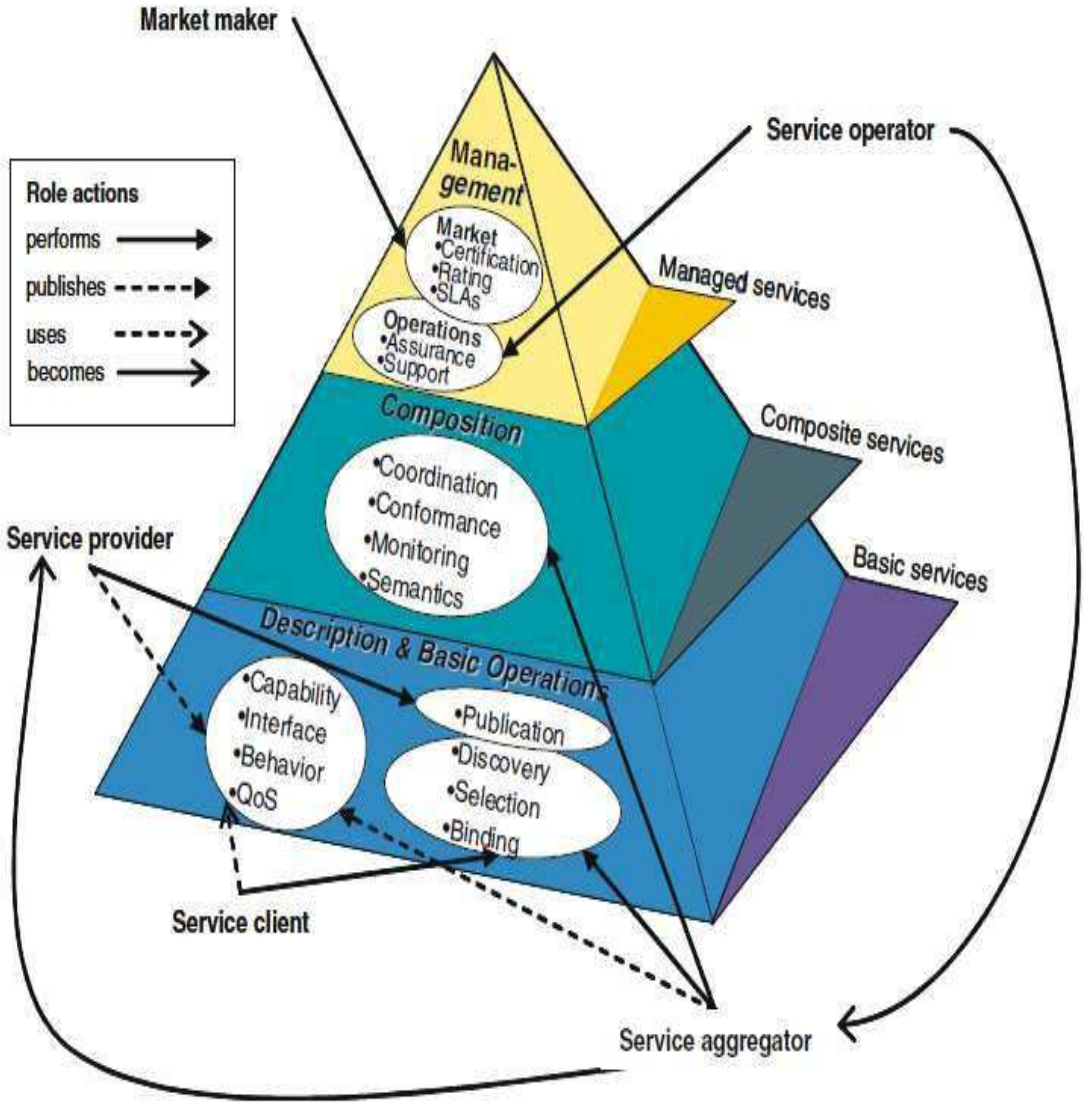


FIGURE 2.2 – Architecture étendue des services web

et elle se préoccupe des aspects relatifs à la composition de services à savoir : la coordination, la supervision, la sémantique, etc.

– Couche de gestion : est la couche supérieure et elle s'intéresse à la configuration, au déploiement et à la maintenance des services web, etc.

L'architecture étendue attache à chacune des ces couches des aspects spécifiques à l'ingénierie des services ; Celle-ci est définie comme étant l'activité qui consiste à analyser, modéliser et développer des techniques et des méthodologies nécessaires

pour les approches à services basiques et/ou composites.

Outre les nouvelles couches de composition et de gestion, l'architecture étendue prend également en compte les aspects importants de propriétés comportementales et non-fonctionnelles d'un service qui ne sont pas explicitement abordée par l'architecture de référence. Pour cette fin, la couche d'infrastructure de base raffine les exigences décrites au niveau de l'architecture de référence pour y inclure également une description du comportement du service et de ses propriétés non-fonctionnelles.

2.4 Les modèles de compositions de services

Le paradigme SOC réfère à l'ensemble de concepts, de principes et de méthodes qui représentent un traitement en SOA dans lesquelles des applications logicielles sont construites à base des services composants indépendants avec des interfaces standard. L'idée principale de SOC / SOA est de séparé explicitement l'ingénierie logicielle de la programmation, à mettre l'accent sur l'ingénierie logicielle et à désaccentuer sur la programmation. SOC sépare le développement de logiciels en trois parties indépendantes : les concepteurs d'application (par des ingénieurs de logiciels), fournisseurs de services (par des programmeurs) et les annuaires, intermédiaires, de services (par des organisations industrielles et économiques, gouvernementales et non gouvernementales, de standardisation).

1. Les fournisseurs de services : Ils utilisent un langage de programmation traditionnel tels que Java, $C++$, etc. pour écrire des composants logiciels (programme). Tous les composants seront enveloppés avec des interfaces standard ouvert, appelées services, ou des services web si les services sont disponibles sur Internet, de tel sorte que les concepteurs d'applications peuvent simplement utiliser les services sans aucune autre communication avec les fournisseurs de services. un mêmes services peuvent être utilisés par de nombreuses applications.

2. Les annuaires de services : Permettre aux services d'être enregistrés et publiés pour un accès public. Aider les concepteurs d'applications pour trouver les services dont ils ont besoin.

3. Les concepteurs d'application : Au lieu de construire des logiciels à partir de zéro en utilisant un langage de programmation de base, les concepteurs d'applications représentent les derniers utilisateurs qui spécifient l'application logique dans une spécification d'un langage de haut niveau, en utilisant des services standard comme des composants. Par conséquent, les concepteurs d'applications sont des ingénieurs de logiciels qui ont une bonne compréhension de l'architecture de logicielle et du domaine d'application.

D'une part, comme le nombre et les types de services disponibles augmentent, la nécessité d'écriture de nouveaux services et la nécessité pour des programmeurs vont diminuer. D'autre part, comme les applications informatiques se déplacent dans plus en plus de domaines, la nécessité pour des concepteurs d'applications va augmenter. En prenant comme exemple, les fournisseurs de services sont des propriétaires des hôtels et des compagnies aériennes, tandis que les concepteurs d'applications sont les vendeurs des plans de tourisme (architectes) qui utilisent des hôtels et des compagnies aériennes pour construire des millions de différentes plans de bases. Nous n'avons pas besoin de beaucoup de gens qui peuvent concevoir différents types d'hôtels et des compagnies aériennes, mais nous avons besoin de beaucoup d'architectes pour construire des plans différents.

Le SOC utilise les services comme des constructeurs pour supporter le développement rapide, de faible coût et de composition aisée des applications distribuées. Les services sont autonomes, des entités de traitement indépendantes de plateforme qui peuvent être utilisés de manière autarcique. Les services peuvent être décrites, publiées, découverts, et dynamiquement assemblés pour développer massivement des systèmes distribués, interopérables et évolutifs. Les services accomplissent des

fonctions qui peuvent se varier d'une simple réponse à l'exécution des processus métiers sophistiquées nécessitant des relations pair-à-pair entre éventuellement plusieurs couches des fournisseurs et clients de services. Toute partie de code et tout élément d'une application déployée sur un système peuvent être réutilisés et transformés en un ensemble de services disponibles. Les services expriment une approche de programmation «orientée services», basée sur l'idée de composer des applications par la découverte et l'invocation d'un ensemble de services disponibles au lieu de construire de nouvelles applications ou bien par invocation des applications disponibles pour accomplir certaines tâches [Papazoglou et Georgakopoulos., 2003]. Les services sont le plus souvent construits d'une façon qui est indépendante du contexte dans lequel ils sont utilisés. Ce qui signifie que le fournisseur et les clients de services sont faiblement couplés.

En résumé, certains des aspects clés de l'orientation service sont :

1. Couplage faible (loose coupling) : les services maintient une relation qui minimise les dépendances entre services, ils conservent seulement une prise de sensibilisation de l'autre.
2. Contrat de service (service contract) : le service adhérer à un accord de communication, tel que défini collectivement par une ou plusieurs descriptions de service et les documents connexes.
3. Autonomie (Autonomy) : les services ont un contrôle sur la logique qu'ils encapsulent.
4. Abstraction (Abstraction) : sauf de ce qui est décrit dans le contrat de service, les services cachent la logique au monde extérieur.
5. Réutilisation (Reusability) : la logique est divisée en services avec l'intention de promouvoir la réutilisation.
6. Composabilité (Composability) : des collections de services peuvent être coordonnés et assemblés pour former des services composites.

7. Apatridie (Statelessness) : les services minimisent le retenir des informations spécifiques à une activité.

8. Possibilité de découverte (Discoverability) : les services sont conçus d'être descriptive à l'extérieur de telle sorte qu'ils peuvent être trouvés et évalués via des mécanismes de découverte disponibles.

La composition de services regroupe les rôles et les fonctionnalités nécessaires pour l'agrégation de plusieurs services en une seule composition. Les services composites résultantes peuvent être utilisées par un processus comme des services de base dans d'autres compositions de services ou peuvent être offerts comme des applications / solutions complets aux clients.

Des agrégateurs de services accomplissent cette tâche. Les agrégateurs de services deviennent ainsi des fournisseurs de services en publiant les descriptions de service des services composites qu'ils construisent. Ces agrégateurs de services élaborer des spécifications et/ou de code qui permettent un service composite d'exercer des fonctions qui sont basés sur des caractéristiques telles que des descriptions de métadonnées, terminologies standards et des modèles de référence et un service de conformité. Ils effectuent la coordination des services pour contrôler l'exécution des services composites (par exemple, processus), les transactions de services et de gérer à la fois le flux de données ainsi que le flux de contrôle entre les services composites. Ils appliquent aussi des politiques sur les invocations de service agrégé.

Actuellement, il existe des initiatives concurrentes de développement des spécifications pour la définition de processus métier, qui visent à définir et gérer les activités de processus métier et des protocoles d'interaction de métiers comprenant des services de collaboration. Les termes conversation, orchestration et chorégraphie ont été largement utilisés pour décrire des protocoles d'interaction métiers comprenant des services de collaboration [Barros et al., 2006].

2.4.1 Chorégraphie

Un modèle de chorégraphie décrit une collaboration entre un ensemble de services pour atteindre un objectif commun. Il capture les interactions dans lesquelles les services participants s'engagent pour achever cet objectif et les dépendances entre ces interactions, y compris : les dépendances et les flux de contrôle causales (i.e. qu'une interaction donnée doit se produire avant qu'une autre, ou bien qu'une interaction provoque une autre), les dépendances d'exclusion (qu'une interaction donnée exclut ou remplace une autre), les dépendances de flux de données, corrélation d' interaction, les contraintes de temps, les dépendances transactionnelles, etc.

Une chorégraphie ne décrit pas une action interne d'un service participant qui ne résultent pas directement par un effet visible de l'extérieur, comme un traitement interne ou une transformation de données. Une chorégraphie capture les interactions dans une perspective globale ce qui signifie que tous les services participants sont traités de manière équitable. En d'autres termes, une chorégraphie englobe toutes les interactions entre les services participants qui sont pertinents par rapport à l'objectif des chorégraphies. La chorégraphie de service web est décrite plus formellement par le Groupe de travail, de Chorégraphie de services web, W3C comme le comportement observable externe entre plusieurs clients (qui sont généralement des services web, mais pas exclusivement) dans lequel le comportement observable externe est défini comme la présence ou l'absence de messages qui sont échangés entre un service web et ses clients [Kavantzas et Burdett., 2004].

Une chorégraphie de services d'un scénario d'interactions bien connu est montrée sous la forme d'un diagramme d'activités UML dans la Figure 2.3. Trois services sont impliqués dans cette chorégraphie : service client, service fournisseur et service magasin. Les actions élémentaires dans le diagramme représentent les activités métiers qui résultent dans les messages d'être envoyés ou reçus. Par exemple, l'action

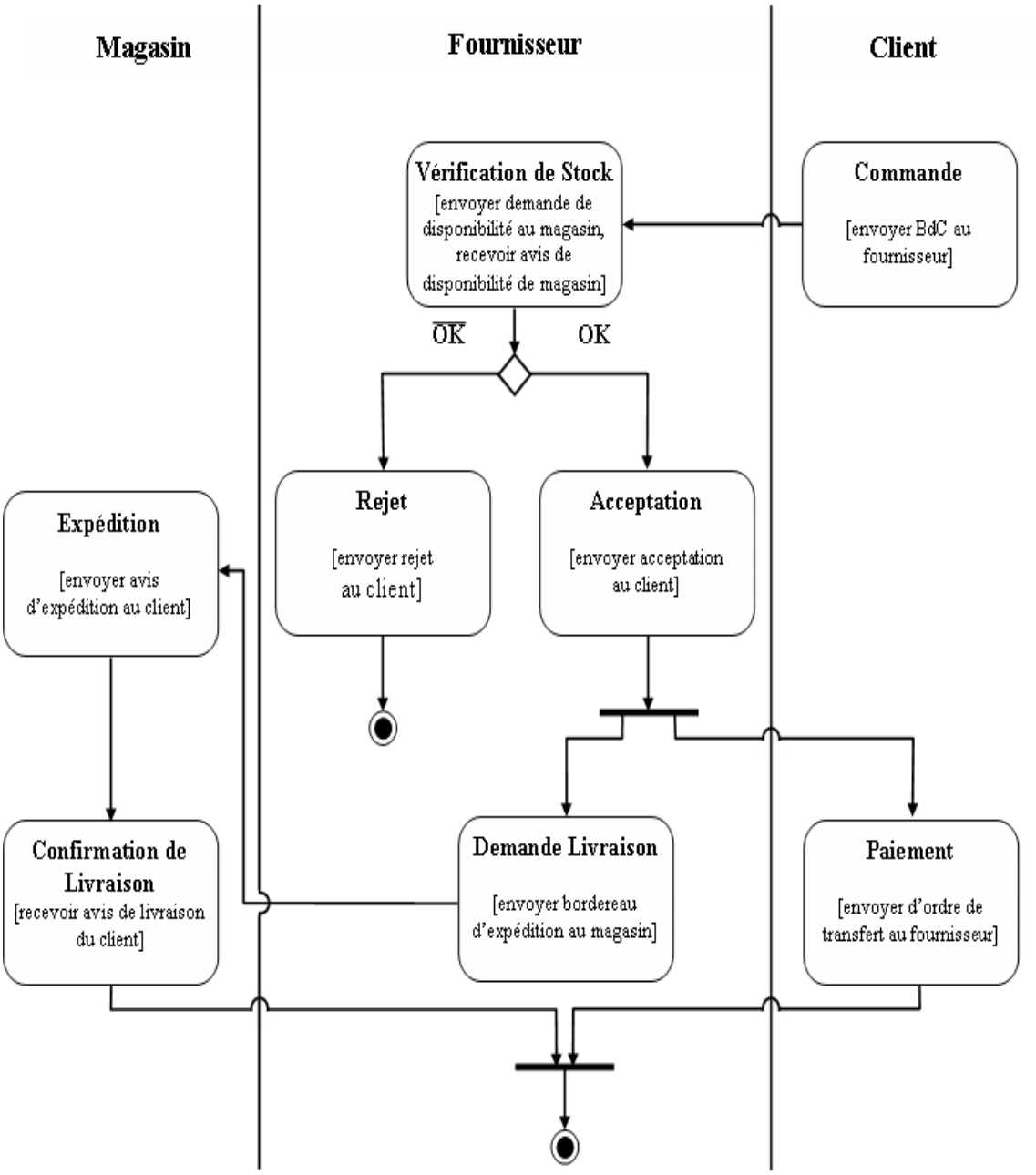


FIGURE 2.3 – Exemple de Chorégraphie

«Commande de marchandises» menées par les résultats des clients dans un message envoyé au fournisseur (ce qui est décrit comme une note textuelle ci-dessous du nom de l'action). Bien sûr, chaque action d'envoi de message a une action de réception de message correspondante mais pour éviter d'encombrer le diagramme, seulement l'action d'envoi ou de réception (pas les deux) est indiquée pour chaque échange de

messages. Par exemple, l'action «*envoyer BdC au fournisseur*» de l'activité «*Commande*» implique qu'il ya une action correspondante «*recevoir BdC du Client*» sur le côté de fournisseur, mais cette dernière action n'est pas indiqué sur le diagramme.

Notez que la Figure 2.3 ne comprend pas les activités et les chemins alternatifs nécessaires pour traiter les erreurs et les exceptions qu'on pourrait raisonnablement attendu pour le scénario en question. On incluant cette information serait augmente considérablement la complexité du modèle.

Des implémentations des standards de chorégraphie sont actuellement sous la forme de WS-CDL (Web Service Choreography Description Language) [Kavantzas et al., 2004] et WSCI (Web Service Choreography Interface) [Arkin et al., 2002]. Ces spécifications ont été introduites comme une partie d'un modèle orientée services aligné avec les mêmes groupes de travail du W3C.

2.4.2 Conversation

Une conversation capture les aspects comportementaux des interactions dans lesquelles un service particulier peut s'engager pour achever un objectif. Elle complète les descriptions structurelles de l'interface telles que celles supportées par le WSDL, qui capture les interactions élémentaires dans lesquelles un service peut s'engager et les types de messages ainsi que les stratégies sous lesquelles ces messages sont échangés.

Une conversation capture des dépendances entre des interactions élémentaires telles que les dépendances de flux de contrôle (par exemple, qu'une interaction donnée doit précéder une autre), les dépendances de flux de données, les contraintes de temps, corrélations de message, les dépendances transactionnelles, etc. Elle se concentre sur la perspective d'une seule partie. En conséquence, une conversation ne capture pas des interactions completes, puisque les interactions impliquent nécessairement deux parties. Plutôt, une conversation capture des interactions de la

perspective de l'un des participants, et donc peut être considéré comme constitué des actions de communication effectuées par ce participant. En outre, les conversations ne décrivent pas des tâches internes telles que les transformations de données internes.

Les Figures 2.4, 2.5 et 2.6 montrent des exemples de conversations correspondant aux rôles : fournisseur, magasin et client dans la chorégraphie de la Figure 2.3.

Notez qu'un rôle défini dans une chorégraphie peut être associé à plusieurs com-

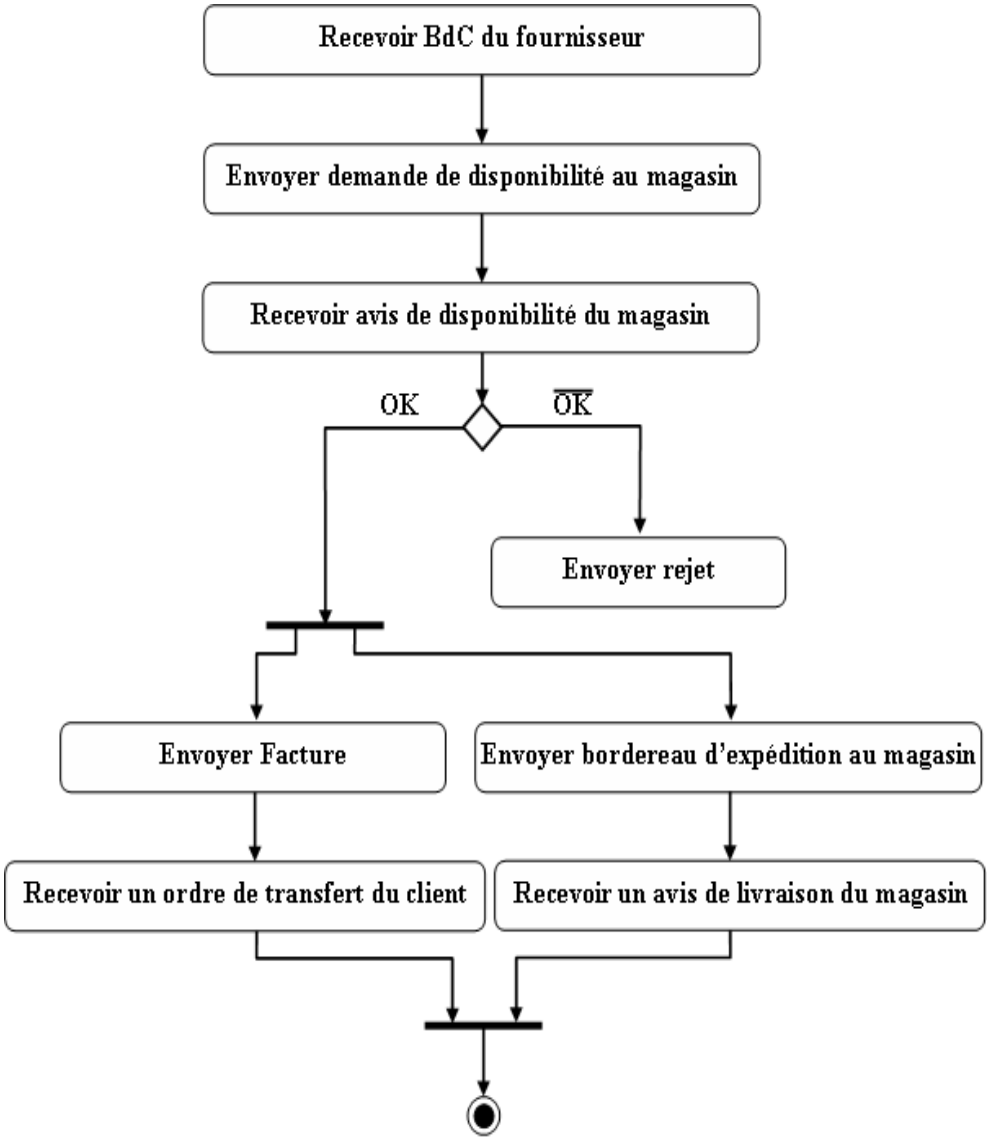


FIGURE 2.4 – Exemple de Conversation pour Fournisseur

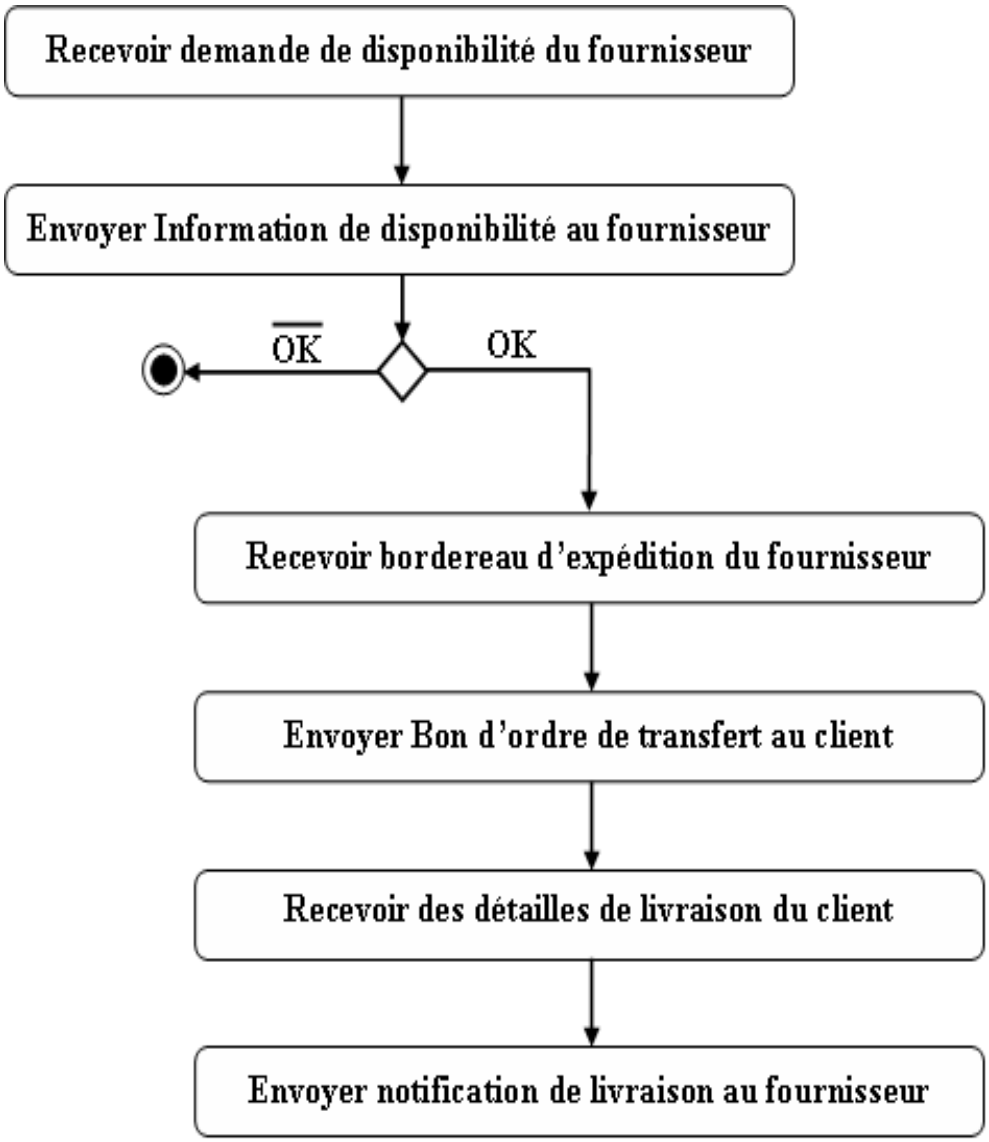


FIGURE 2.5 – Exemple de Conversation pour Magasin

portements et plusieurs interfaces WSDL. En outre, pour un rôle donné dans une chorégraphie, un nombre arbitraire de conversations peut être définie offrant la même fonctionnalité, mais pas nécessairement d'utiliser les mêmes interactions ou le même ordre d'interactions. Par exemple, dans la Figure 2.4 «*un bordereau d'expédition*» est envoyé au magasin en parallèle à celui-ci dans lequel des détails du paiement sont reçus à partir du client. Une autre solution serait que le paiement est reçu par le client avant que la commande de livraison soit envoyée.

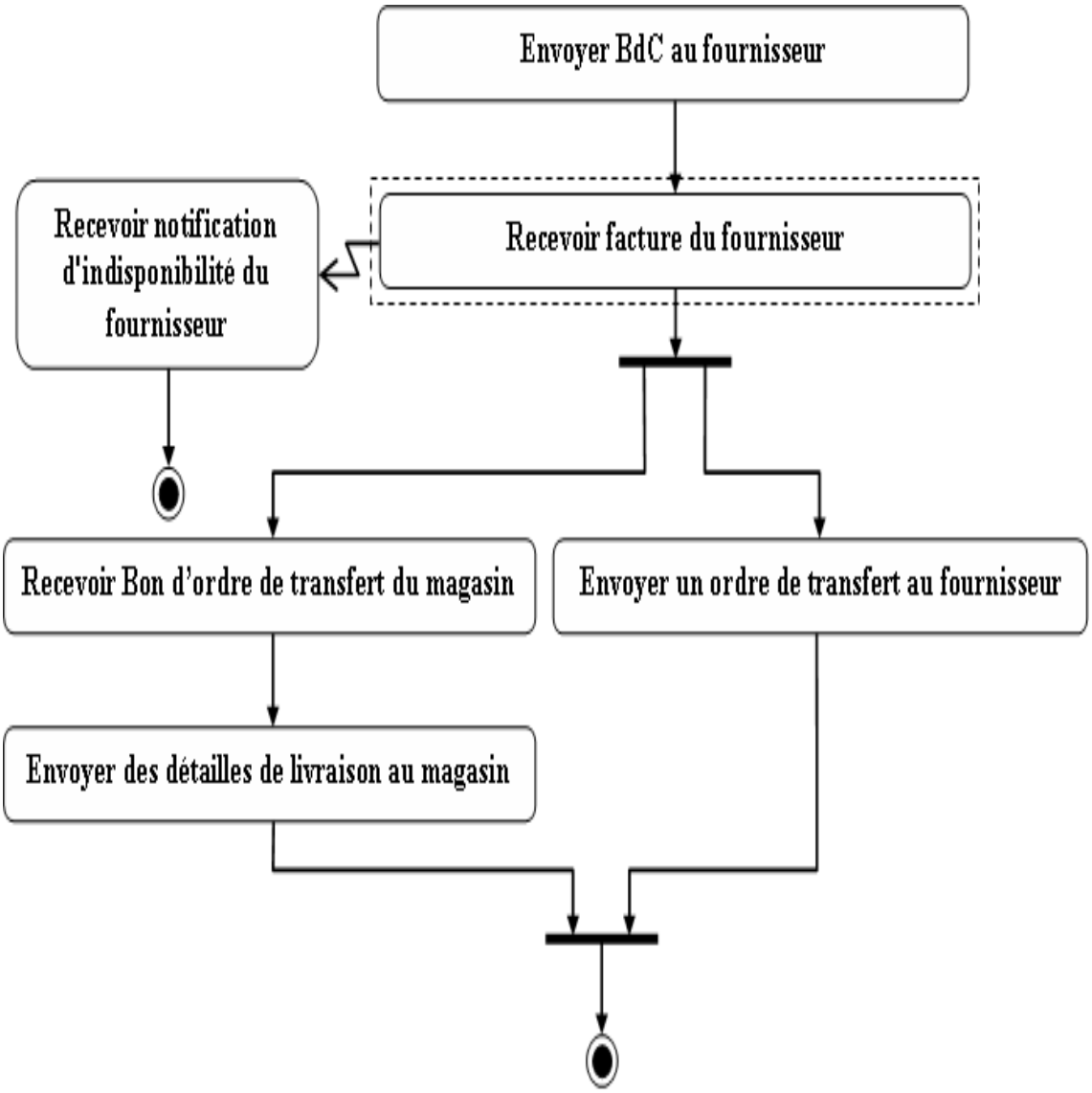


FIGURE 2.6 – Exemple de Conversation pour Client

Selon que l'interface capture une situation comme est (as is) ou à être (to be), alors une distinction peut être établie entre des interfaces offertes et prévues (ou requises). Une interface (comportementale) offerte est une abstraction de la manière qu'un service donné interagisse avec le monde extérieur. Une interface (comportementale) prévue capte une anticipation de la façon qu'un service doit se comporter afin d'effectuer un rôle donné dans une chorégraphie. Ainsi, une interface prévue correspond à un contrat qu'une partie donnée à besoin d'être satisfaite pour collaborer, avec suc-

cèes, avec d'autres parties. Idéalement, les interfaces offertes et prévues de services coïncident. Cependant dans la pratique, il peut arriver que l'interface fournie par un service soit différent de l'interface anticipée pour assurer un scénario donné. Dans ce cas, le fournisseur du service est responsable de la médiation entre l'interface prévue à fournir et à celle que l'implémente effectivement. Ce processus de médiation (ou d'adaptation) a fait l'objet de plusieurs travaux de recherche [Schmidt et Reussner., 2002].

Des implémentations de la conversation sont actuellement sous la forme de WSCL (Web Services Conversation Language) [Banerji et al., 2002] qui propose un, simple, langage standard de conversation qui peut être utilisé pour différents protocoles de services web, et des frameworks. Il est basé sur la modélisation de l'enchaînement des interactions ou des opérations d'une seule interface. Il comble les lacunes entre des langages de définition des interfaces simples qui ne spécifient aucune chorégraphie et des langages de flux ou de processus plus complexe qui décrivent des processus et des conversations multi-partie complexes globales.

2.4.3 Orchestration

Orchestration décrit comment les services peuvent interagir les uns avec les autres au niveau de message, y compris la logique métier et l'ordre d'exécution des interactions sous le contrôle et la perspective d'un seul point final (single endpoint). Un modèle d'orchestration décrit à la fois les actions de communication et les actions internes dans lequel un service s'engage. Les actions internes incluent des transformations de données et des invocations pour des modules logiciels internes. Une orchestration peut également contenir des actions de communication ou des dépendances entre des actions de communication qui n'apparaissent à aucune des interfaces comportementales de services. C'est parce que les conversations peuvent être mis à la disposition des parties externes et par conséquent, ils ne doivent montrer

que l'information qui doit effectivement être visible pour ces parties. Orchestration réfère à un processus métier exécutable qui peut entraîner un modèle de processus multi-étape, transactionnel, et à long terme (long-lived). Avec l'orchestration, les interactions d'un processus métiers sont toujours contrôlées de la perspective (privée) de l'une des parties métiers impliquées dans le processus.

La Figure 2.7 montre une orchestration d'un service fournisseur. Cette orchestration comprend une action interne pour la validation du paiement, représentée en lignes pointillées sur le diagramme. Ceci peut correspondre par exemple à une interaction avec un service qui n'est pas exposée au monde extérieur. D'autres actions internes peuvent être inclues dans cette orchestration. L'orchestration de la Figure 2.7 prend également en charge la possibilité d'une requête d'annulation d'une commande en cours de réception par le client - à tout moment - avant le paiement, conduisant à la terminaison (fin) du processus.

Orchestration est ciblée par une famille de langages standards, les plus représentatifs à base de XML, de définition de processus parmi ceux est le WS-BPEL (Business Process Execution Language for Web Services) [Andrews et al., 2003]. Cependant, BPEL est destiné à adresser à la fois les points de vue de'orchestration et de conversation. Exactement, la partie abstraite du processus BPEL peut être utilisée pour décrire les points de vue d'une conversation, et la partie exécution peut être utilisée pour décrire d'une orchestration.

2.4.4 Évaluation des modèles de composition

Nous présentons dans cette section un tableau synthétisant les principales caractéristiques des modèles de composition de services web. Le tableau récapitule les différents modèles de composition en spécifiant un certains nombre de critères afin de mettre en évidence les points forts et les points faibles de chaque modèle :

– **Mono-Control** : Un processus est toujours contrôlé par une seule partie

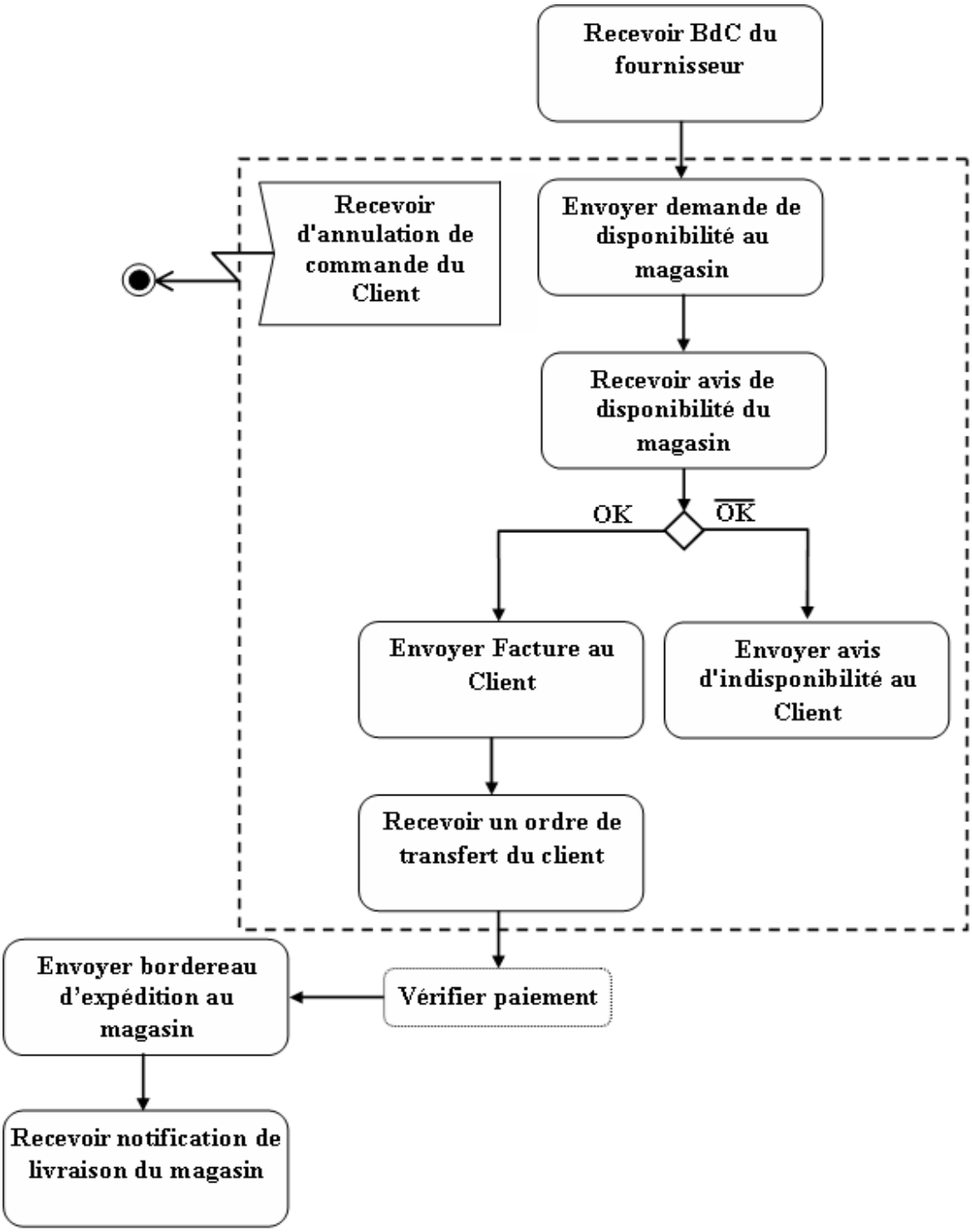


FIGURE 2.7 – Exemple d'Orchestration

- **Autonomé** : Chaque service est autonome des autres
- **Plan Pré-Défini** : Les services suivent un plan d'interactions pré-défini
- **Savoir de Participation** : Les services web concernés ne savent pas qu'ils

sont impliqués par une composition

- **Inclusion** : Inclusion des services web internes et externes
- **Echange Public** : Définition des messages publics échangés entre les services web participant à la composition
- **Suivi Collaboratif** : Suivi de la séquence de messages impliquant plusieurs parties et sources
- **Réaction à l'Anomalie** : Chaque participant est responsable du comportement adaptatif pour répondre aux anomalies
- **Description d'État** : Un message échangé doit contenir toutes les informations d'état nécessaires pour évaluer l'action suivante
- **Compensation** : La capacité de compensation des anomalies en temps réel
- **Processus** : La description des flux de processus de services

Le tableau 2.1 résume les modèles de composition de services. Le symbole $(+)$ signifie que la propriété sur cette colonne est supporté par le modèle de composition. Le symbole $(-)$ signifie que la propriété n'est pas prise en charge. Le signe $(+/-)$ signifie que la propriété est moyennement supporté. Enfin, le symbole N/A signifie non applicable, c.-à-d. la propriété ne s'applique pas au modèle avec lequel il a été associé.

Orchestration diffère de la chorégraphie en ce qu'il exprime un corps de la logique des processus métier qui est généralement détenu (possédée) par une seule organisation. Une orchestration établit un protocole métier qui définit formellement une définition de processus métier. La logique de workflow au sein d'une orchestration se décompose en une série d'activités de base et structurées qui peuvent être organisées en séquences et en paralléle. Plus collaboratif par nature, la chorégraphie suit la séquence de messages impliquant plusieurs parties, où aucune partie «possède» vraiment la conversation. Elle est associée à des échanges de messages publics qui se produisent entre plusieurs services web participants qui peuvent assumer des rôles

Approche / Critère	**Chorégraphie**	**Conversation**	**Orchestration**
Mono-Control	−	+	+
Autonomé	+	−	−
Plan Pré-Défini	+	−	−
Savoir de Participation	−	+	+
Inclusion	−	+/−	+
Echange Public	+	+/−	−
Suivi Collaboratif	+	−	−
Réaction à l'Anomalie	−	+/−	+
Description d'État	+	−	−
Compensation	−	−	+
Processus	−	+	+

TABLE 2.1 – Comparaison des modèles de composition de service

différents et qui ont des relations différentes. Si nous avons une analogie avec le trafic urbain, l'orchestration est semblable aux feux de circulation où les événements sont centralement contrôlés, tandis que la chorégraphie est plus comme un rond-point, où chaque participant suit un ensemble de règles pré-établies.

Une conversation exprime le comportement d'un fournisseur de service particulier ou d'un client de service dans sa communication avec un autre service fournisseur ou utilisateur pour accomplir un objectif particulier. Puisqu'une conversation décrit

seulement le comportement d'un seul service fournisseur, il comprend seulement un seul rôle dans la conversation de services. Une conversation peut être utilisée comme un point de départ pour générer un squelette, d'orchestration, qui peut ensuite être rempli de détails concernant les tâches internes et raffiné dans une orchestration complète. Ceci présente l'avantage que, une fois l'orchestration est entièrement raffinée, la conversation offerte du service sera coïncide avec la conversation prévue. D'autre part, une orchestration existante peut être utilisée pour générer la conversation offerte d'un service par des actions invisibles pour ne pas être exposées.

2.5 Langages de composition de services web

Dans ce qui suit, nous allons présenter un certain nombre de langages de conversation, d'orchestration et de chorégraphie de services web, que nous allons ensuite comparer. Les langages que nous avons choisi de présenter et de comparer sont des langages de composition récents, très présents dans le domaine de la recherche et de l'industrie car ils sont pour la pluparts proposés par de grands consortiums (par exemple W3C, BPMI [5], ou de grands organismes industriels (par exemple Sun, HP, IBM, Microsoft). Une présentation et une comparaison entre ces différents langages est donc intéressante, car elle peut nous aider à choisir un langage plutôt qu'un autre.

2.5.1 Langages de conversation de services

WSCL

WSCL (Web Services Conversation Language) [Banerji et al., 2002], est une proposition de vocabulaire XML représentant les interactions que publie un service web. WSCL à été proposé par Hewlett-Packard au W3C en février 2002. Un document

5. BPMI : Business Process Management Initiative

WSCL est constitué de trois blocs :

- les schémas XML des documents échangés durant l'interaction ;
- la description des interactions elles-mêmes (émission d'un message, réception d'un message, ou réception-puis-émission d'un message) ;
- la description de transactions gérant le passage d'une interaction à l'autre.

WSCL se traduit par une interface qui décrit le comportement d'un service, celle-ci peut être publiée dans un annuaire UDDI en vue d'être retrouvée puis prise en compte par d'autres services web. Contrairement aux langages d'orchestration et de chorégraphie, qui décrivent les connexions et les interactions entre un ensemble de services web, WSCL décrit les interactions d'un service web avec un seul autre service.

WSCI

WSCI (Web Services Choreography Interface) [Arkin et al., 2002], est un langage basé sur XML proposé en 2002 par les éditeurs de logiciels Sun, Intelio, BEA et SAP, qui étend WSDL et qui permet de décrire d'une part la conversation d'un seul service et d'autre part la chorégraphie d'un ensemble de services web. La conversation d'un service web est décrite via la construction «Interface». Cette construction utilise les opérations décrites dans le fichier WSDL, et les organise avec des opérateurs de contrôle, pour générer un flux d'échange de messages qui réalise une fonctionnalité donnée. La chorégraphie d'un ensemble de services web est décrite dans WSCI via la construction «Model» qui permet de construire à partir d'un ensemble de constructions «Interface», un schéma de collaboration entre un ensemble de services web.

OWL-S

OWL-S est un langage de description de services web [OWL-S., 2004, Claro et al., 2005, Lara et al., 2004], il utilise les ontologies OWL pour enrichir les descriptions de ces services par l'informations sémantiques décrivant leurs fonctionnalités, les données qu'ils échangent ainsi que d'autre informations caractérisant un service web telles que des propriétés non fonctionnelles. OWL-S permet également de décrire la conversation d'un service web et l'orchestration d'un ensemble de services web sous forme d'un processus décrit dans la construction «Process Model».

2.5.2 Langages d'orchestration de services

BPEL4WS et BPML

Les langages BPEL4WS (Business Process Execution Language for Web Services) et BPML (Business Process Modeling Language) sont des langages les plus utilisés pour la description de l'orchestration d'un ensemble de services web. Le langage BPEL4WS est le résultat de la fusion des deux langages de description de Workflow : XLANG développé par Microsoft et WSFL développé par IBM. Le langage BPEL4WS permet de décrire d'une part l'orchestration d'un ensemble de services web à travers la description de processus exécutables (executable processes), et d'autre part, la chorégraphie d'un ensemble de services web à travers la description de protocols abstraits (abstract protocols).

BPML est un langage proposé par le consortium BPMI pour la modélisation des processus métiers. La première version de BPML a été rendue publique en mars 2001. La comparaison de ces deux langages d'orchestration révèle qu'ils ont un grand nombre de concepts en commun, tels que la gestion d'erreurs, recouvrement etc. Il y a néanmoins quelques concepts qui les différencient, cependant ces différences sont minimes.

Orc

Orc (**Orc**hestration) [Misra., 2010, Orc, 2011] est utile pour la modélisation de traitements distribués mais principalement ciblé pour la modélisation d'orchestration de services web. Le langage est basé sur un framework mathématique pure et il a une sémantique formelle solide. La simplicité de sa syntaxe, et sa grande capacité d'expression, il le rendre très intéressant pour spécifier des orchestrations de services. *Orc* implémente la programmation arborescente (tree programming) pour des orchestrations où une requête initiale peut être transmis dans des branches parallèles ou cascadées en séquence pour réaliser la tâche requise. Les résultats des sous-requêtes peuvent être collectés ensuite transmit à d'autres sous-requêtes et enfin retournés à l'appelant initial. Ce genre de paradigme correspond convenablement à la notion d'orchestration. Le calcule de *Orc* (*Orc* calculus) est basé sur l'exécution des expressions *Orc*. Les expressions sont construites de manière récursive en utilisant des combinateurs de *Orc*. Lorsqu'il est exécuté, une expression *Orc* invoque des services et peut publier des valeurs. Les expressions *Orc*s utilisent des sites pour faire référence aux services externes. Un site peut être implémenté sur une machine d'un client ou une autre machine distante. Un site peut offre n'importe quelle service ou être un proxy pour l'interaction avec un utilisateur.

2.5.3 Langages de chorégraphie de services

La majorité des langages que nous avons présentés précédemment proposent de modéliser la chorégraphie de services web en plus d'un autre aspect tel que la conversation ou l'orchestration (par exemple WSCI et BPEL4WS). Cependant, un langage proposé par le W3C en Avril 2004 est susceptible de prendre le dessus pour la spécification de la collaboration entre services web ; c'est le langage WS-CDL (Web Services Choreography Description Language). WS-CDL est un langage basé sur XML pour la description de la collaboration pair-à-pair entre un ensemble de

services web. Il permet de définir le comportement externe, en termes de messages échangés, d'un ensemble de services web qui collabore pour atteindre un objectif commun. Les auteurs de WS-CDL, se sont inspirés des langages WSCI, XLANG, WSFL, BPEL, BPML et XPDL pour construire un langage complet répondant à la majorité des besoins des entreprises en terme de puissance de modélisation (expressiveness).

2.5.4 Évaluation des langages de composition

Nous présentons dans cette section un tableau synthétisant les principales caractéristiques des langages de composition. Le tableau récapitule les différents langages de composition de services web en spécifiant un certains nombre de critères afin de mettre en évidence les points forts et les points faibles de chaque langage. Ce qui nous permet de choisi et de combler la lacune de celui le plus approprié pour notre travaille :

- **Conversation** : L'adoption de la conversation comme un modèle de composition.
- **Orchestration** : L'adoption de l'orchestration comme un modèle de composition.
- **Chorégraphie** : L'adoption de chorégraphie comme un modèle de composition.
- **Sémantique** : la capacité de spécification des descriptions sémantiques autant explicites et précises.
- **QoS** : la prise en compte des caractéristiques paramétriques non fonctionnelles comme (par exemple Temps de Réponse, Disponibilité, Fiabilité, etc.).
- **Graphe** : Le support de représentation graphique de la composition.
- **WSDL** : L'adoption des fichiers WSDL comme des services concrèts impliqués par la composition.

Le tableau 2.2 résume les langages de composition de services présentés. Le symbole (+) signifie que la propriété sur cette colonne est supporté par le langage de composition. Le symbole (−) signifie que la propriété n'est pas prise en charge. Enfin, le signe (+/−) signifie que la propriété est moyennement supporté.

Langage / Critère	WSCL	WSCI	OWLS	BPEL	BPML	WSCL	Orc
Conversation	+	+	+	−	−	−	−
Orchestration	−	−	+	+	+	−	+
Chorégraphie	−	+	−	−	−	+	−
Sémantique	−	−	+	−	−	+	−
QoS	−	−	+/−	−	−	−	−
Graphe	−	−	−	+	−	−	−
WSDL	+/−	+	+/−	+/−	+/−	+/−	−

TABLE 2.2 – Comparaison des langages de composition

2.6 La représentation du processus de services

Cette section présente quatre formalismes de modélisation des processus métier. Ces formalismes offrent des mécanismes de représentation pour décrire, échanger et exécuter des processus métiers.

2.6.1 Algèbres de processus

Les processus peuvent être modélisés en utilisant les algèbres de processus. Des exemples de modèles d'algèbre de processus sont CCS (calculus of communicating systems) [Milner., 1982] , CSP (communicating sequential processes) [Hoare., 1985] et $\Pi - calcul$ [Milner., 1999] . Ces langages fournissent des algèbres pour la spécification et le raisonnement sur des systèmes concurrentiels ou des processus. Ils fournissent des ensembles de termes, d'opérateurs et d'axiomes pour l'écriture et la manipulation d'expressions algébriques. Le comportement des systèmes modélisés peut être analysé sur la base des opérateurs définis. Le $\Pi - calcul$ est un modèle mathématique de processus dont les interconnexions changent comme ils interagissent [Bergstra et al., 2001] . L'étape de calcul de base est le transfert d'un lien de communication entre deux processus, le destinataire peut alors utiliser le lien pour plus d'interaction avec d'autres parties. Ce qui rendre le $\Pi - calcul$ particulièrement approprié de représentation lorsque les ressources accessibles varient avec le temps. Le $\Pi - calcul$, en plus de la modélisation de systèmes concurrents, peut également exprimer des processus et des techniques versatiles pour analyser leur comportement. Certaines des opérations supportées par ces langages sont la simulation et bisimulation des instances de processus. Par exemple, étant donné deux descriptions de processus de notation $\Pi - calcul$, il est possible de vérifier leur équivalence en utilisant une opération de bisimulation.

Dans [Mazzara et al., 2006] , les auteurs affirment que BPEL n'est pas doté de la sémantique formelle (ainsi que les autres propositions existantes) et puisqu'il comprend un grand nombre d'aspects, il est difficile de raisonner formellement sur le comportement des processus. A la lumière de cette observation, la sémantique d'un fragment BPEL est formellement traitée dans [Mazzara et al., 2006] . La proposition représente une contribution significative dans les deux sens. Premièrement,

formalise un nouveau langage d'orchestration, *web*$\Pi\infty$ qui représente lui-même une simplification de WS-BPEL y compris une spécification claire et non ambigu, ce qui rendre possible de raisonner formellement sur les processus d'orchestration. Deuxièmement, un réalisateur d'un moteur d'orchestration BPEL effectif peut implémente d'une manière très simple ce mécanisme unique fournissant tous les autres mécanismes par compilation. Le langage est composé d'un petit ensemble d'opérateurs qui peuvent répondre aux comportements BPEL, offrant une simplicité raisonnable pour les concepteurs des applications. Ce langage comprend : un opérateur parallèle qui permet la concurrence explicite, un opérateur de restriction permettant la création de compositionnalité et de ressources explicite ; une récurrence ou un opérateur de définition de processus, un opérateur de séquence permettant la relation causale entre les activités, un opérateur de l'inaction qui est juste un terme de masse pour la définition inductive sur le séquençage ; le passage de message et surtout les opérateurs de passage de noms permettant la communication et la mobilité de lien.

2.6.2 Réseaux de pétri

Un réseau de Pétri est un langage formel et graphique pour la modélisation des systèmes et des processus [Peterson., 1981]. Il comprend des places, des transitions, des arcs et des jetons. Les places représentent des états d'un réseau de Pétri et ils peuvent contenir des jetons. Les arcs relient les places avec les transitions. L'arc d'entrée reliant la place à la transition, alors que l'arc de sortie reliant la transition à la place. L'état courante du système modélisé, appelé le marquage, est donné par le nombre de jetons dans chaque place. Lorsqu'un jeton est présent dans chacune des places en entrée d'une transition, la transition est dite active et peut alors s'exécuter. L'exécution d'une transition consomme des jetons pour une place en entrée et les place dans une place en sortie, c.à.d. des jetons sont retirés des places d'entrées et inséré dans des places de sortie d'une transition. Les transitions ne peuvent être

franchies si elles sont actives, c.à.d. s'il ya des jetons prêts à franchir dans les places d'entrée.

Les réseaux de Pétri offre un outil pour la description des systèmes qui sont caractérisées comme étant concurrente, asynchrone, distribué et non déterministe. Du point de vue graphique, les réseaux de Pétri peut être utilisé comme une aide de communication visuelle d'une manière similaire à ce que les notations de la conception structurée de l'analyse des systèmes traditionnelle et des méthodologies de conception. Cependant, le langage de Pétri fournit une base mathématique solide pour la description et l'analyse des équations d'état, algébrique et d'autres modèles mathématiques. On obtient ainsi une notation pratique pour décrire le comportement de processus du système, telles que celle donnée pour une alternative un peu simplifiée [6].

Un formalisme lié au réseaux de Pétri est WF-net (Workflow Net) [Aalst., 2002]. WF-Nets ont été utilisés pour modéliser les flux d'activités inter-organisationnelle [Aalst., 1999] . Comme les réseaux de Pétri, les WF-Nets sont basée sur des jetons et contiennent des places et des transitions, mais ils contiennent en plus, une place initial et final unique. Ils ont de meilleures propriétés de calcul que les réseaux de Pétri, mais sont utilisés dans les modèles de communication asynchrones où les messages peuvent arriver dans un ordre différent de celui dans lequel ils ont été envoyés. Les auteurs dans [Hamadi et al., 2003] proposent une algèbre basée sur réseaux de Pétri pour la composition de services web. La sémantique formelle des opérateurs de composition est exprimée en termes de réseaux de Pétri en fournissant un mappage direct de chaque opérateur à une construction réseau de Pétri. Ainsi, tout service exprimé en utilisant les constructeurs algébriques peuvent être convertis en une représentation de réseau de Pétri. Grâce à un ensemble de propriétés algébriques, l'approche est capable de transformer et d'optimiser les expressions de service web en garantissant les mêmes sémantiques des expressions initiales. En outre, l'utili-

6. www.tml.tkk.fi/Opinnot/Tik-110.501/1995/intfo.html

sation d'un modèle formel permet la vérification de propriétés et la détection des incohérences à l'intérieur et entre des services.

2.6.3 Automate d'état finis

Un AEF est définie par un ensemble fini de messages, états, un ensemble de transitions, un état initial et un ensemble d'états finaux (ou états acceptées) [Hopcroft et al., 2001]. Il peut être représenté comme un graphe avec un seul état initial, où les noeuds représentent les états, les arcs représentent les transitions reliant deux états. Les transitions sont étiquetées avec des messages tirés de l'ensemble de messages. Les graphiques des AEF sont traversées de l'état initial. Les états finaux sont marqués avec des cycles concentriques, et ils représentent la séquence de messages acceptée par AEF. Les AEFs sont comparés avec une opération d'intersection et ont des algorithmes en temps polynomial pour le test de vide et d'intersection. Cependant les AEFs dans leur forme originale ne peuvent pas représenter les sémantiques obligatoires de séquences de messages. Par conséquent AEF n'ont pas la sémantique d'exécution parallèles, comme prévu par des approches plus expressives comme les réseaux de Pétri.

Dans [Berardi et al., 2003], les auteurs décrivent un modèle conceptuel pour représenter le comportement e-service et des contraintes temporelles, à base des AEFs. Les AEFs permet de capturer une grande classe de e-Services et de vérifier formellement des propriétés importantes de e-Services [Weikum., 1999] tels que l'exactitude, la sécurité (i.e., à chaque point d'exécution d'un e-service, certains invariantes logiques sont capturées), vivacité (i.e., un e-service est assuré de se déplacer vers un point où un but peut être atteint). En outre, cette approche introduit un nouveau langage, WSTL (Web Service Transition Langage), qui s'appuie sur un tel modèle conceptuel. La nouveauté de cette approche est que WSTL fournit des constructeurs dont les éléments correspondants des ASFs peuvent être carrément mappés

2.6.4 Graphes

Les graphiques sont une structure générale et puissante de données pour la représentation des objets et des concepts. Dans une représentation graphique, les noeuds représentent généralement des objets ou parties d'objets, tandis que les arêtes décrivent des relations entre des objets ou parties d'objets. Les graphes ont des propriétés d'invariances intéressantes. Par exemple, si un graphe est établi sur papier, au sens mathématique (s'il est traduit, pivoté, et transformé en son image miroir) il est toujours le même graphe. Ainsi que, le fait que les graphiques sont bien suitables pour la modélisation d'objets en termes de portions et de leurs relations. Ces propriétés d'invariance et de modélisation les rendre très intéressant pour diverses applications [Bunke., 2000]. Dans les applications telles que la reconnaissance des formes et la vision par ordinateur, l'opération de similarité d'objets est un élément important, où une base de données des objets connus et une requête, la tâche est de trouver un ou plusieurs objets de la base de données qui sont similaires à la requête. Si les graphes sont utilisés pour la représentation de l'objet ce problème se transforme en déterminer la similarité de graphes, qui est généralement appelée matching de graphes. Dans [Gerbe et al., 1998], les auteurs présentent un méta-modèle sur la base de graphes conceptuels pour représenter les processus qui répond aux exigences de mémoire institutionnelle (acquisition, le stockage, l'évolution et la diffusion des connaissances acquises par l'organisation). Le méta-modèle est composé de trois concepts de base : activité, processus et événement. Une activité est définie par ses entrées et sorties, les agents qui supportent l'activité et des pré- et post-conditions. Les pré-conditions définissent des conditions ou états qui doivent être vérifiées pour déclencher l'exécution de l'activité ; les post-conditions définissent des états ou conditions qui résulteront de l'exécution de l'activité. Un événement est un point dans le temps qui marque la fin d'une activité, il marque la réalisation de

la post-condition de l'activité. Un processus est défini comme un ensemble d'événements qui représentent l'exécution d'un ensemble d'activités.

Dans [Mendling et al., 2005], les auteurs ont introduit une transformation de BPEL à EPCs (Event-Driven Process Chains). Un schéma EPC plat est défini comme un graphe orienté et cohérent avec les contraints type et cardinalité. Construit sur un mapping conceptuelle, cette approche présente un programme de transformation qui est capable de générer automatiquement des modèles EPC sous forme des fichiers EPML (format d'échange à base XML pour EPCs) à partir des définitions de processus BPEL. Une telle transformation permet de communiquer des processus BPEL aux analystes d'affaires qui sont souvent impliqués dans l'approbation du logique métier. La visualisation EPC est basée sur le comportement dynamique du modèle BPEL. Les constructeurs BPEL (c.-à-d. les activités de base et structurés) sont transformés en blocs d'éléments EPC qui offrent des sémantiques équivalentes. Les éléments EPC obtiennent des noms qui sont générés à partir des noms des éléments BPEL correspondantes. En outre, le programme peut être utilisé pour la réingénierie des processus BPEL. Enfin, le concept transformation est général de manière à ce qu'il puisse être facilement adapté pour générer une sortie d'un autre langage de processus à base de graphe qui est codé en XML.

2.6.5 Évaluation des formalismes de représentation de processus

Nous présentons dans cette section un tableau synthétisant les principales caractéristiques des formalismes de représentation de processus de services web. Le tableau récapitule les différents formalismes de représentation de processus de services web en spécifiant un certains nombre de critères d'évaluation afin de mettre en évidence les points forts et les points faibles de chaque formalisme.

– **Complétude** : est un ensemble complet de la sémantique et de la maturité

de la notation.

- **Composabilité** : est une propriété qui permet de raisonner sur un système composé sur la base de ses éléments constitutifs sans besoin supplémentaire d'information sur la mise en oeuvre de ces parties
- **Parallélisme** : est un aspect clé des formalismes qui doit être remplie pour modéliser avec précision la composition de services web,
- **Complexité** : complexité d'exécution d'appariement est polynômiale ou non.

Le tableau 2.3 résume les formalismes de représentation de services. Le symbole $(+)$ signifie que la propriété sur cette colonne est supporté par le formalisme de représentation. Le symbole $(-)$ signifie que la propriété n'est pas prise en charge. Le signe $(+/-)$ signifie que la propriété est moyennement supporté. Enfin, le symbole N/A signifie non applicable, c.-à-d. la propriété ne s'applique pas au modèle avec lequel il a été associé. Par rapport à nos travaux qui portent sur l'aspect comporte-

Critère / **Formalisme**	**Complétude**	**composabilité**	**Parallélisme**	**Complexité**
AdP	+	+	+	N/A
RdP	+	−	+	+
AEF	+	+	+/−	−
Graphes	+	+	+	+

TABLE 2.3 – Comparaison des formalismes de représentation de processus

mental pour la composition personnalisée de services web, il s'avère qu'aucune de ces approches n'est complètement mature pour prendre en charge totalement et effica-

cement l'ensemble du processus de composition de services web. En outre, toutes ces formalismes, excepte des AEFs, souffrent du problème de complexité ce qui influe sur la flexibilité et l'assouplissement du processus de composition. Cependant, au vu de tous les éléments présentés dans cette section, il nous paraît que le formalisme AEF est celui qui présente le moins de complexité et également il offre une notation simple et efficace pour exprimer la sémantique du service, ce qui nous à pousser de le considérer comme le plus mature malgré qu'ils, dans leur forme originale, ne peuvent pas représenter la sémantique d'exécution parallèle de messages. Cependant les AEFs peut être étendue afin de résoudre ce genre de problèmes surtout par le biais de sa représentation grammaticale comme nous venons le voir dans ce qui suit. C'est la raison fondamentale qui nous a amené à nous baser sur ce model comme fondement de base pour les travaux que nous menons dans le cadre de la composition personnalisé de services web.

2.7 Coclusion

Des standards ont été proposés pour la description de services web, leur découverte ainsi que leur composition. Ainsi, le langage WSDL est couramment utilisé pour décrire l'interface des services. L'annuaire UDDI a été proposé pour résoudre le problème de découverte. Enfin, le langage BPEL a su s'imposer dans le domaine de la composition de services web.

Bien que BPEL semble être accepté par la majorité comme langage exécutable pour la composition de services, celui-ci ne support pas la composition dynamique de services web. Il existe donc un manque à combler dans le domaine de la composition dynamique des services. Ce manque ne permet pas le développement de services composés à la volée.

Dans ce chapitre :

– Nous avons présenté les concepts de base du service web, la composition de

services, l'architecture de référence ainsi que l'architecture étendue pour les services web.

- Nous avons passé en revue les modéles de composition de services, et également une évaluation de ces modéles a été introduite.
- Nous avons présenté des différents langages de composition de services web selon les trois modèles de composition de services. Ensuite, une évaluation de ces langages de composition a été effectuée. Le but de cette comparaison était de fournir des critères pour la description et l'exécution de la composition de service web. Nous avons conclu que le langage $OWL-S$ fournit une notation sémantique et mature pour exprimer la sémantique des services composites.
- Puisque notre approche de sélection et de composition de services opère sur des modèles comportementaux de services, dans ce chapitre, nous avons illustré des représentations formelles qui permettent de décrire, l'échange et l'exécution des comportements de service. Les représentations formelles présentées sont : l'algèbres de processus, Réseaux de pétri, Automate d'état finis et les Graphes. Enfin, une évaluation de ces représentations formelles a été effectuée. Le but de cette comparaison était de fournir des critères pour la modélisation de la composition de service web. Nous avons conclu que le modèle de représentation AEF fournit une notation simple et plus ou moins mature pour exprimer la sémantique de service. En outre, un AEF, par le biais de sa représentation grammaticale, permet de raisonner sur un système composé sur la base de ses parties constituantes, représentées par les différentes de règles de productions des différentes non-terminaux, sans avoir besoin d'information supplémentaire sur la mise en oeuvre de ces parties.

Chapitre 3

Appariement et Composition Dynamique de Services web : État de l'Art

Sommaire

3.1 Introduction

Actuellement, les services web sont très utilisés notamment par les entreprises pour rendre accessible leurs métiers ou leurs données via le web. L'émergence de ces services a permis aux applications web d'être vues comme un ensemble de services

métiers bien structurés et correctement décrits, plutôt qu'un ensemble d'objets et de méthodes. En effet, cela facilite les échanges entre les applications d'une même organisation, mais permet aussi l'ouverture vers les applications des autres organisations d'une façon distribuée ; ce qui permet d'offrir des avantages primordiaux en termes d'interopérabilité et de maintenance de telles applications. La composition de services sert à définir et à faciliter le processus d'intégration et de collaboration entre plusieurs services distribués. Cette technologie émergente présente beaucoup d'intérêts, tant dans le domaine de la recherche que dans le domaine industriel. L'une des plus importantes et complexes tâches de l'architecture SOA est de créer des services à valeur ajoutée ; c'est-à-dire des services composés qui sont le résultat de la composition de plusieurs services existants. La complexité de cette tâche provient de la nature dynamique de l'environnement des services web. En effet, le nombre de fournisseurs de services augmente constamment et, de nouveaux services deviennent, de plus en plus disponibles. Idéalement, les services à composer devraient être capables de s'adapter aux changements de l'environnement et aux exigences des consommateurs. Par conséquent, la difficulté majeure réside dans la composition dynamique de services, en particulier lorsqu'une tâche, indispensable pour un scénario, ne peut pas être réalisée par le service existant.

Dans ce chapitre nous allons présenter les différentes recherches effectuées dans le domaine de composition de services web. Nous allons voir également, dans la section 3.2 les approches d'appariement de services web. Ensuite, nous adressons dans la section 3.3 les approches de composition dynamiques de services. Enfin, Nous discutons dans la section 3.4 les différents travaux présentés pour nous permettre de mieux positionner notre approche, et voir comment il serait possible d'adresser le problème de personnalisation de la composition de services web en tenant compte non seulement les aspects non-fonctionnels (par le biais des attributs QoS) mais aussi d'autres aspects comportementaux (par le biais de processus de service).

3.2 L'appariement de services

Dans cette section, nous expliquons les techniques de matching des services qui sont basées sur les formalismes de modélisation de processus. Ensuite, une évaluation des techniques existantes est donnée. Plus précisément, nous montrons les différentes techniques de matching de service selon une classification de trois catégories : à base d'interfaces, à base de sémantique, et à base de comportement.

3.2.1 Appariement de service à base d'interfaces

Actuellement, les algorithmes de découverte des services web dans les registres comme UDDI ou ebXML sont basés sur une recherche par mots clés ou des tables de correspondance des couples « clé-valeur ». Pour supporter un processus de découverte et de composition de services plus précis, des mécanismes basés sur le matching d'interface ont été proposées [Stroulia et al., 2005, Kokash et al., 2006, Kritikos et al., 2013, Sana et al., 2013, Hamdi et al., 2014]. De telles approches de matching de service web tendent à aborder le matching syntaxique et ou sémantique. Pour analyser leurs mérites, il est utile de les classer en uniforme ou hybride. Les approches de matching uniformes se réfèrent à des techniques de matching atomiques qui ne peuvent être décomposés encoure en ne granularité plus fins. Les approches de matching hybrides peuvent combinent plusieurs méthodes de matching (par exemple, syntaxiques et sémantiques) dans un algorithme globale.

Dans [Stroulia et al., 2005], les auteurs discutent un ensemble de méthodes complémentaires pour évaluer la similarité des spécifications d'interface WSDL, permettant d'ordonner les services potentiellement utiles en fonction de leur pertinence à la requête de demandeur. Pour apprécier la similitude entre deux spécifications WSDL, ces méthodes utilisent, d'une part, la sémantique des identifiants et des descriptions en langage naturel des spécifications WSDL, et d'autre part, la structure de leurs

opérations, les messages et les types. Plus précisément, les auteurs décrivent une suite de méthodes de découverte de services web qui combinent une technique traditionnelle inspirée du domaine de recherche d'information et deux autres techniques basées sur WordNet avec un algorithme de matching structure à base de structure de spécification d'un service, bâti sur XML, WSDL.

Dans [Kokash et al., 2006] les auteurs proposent l'algorithme WSDL-M2 qui combine deux techniques : une de matching lexical pour calculer la similarité linguistique entre les descriptions de concept, et une autre de matching structurel pour évaluer la similarité globale entre des concepts composites. Le processus global de matching est implémenté en trois étapes : (i) d'abord, tous les fichiers de la collection des spécifications WSDL sont analysés afin de permettre l'extraction de leur contenu structuré. (ii) Ensuite, le document analysé est tagué (étiqueté) pour permettre une analyse lexicale. (iii) Enfin, les spécifications WSDL tagués peuvent encore être analysés et ensuite indexées à partir de différents modèles de recherche d'information (VSM : Vector-Space Model et $tf - idf$ measure). Pour surmonter les principales lacunes de VSM une combinaison de VSM et WordNet est proposée. Le matching structurelle est traité comme un matching de poids maximum bipartite pour calculer la similarité sémantique entre des descriptions de concepts, ainsi que la similarité des concepts WSDL complexes en tenant compte leurs constituants (sous-types).

Sur un autre plan, récemment, il y a eu une prolifération de moteurs de recherche de services web sur Internet. Ceux-ci peuvent être regroupés en deux types. Le premier (par exemple Bindingpoint, NET XML web Services Repertory, web service X.NET, Web Service List et SalCental) accepte comme entrée, les mots clés qu'ils utilisent pour chercher dans les descriptions WSDL de services (voir [Saboua et Panb., 2007]). Le deuxième (par exemple Woogle) surpasse un simple matching, par mots-clés, des contenus des documents WSDL en effectuant une recherche de similitude sur les opérations WSDL des services web, en tenant compte le nom d'opération

et les paramètres d'entrées / sorties (voir [Dong et al., 2004]). Dans [Dong et al., 2004], le moteur de recherche combine plusieurs sources de preuve pour détecter la similarité : descriptions textuelles des opérations et des services web entiers et la similarité entre les noms de paramètres des opérations. Un algorithme sous-jacent est basé sur une technique qui regroupe les noms de paramètres dans la collection de services web en des concepts sémantiquement utiles. Ces concepts sont utilisés dans la comparaison des paramètres d'entrées et de sorties. En résumé, les approches utilisées par les moteurs de recherche de service web ne peuvent faire le matching seulement pour des services simples, donc ne traitent pas les aspects d'exécution, comportement, de services. Les auteurs dans [Kritikos et al., 2013, Hamdi et al., 2014], proposent une approche à base de la logique floue pour le matching de services web. L'appariement s'étend sur une étape, dépendante du domaine, qui produit des règles de classification pour les services web. Les règles élaborées tirer parti d'un ensemble d'attributs fonctionnels et non-fonctionnels, qui est extrait en utilisant la théorie des ensembles. En outre, ces règles sont mises à profit pour classer les services web dans des catégories, pour réduire l'espace de matching.

3.2.2 Appariement de service à base de sémantique

Dans le framwork du web sémantique, une logique de description a été proposée pour une description formelle, plus riche et précise, de services. Ces langages permettent la définition des ontologies, tel que par exemple OWL-S. OWL-S [David et al., 2004] a proposé une ontologie pour décrire des services web basés sur le langage d'ontologie web (OWL) [7]. OWL-S est structuré en trois types d'information essentiels pour un service : le profil de service, modèle de service et le grounding de service. Le profil de service décrit ce que réalise le service. Le modèle de service décrit les services en termes d'entrées, de sorties, de pré-conditions et d'effets de service ;

7. www.w3.org/TR/owl-ref

les processus sont décrits en termes de leurs états, y compris des informations telles que l'activation initiale, l'exécution et l'achèvement. le grounding de service décrit comment accéder au service.

Par la description de la fonctionnalité (opérations) de service en utilisant OWL-S, il est possible de trouver des services web à partir de la perspective sémantique. Plusieurs prototypes de matching de services web ont été implémenté en utilisant cette approche, par exemple [Chiat et al., 2004, Li et al., 2003, Paolucci et al., 2002, Bernstein et al., 2002, Benatallah et al., 2003, Trastour et al., 2001]. Des travaux relatifs peuvent être trouvée dans [Abhijit et al., 2004, Sivashanmugam et al., 2003], où des approches d'annotation de services web avec des informations sémantiques pour la découverte de service sont décrits. Dans [Paolucci et al., 2002, Benatallah et al., 2003], un service publié est compatible avec un service requis lorsque les entrées et les sorties du service requis sont compatibles avec les entrées et les sorties du service publié (c'est à dire, ils ont le même type ou l'un est une généralisation de l'autre). Dans [Kawamura et al., 2003], des filtres indépendants sont définis pour la découverte de service : l'espace de nom, la description textuelle, le domaine de l'ontologie utilisé, les types d'entrées / sorties et les contraintes.

Les approches présentées dans [Cardoso et Sheth., 2003, Plebani et al., 2009, Segev et al., 2009, Iosif et al., 2010, Farrag et al., 2011, Farrag et al., 2012, Cassar et al., 2013] prend en compte les propriétés opérationnelles telles que le temps d'exécution, le coût et la fiabilité. Les auteurs de [Wu et al., 2005] fournissent une comparaison sémantique léger des méthodes d'évaluation de la similarité à base d'interfaces (similarité lexical, d'attribut, d'interface et de QoS). Les approches sémantiques mentionnées ci-dessus sont très efficaces pour le matching de services simples basés sur des descriptions sémantiques de leurs capacités. D'ailleurs, il n'est pas clair comment ces approches peuvent faire le matching des processus métier complexes, qui considère l'information structurelle des processus compris dans les services web.

Dans [Chen et al., 2011], une approche de sélection de service web basé sur la similitude entre les entrées et les sorties des services. Ces similitudes sont calculées en fonction du résultat de l'algorithme de règles associatives. Le concept de similarité permet de mesurer le degré de correspondance entre un service requis et d'autres offerts, où le plus haut degré de similitude mesurée est choisit comme une meilleur préférence pour un client donné.

3.2.3 Appariement de Service à base de comportement

La sélection et la composition de service à base des mots clés ou des attributs sémantiques n'est pas satisfaisante pour un grand nombre d'applications. La tendance des travaux récents est d'exploiter de plus en plus de connaissances sur les éléments de service et son comportement. La nécessité de prendre en compte le comportement de service décrit par un modèle de processus a été soulevée par plusieurs chercheurs [Trastour et al., 2001, Bansal et Vidal., 2003, Bernstein et al., 2002, Piccinelli et al., 2001, Wombacher et al., 2004, Cao et al., 2012, Cao et al., 2013]. Dans [Bernstein et al., 2002, Cao et al., 2012, Cao et al., 2013], afin d'améliorer la précision de découverte de service web, le modèle de processus est utilisé pour capturer le comportement d'un service. Un langage de requête pour les services est défini qui permet de trouver des services en spécifiant les conditions sur les activités qui les composent, les exceptions traités, et le flux de données entre les activités. Dans le monde académique, plusieurs travaux de recherche ont traité la similarité et la compatibilité à différents niveaux d'abstraction d'une description de service (par exemple, [Benatallah et al., 2004, Bordeaux et al., 2004, Dong et al., 2004, Wombacher et al., 2004]). En termes de protocole et d'analyse, les approches existantes fournissent des modèles (par exemple, sur la base de $\Pi - calcul$, réseaux de Pétri ou des automates d'état) et des mécanismes pour comparer des spécifications (par exemple, les protocoles de vérification de compatibilité).

[Ould et al., 2006], propose une approche de binding dynamique du processus BPEL (binding dynamique des services avec des instances WS-flow au moment d'exécution, c'est à dire la substitution, en trois étapes, d'un service participant avec une instance de WS-flow). Premièrement, la proposition d'une description de haut niveau de processus. Deuxièment l'abstraction du comportement de processus en utilisant des SOG (symbolic observation graphs) en se basant sur, une forme spécifique de réseaux de Pétri, le Wf-nets (workflow nets). Enfin la proposition d'un algorithme pour le matching de SOG utilisé pour le binding dynamique des processus métiers.

Dans [Kuang et al., 2006], le modèle $\Pi - calcul$ est utilisée pour formaliser le comportement d'un service web et de la requête. Plus précisément, des opérations simples impliquant des échanges de messages sont exprimés en $\Pi - calcul$, différenciés par quatre types de transmission, unilatéral (one-way), notification, demande-réponse et solliciter-réponse. En outre, les contraintes entre les opérations, d'un service ou d'une requête, qui définissent l'ordre d'exécution autorisé sont exprimés en $\Pi - calcul$. Après avoir exprimé le service et la requête en $\Pi - calcul$, le matching de service entre une description de la requête et une description du service est realisé grâce à la capacité de $\Pi - calcul$.

Dans [Wombacher et al., 2004], les auteurs proposent une sémantique formelle de matching de processus métier basés sur des automates d'états finis étendus par des expressions logiques associées à des états. Les auteurs dans [Mahleko et al., 2006] présentent une approche d'indexation pour interroger des processus métier cycliques en utilisant des systèmes de bases de données traditionnelles. Un autre modèle de comportement pour les services web est présenté dans [Shen et al., 2005] qui associent les messages échangés entre les participants avec des activités réalisées dedans le service. Des profils d'activité sont décrits en utilisant OWL-S (Web Services Ontology Language). Les services web sont modélisés comme des automates d'états finis non-déterministes où nouveau langage de requête est développé pour exprimer des

propriétés temporelles et sémantiques pour les comportements de service.

3.2.4 Évaluation des techniques d'appariement

Nous présentons dans cette section un tableau synthétisant les principales caractéristiques des approches d'appariement de services web. Le tableau récapitule les différentes approches d'appariement de services web en spécifiant un certains nombre de critères afin de mettre en évidence les points forts et les points faibles de chaque approche :

- **Stateful (Avec état)** : signifie que la technique assure le suivi de l'état de l'interaction, généralement en définissant des valeurs dans un champ de stockage désigné à cet effet.
- **Stateless (Sans état)** : signifie qu'il n'existe aucune trace sur des interactions déjà effectué et chaque demande d'interaction doit être traitée entièrement en se basant seulement sur l'information qui vient avec elle.
- **Sémantique** : la capacité de spécification des descriptions sémantiques autant explicites et précises.
- **QoS** : la prise en compte des caractéristiques paramétriques non fonctionnelles comme (par exemple Temps de Réponse, Disponibilité, Fiabilité, etc.).

Le tableau 3.1 résume les techniques d'appariement de services. Le symbole (+) signifie que la propriété sur cette colonne est supporté par la technique d'appariement. Le symbole (−) signifie que la propriété n'est pas prise en charge. Enfin, le signe (+/−) signifie que la propriété est moyennement supporté.

Les paramètres Stateful et Stateless sont des propriétés qui décrivent si une approche est conçu pour noter et rappeler un ou plusieurs événements déjà accomplis dans une séquence d'interactions donnée avec un utilisateur d'un dispositif ou d'un autre élément extérieur. Stateful et Stateless sont dérivés de l'utilisation de l'état comme un ensemble de conditions à un moment donné.

Approche / Critère	Interface	Sémantique	Comportement
Stateful	−	−	+
Stateless	+	+	+
Sémantique	+/−	+	+/−
QoS	+/−	+/−	+/−

TABLE 3.1 – Comparaison des techniques d'appariement de service

Dans le tableau 3.1, nous pouvons voir que les techniques de matching de services basés sur des interfaces, supporte des services stateless mais pas stateful et supporte un peut la description sémantique de services ; parce que les moteurs de recherche ne peuvent matcher que des services simples, donc ne manipulent pas les aspects processus de services. Enfin, les descriptions sémantiques sont assez supporté, puisque des moteurs de recherche comme Woogle (voir [Dong et al., 2004]) a un niveau d'évaluation de la sémantique.

La technique de matching de service basé sur la sémantique supporte les états de services (stateless services) et les descriptions sémantiques. En outre, de telle technique ne supporte pas les services stateful, à cause de cela il est très efficace pour trouver des services simples en faisant un matching de descriptions sémantiques de leurs capacités, mais il n'est pas claire de savoir comment la traduire pour matché des processus métier complexes pour vérifier des bilatérals ou multilatérals collaborations.

Une technique de matching de services basés sur le comportement supporte fidèlement la description sémantique, mais certaines approches comme [Shen et al., 2005] insèrent la sémantique au matching de comportement. En revanche, de telle technique supporte les services stateless et stateful, grâce aux descriptions formelles de services et à la technique de matching utilisée en tenant compte l'ordre dans lequel les services sont exécutés.

3.3 La composition dynamique de services web

Une grande partie de recherche dans la littérature a été faite ces dernières années sur les approches de composition de services web, mais malheureusement il considère que les services web sont primitifs (atomiques) même s'il s'agit des services composites). Dans ce qui suit nous allons présenter certains travaux connexes en les classifiant selon la perception de services impliqués (soit par un processus de réalisation de composition, soit par un autre service qui utilise cette composition).

3.3.1 Vision atomique

Dans ce cas, les services web sont considérés comme des composants atomiques (décrits en termes d'entrées, de sorties, de pré-conditions et d'effets) à enchaînés et intégrés. [Constantinescu et Faltings, 2004] présente un algorithme de chaînage en avant de services pour trouver la composition de services web en enchaînant répétitivement des services web en se basant sur les paramètres d'entrée de la requête. En revanche, le chaînage en arrière de services web introduit par [Aversano et al., 2004] sert de construire la composition de services web en enchaînant répétitivement des services web en se basant cette fois-ci sur les paramètres de sortie de la requête.

Les insuffisances principales de ce genre de travaux de recherches est l'inefficacité de ces algorithmes de composition proposés, puisque ils ne prennent pas en compte ni le comportement de service, ni le comportement de la tâche, ce qui ne garantit

jamais ni les contraintes exigées par les fournisseurs, ni les préférences demandées par les utilisateurs.

La composition de services web à base de tâche d'utilisateur (workflow) a été proposée pour pallier les carences des algorithmes de chainages avant et arrière. [Benatallah et al., 2003, Chakraborty et al., 2005] ont proposent des approches de sélection de services primitifs. Les services sont comparés avec des fonctionnalités (opération abstraits) spécifiées dans la tâche de l'utilisateur.

Dans [Majithia et al., 2004], la requête d'utilisateur est automatiquement mappé à un workflow abstrait. Les instances de service qui correspondent à celles décrites dans le workflow abstrait - en termes de paramètres d'entrées, de sorties, de pré-conditions et d'effets - sont découverts pour constituer une description concrète de workflow. Dans [Hwang et al., 2007], deux stratégies sont proposées pour sélectionner des services web composants qui sont susceptibles d'accomplir avec succès l'exécution d'une séquence d'opérations donné. [Harney et al., 2008] introduit une méthode myope (à courte vue), pour examiner des services pour leurs paramètres modifiés en utilisant la nouvelle valeur de l'information, pour mettre à jour les paramètres du modèle et ainsi pour composer à nouveau les services web en temps réel. Dans [Gamha et al., 2008], les contraintes de l'utilisateur (ou préférences) sont considérées lors de la composition, et sont exprimées comme un ensemble fini de formules logiques spécifiées en langage KIF (knowledge interchange format). Dans [Lecue et al., 2009], la composition de services web est vue comme une composition des liens sémantiques qui font référence à une équivalence sémantique entre les paramètres de service web (i.e., les sorties et les entrées) afin de modéliser leur connexion et leur interaction. [Tang and Zou, 2010] ont présenté une approche pour identifier des patterns (modèles) de composition de services à partir des logs (historiques) d'exécution. Ils désignent un ensemble de services associés en analysant l'ordre d'invocations de services pour récupérer les flux de contrôle entre ces

services. [Ganapathy et al., 2010] propose une approche d'optimisation pour identifier dynamiquement un ensemble réduit de services candidats au moment de la composition. [Wang et al., 2010] présente une approche fondée sur un processus de segmentation progressive en prenant en compte à la fois les limites fonctionnelles de service et l'utilisation de l'historiques de données, pour assurer l'équilibre entre les degrés de satisfaction de ces besoins séquentiels temporellement. [Barhamgi et al., 2010] propose une approche de réécriture de requête pour composer automatiquement des services de données (data-providing services). Les services de données sont décrits comme des vues RDF sur une ontologie médiatrice. Ces vues sont enrichies avec des contraintes sémantiques RDFS et utilisés pour annoter des descriptions WSDL (des paramètres d'entrées et de sorties). L'approche du [Luo et al.,] est basée sur la gestion adaptative de la composition de service avec des propriétés QoS dans des environnements interactifs (grid environments). Les auteurs ont proposé un algorithme heuristique pour sélectionner des services candidats pour la composition. Les auteurs dans [Fan et al., 2010] ont proposé une solution hiérarchique pour la composition dynamique de services à base de réseau de pétri pour caractériser avec précision les préférences d'utilisateur. [Li et al., 2011] introduit une définition formelle pour la similarité hors-context (context-independent similarity), et montre qu'un service web peut être substitué par un autre pair alternatif avec un comportement similaire sans intervenir d'autres services web dans la composition. [Kwon et Lee, 2012] propose un algorithme a deux phases (une phase, de chainage, en avant et une phase en arrière) fondée sur une approche de recherche de composition de services web non-redondants. [Hatzi et al., 2012] introduit une approche en utilisant des techniques d'IA pour adresser la composition automatique de services web.

L'un des inconvénients majeurs des approches de composition de services web à base de tâche utilisateur, c'est qu'il ne satisfait que les préférences d'utilisateur en ignorant les contraintes de fournisseurs.

3.3.2 Vision composite

Récemment, plusieurs issues s'orientent vers des services web composites ont reçu une attention pour l'analyse, la vérification, la sélection et l'agrégation de QoS. Dans ce cas, un service web est perçu comme un composant complexe décrit par un processus métier beaucoup plus complexe.

Les auteurs dans [Xia et al., 2012] ont proposé une approche basée sur la traduction pour l'analyse des processus BPEL, qui utilise un réseau de Pétri stochastique comme une représentation intermédiaire. Ils ont introduit une méthode à base d'état pour calculer le temps de réussite probable pour l'exécution normale du processus (expected-process-normal-completion-time).

Dans [Quan et al., 2012, Jamal et al., 2013], une approche a été présentée qui sépare les comportements de services en des comportements opérationnels et de contrôles, afin de vérifier la conception de services. [Salaün et al., 2012] introduit un codage des diagrammes de collaboration dans l'algèbre de processus LOTOS pour l'analyse de la chorégraphie de services, afin de vérifier les pairs impliqués dans la spécification de chorégraphie qui produisent exactement le même comportement. [Rathore et Suman, 2011] propose un modèle de processus via un QoS-broker pour la composition dynamique de services web. Le QoS-broker support la découverte, en prenant en compte des propriétés de QoS, de services web composites avec l'enregistrement, la vérification, la certification et la confirmation, l'optimisation, la sélection des services individuels pour la réalisation de la composition. Le modèle assure l'utilisation de services offerts avec des paramètres de QoS et aussi il résout des problèmes tels que la mise à jour de service en publiant seulement sa qualité de service dans le registre. Dans [Li et al., 2012], une méthode basée sur le problème de couverture de réseau de Petri (Petri net coverability problem) où une fonction d'utilité de service web est proposée pour assurer la composition automatique de

service. [Sherry et Zhao, 2012] propose une approche à base de décomposition pour la composition de services, dont l'utilité d'un service composite peut être calculée à partir des services composants et les contraintes de services composants peuvent être déduites à partir des contraintes du service composite.

Toutes les approches présentées ci-dessus, suivant la vision composite, traitent que les aspects de vérification, d'optimisation, de sélection ou d'agrégation des paramètres QoS des services web composites et non pas leur composition. Cependant, l'inconvénient de ces approches est que, malgré son importance pour l'amélioration de la composition dynamique de services web, l'aspect comportement de service web n'a pas été profondément étudié et exploité. Nous présentons dans le chapitre suivant notre approche, en exploitant le comportement de services pour la composition dynamique de services web. L'approche proposée est basée sur la reconfiguration et l'enrichissement de processus de service pour la combinaison flexibles de services web au moment de l'exécution. On suppose que les services web offerts et requis sont décrits par des comportements beaucoup plus complexes, afin d'assurer à la fois les contraintes de fournisseurs et les besoins des utilisateurs.

3.3.3 Évaluation des techniques de composition

Nous présentons dans cette section un tableau synthétisant les principales caractéristiques des approches de composition dynamique de services web. Le tableau récapitule les différentes approches de composition dynamique de services web en spécifiant un certains nombre de critères afin de mettre en évidence les points forts et les points faibles de chaque approche, et également de bien positionner notre approche :

- **Préférences d'utilisateurs** : la capacité de spécification des préférences des utilisateurs afin de permettre une réalisation de la composition comme il est souhaité par son utilisateur.

- **Contraintes de fournisseurs** : la capacité de spécification des contraintes de fournisseurs afin de permettre une consommation de services comme il est exigé par leur fournisseur.
- **Sélection** : la composition par le biais de sélection de services composites.
- **Intégration** : la composition par le biais d'intégration de services composites.
- **Chevauchement** : la composition par le biais de chevauchement de services composites.
- **Sémantique** : la capacité de spécification des descriptions sémantiques autant explicites et précises.
- **QoS** : la prise en compte des caractéristiques paramétriques non fonctionnelles comme (par exemple Temps de Réponse, Disponibilité, Fiabilité, etc.).

Le tableau 3.2 résume les techniques de composition dynamique de services présentés. Le symbole $(+)$ signifie que la propriété sur cette colonne est supporté par la technique de composition. Le symbole $(-)$ signifie que la propriété n'est pas prise en charge. Le signe $(+/-)$ signifie que la propriété est moyennement supporté. Enfin, le symbole (N/A) signifie non applicable, c.-à-d. la propriété ne s'applique pas à la technique avec laquelle elle a été associée.

3.4 Conclusion

Les services web sont actuellement très utilisés par les entreprises pour rendre accessible par le réseau leurs données et leur savoir-faire. Cette technologie émergente facilite également la collaboration inter-entreprise et intra-entreprise. Ce processus d'intégration et de combinaison de plusieurs services web est fait appel à la composition de services. La composition de services est une approche de développement, très populaire dans l'industrie logicielle, qui se concentre sur la réalisation de modèles abstraits plutôt que sur des concepts algorithmiques. La phase de spécification de services est donc particulièrement importante dans une approche de composi-

Approche / Critère	Vision Atomique Á Base de		Vision Composite
	Chaînage	Tâche	
Préférences d'utilisateurs	−	+	N/A
Contraintes de fournisseurs	−	−	N/A
Sélection	−	−	+/−
Intégration	−	−	−
Chevauchement	−	−	−
QoS	+/−	+/−	+/−
Sémantique	+/−	+/−	+/−

TABLE 3.2 – Comparaison des approches de composition de services

tion et représente une partie conséquente du cycle de développement. Cela permet aux développeurs de se concentrer sur le comportement souhaité du service, sans se soucier de la manière de l'implémenter. Pour mettre en oeuvre cette collaboration, de nombreux langages ont été proposés pour la description et l'exécution de services composés. Cependant, les langages de composition proposés manquent de formalisme puis qu'ils sont simplement décrits en prose. Il n'est donc pas possible d'appliquer directement de méthodes mathématiques sur ces langages, rendant ainsi la composition difficile voir inappropriée pour ne pas dire impossible. Ces langages supposent que les services composants impliqués par un processus de composition sont atomiques. La description comportementale des services composants est très importante puisqu'elle permet de préciser les différentes interactions avec les services pour réaliser des descriptions plus clairs, aidant à la compréhension globale des services offerts. Ce genre de descriptions modélise précisément le comportement

de services et permet de s'assurer, notamment auprès des clients, que celui-ci répond bien aux attentes et réalise bien le besoin demandé. De plus, cette description est généralement suffisamment expressive pour servir de base à l'avalisation des contraintes d'utilisation de services exigées par leur fournisseur.

A travers de ce qui a été présenté au niveau des chapitres 2 et 3, il ressort que, la composition dynamique de services web dépend donc de quatre éléments primordiale : il s'agit de la description de services que ce soit offerts ou requis, la représentation formelle de services qui automatise et facilite le processus d'appariement de services pour leur sélection et leur combinaison, l'algorithme d'appariement adopté qui met en correspondance la description des services offerts avec la description de celui requis, et le processus de combinaisons qui permet de récupérer les descriptions concrètes des services choisis lors du processus d'appariement, et de procéder aux différentes combinaisons de ces services pour réaliser le service requis. En outre, après avoir étudié les différents travaux de recherche liés à la sélection et la composition des services web, un certain nombre de besoins doivent être spécifiés explicitement au niveau des descriptions des services afin de bien guider et simplifier de plus en plus un processus de composition. Parmi ces besoins on peut citer : la nécessité à la fois d'une description sémantique ainsi que d'une signature comportementale. En effet, des descriptions fonctionnelles par le biais des paramètres d'entrées et de sorties, comme celles offertes par des descriptions WSDL d'un service web, sont insuffisantes pour faire guider d'une manière appropriée un processus de composition de services, et également de bien garantir des contraintes de fournisseurs ainsi que des préférences d'utilisateurs. A titre d'exemple, la description d'un service par plusieurs sous-services (opérations), et des signatures (paramètres d'entrées et de sorties) ne détermine pas l'ordre de leurs invocations pour réaliser une fonctionnalité donnée. C'est pour quoi, il devient nécessaire d'avoir, en plus de cette description, une description comportementale, par le biais de coordination des

opérations qui détermine leur ordre d'exécution, selon les exigencies du fournisseur ou les preferences de l'utilisateur. D'autre part, une description comportementale syntaxique est insuffisante puisque des descriptions comportementales offertes par les services peuvent être hétérogènes ce qui peut expliquer la présence de multiples conflits d'ordre sémantique qui peuvent exister et qui doivent être adressés afin d'assurer une bonne composition de services.
Pour notre travail, nous proposons une approche mettant en oeuvre à la fois la sélection et la combinaison guidée par la description comportementale sémantique de services ainsi que la représentation formelle pour la composition approprié et flexible de services au moment d'exécution. Notre approche vise à favoriser l'assouplissement de la composition des services déjà composés et également la collaboration inter-organisationnelles, par l'adoption des AEF comme modèle de représentation formel, et $OWL-S$ comme langage de description sémantique.

Dans ce chapitre :

- Nous avons expliqué les différentes techniques de matching pour la composition de service web. Comme notre approche de composition de services est fondée sur la matching comportemental, nous avons présenté un état de l'art des différentes techniques de matching de services. Nous avons regroupé les techniques existantes en trois ensembles : le matching de service à base d'interface, de sémantique et de comportement. Enfin, une évaluation des techniques existantes a été fournit. Les approches à base d'interface de service web ne peuvent considéré que des services simples, donc ne manipule pas les aspects d'exécution de services. Les approches fondées sur la sémantique sont très efficaces pour le matching des services simples basés sur des descriptions sémantiques de leurs capacités. D'ailleurs, il n'est pas clair comment ces approches peuvent faire le matching des processus complexes, qui prennent en compte l'information structurelle des processus compris dans les services.

– Enfin, nous avons détaillé les travaux connexes à l'égard des techniques de composition dynamiques de service web. Comme notre approche de composition de services est fondée sur l'aspect comportemental, nous avons présenté un état de l'art des différentes techniques de composition de services. Nous avons regroupé les approches existantes en deux ensembles : la vision, de service, atomique et la vision composite. Enfin, une évaluation de ces approches a été donnée à la fin du chapitre.

Chapitre 4

Ws-BeC : Composition Personnalisée de Services Web

Sommaire

4.1 Introduction

Le SOC est un nouveau paradigme de traitement où des services sont utilisés comme des briques de base pour soutenir le développement d'une composition (rapide, de faible coût, interopérable, évolutive et facile) des applications massivement distribués. L'objectif principal du paradigme service web est de réaliser l'interopérabilité entre des applications hétérogènes et distribués. Un service web a été identifié comme l'élément de base de développement pour la génération future des solutions métiers basées sur le web, qui est caractérisé par l'indépendance d'application, de plate-forme, et de fournisseur. Depuis son émergence, de nombreux spécialistes ont prédit que le service web va révolutionner le paradigme de l'informatique distribuée. Un des aspects les plus importants d'utilisation des technologies de services web est la composition de services web pour créer des services à valeur ajoutée. La composition de services web est une nouvelle méthodologie pour construire des applications illimités à valeur ajoutée en agrégeant un ensemble existant de composants de service limité ainsi que selon les besoins dynamiques de métiers, qui a le potentiel de réduire l'effort et le temps de développement pour de nouvelles applications.

Ce chapitre est consacré à la description détaillée de notre approche. 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 proposé, à savoir : la formalisation et la description comportementale de la composition, l'appariement et la sélection fonctionnelle et extra fonctionnelle de services, la combinaison flexible et appropriée de services, et enfin la transformation et la concrétisation de la composition sous forme d'un fichier de description d'orchestration très riche.

4.2 Présentation de l'approche

Notre approche est basée sur l'idée qu'un service web obtenu par un mécanisme de composition peut être à son tour impliqué une nouvelle fois dans d'autres opérations de compostions. Ce genre de services oblige d'exploiter toute leur richesse afin d'augmenter de plus en plus la chance de réussite de la composition dans laquelle ils sont impliqués. Ceci nous a poussé à proposer une approche à fin de permettre un enrichissement comportemental de services en analysant et en reconfigurant leurs processus métiers au moment de la conception. Cette nouvelle description comportementale sera exploitée par la suite au moment d'exécution pour favoriser la combinaison flexible (par sélection, intégration et chevauchement) tout en garantissant à la fois les contraintes de leur fournisseur ainsi que les préférences des utilisateurs.

Notre approche s'appuie donc sur l'analyse des processus métiers de service pour l'enrichissement comportemental afin de fournir un mécanisme de composition de services web fiable, flexible et approprié.

4.2.1 Motivations de la contribution

La *QoS* est un élément important des attributs non-fonctionnels de services, qui décrit l'utilité et la fiabilité de service et joue un rôle important dans la composition dynamique de services multiples. Avec la croissance du nombre de services offrants des fonctionnalités similaires disponibles sur le web, il est nécessaire d'être en mesure de les distinguer en utilisant un ensemble de critères de qualité de service (QoS) bien définis.

Dans ce document, nous considérons l'aspect comportemental de service pour la composition de services web. Malgré son importance, il n'est pas vraiment traité jusqu'à présent. Nous faisons référence à la composition de services web dirigée par des contraintes comportementales par une composition comportementale appropriée.

Par exemple, (i) un fournisseur de services web peut ne pas accepter les demandes pour des utilisations partielles d'un service donné. (ii) un client de service web peut exige que les services soient du même fournisseur pour un besoin donné. Ce genre de contraintes doit être soigneusement considéré lors du processus de composition de services web. Pour assurer une composition appropriée de services web, les services de différents fournisseurs doivent être vérifiés et analysés afin de : (i) satisfaire convenablement les contraintes des fournisseurs, (ii) bien rependre aux besoins des utilisateurs, (iii) ainsi que, améliorer la chance de réalisation des tâches des utilisateurs. Plus précisément, nous analysons la compatibilité comportementale de services web participants pour permettre une utilisation/combinaison la plus appropriée de services. La compatibilité comportementale signifie que les services web interactifs se coïncident en termes d'opérations à exécuter, des résultats à fournir, et des messages à échanger (à envoyer et à recevoir). Un comportement de service se réfère aux séquences de messages que des services web peuvent échanger mutuellement afin de rependre à des besoins ou des dépendances métiers bien déterminés [Li et al., 2011]. Dans cette thèse, nous nous concentrons sur le comportement de services web, et nous adoptons un matching comportemental flexible pour la sélection, l'intégration et le chevauchement adéquats pour résoudre le problème de la personnalisation et de la composition dynamique de services web.

Ce travail à pour but de favoriser la réalisation au moment d'exécution des besoins des utilisateurs. Notre solution assure la réalisation de la tâche de l'utilisateur, et l'utilisation convenable des services web par le biais des améliorations suivantes [Mekour et Benslimane, 2013] :

- La manipulation flexible du comportement de service web (la sélection, l'intégration et/ou chevauchement), ainsi que la compatibilité des flux de contrôle pour permettre l'exploitation complète de services disponibles au moment de composition,

- La prise en compte de tous les processus de service de fournisseurs comme primitives (non décomposables), qui doivent être invoqués en entier (telqu'ils sont spécifiés par leurs fournisseurs),
- La capacité aux utilisateurs de spécifier leurs processus primitifs requis selon leurs préférences.

4.2.2 Les grammaires formelles

La description dans un langage formel du comportement d'un service web permet l'assouplissement de l'opération d'appariement des services web, par le biais des opérations d'intersection et de fusion, et également du processus de composition en entier. Un certain nombre de solutions ont été proposées pour formaliser le comportement de service en utilisant l'algèbre de processus. Dans notre travail, nous avons besoin de représenter le processus métier qu'offre un service web, sans pour autant modéliser certains détails tels que les canaux de communications, et les évènements internes des processus. Le processus métier d'un service web est ainsi vu comme le langage publié par ce service dans lequel les opérations constituent l'alphabet de base. Un comportement cohérent d'un service web doit donc être un mot reconnu par le langage du service, en invoquant dans un certain ordre bien déterminé des opérations offertes par ce service. Les automates d'états finis et également leur représentation grammaticale qui permet la représentation des langages sont tout à fait appropriés pour générer le processus de service.

Definition 4.2.1 (Grammaire)**.** *Une grammaire G est définit comme un quadruplet $G = (T, N, P, S)$, avec :*

- T est un ensemble fini de symboles terminaux : l'alphabet (les lettres) sur lequel le langage généré est défini. Les terminaux notés conventionnellement par des minuscules

- N est l'ensemble des symboles non-terminaux. Les non-terminaux notés conventionnellement par des majuscules.
- $P \subseteq (V^+ \times V^*)$ où $V = T \cup N$ est un ensemble fini de règles de productions (ou règles de réécriture). Les règles sont donc de la forme $\alpha \longrightarrow \beta$ où $\alpha \in V^+$ et $\beta \in V^*$. α est appellée partie gauche (source) de la production, et β partie droite (cible) de la production.
- $S \in N$ appelé Axiome, est le symbole de départ.

D'autre part :

- L'opérateur | représente le choix entre deux règles de productions.
- L'opérateur $\bullet$ représente la concaténation de deux symboles.

Par conséquence

- La règle de production $\alpha \longrightarrow \beta \mid \gamma$ représente le choix entre les deux règles de productions $\alpha \longrightarrow \beta$ et $\alpha \longrightarrow \gamma$.
- La règle de production $\alpha \longrightarrow \beta \bullet \gamma$ représente la concaténation des deux symboles β et γ. Pour des raisons de simplification d'écriture l'opérateur $\bullet$ peut être supprimé ainsi les productions $\alpha \longrightarrow \beta \bullet \gamma$ et $\alpha \longrightarrow \beta\gamma$ sont également équivalentes.

4.2.3 Méthodologie

Notre approche de composition comportementale de services web se déroule en deux étapes fondamentales (voir Figure 4.1) :

- Une étape de description comportementale pour la représentation riche de services. Cette étape débute par une phase d'analyse des descriptions de services que ce soit concrètes ou abstraites (tâches à réaliser) pour l'extraction des caractéristiques comportementales de services, suivie d'une phase de reconfi-

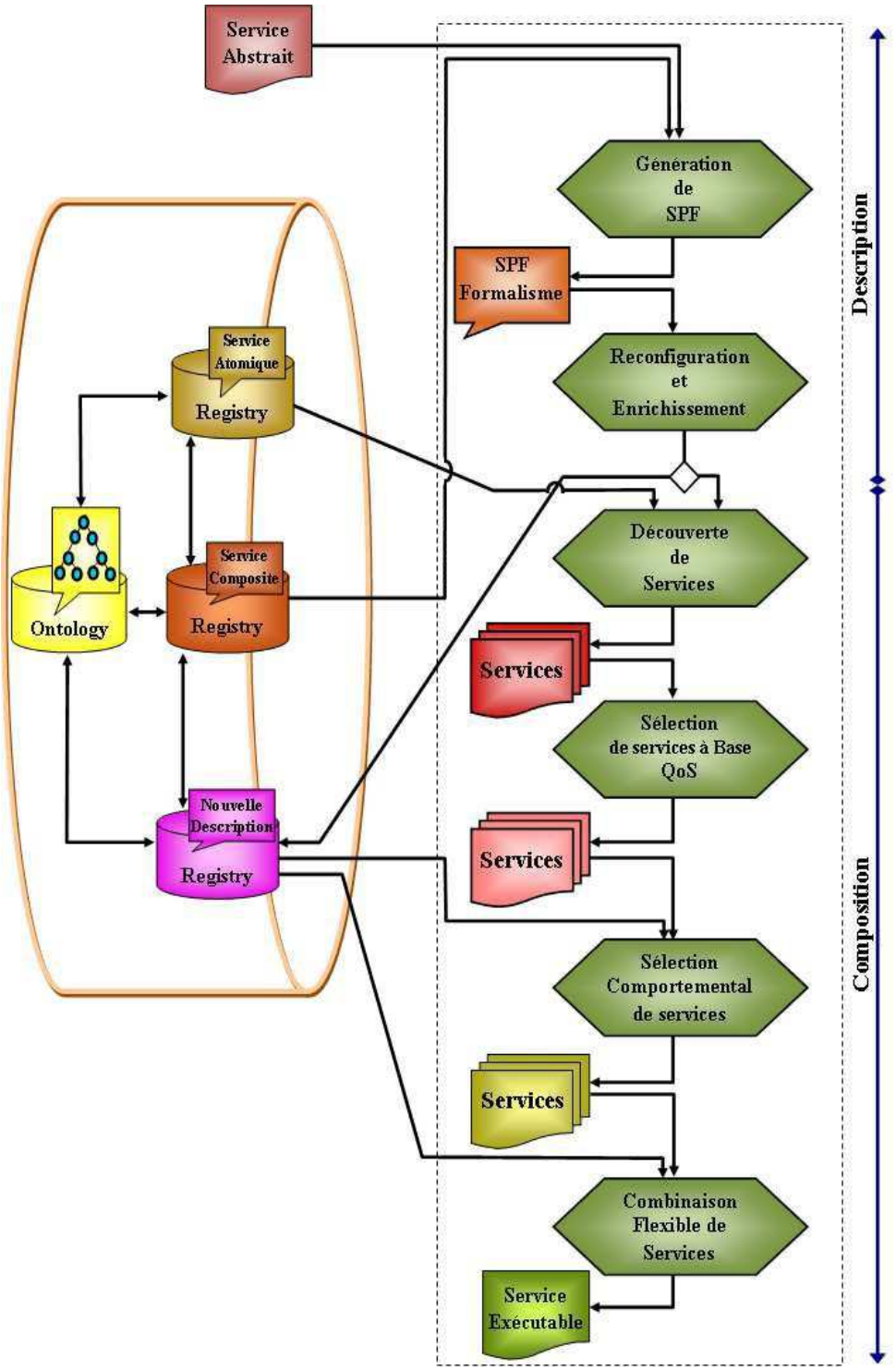


FIGURE 4.1 – Processus de composition personnalisée de services web avec des propriétés comportementales

guration et d'enrichissement de leur comportement pour par la suite stocker dans un annuaire afin d'être trouvés dynamiquement [Mekour et Benslimane,

2011a].

– Une étape de composition pour la construction de fichier d'orchestration en adoptant une combinaison appropriée et flexible des services disponibles. Cette étape effectue d'abord une phase de découverte de sévices en se basant sur leur signature fonctionnelle, suivie d'une phase de sélection non fonctionnelle en se basant sur leurs paramètres de QoS pour minimiser le nombre de service candidat à la composition. Une autre phase de sélection non fonctionnelle en se basant cette fois-ci sur leurs caractéristiques comportementales pour satisfaire à la fois des contraintes fournisseurs ainsi que des préférences utilisateurs, ce qui minimise de plus en plus, le nombre de services candidats à la composition. Une dernière phase de combinaison flexible de services candidats afin de rependre à un besoin beaucoup plus complexe d'une manière fiable et plus appropriée [Mekour et Benslimane, 2011b, Mekour et Benslimane, 2013].

4.3 Description comportementale de service

Dans notre approche, nous considérons que les descriptions des services web sont enrichies d'informations sémantiques décrivant la fonctionnalité des services et les données qu'ils échangent. De plus, une description du processus métier offerte par les services est également spécifiée. Le processus d'un service web est une spécification de l'ordre dans lequel les opérations offertes par le service web doivent être exécutées. Cet ordre d'exécution est spécifié en utilisant des opérateurs de construction telles que les structure itératives (boucles) ou conditionnelles. Chaque opération correspond à une opération concrète (WSDL) à laquelle on ajoute des informations sémantiques sur la fonctionnalité de cette opération et les données qu'elle échange.

Nous présentons ci-après notre approche pour décrire le comportement (processus) de services web. L'approche proposée est basée sur la reconfiguration et la combinaison flexible des processus pour une composition de services web au moment

d'exécution. Afin d'améliorer la réalisation dynamique des tâches, nous considérons l'exploitation valide et flexible des services disponibles. Pour satisfaire les contraintes de fournisseurs et de préférences d'utilisateurs.

Dans ce travail, nous supposons qu'un service fourni et celui requis peuvent être complexes et peuvent être décrits par un comportement (processus) beaucoup plus complexe. Le comportement du service fait référence à un ensemble de messages échangés mutuellement par les services web en rependant à certain objectif bien déterminé décrits par un processus métier. En d'autres termes, le comportement du service web est l'ensemble des relations temporelles entre les opérations de service nécessaires pour une interaction avec le service. Le comportement de service est décrit par un ensemble de processus offert par un service. Le processus est un ensemble d'opérations (services atomiques) reliés entre eux par des flux de contrôle [Mekour et Benslimane, 2013].

4.3.1 Formalisation de processus de service

La composition de service tente de sélectionner et d'interconnecter des services offerts par les différents fournisseurs de services sur la base d'un processus de métiers spécifié. Le processus métier du service composite peut être exactement représenté par un workflow pour les services web, qui définit la dépendance potentielle de données entre les tâches. Les structures de contrôle les plus fréquemment utilisés pour l'orchestration de services notamment : *sequence, loop, parallel and choice.*

Pour fournit une perspective détaillé sur un comportement de service, ce dernier peut être perçu comme un processus. Dans cette thèse, un processus peut être atomique ou composite. Les services décrits par un processus atomique / composite sont appelés services simples / composites respectivement. Un processus atomique prévoit un message en entré et renvoie un message en sortie. Un processus composite est celui qui maintient un certain nombre d'états où chaque message envoie des

progrès à travers le processus. En d'autres termes, un processus composite est décrit par plusieurs processus (peuvent être atomiques et/ou composites) interconnectés par des flux de contrôle. Un flux de contrôle est un constructeur comportemental qui fait partie de la description du processus. Il peut être un constructeur de *séquence, non-ordonné, de choix externe, de parallélisme, de synchronisation, ou un constructeur itérative*, etc. Au moment d'exécution de processus, plusieurs chemins d'exécution sont possibles. Chaque chemin est un scénario alternatif qui décrit un sous-processus appelé un processus alternatif.

Toutefois, la représentation du comportement de service doit être fondée sur un modèle formel, afin : (i) de simplifier et de favoriser l'appariement comportementale et la composition flexible, (ii) et aussi, pour améliorer la réalisation d'une tâche utilisateur. En adoptant les grammaires formelles, nous développons le formalisme SPF (Service Process Formalism) comme une infrastructure de description comportementale de services web. Le SPF est décrit par des processus atomiques, des processus composites, et des flux de contrôle. Un processus composite (cp) est décrit par une règle de production (pr), dont la source représente le processus composite, et la cible représente les processus composants. Les constructions itératives, sont décrites à base de deux règles de production pour représenter les comportements répétitifs :

1. $cp \rightarrow p \bullet p \bullet p \cdots \bullet p$, pour un comportement répétitif avec un nombre connu d'itérations,

2. $cp \rightarrow cp \bullet p|p$, pour un comportement répétitif avec un nombre inconnu d'itérations.

Definition 4.3.1 (Formalisme de processus de service)**.** *Un SPF est un quintuple* $SPF = (\sum, T, N, P, S)$*, où :*

– $\sum$ *est un ensemble fini et non vide de tous les éléments d'un service web qui*

décrivent un processus de service. Plus formellement,

$$\sum = \{x \mid (x \in AP) \vee (x \in CP) \vee (x \in CF)\} \tag{4.1}$$

Où AP désigne un ensemble fini et non vide de tous les processus atomiques d'un service web, CP représente un ensemble fini de tous les processus composites d'un service web, et CF désigne un ensemble de tous les flux de control (sauf ceux qui sont itératifs) utilisés dans une description de service web ($\bullet, *, |, \|, \Vdash$) et qui représentent respectivement le flux de contrôle sequence, unordered, external choice, parallel et synchronization.

- *T est un ensemble fini et non vide d'éléments terminaux qui représentent tous les flux de contrôle et les processus atomiques d'un service web. Plus formellement,*

$$T = \{x \in \sum \mid (x \in CF) \vee (x \in AP)\}$$

- *N est un ensemble fini d'éléments non-terminaux qui représentent tous les processus composites d'un service web. Plus formellement,*

$$N = \{x \in \sum \mid x \in CP\}$$

- *P est un ensemble de toutes les règles de production des processus composites. (pr). Plus formellement,*

$$P = \{pr \mid \forall x \in N, \exists(\rho, \omega) \in CF \times (N \cup AP)^+, pr : x \to \rho\omega\}$$

- *S est un axiome qui représente le processus principal d'un service web. C'est le seul processus composé récursivement par tous les autres processus (p) de même service (en d'autre termes, c'est le processus qui n'apparaitre à aucune*

partie cible d'aucune règles de production). Plus formellement,

$$S \in \{x \in \sum |\forall(\rho,(y,z),(\alpha,\beta)) \in CF \times N^2 \times [(N \cup AP)^*]^2, \exists! x \in N \\ (y \to \rho\alpha z\beta) \wedge (x \neq z)\}$$

L'algorithme de représentation formelle du comportement de service (voir Algorithm 1) génère le formalisme SPF à partir de la description de processus principal de service (SPD). Le SPD est le fichier de description d'une composition qui contient la description fonctionnelle (les entrées et les sorties, les pré-conditions et les effets), ainsi que la description comportementale (les processus et les flux de contrôle).

A titre d'illustration, nous considérons le scénario d'organisation de voyage illustré par la Figure 4.2. Le processus principal *Organisation de Voyage* est une séquence de deux autres processus composites : *Réservation de Transport*, et *Réservation d'Hôtellerie*. Le premier est une séquence de trois processus : le processus composite *Sélection d'Itinéraire*, et les deux processus atomiques *Acquisition de Ticket*, et *Paiement*. Le processus composite *Sélection d'Itinéraire* est un choix alternatif de quatre processus atomiques : *Sélection de Vol, Sélection de Bateau, Sélection de Metro, et Sélection de Bus*. Le deuxième est une séquence de deux processus atomiques : *Acquisition de Chambre*, et *Paiement*.

- Soient A, B, C, D les processus composites *Organisation de Voyage, Réservation de Transport, Réservation d'Hôtellerie et Sélection d'Itinéraire* respectivement.
- Soient a, b, c, d, e, f, g les processus atomiques *Acquisition de Ticket, Paiement, Acquisition de Chambre, Sélection de Vol, Sélection de Bateau, Sélection de Metro et Sélection de Bus* respectivement.

En se basant sur les définitions présentées ci-dessus, nous constituons le formalisme de processus de service SPF comme suit :

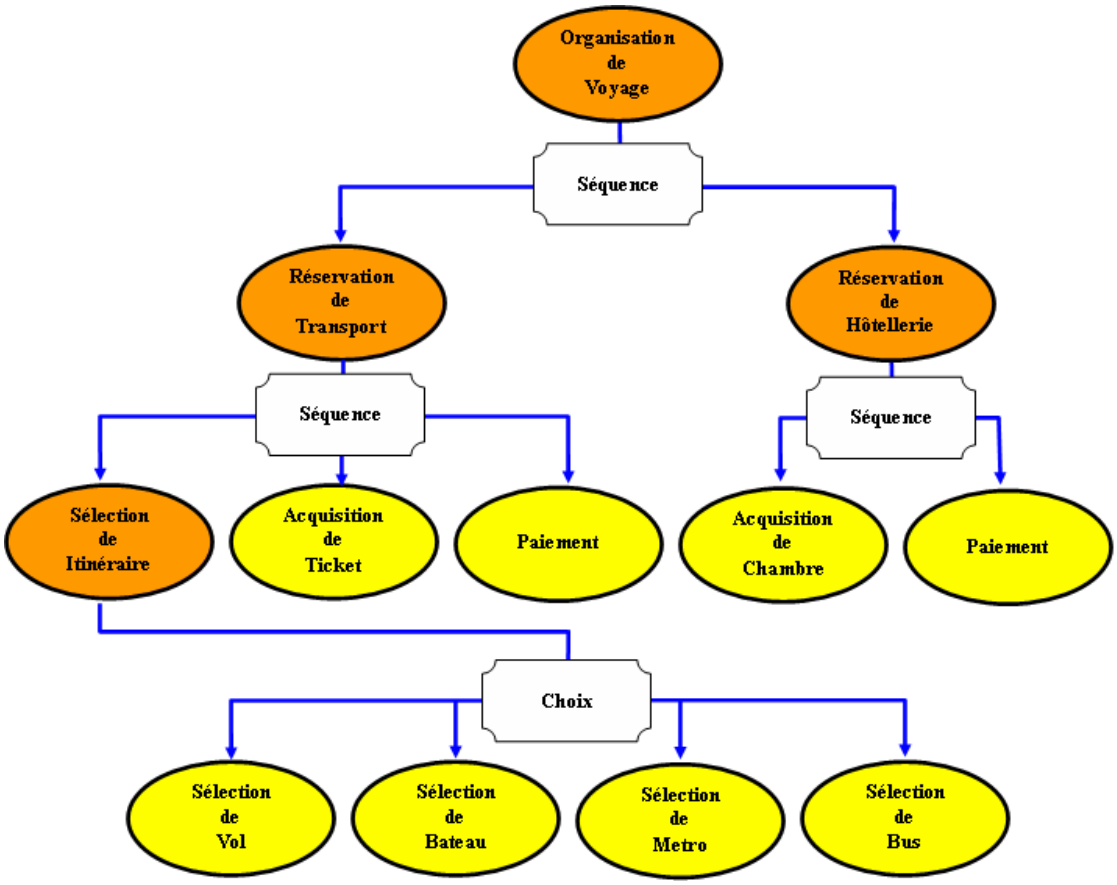


FIGURE 4.2 – Processus de service d'organisation de voyage

– L'ensemble fini et non vide AP.

$$AP = \{a, b, c, d, e, f, g, \bullet, |\}$$

– L'ensemble fini CP.

$$CP = \{A, B, C, D\}$$

– L'ensemble fini CF.

$$CF = \{\bullet, |\}$$

– L'ensemble fini et non vide $\sum$.

$$\sum = \{A, B, C, D, a, b, c, d, e, f, g, \bullet, |\}$$

– L'ensemble fini T.

$$T = CF \ \mathbf{U} \ AP$$
$$= \{\bullet, |, a, b, c, d, e, f, g\}$$

– L'ensemble fini et non vide N.

$$N = \{A, B, C, D\}$$

– L'ensemble fini et non vide P .

$$P = \{A \rightarrow B \bullet C,\ B \rightarrow D \bullet a \bullet b,\ D \rightarrow d \mid e \mid f \mid g, C \rightarrow c \bullet b\}$$

– L'axiome S.

$$S = A$$

Par conséquent, le SPF sera comme suit :

$$SPF(\{A, B, C, D, a, b, c, d, e, f, g, \bullet, |\}, \{A, B, C, D\},$$

$$\{\bullet, |, a, b, c, d, e, f, g\}, \{A \rightarrow B \bullet C,\ B \rightarrow D \bullet a \bullet b, \tag{4.2}$$

$$D \rightarrow d \mid e \mid f \mid g,\ C \rightarrow c \bullet b\}, A)$$

4.3.2 Représentation de processus de service

Une fois que le SPF est défini, les différents scénarios comportementaux de service peuvent être engendrés par la substitution des éléments non-terminaux par leurs règles de productions. Pour permettre la production de tous les scénarios qui

Algorithm 1: Formalisme de processus de service

Input : SPD
Output: SPF ;

$N \leftarrow \varnothing$; $T \leftarrow \varnothing$; $P \leftarrow \varnothing$;
for All cp **In** SPD **do**
  $\rho \leftarrow$ `GetCF(`cp`)` ; `/* GetCF gets the cp control flow */`
  `ProcessList` $\leftarrow$ `GetPL(`cp`)` ; `/* GetSL gets the list of all cp processes */`
  $p \leftarrow$ `ProcessList(`1`)` ; `/* ProcessList(1) is the first process of the list */`
  switch ρ **do**
    case *Construct with known iterations times :*
      $P \leftarrow P \cup \{cp \rightarrow p \bullet p \ldots \bullet p\}$;
    case *Construct with unknown iterations times :*
      $P \leftarrow P \cup \{cp \rightarrow cp \bullet p | p\}$;
    otherwise
      $P \leftarrow P \cup \{cp \rightarrow$ (`ProcessList(`1`)` ρ `ProcessList(`2`)` $\ldots \rho$ `ProcessList(`n`))`$\}$;
  $AP \leftarrow$ `GetAS (ProcessList)` ; `/* GetAS gets all atomic processes set */`
  $T \leftarrow T \cup \{\rho\} \cup AP$;
  $N \leftarrow N \cup \{cp\}$;
$S \leftarrow$ `GetMS(`P`)` ; `/* GetMS gets the main process */`

peuvent être fournis par un comportement de service, le processus de substitution doit être lancé à partir de l'axiome S qui représente le processus principal de service.

Cependant, le nombre d'itérations pour certains flux de contrôles itératifs (constructeurs itératifs) comme *iterate, repeat-while, repeat-until* est connue qu'après l'exécution effective soit terminée, ce genre de contraintes empêche le processus de génération de scénarios.

Les auteurs dans [Canfora et al., 2005, Zeng et al., 2004] ressoudent ce défi par l'adoption des techniques de prédiction. Ces techniques sont basées sur l'historique d'invocation de service pour prévoir le nombre d'itérations. Cependant, ce genres d'anticipation n'est pas vérifié tout le temps, et peut varié d'une invocation à une autre, selon certains facteurs comme par exemple les entrées, les pré-conditions, voir

même le service impliqué. En outre, le choix du service approprié est plus important par rapport à l'estimation du nombre d'itérations. Ainsi, deux services sont dits équivalents, s'ils ont le même nombre d'itérations sous les mêmes contraintes d'invocation. Par conséquent, pour résoudre ce problème, nous avons conservé la règle de production $cp \rightarrow p$ du constructeur itératif avec un nombre d'itérations inconnus, avec laquelle on peut effectuer le processus de substitution des non-terminaux. L'algorithme suivant (voir Algorithm 2) formalise le processus de représentation du comportement de service.

Par exemple, si nous reprenons le SPF ci-dessus (voir équation 4.2), alors nous avons initialement la règle : $A \rightarrow B \bullet C$. Après la substitution des règles de production du non terminale B, nous trouvons $A \rightarrow D \bullet a \bullet b \bullet C$. Ensuite, après avoir substitué les règles de production du non terminal D, nous trouvons les quatre règles de production suivantes :

$$A \rightarrow d \bullet a \bullet b \bullet C \mid e \bullet a \bullet b \bullet C \mid f \bullet a \bullet b \bullet C \mid g \bullet a \bullet b \bullet C$$

Ainsi que, après la substitution des règles de production du non terminal C, nous trouvons la règle globale GPR suivante qui contient seulement des processus atomiques.

$$A \rightarrow d \bullet a \bullet b \bullet c \bullet b \mid e \bullet a \bullet b \bullet c \bullet b \mid f \bullet a \bullet b \bullet c \bullet b \mid g \bullet a \bullet b \bullet c \bullet b \tag{4.3}$$

4.3.3 Enrichissement de la description de processus de service

Le comportement de service est décrit par une seule règle de production globale (GPR), qui représente tous les différents scenarios pouvant être fournits par un web service. La règle (GPR), peut être munie par un ensemble de règles de production pour des constructeurs comportementaux avec un nombre d'itération inconnue. La

règle (GPR), ne contient récursivement aucun élément non-terminal (c.-à-d. aucun processus composite) ce qui permet de simplifier la reproduction de tous les scénarios pouvant être fournits par un service web. Pour simplifier encore le processus de matching comportemental de services, ainsi que pour générer tous les comportements alternatifs pouvant être offerts par un service, nous nous basons sur l'opérateur de choix externe, choice construct, "|" pour decomposer la règle GPR en un ensemble de comportement de services alternatifs (SB). Ensuite chaque SB est raffiné, adapté, et réorganisé pour le décomposer à son tour en un ensemble d'unité de conversation homogène dite CU.

Definition 4.3.2 (Unité de conversation)**.** *CU représente un ensemble fini de processus p_i interconnectés par un même flux de contrôle δ. Un processus dans un CU peut être atomique ou composite.*

Pour mieux représenter une CU, nous adoptons une notation préfixée comme suit :

$$CU = \delta p_1 p_2 \dots p_m \tag{4.4}$$

Par exemple, lorsque nous reprenons la régle GPR précédente (voir équation 4.3).Chaque fois que nous nous concentrons sur le flux de contrôle de choix externe "|", nous serons capables de diviser cette GPR en quatre comportements alternatifs de service SBs. Dans cet exemple, chaque SB se compose de cinq services atomiques interconnectés par le même flux de contrôle "•".

$SB_1 = d \bullet a \bullet b \bullet c \bullet b$,

$SB_2 = e \bullet a \bullet b \bullet c \bullet b$,

$SB_3 = f \bullet a \bullet b \bullet c \bullet b$,

$SB_4 = g \bullet a \bullet b \bullet c \bullet b$.

Par conséquent, Si on reprend l'exemple précédent, puisque chaque SB se com-

Algorithm 2: Service Process Rewriting

Input : P ; /* Set of production rules */
Output: GPR ; /* Global production rule that describe the service behavior */

1 **for All** pr **In** P **do**
2 **repeat**
3 $CanSubstitute \leftarrow$ False;
4 $CPSet \leftarrow$ `GetCP(`pr`)` ; /* $GetCP$ extracts all pr composite processes */
5 $\rho \leftarrow$ `GetCF(`pr`)` ; /* $GetCF$ gets cp control flow described by pr */
6 **for All** cp **In** $CPSet$ **do**
7 $\rho^{'} \leftarrow$ `GetCF(`cp`)` ; /* $GetCF$ gets the cp control flow */
8 **if** $((\rho^{'} = \rho) or (\rho^{'} = \mid))$ **then**
9 $pr \leftarrow$ `SubtCP` (cp, p); /* $SubtCP$ substitutes all cp production rules in pr */
10 $P \leftarrow$ `Delete` (cp, p) ; /* $Delete$ removes all cp production rules from P */
11 $CanSubstitute \leftarrow$ True;
12 **until** ($CanSuptitute$ =*False*);
13 $GPR \leftarrow$ `Replace(`P`)`; /* $Replace$ substitutes all production rules in the main rule */

pose d'une seule unité de conversation CU, alors les différentes CUs peuvent être trouvées, comme suit :

Les CU_s du SB_1 : $CU_1 = \bullet dabca$,

Les CU_s du SB_2 : $CU_2 = \bullet eabca$,

Les CU_s du SB_3 : $CU_3 = \bullet fabca$,

Les CU_s du SB_4 : $CU_4 = \bullet gabca$.

La description de service sera donc, en plus de sa description fonctionnelle et non fonctionnelle préexistante, une annotation par les différentes SB ainsi que leurs CU associées. Ce qui favorise le mécanisme de sélection, d'intégration et de chevauchement des SB offerts et leurs CU associées. Il augmente également la chance de

réalisation de la composition.

4.4 Appariement de processus de services

L'algorithme d'appariement que nous proposons est un algorithme de maching sémantique et comportemental. Cela signifie que les descriptions comportementales publiées par les services web sont utilisées pour reconstituer la tâche d'utilisateur, et que les informations sémantiques présentées dans les descriptions des services web et la tâche à réaliser sont utilisées pour réconcilier les données échangées, et les fonctionnalités offertes. En d'autres termes, l'objectif est de reconstituer la séquence d'échange de messages qu'effectue la tâche pour réaliser une fonctionnalité donnée, en sélectionnant et en intégrant les différentes séquences d'échanges de messages publiées par les services web. Il existe de nombreuses approches pour l'appariement comportemental, cependant, la plupart de ces approches supposent d'avoir une description abstraite des services web impliqués dans la composition. Lors de l'exécution du service composite, des instances, services, concrètes décrits sont recherchées. En d'autres termes, elles sont basés sur la sélection des services perçus comme atomiques même s'il s'agit des services composites pour répondre à une opération abstraite décrite dans un fichier d'orchestration conçu statiquement au moment de la conception. Dans notre approche, nous ignorons le nombre de services à utiliser pour reconstituer dynamiquement la composition. La même tâche peut être reconstituée en utilisant des services différents dans deux instants de composition. Ainsi, La même tâche peut être reconstituée par des configurations différentes dans un même instant de composition par le biais de processus de combinaison flexible (sélection, intégration, chevauchement). Cette solution permet de profiter pleinement de la richesse des services disponibles au moment d'exécution à un instant donné.

4.4.1 Appariement fonctionnel

Dans notre travail, la composition est vue comme un processus abstrait (sans aucune référence concrète à un service particulier). Ce processus est constitué d'un ensemble d'opérations abstraites gérées par des opérateurs de contrôle (par exemple boucles, conditionnelles, etc.). Chaque opération est décrite sémantiquement par la fonctionnalité qu'elle réalise et les données qu'elle échange. Une telle description peut être vue comme une représentation abstraite, du processus métier, offerte par la tâche. Les informations sémantiques au niveau de la description de la tâche et des services web permettent de favoriser le processus de sélection, d'intégration et de chevauchement dynamique et flexible.

Matching de données

Une donnée est un flux d'entrée / sortie décrit par un concept ontologique. Une valeur de donnée est une instance (individu) de concept dans une ontologie de domaine. Le matching des paramètres (d'entrée / de sortie) de processus correspondant est présenté dans la définition suivante.

Definition 4.4.1 ($DSim$)**.** *DSim est la similarité sémantique de données des processus entre une donnée (D_r) requise et une autre (D_o) offerte.*

La similarité sémantique de données est définie par la formule suivante :

$$DSim(D_r, D_o) = \frac{Depth(C_o)}{Depth(C_r)} \tag{4.1}$$

Où D_r (respectivement D_o) est une instance du concept C_r (respectivement C_o) dans une ontologie de domaine. La Profondeur $Depth(Ci)$ est le nombre d'arcs du chemin emprunté du noeud racine de l'ontologie au concept Ci de la même ontologie. Pour ce matching, nous supposons que les concepts sont impliqués par une relation de subsomption (une relation qui lie un concept plus spécifique à un concept plus

générique). En d'autres termes, les concepts sont contenus dans le même chemin dans l'ontologie.

Matching de prédicats

Un prédicat est un paramètre de pré-condition / effet. Un prédicat est décrit par un littérale, un opérateur de comparaison et une valeur. Le littérale est décrit par un concept ontologique. Le littérale et sa valeur sont définis par un même domaine de définition. Nous utilisons le matching de prédicats pour la réconciliation des pré-conditions / effets des processus. Le matching de prédicats est présenté dans la définition suivante :

Definition 4.4.2 ($CSim$)**.** *$CSim$ est la similarité sémantique de prédicat entre une condition requise ($Cond_r$) et une autre offerte ($Cond_o$) qui représente la pré-conditions / effets d'un processus.*

La similarité sémantique de prédicat est définie par la formule suivante :

$$CSim(Cond_r, Cond_o) = \frac{w_L * DSim(L_r, L_o) + w_{op} * OPSim(op_r, op_o)}{w_L + w_{op}} \tag{4.2}$$

Où L est la partie littérale d'un prédecat, op est la partie opérateur de comparaison ($op \in \{<, <=, >, >=, =, \neq\}$) et $OPSim$ est la similarité des opérateurs op (voir table 4.1). Ces valeurs sont basées sur certaines expériences, ainsi que l'équivalence logique des opérateurs relationnels. Par exemple, l'expression $a \geqslant 5$, est équivalente à :

- $a \geqslant 5$, avec un matching exacte,
- $a > 5$, avec un matching partiel (considère toutes les valeurs sauf la valeur 5),
- $a = 5$, avec un matching faible (considère seulement la valeur 5),
- $a \leqslant 5$ avec un matching très faible (considère seulement la valeur 5, et d'autres

valeurs indésirables).

– $a < 5$, échec de matching.

$\mathbf{op}_r$ \ $\mathbf{op}_o$	$=$	$>$	$\geqslant$	$<$	$\leqslant$
$=$	1	0	0.2	0	0.2
$>$	0	1	0.8	0	0
$\geqslant$	0.2	0.8	1	0	0
$<$	0	0	0	1	0.8
$\leqslant$	0.2	0	0	0.8	1

TABLE 4.1 – La similarité des opérateurs de comparaison $OPSim(op_r, op_o)$

Un processus est décrit par des entrées, sorties, pré-conditions et/ou effets. Pour la réconciliation des processus, nous utilisons le matching présenté dans la définition suivante.

Definition 4.4.3 ($PSim$)**.** *$PSim$ est la similarité sémantique de processus entre un processus atomique (p_r) requis et et un autre (p_o) offert.*

La similarité sémantique de processus est définie par la formule suivante :

$$PSim(p_r, p_o) = \frac{\sum_i^M \frac{w_{M_i}}{n_{M_i}} \sum_j^{n_{Mi}} MSim(M_{r_j}, M_{o_j})}{\sum_i^m w_{M_i}} \tag{4.3}$$

Où M_r and M_o sont des mesures requises et offertes de processus respectivement, qui peuvent-être un paramètre de donnée ou de prédicat. La mesure de similarité

sémantique MSim est calculée comme suit :

$$MSim(M_r, M_o) = \begin{cases} DSim(M_{rj}, M_{oj}) & \text{Si } M \text{ est un paramètre de donné} \\ CSim(M_{rj}, M_{oj}) & \text{Autrement} \end{cases} \tag{4.4}$$

4.4.2 Appariement Comportemental

Pour faire le matching des unités de conversation CUs, nous effectuons la similarité entre les flux de contrôle, ainsi que la similarité entre les processus (peuvent-être atomiques et/ou composites) constituant les CUs impliquées. En outre, afin d'augmenter la chance de réalisation des unités de conversation requises (RCU), nous exploitons la réconciliation de l'ordre d'exécution.

Par exemple, les CUs suivantes peuvent être considérées comme égales lorsque δ n'est pas un constructeur de séquence : $\delta p_1 p_2 p_3$, $\delta p_1 p_3 p_2$, $\delta p_3 p_1 p_2$. Pour réaliser fortement la RCU, nous profitons également de la compatibilité des flux de contrôle. Par exemple, si le flux de contrôle requis est un *"unordered"* et celui fourni est un constructeur *"split"*, il sera trop utile de les considérer comme compatibles.

En se basant sur les ontologies comme formalisme de description sémantique, nous proposons le matching comportemental entre des CUs requises et celles fournies par la définition 4.4.4.

Une unité de conversation est décrite par les processus interconnectés par un flux de contrôle ρ. En se basant sur le matching des processus et des contrôles des flux, nous définissons la similarité des unités de conversation comme suit.

Definition 4.4.4 ($CUSim$)**.** *$CUSim$ est la similarité sémantique entre une unité*

de conversation requise et une autre offerte.

La similarité sémantique d'unités de conversation est définie par la formule suivante :

$$CUSim(CU_r, CU_o) = \frac{\frac{w_P}{n_P}\sum_i^{n_P} PSim(p_{ri}, p_{oi})}{w_P + w_\rho} + \frac{w_\rho * FSim(\rho_r, \rho_o)}{w_P + w_\rho} \quad (4.5)$$

Où ρ est un flux de contrôle, et $FSim$ est la similarité des flux de contrôle comme il est indiqué par la table 4.2. Ces valeurs sont basées sur certaines expériences, ainsi que l'équivalence logique des flux de contrôle. Par exemple, l'expression $*abcd$, est équivalente à :

- $*abcd$, avec un matching exacte,
- $\bullet abcd$, peut être considéré comme un cas particulier d'une exécution non ordonnée,
- $|\, abcd$, avec un échec de matching,
- $\|$, peut être considéré comme un cas particulier d'une exécution non ordonnée,
- $\Vdash abcd$, peut être considéré comme un cas particulier d'une exécution non ordonnée.

4.5 Sélection à base de QoS

Les propriétés non-fonctionnelles peuvent également jouer un rôle important dans toutes les tâches associées aux services, en particulier au niveau de la découverte, de la sélection et de la substitution de services. Nous imaginons un scénario où l'on pourrait faire un choix entre plusieurs services, pouvant répondre à la requête d'un utilisateur et offrant essentiellement la même fonctionnalité, en fonction de certaines propriétés non fonctionnelles. Par exemple, si nous considérons un service de réservation de vol. La fonctionnalité (achat d'un billet d'avion) pourrait être

ρ_r \ ρ_o	$\bullet$	$*$	$\mid$	$\parallel$	$\Vdash$
$\bullet$	1	0	0	0.6	0.8
$*$	1	1	0	1	1
$\mid$	0	0	1	0	0
$\parallel$	0.4	0.6	0	1	0.8
$\Vdash$	0.3	0.5	0	0.8	1

TABLE 4.2 – La similarité des flux de contrôle $FSim(\rho_r, \rho_o)$

limitée par l'utilisation d'une connexion sécurisée (la sécurité étant une propriété non-fonctionnelle) ou par la tranche horaire dans laquelle les services sont invoqués (la disponibilité étant une propriété non-fonctionnelle).

Comme les compositions de services exigent de combiner les fonctionnalités de différents services, constituer la QoS de ces services devient aussi important. Les valeurs de QoS des services impliqués par la composition doivent être comparées et agrégées pour produire en sortie la QoS de bout en bout. La composition de QoS est le processus d'agrégation de la QoS des services invoqués pour que l'orchestration puisse estimer son propre résultat global.

Cependant, s'agissant des services composites qui invoquent divers services variant suivant leurs caractéristiques fonctionnelles et non fonctionnelles, les caractéristiques de bout en bout sont intimement liées au contrôle du flux de l'orchestration. La performance fonctionnelle de bout-en-bout repose sur une grande partie des services qui exécutent des tâches spécifiques et qui transmettent les données nécessaires au sein du contrôle de flux. Cependant, dans la plupart des orchestrations,

les contraintes sur les propriétés non-fonctionnelles (expiration de délai, niveaux de sécurité, etc.) affectent aussi la performance fonctionnelle de bout en bout. Cette influence de performance fonctionnelle par des caractéristiques non fonctionnelles n'est pas traitée dans notre thèse.

Dans notre solution de sélection de services web, le modèle de qualité dépend d'un ensemble d'attributs de qualités catégorisés en deux types principaux : les attributs positives et les attributs négatives. L'objectif est de maximiser autant que possible les valeurs des propriétés positives (par exemple, le débit et la disponibilité, etc.), tandis que les valeurs des propriétés négatives doivent être réduits au minimum (par exemple le prix et le temps de réponse, etc.). Notre solution pour le problème $O-QSC$ (*QoS-Aware Service Composition*) est formulée à base des définitions suivantes.

Definition 4.5.1 (Service atomique). *s_{ij} est un service composant, processus atomique, d'un service composite associé avec un ensemble de paramètres QoS.*

Definition 4.5.2 (Tableau qualitative). *Q est un tableau de trois dimensions, $(n * m * n_i)$ où $(i = \overline{1,n})$, dans lequel l'entrée Q_{ikj} définit la valeur de la propriété-qualitative k (par exemple le prix) pour le service web j qui est associée à la tâche i, n est le nombre de tâches (les classes de service), m est le nombre de propriétés-qualitative et n_i est la taille de la classe de service i.*

Definition 4.5.3 (Matrice de poids). *W est une matrice de taille $(n * m)$ dans laquelle l'entrée W_{ik} définit le poids de la propriété-qualitative k concernant les préférences de l'utilisateur pour la tâche i.*

Lemma 4.5.4 (Poid de propriété-qualitative). *si W_{ik} est le poid de la propriété qualitative k concernant les préférences de l'utilisateur pour la tâche T_i, alors $\sum_k^m W_{ik} = 1$.*

Definition 4.5.5 (Fonction d'utilité)**.** *Chaque service candidat a une fonction d'utilité associée F, qui est définie par un ensemble d'attributs de QoS tels que le temps de réponse, la disponibilité, la fiabilité, etc (voir l'équation* 4.6*).*

Definition 4.5.6 (Classe de service)**.** *(SC_i) englobent toute la collection de services atomiques qui ont la même fonctionnalité avec des propriétés non fonctionnelles (QoS), et sont candidats pour être associés à la tâche T_i.*

Le principe du problème $O-QSC$ pour la sélection et l'intégration des services est représenté dans la Figure 4.3. A la réception d'un workflow (WF) au moment d'exécution, le moteur de composition examine le registre de services $UDDI$ pour trouver des services candidats pour chaque tâche abstraite dans WF en utilisant un matching fonctionnel grâce aux descriptions de service. Ainsi, pour chaque tâche abstraite, un ensemble de services candidats soit trouvé. L'objectif de sélection et d'intégration de services vise à lier chaque tâche abstraite dans le workflow WF à un service concret afin de former un service-composite concret (CCS), dont les valeurs QoS agrégées sont optimales sur la base des préférences QoS des utilisateurs (W) ainsi que des contraintes QoS de bout en bout sont également satisfaits.

Malheureusement, le $O-QSC$ est connu comme un problème NP-difficile (NP-hard) [Zeng et al., 2004, Canfora et al., 2008, Wada et al., 2012]. Pour remédier à ce problème certains travaux de recherche ont adopté les graphes acycliques directs, et le traitent comme un problème de recherche de plus court chemin comme suit :

Etant donnée un workflow constitué de n tâches et un ensemble de services candidats susceptibles de satisfaire la fonctionnalité de la tâche i^{th} dans le workflow est noté comme un service candidate S_i ($i = \overline{1,n}$). Le j^{th} dans S_i est noté comme s_{ij} ($j = \overline{1,n_i}$). Le problème de la composition dynamique de services est modélisé comme un graphe acyclique direct $G(C,E)$, où C est l'ensemble des de services candidats, plus le noeud initial et le noeud final. E est l'ensemble des arêtes reliant les services candidats des deux ensembles adjacents S_i and S_{i+1}. Nous illustrons

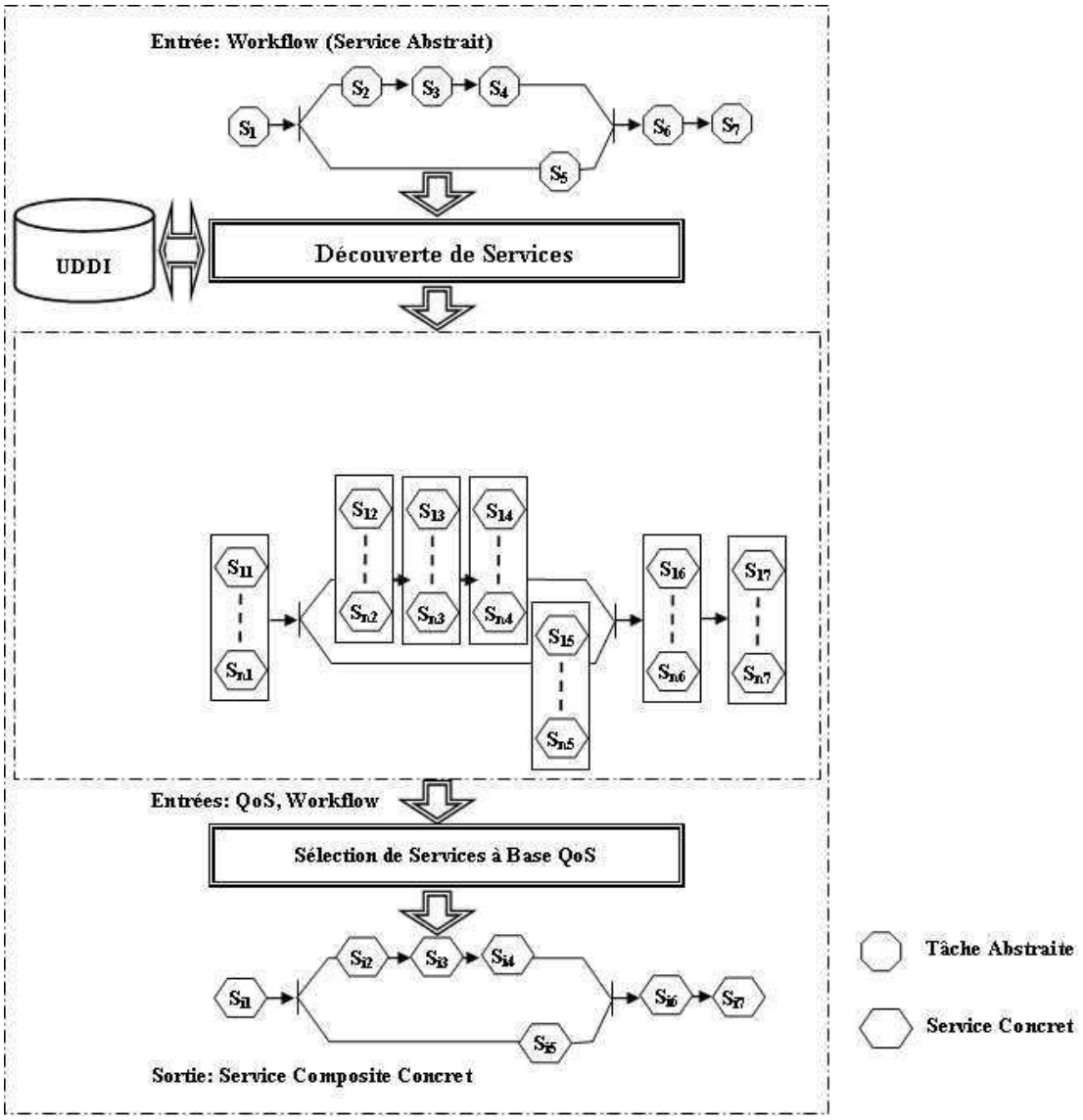


FIGURE 4.3 – Principe du $O-QSC$ pour la sélection de services

le problème de la composition dynamique de services comme un graphe acyclique directe dans la Figure 4.4. Donc, le problème se transforme en une recherche du plus court (meilleur) chemin satisfaisant les contraintes globales à partir du noeud initial au noeud final.

Cependant, via-à-vis le nombre considérable de tâches à réaliser, ainsi que la quantité énorme de services candidats disponibles pour réaliser ces tâches, le $O-QSC$ est devenu beaucoup plus difficile. Ce problème a été montré comme un pro-

Algorithm 3: L'algorithme de sélection de services

Input : RSB,$\{RCU\}$, $\{PSB\} \cup \{PCU\}$
Output: RSB ; /* New Concrete RSB */

```
1  for All RCU In RSB do
2      for All T_i In RCU do
3          for All q Of T_i do
4              F_ij = [0];
5              for All S_ij In SC_i do
6                  Q^max = Max [Q_ikj];
7                  Q^min = Min [Q_ikj];
8                  if q is Positif QoS then
9                      q = Q^min;
10                 else
11                     q = Q^max;
12                 if Q^max ≠ Q^min then
13                     Q_ikj = |q−Q_ikj| / (Q^max − Q^min);
14                 else
15                     Q_ikj = 1;
16                 F_ij += W_ik * Q_ikj;
17             BindTo(T_i,Max [F_ij]) ; /* Bind T_i to S_ij, of SC_i, that
                   have the Max Value for F_ij                    */
```

blème NP-complet [Martello et Toth., 1987, Yu et Zhang., 2007]. Pour l'adresser, plusieurs approches ont été proposées (par exemple, la programmation linéaire, algorithmes évolutionnaires, heuristique, etc.).

D'une part, puisque tous les travaux de recherche ont montré que la sélection de services selon la stratégie locale est optimale en temps et en espace, ainsi que les auteurs qui adopte cette stratégie sont d'accord sur le fait que la sélection des services optimaux garantie l'optimisation globales de la composition de ces services. D'autre part, comme notre travail est orienté beaucoup plus sur l'aspect comportementale, pour la sélection de services par les propriétés QoS, nous avons adopté la stratégie de sélection locale pour traiter ce problème. Par conséquence, nous avons proposé un algorithme déterministe (voir Algorithm 3). Le problème $O-QSC$ sera

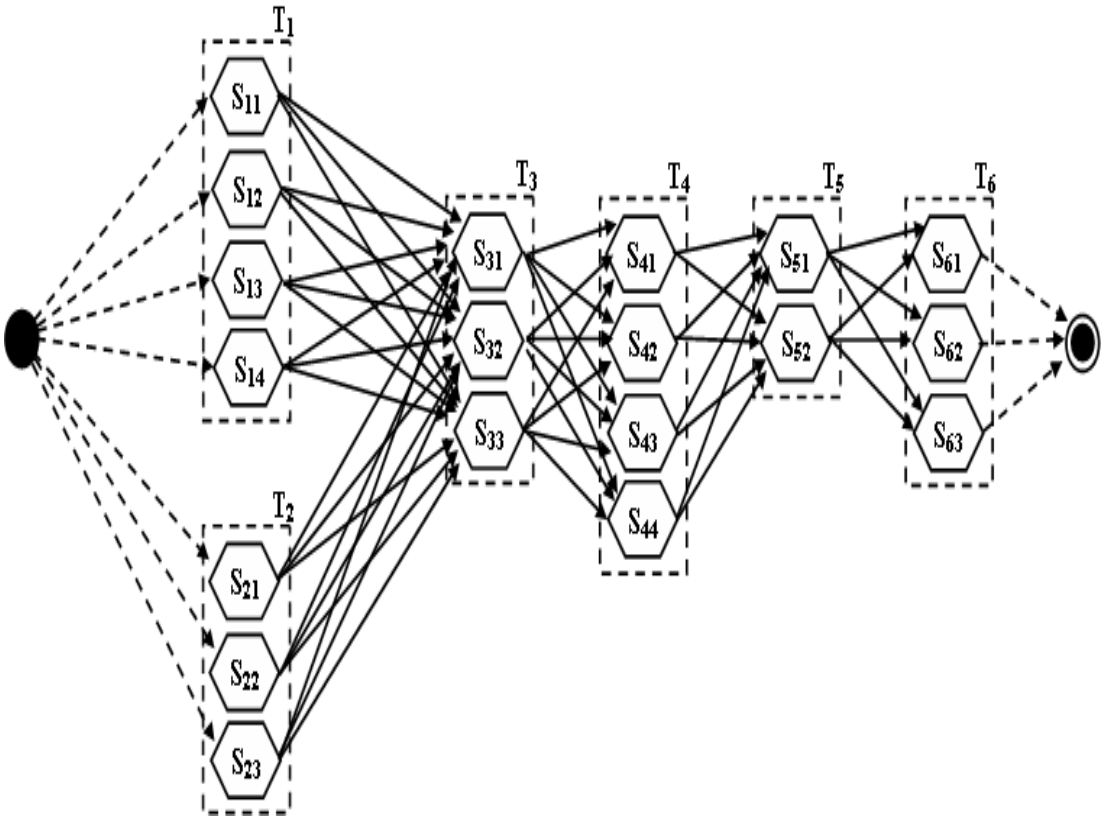


FIGURE 4.4 – Graphe de services candidats

donc une sélection des services composants, jugés comme les plus pertinents à la composition (CU dans notre cas), pour chaque classe i. L'objectif des algorithmes de sélection de services consiste à sélectionner les services individuels qui répondent aux contraintes de QoS et fournissent la meilleure valeur pour la fonction d'utilité défini par l'utilisateur F.

Supposons qu'il existe un ensemble d'attributs positifs et négatifs de QoS. Les attributs positifs doivent être maximisés et les attributs négatifs doivent être minimisés. La fonction d'utilité F pour le service s_{ij} est défini comme suit :

$$F_{ij} = \sum_{k} W_{ik} * Q_{ikj} \tag{4.6}$$

Où W_{ik} sont les coefficients de pondération pour chaque attribut Q_{ik} qualitatif,

et Q_{ikj} est le k^{me} attribut de QoS pour le j^{me} service associé à la tâche i.

Dans l'équation de la fonction d'utilité, tous les attributs de qualité de service sont pondérés par leur importances (poids). Ils sont également normalisés par l'équation suivante :

$$Q_{ikj} = \begin{cases} 1 & \text{si } Q_{ik}^{max} = Q_{ik}^{min} \\ \frac{Q_{ik}^{max} - Q_{ikj}}{Q_{ik}^{max} - Q_{ik}^{min}} & \text{si } (Q_{ik}^{max} \neq Q_{ik}^{min}) \ \& \ (Q_{ikj} \text{ est un attribut négatif}) \\ \frac{Q_{ikj} - Q_{ik}^{min}}{Q_{ik}^{max} - Q_{ik}^{min}} & \text{si } (Q_{ik}^{max} \neq Q_{ik}^{min}) \ \& \ (Q_{ikj} \text{ est un attribut positif}) \end{cases} \tag{4.7}$$

Où Q_{ik}^{max} et Q_{ik}^{min} sont évaluées par les équations suivantes, respectivement.

$$Q_{ik}^{max} = Max \left[Q_{ikj} \right] \tag{4.8}$$

$$Q_{ik}^{min} = Min \left[Q_{ikj} \right] \tag{4.9}$$

4.6 Sélection et Composition comportementales de services

Dans cette section, nous présentons la notion de combinaison comportementale flexible et appropriée. La combinaison comportementale flexible (sélection, intégration et/ou chevauchement) améliore la chance de réalisation de la tâche. La combinaison comportementale appropriée satisfaite à la fois les contraintes de fournisseurs ainsi que les préférences des utilisateurs. De telles contraintes doivent être considérées attentivement lors de la composition des services web. Malheureusement, elles sont rarement prises en compte par les travaux courants. Afin de bien détailler le processus de sélection et composition comportemental, nous allons tout d'abord présenter la codification des services (processus) participants à une composition que nous allons les utiliser durant cette section comme suit : ***Vol*** : p_1, ***Voiture*** : p_2, ***Hôtel*** : p_3, ***Météo*** : p_4, ***Bateau*** : p_5, ***Moto*** : p_6.

4.6.1 Combinaison flexible

Comme nous avons déjà évoqué, le processus de service est un ensemble de comportements alternatifs de services SBs. Chaque comportement de service requis (RSB) peut être réalisé soit par une sélection, une intégration et/ou un chevauchement de certains comportement de service offerts (RSBs). Par exemple, le ($RSB = \bullet p_1 p_2 p_3 p_4$) indiqué par la Figure 4.5 peut être réalisé soit par :

1. la sélection du ($PSB = \bullet p_1 p_2 p_3 p_4$) (voir Figure 4.6($a$)) ;
2. l'intégration des ($PSB = \bullet p_1 p_2$) et ($PSB = \bullet p_3 p_4$) (voir Figure 4.6(b)) ;
3. le chevauchement des ($PSB = \bullet p_1 p_3$) et ($PSB = \bullet p_2 p_4$) (voir Figure 4.6(c)).

Néanmoins, l'accomplissement de tous les SBs alternatifs est une réalisation complète d'un processus de service web. Toutefois, cette réalisation sera impossible

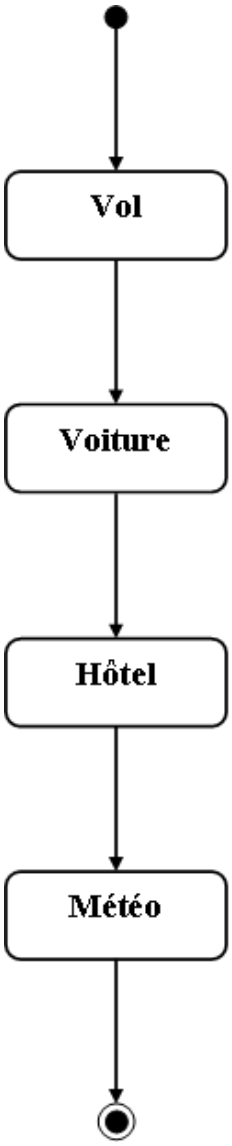


FIGURE 4.5 – Exemple d'un RSB à réaliser

lorsque certains services ne sont pas disponibles. Dans ce cas, l'accomplissement d'un seul SB est en fait une solution partielle et suffisante, puisque durant l'invocation effective de service, seulement un seul SB est invoqué. Par exemple, le processus de service requis indiqués par la Figure 4.7 peut être décrit comme nous avons vu par $(RSB_1 = \bullet p_1 p_2 p_4)$ et $(RSB_2 = \bullet p_5 p_2 p_4)$ peut être réalisé par la réalisation de l'un entre eux comme une réalisation partielle et suffisante.

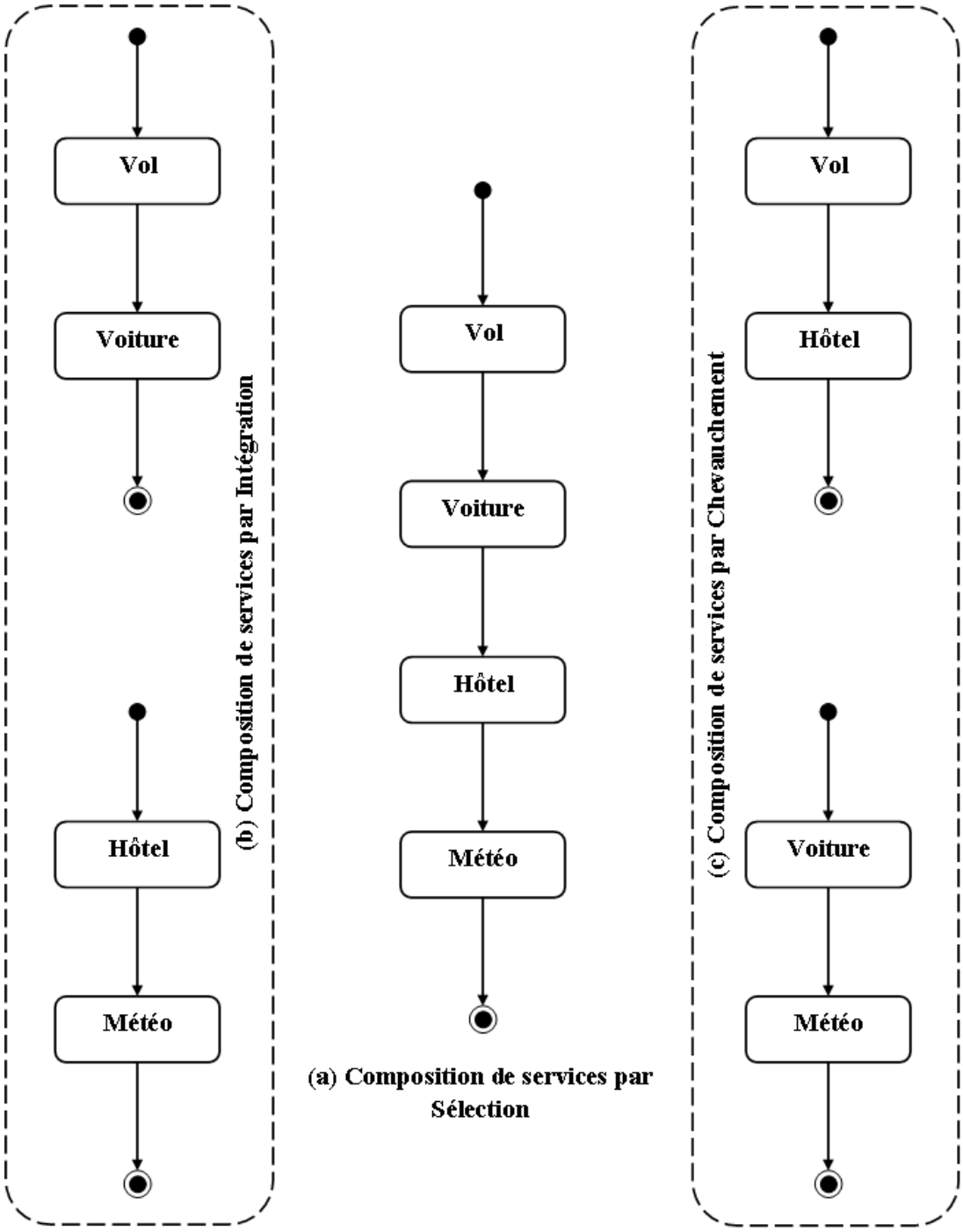


FIGURE 4.6 – Combinaison Comportementale Flexible

En outre, pour notre solution, un SB est une liste de CUs. Ainsi une CU requise peut être trouvée soit par sélection, intégration et/ou chevauchement de certains CU offertes. Par exemple, le processus de service requis indiqué par la Figure 4.8

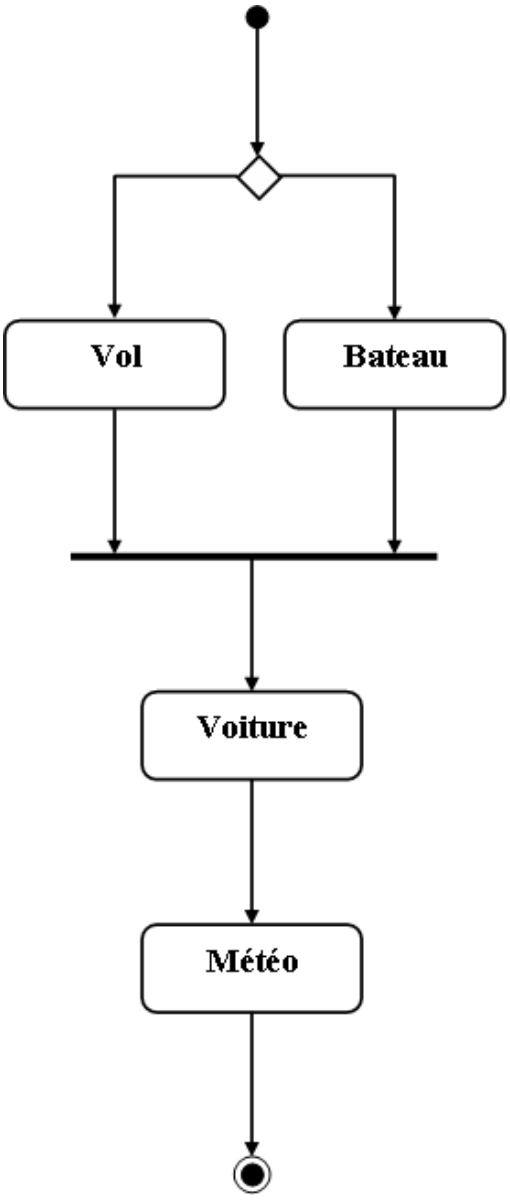


FIGURE 4.7 – Exemple d'un RSB qui support la réalisation partielle et suffisante

peut être décrit par un seul PSB (PSB : $PCU_1 \bullet PCU_2$ où $PCU_1 = \|p_1p_3$ et $PCU_2 = \|p_2p_4$), alors l'unité de conversation requise ($RCU = \bullet p_1p_2p_3p_4$) indiquée par la Figure 4.5 peut être accomplie par ce PSB lorsqu'en profitant de la notion d'équivalence entres les flux de contrôles.

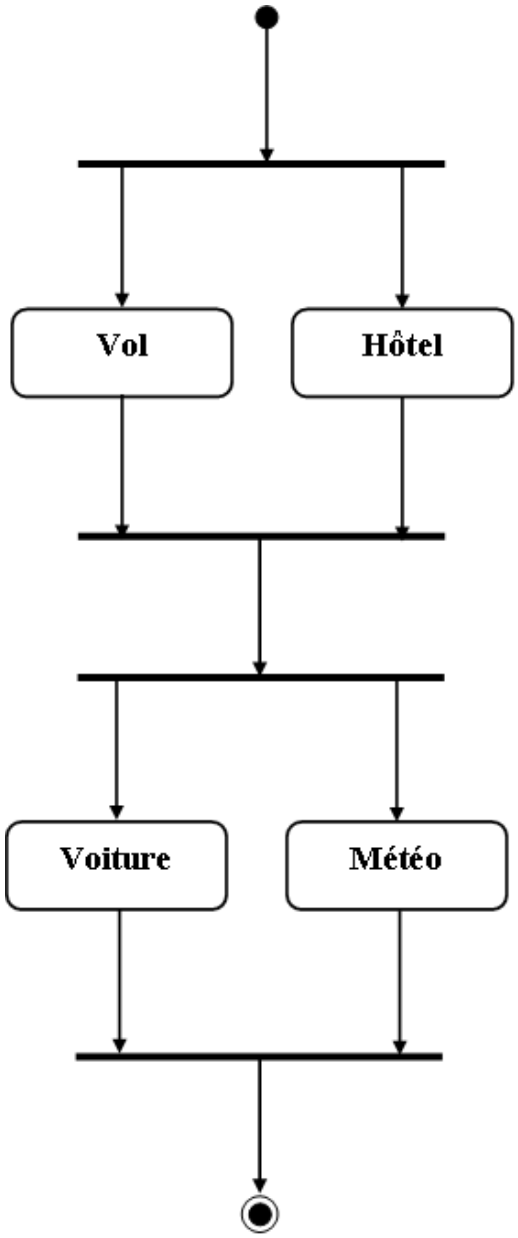
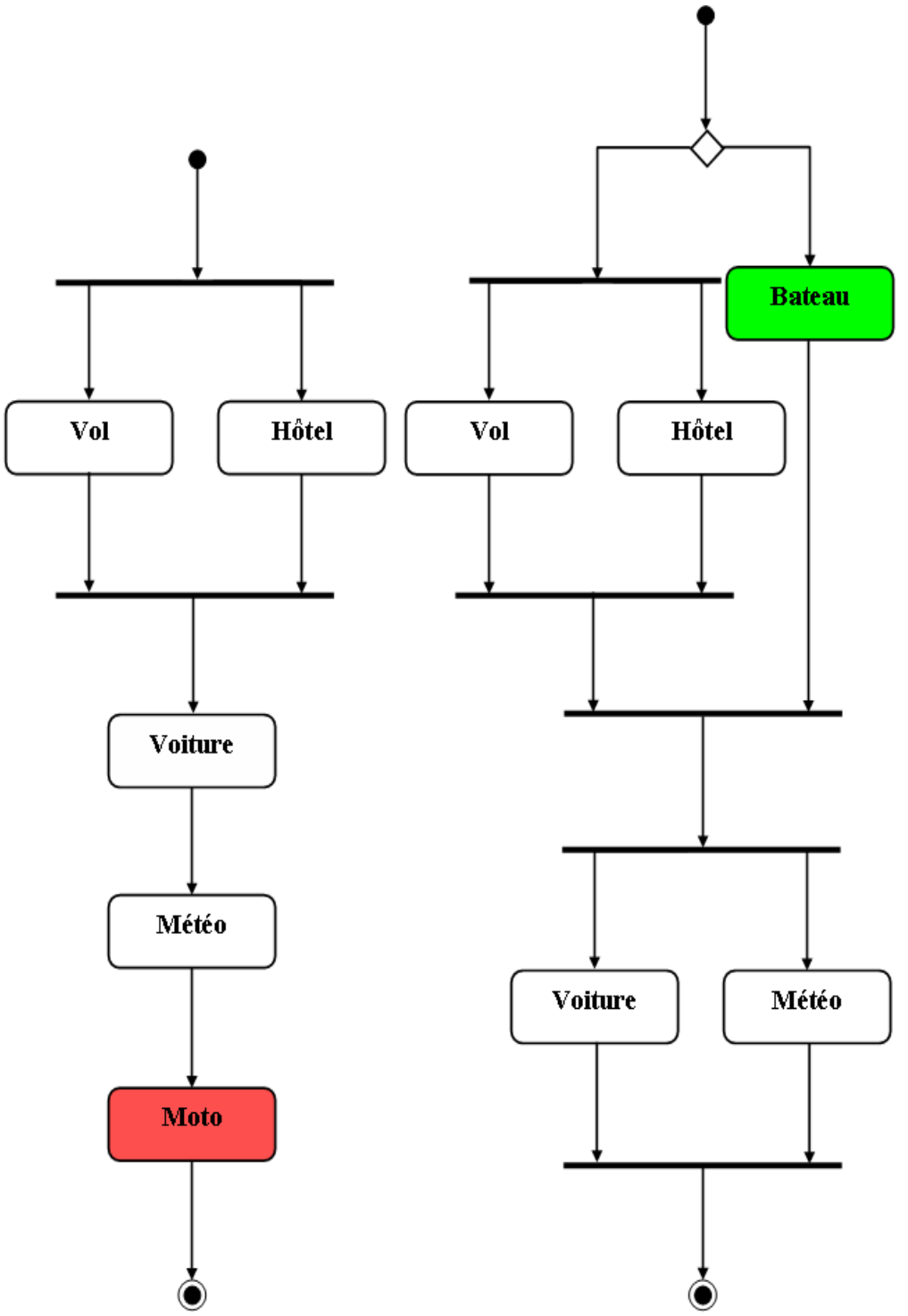


FIGURE 4.8 – Exemple d'exécution parallèle de services composants

4.6.2 Combinaison appropriée

Afin d'assurer une combinaison comportementale appropriée, il doit y avoir une vérification double : l'une au niveau de fournisseur, et l'autre au niveau d'utilisateur.

Pour le niveau de fournisseur, nous considérons que chaque comportement de service (une liste des $PCUs$) comme un comportement (processus) primitif et non



(a) Consommation de service non autorisée (b) Consommation de service autorisée

FIGURE 4.9 – Combinaison Comportementale Appropriée

décomposable, et il doit être entièrement utilisé comme il est spécifié par son fournisseur.

Par exemple, si on considère le RSB, indiqué par la Figure 4.5, constitué d'une seule ($RCU = \bullet p_1 p_2 p_3 p_4$) et le processus de service, indiqués par la Figure 4.9(a), peut être décrit par un PSB_1 qui est décrit à son tour par ($PSB_1 : PCU_1 \bullet PCU_2$ où $PCU_1 = \| p_1 p_3$ et $PCU_2 = \bullet p_2 p_4 p_6$). Comme nous avons considéré que tout PSB est primitif et non décomposable, alors ce PSB_1 ne peut être utilisé, parce que son processus p_6 doit être ainsi utilisé parallèlement avec les processus p_1, p_2, p_3 et p_4, en d'autres termes le PSB_1 doit être consommé entièrement et non pas partiellement. Par conséquent, le RSB ne peut être réalisée malgré que tous ses processus constituants sont disponibles, puisque la sous-consommation du PSB_1 n'est pas autorisée par son fournisseur. Alors que, le processus de service, indiqué par la Figure 4.9(b), peut être décrit par les deux PSB_1 et PSB_2 qui sont décrits à leur tour par ($PSB_1 : PCU_1 \bullet PCU_2$ où $PCU_1 = \| p_1 p_3, PCU_2 = \| p_2 p_4$) et ($PSB_2 : PCU_3 \bullet PCU_4$ où $PCU_3 = p_5, PCU_4 = \| p_2 p_4$). Le processus dans ce cas peut être utilisé, malgré que son PSB_2 n'est pas été impliqué par la composition, par ce que sa sous-consommation entière du PSB_1 est autorisée par son fournisseur.

Pour satisfaire les préférences des utilisateurs, les comportements primitifs requis ($PRSB$) doivent être trouvés comme ils sont indiqués par leur utilisateur. De tels comportement primitif requis $PRSB$ permet à la fois de réaliser et de contrôler les opérations d'intégration et/ou de chevauchement comportementales.

Par exemple, si on suppose que le RSB, indiqué par la Figure 4.5 constitué d'une seule ($RCU = \bullet p_1 p_2 p_3 p_4$), munie par le $PRSB$ décrit par ($RCU_2 = \bullet p_1 p_2 p_3 p_4$). Si on suppose de plus que les PSB_1 et PSB_2, indiqués par la Figure 4.6(b), décrits par ($PCU = \bullet p_1 p_2$) et ($PCU = \bullet p_3 p_4$) respectivement. Par conséquent, la combinaison des PSB_1 et PSB_2 ne peut pas être acceptée malgré qu'elle constitue en entier le RSB demandé, puisque l'utilisateur via sa $PRSB$ n'accepte pas de décomposer le

RCU_2 en certain nombres de PCU_s, mais il préfère de le réalisé par une seule PCU.

L'algorithme suivant (voir Algorithm 4) formalise le processus de combinaison comportementale flexible et appropriée de services .

Algorithm 4: Combinaison comportementale flexible et appropriée

```
Input  : RSB, PRSB, SetPSB ;   /* SetPSB is a set of provided
         services behaviors */
Output: NewSB ;                /* New service behavior */
1  NewSB ← Null;
2  for All RCU In RSB do
3      Max ← 0;
4      Rang ← 1;
5      Depth ← GetDepth(RCU); /* GetDepth computes the RCU size
       "processes number" */
6      repeat
7          SCombs ← GetAllCombs (Rang, SetPCU, PRSB);
           /* GetAllCombs builds all PCUs */
                /* appropriate combinations set with rank equals to
           Rang */
8          for All Comb In SCombs do          /* Comb is a
              Rang rank combination                          */
9
10             Sim ← GetSim(RCU, Comb);    /* GetSim computes the
               RCU, Comb similarity */
11             if (Sim > Max) then
12                 Max ← Sim;
13                 NewCU ← Comb ;  /* NewCU is a New Conversation
                   unit */
14         Rang ++;
15     until (Rang = Depth + 1) Or (Max ≠ 0);
16     NewSB.add(NewCU) ;
```

4.7 Conclusion

Ce chapitre a présenté notre approche de composition dynamique et comportementale de services web sémantiques. La solution proposée améliore la chance de réalisation de la tâche utilisateur, par le biais de la combinaison flexible (sé-

lection, intégration et/ou chevauchement) de comportement de service au moment d'exécution. En tenant compte du matching de flux de contrôle, de telles propriétés permettent l'exploitation totale de services web disponibles au moment de processus de composition. En plus notre approche assure l'utilisation appropriée de services web, en supposant que tous les comportements de services fournis sont considérés comme primitives. Ainsi, l'approche proposée garantit la satisfaction des préférences des utilisateurs en les spécifiant sous forme de comportements primitives requis.
Plus particulièrement, le modèle proposé permet de représenter un nombre potentiellement grand de services répondant à un même ensemble de besoins fonctionnels et se distinguent par des propriétés non fonctionnelles. Ceci fournit un cadre pour la sélection d'un ou de plusieurs services portant sur leurs caractéristiques non fonctionnelles. Cette sélection peut aussi bien être effectuée au moment de conception de composition qu'au moment de sa réalisation. Le modèle de composition de services que nous avons proposé s'appuie sur cette dernière caractéristique. L'objectif est de permettre au concepteur d'associer des propriétés extra fonctionnelles et comportementales à une composition au moment de sa conception, puis de sélectionner, à l'exécution, les services participant afin de choisir ceux qui répondent mieux aux besoins exprimés ainsi par des propriétés non fonctionnelles paramétriques et comportementales.
L'étude rapportée ici ouvre plusieurs perspectives de recherche. La sélection proposée s'appuie sur un modèle de qualité qui doit être enrichi, par exemple pour couvrir un éventail plus large de critères de qualité ou bien pour permettre de réaliser des conversions entre des valeurs d'un même critère exprimées dans différentes unités (monétaires, temporelles, etc.). Dans l'approche que nous avons proposée, si certains services offrent un modèle de qualité qui n'inclut pas tous les critères utilisés dans une sélection, ils ne sont pas considérés par cette sélection. Cette situation doit être traitée, surtout lorsqu'aucun service enregistré n'est muni d'un modèle de qualité

correspondant à tous les critères utilisés dans la sélection. De façon plus générale, nous avons adopté une vision simplifiée du choix multicritère en nous ramenant, par l'introduction de pondération, à un choix monocritère.

Chapitre 5

Implémentation et Évaluation

Sommaire

5.1 Introduction

Nous venons de terminer la présentation théorique de nos travaux dans le cadre de notre contribution pour la composition personnalisée de services web. Désormais, avant de conclure cette thèse, il nous reste à exposer les moyens que nous avons pu mettre en oeuvre pour la phase applicative et expérimentale de nos travaux de recherche. C'est pourquoi dans ce chapitre, nous allons tout d'abord décrire l'architecture de l'approche proposée dans la section 5.2. Ensuite, les principales caractéristiques du prototype que nous avons développé pour valider l'approche que nous proposons au travers du framework Ws-BeC sont présentées dans la section

5.3. Nous avons décliné la mise en oeuvre de ce prototype en deux parties : (i) une qui correspond à l'architecture logique dans la section 5.3.1 (ii) et une autre partie qui correspond à l'interface d'utilisateur dans la section 5.3.2. L'évaluation expérimentale de la mise en oeuvre du framework Ws-BeC est introduite dans la section 5.4. Finalement la section 5.5 conclut en discutant particulièrement les questions soulevées par le biais de mise en oeuvre du framework $Ws - BeC$.

5.2 Architecture de l'approche proposée

La Figure 5.1 montre l'architecture du framework de composition, constituée de sept modules permettent d'assurer les différentes phases proposées dans le processus global (voir La Figure 4.1 du chapitre précédent). Ainsi, les modules de description sont utilisés dans la phase de conception, alors que ceux responsables de la constitution de composition et de la transformation dans un fichier d'orchestration annoté sont dans la phase d'exécution. Le registre d'ontologie, de services atomiques et de services composites avec des descriptions comportementales offre quant à lui des données utilisées au moment de la conception et d'autres utilisées au moment de l'exécution. Les moteurs de publication et d'exécution de services sont des entités en dehors de l'architecture proposée mais avec lesquelles Ws-BeC interagit. Dans l'état actuel d'avancement du développement, en plus de la mise en oeuvre des modules de publication (stockage) et d'invocation de services, les modules de description, de sélection (comportementale et à base de QoS) de services et de composition ont été réalisés. Dans la suite, nous introduisons chacun de ces modules.

Le module SPF Maker Ce module interagisse avec l'utilisateur ainsi que le registre de services afin de concevoir le formalisme SPF comme une infrastructure de base pour l'extraction de la description comportementale de service que ce soit offerte (PCS : provided composite service) ou bien requise (RCS : required composite

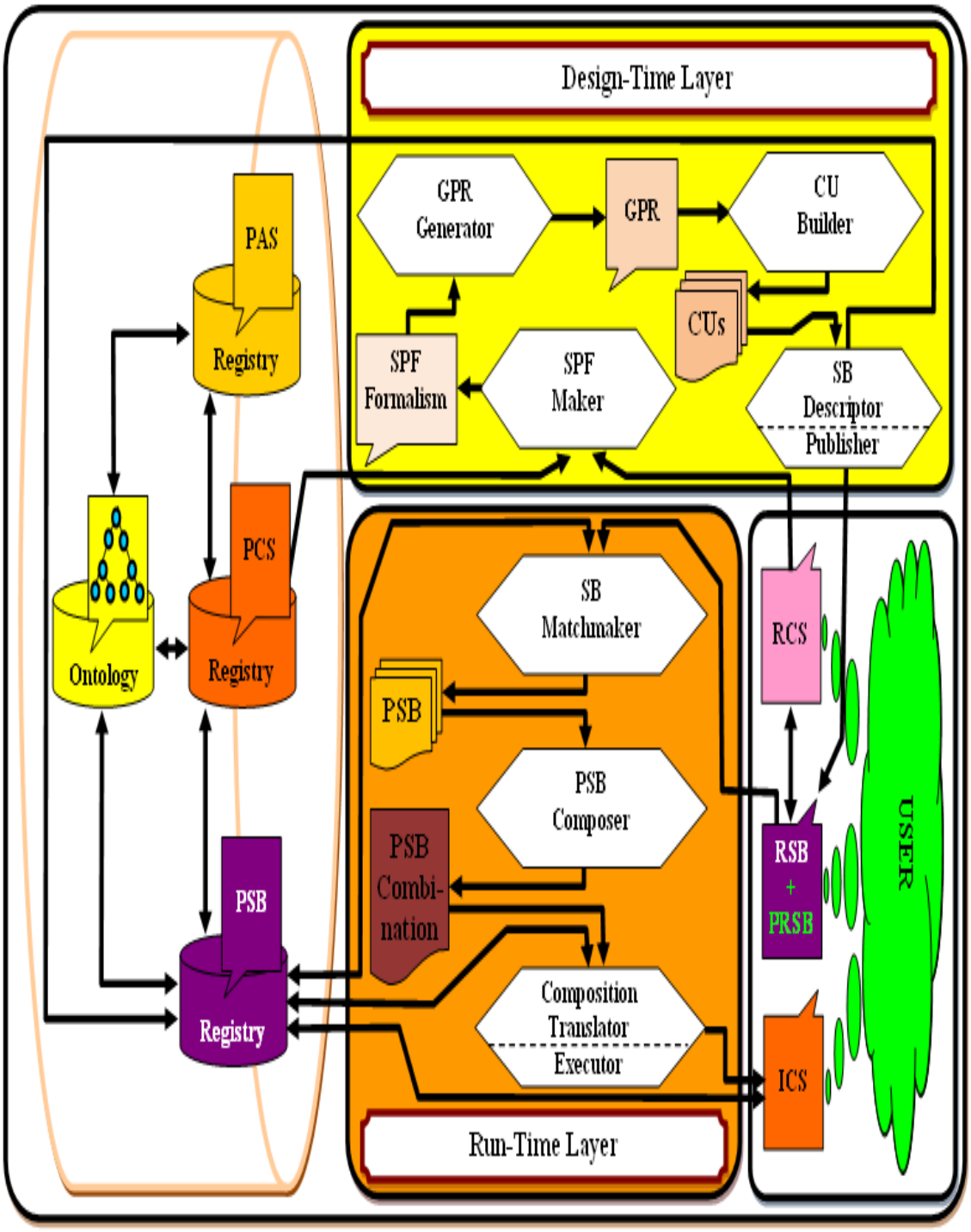


FIGURE 5.1 – Architecture d'adaptation et de combinaison comportementale

service). Bien qu'un PCS pour son interaction peut communiquer avec ses services composants PAS (provided atomic service) ainsi que l'ontologie de domaine OWL.

Le module GPR Generator Ce module, comme son nom l'indique, adopte le processus de substitution des non-terminaux par leurs règles de production afin de générer la règle globale GPR (global production rule) qui représente tous les scénarios de communication avec les services composants.

Le module CU Builder Ce module utilise le constructeur (l'opérateur) de choix externe pour scinder la règle GPR en un ensemble de SBs alternatifs. Il se base sur la perspectif d'un seul opérateur de contrôle par une UC afin d'extraire les différentes UC pour chaque SB.

Le module SB Descriptor Ce module crée la nouvelle description comportementale de service d'une manière beaucoup plus riche, ensuite il déploie la description PSB dans l'annuaire de services et stocke la description RSB dans la machine utilisateur.

Le module SB Matchmaker Ce module, à la réception d'une requête utilisateur, au moment d'exécution, exprimée sous forme d'un ensemble de RSB (required services behavior) il interagisse avec le registre PSB *registry* afin de rechercher et de sélectionner les services les mieux adaptés à la composition en cours par le bais d'appariement fonctionnel (à base des entrées, sorties , pré-conditions et des effets), non fonctionnel (à base des paramètres QoS) et comportemental.

Le module PSB Composer Ce module utilise les descriptions comportementales PSB des services candidats à la composition choisie par le module *SB Matchmaker* pour construire la combinaison la plus appropriée qui satisfait à la fois les contraintes de fournisseurs ainsi que les préférences d'utilisateur.

Le module Composition Translator Comme son nom l'indique, ce module transforme la combinaison trouvée par le module PSB Composer en une description de

service exécutable ICS (Invocable composite service) sou forme d'un fichier de composition reconnue par un langage de composition donné (OWL-S pour notre expérimentation).

5.3 Prototype et description du fonctionnement

Dans cette section nous allons présenter l'architecture logique ainsi que l'implémentation de différents modules du framework Ws-BeC

5.3.1 Architecture logique

L'architecture logique organise les différentes classes du prototype développé en packages, sous-systèmes et couches. Celle-ci est implémentée en trois couches différentes (d'application, de médiation et de Fondation).

La Figure 5.2 montre les couches de l'architecture, l'interaction entre elles, ainsi que les packages les plus importants qui composent chaque couche.

Couche d'Application

La couche application gère les packages qui implémentent les fonctionnalités de prototypes. Cette couche est constituée par les packages suivants :

- **Service Parser** : ce package permet d'enregistrer la nouvelle description comportementale de services (composite) conçue à partir d'une représentation du FSP en modifiant des descriptions préexistantes. Par exemple, pour Ws-BeC nous avons implémenté des annotations comportementales des descriptions OWL-S, mais il est possible de manipuler d'autres descriptions comme BPEL, Orc etc.
- SPF **Maker** : ce package utilise le package OWL-S pour générer le formalisme SPF à partir d'une description ordinaire de service OWL-S.
- **Behavioral Description** : ce package utilise les packages, *Service Parser*

et *SPF Maker*, pour transformer une description ordinaire de service (par exemple OWL-S) en une nouvelle représentation sous forme des *SB*s et *UC*s pour faire l'assouplissement du processus de composition comportementale.

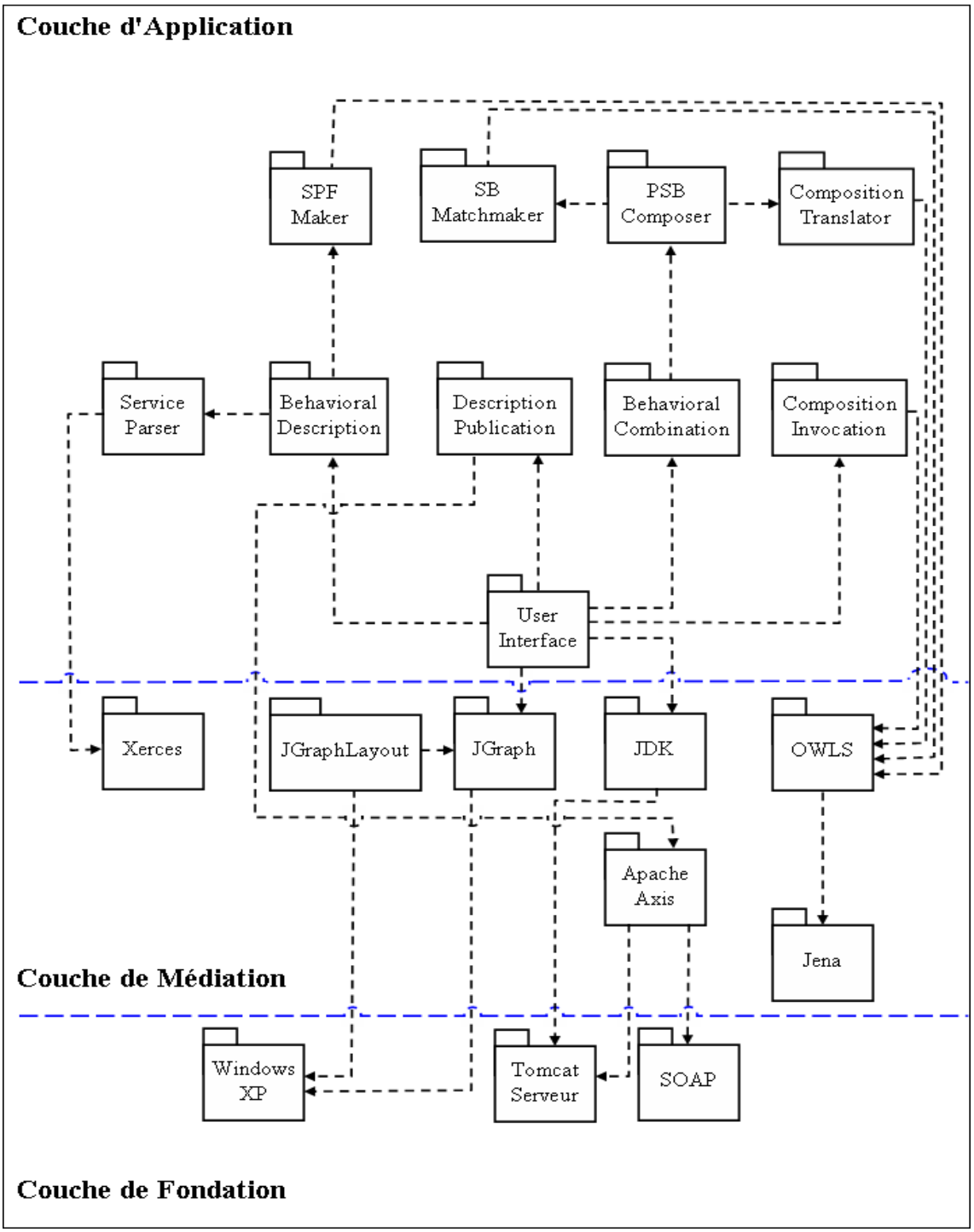


FIGURE 5.2 – Architecture logique du prototype Ws-BeC

- **Description Publication** : ce package utilise *Apache Axis* pour enregistrer la nouvelle description pour être utilisée par la suite pour l'appariement comportemental flexible de services, ainsi que la composition de services établie en tant que service web OWL-S beaucoup plus complexe.
- **Composition Translator** : ce package utilise le package $OWL-S$ pour traduire la composition de services établie en un fichier d'orchestration exécutable exprimé en $OWL-S$.
- **Composition Invocation** : ce package utilise le package OWL-S pour permettre l'invocation de la composition réalisée.
- *SB* **Matchmaker** : Ce package prend en entrée des PSBs/PCUs et un RSB/RCUs pour faire l'appariement fonctionnel et extra-fonctionnel en calculant les différentes mesures de similarité. Il utilise le package $OWL-S$ pour manipuler les différentes paramètres de service.
- *PSB* **Composer** : ce package prend en entrée un RSBs/RCUs, ensuite il fait la découverte, la sélection et la composition des PSBs/PCUs. Il utilise le package *SB Matchmaker* pour les différentes mesures de similarité afin de sélectionner la combinaison la plus appropriée.
- **Behavioral Combination** : ce package contient les classes qui implémentent la composition comportementale de services. Il utilise le package *PSB Composer* pour faire la sélection, l'intégration et le chevauchement des PSB/UC afin de réaliser un RSB/RCUs.
- **User Interface** : Afin de réaliser une représentation visuelle, ce package contient toutes les classes qui implémentent les interfaces graphiques du prototype. Il utilise tous les packages d'une manière directe ou indirecte.

Couche de Médiation

La couche médiation contient toutes les APIs (Application Program Interfaces) utilisées par le prototype. La couche est composée par les packages suivants :

- **Apache Axis** : est un open source framework de service web bâti sur XML. Il constitue, sous Java et C++, une implémentation de serveur SOAP, de plusieurs utilitaires et des APIs, pour générer et également de déployer des applications de service web. En utilisant l'*Apache Axis*, la composition de services peut être publiée à son tour en tant que service web beaucoup plus complexe.
- **Xerces** : est une famille de packages logiciels de parsing et de manipulation des documents XML. La bibliothèque implémente un certain nombre des APIs standard pour parser des documentrs XML, y compris les DOM (Document Object Model), SAX (Simple API for XML) et SAX2. De cette manière, le package Xerces supporte le package *Service Parser* pour parser des services.
- **Jena** : il offre une abstraction pour les ontologies OWL(Web Ontology Language) et il permet la manipulation des ces dernières à l'aide d'un raisonneur. Il constitue un framework open-source applicatif pour le développement des applications pour le web sémantique. Il fournit ainsi un environnement de programmation pour RDF(Resource Definition Language), RDFS (RDF Schema) et OWL, ainsi qu'un moteur d'inférence à base de règles. Le framework inclut des APIs RDF, OWL et RDQL (Resource Definition Query Language). Le sous-système d'inférence de Jena est conçu pour permettre à certains moteurs d'inférence ou raisonneurs d'être connecté à Jena. Le terme inférence est utilisé pour se référer au processus abstrait de dérivation d'informations supplémentaires, et le terme raisonneur pour faire référence à un code objet spécifique qui effectue cette tâche.
- **OWL-S** est une ontologie construite au-dessus du langage pour les ontologies

OWL. Elle remplace l'ancien ontologie DAML-S. OWL-S est intégrée dans le framework de web sémantique bâti sur OWL, pour décrire des services web sémantiques. Elle permettra aux utilisateurs et aux agents logiciels de découvrir, d'invoquer, de composer, et de surveiller des ressources web offrant des services, sous certain contraintes bien spécifiées.

- **JDK (Java Development Kit)** : est un environnement de développement intégré (IDE : integrated development environment) pour le développement des applications Java. Il constitue un environnement d'exécution sous un système d'exploitation qui permettre de compiler, déboguer et exécuter l'outil effectif qui est développé avec le langage Java.
- **JGraph** : est un framework graphique Java qui est conforme pleinement aux principes de conception de la bibliothèque Swing. Il contient toutes les fonctionnalités de visualisation et d'interaction de la bibliothèque de Graph. Ce package supporte la visualisation de l'interface utilisateur graphique.
- **JGraph Layout** : est une bibliothèque graphique de présentation de haute performance pour *JGraph* qui positionne automatiquement le graphique, diagramme, ou le réseau d'une manière visuellement agréable.

Couche de Fondation

La couche fondation comporte le software de base qui permet la réalisation et l'exécution du prototype. Cette couche est composée des packages suivants :

- **SOAP (Simple Object Access Protocol)** : est un protocole, orienté objet bâti sur XML, d'échange de messages entre objets distants, ce qui veut dire qu'il autorise un objet à invoquer des méthodes d'objets physiquement situés sur un autre serveur. Le transfert se fait le plus souvent à l'aide du protocole HTTP/HTTPS, mais peut également se faire par un autre protocole, comme SMTP. SOAP constitue la couche de fondation de la pile de services web,

fournissant un framework de messagerie de base pour que les couches les plus abstraites puissent être construits. Ce package supporte la publication de la composition établie en tant que nouveau service web.

- **Apache Tomcat** : est un Servlet container développé par ASF (Apache Software Foundation). Tomcat met en oeuvre les spécifications des pages Java Servlet et JavaServer (JSP) à partir de Sun Microsystems, et fournit un environnement de serveur web http, pure Java, pour faire tourner le code Java. L'Apache Tomcat inclut des outils pour la configuration et la gestion, mais peut aussi être configuré en éditant les fichiers de configuration qui sont normalement formatés en XML. De cette manière, *Apache Tomcat* supporte le package *Axis*.
- **Windows XP professionnel** : est le système d'exploitation qui supporte le prototype.

5.3.2 Interface utilisateur

Le prototype qui implémente notre approche a été développé entièrement, sous l'environnent de development eclipse, en Java. Le choix de Java a été effectué en raison du nombre d'APIs existant qui ont servi à la réalisation de notre outil. Comme nous avons évoqué auparavant, nous avons utilisé le parseur *Xerces* pour manipuler les documents XML, l'API *axis-bin-1_4* pour la génération et également le déploiement de services, et l'API *OWL-S -1.1* pour l'analyse et l'écriture des services web sémantiques. Le prototype a été implémenté afin d'expérimenter et de vérifier la faisabilité de notre approche. L'outil développé, Ws-BeC, est doté d'une interface graphique GUI (Graphical User Interface) facile à utiliser qui permet la mise en oeuvre des principaux composants de notre architecture. Dans cette section, nous présentons l'interface graphique de Ws-BeC à travers deux captures d'écran (voir Figures 5.3 et 5.4) : la première montre le panneau de conception pour la description

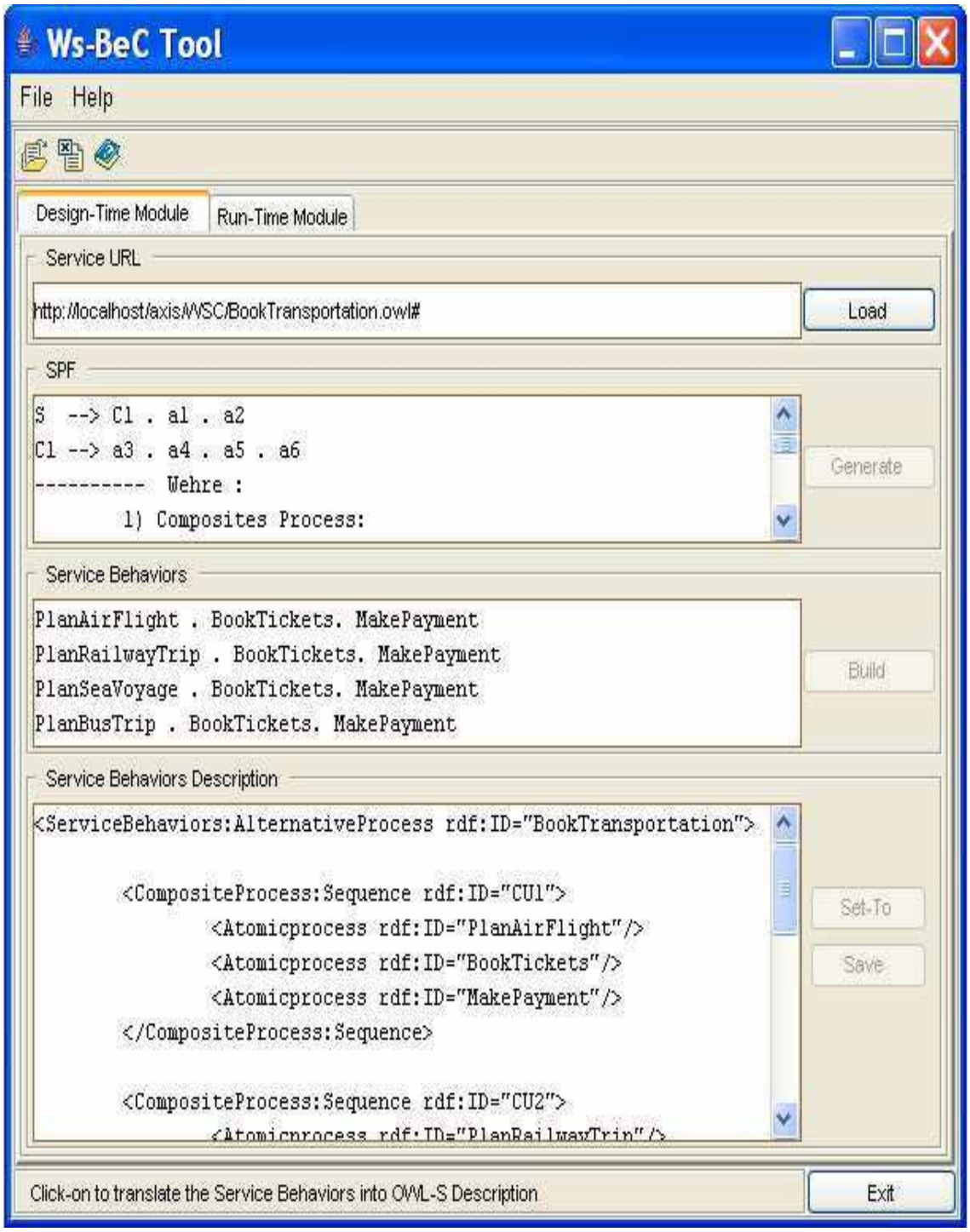


FIGURE 5.3 – Capture d'écran du module de Conception

comportementale de services et la deuxième montre le panneau d'exécution pour la sélection, la composition et l'exécution de services.

– Le premier est utilisé au moment de conception pour la reconfiguration de la

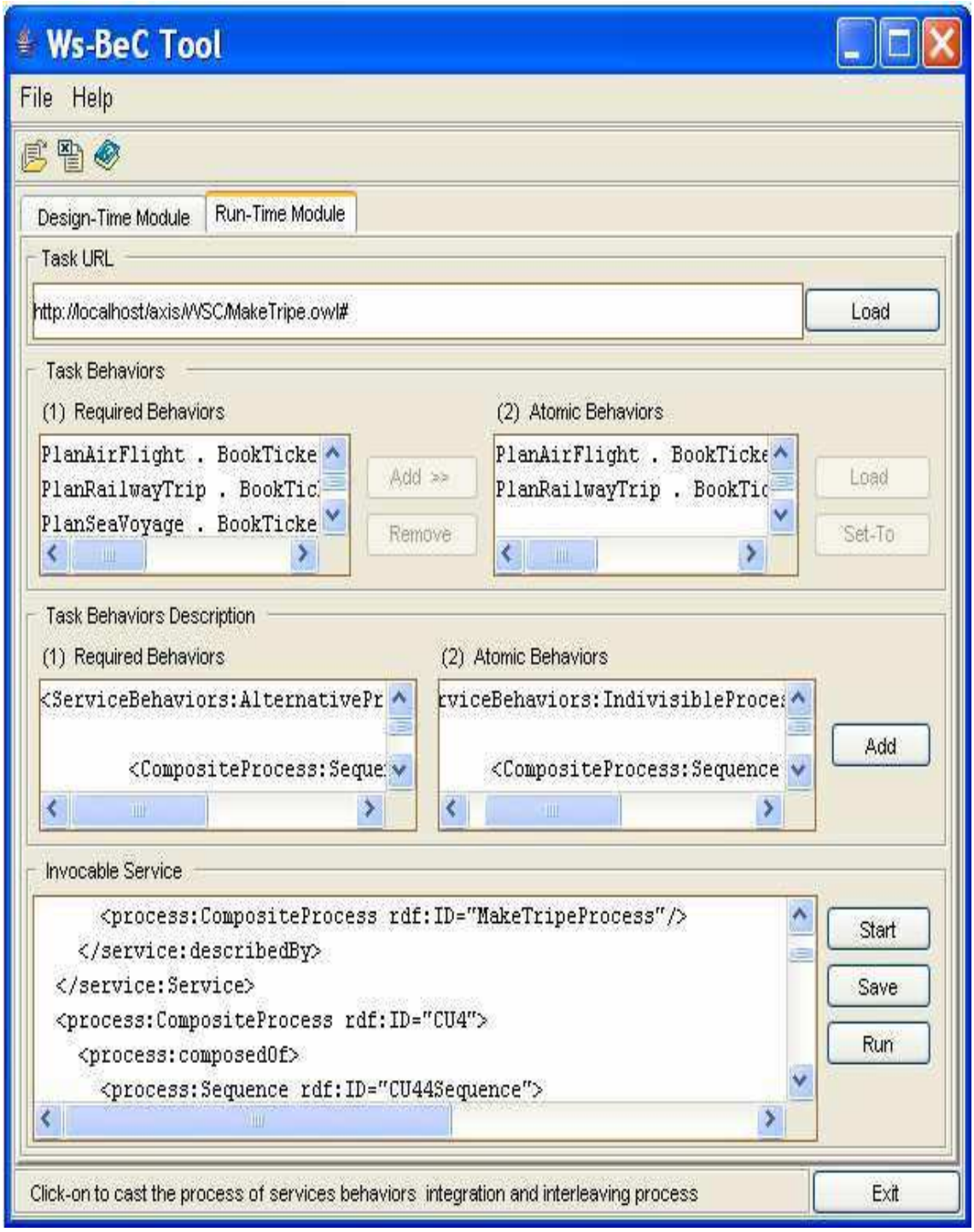


FIGURE 5.4 – Capture d'écran du module d'Exécution

description comportementale de service. Le processus d'analyse et de génération du formalisme SPF commence lorsque l'utilisateur charge un fichier de service OWL-S, puis la nouvelle description comportementale est crée après

avoir généré le GPR et extrait l'ensemble de SBs alternatifs. Ensuite, cette nouvelle description est écrite selon le format du langage OWL-S et publiée par la suite sous forme d'un service, OWL-S, annoté.

– Le deuxième est utilisé au moment d'exécution pour la réalisation et l'invocation de la composition. Le processus de spécification et/ou de chargement des comportements primitifs $PRSB$s et de transformation des RSBs et PSBs dans un format du langage OWL-S commence lorsque l'utilisateur charge un fichier de service OWL-S abstrait. Par la suite, le processus de personnalisation et de composition comportementale est lancé. Si la composition est trouvée, elle sera transformée selon le format du langage OWL-S pour être exécutée et/ou publiée par la suite comme étant un nouveau service exécutable ICS (Invocable composite service).

5.4 Evaluation expérimentale

Dans le cadre de l'évaluation expérimentale du Framework proposé, nous avons effectué plusieurs expériences pour illustrer sa validité et son efficacité, et pour évaluer également ses performances et son évolutivité. Ces expériences sont réalisées par la mise en oeuvre des algorithmes de matching, de sélection et de composition, proposés sur un PC Pentium Dual CPU E2180 2.0 GHz, 2 GB de RAM, 720 GB HDD, sous Windows XP SP3 et avec le langage de programmation Java 2 Version V1.6.0.

D'autre part, pour valider notre approche nous avons utilisé OWLS-TC (OWL-S Service Retrieval Test Collection version 4.0). La base de services OWLS-TC offre 1083 services web sémantiques écrits avec la version OWL-S 1.1. Cependant, l'un des problèmes majeurs rencontrés pour l'évaluation de notre approche de composition service est l'absence d'un benchmark qui concerne à la fois des services atomiques anisi que des services composites. Puisque le développement de ce genre

de benchmark nécessite des données de testes et de nombreuse expérimentations, ce qui consomme beaucoup de temps, nous avons :

- adopté une méthode d'évaluation statistique, pour la sélection non fonctionnelle des services individuels. Etant donnée une RCU avec n tâches (processus atomiques) et k propriétés de QoS, en générant aléatoirement un ensemble de services concrets avec des propriétés QoS et leurs poids associés.
- créé, en utilisant notre framework $Ws - BeC$, notre propre base de services composites pour le traitement comportemental de services afin d'évaluer les différentes stratégies de sélection, d'intégration et de composition de services. La base de services crée contient 150 services web composites en tenant compte pour cette expérimentation seulement les entrées, les sorties et les processus de services.

5.4.1 Objectifs de l'évaluation expérimentale

L'objectif de cette évaluation expérimentale était de caractériser les performances et la qualité du processus d'appariement et de composition de services web. En particulier, nous avons voulu tester l'influence de :

(a)- la chance de réalisation de la composition vis-à-vis :

- la stratégie de la composition adoptée,
- la taille de PCU,
- la taille de RCU,
- et la taille de $PRCU$.

(b)- la performance des algorithmes de matching, de sélection, d'intégration et de cheaveachement vis-à-vis :

- le nombre de tâches (processus) impliquées par une RCU,
- le nombre de services candidats par une RCU,
- et le nombre de paramètres de QoS impliqués par une tâche d'une CU.

5.4.2 Méthodologie d'évaluation et Analyse des résultats obtenus

Comme nous l'avons dit, pour évaluer le prototype Ws-BeC nous avons comparé les résultats obtenus pour les différentes stratégies de compositions. Nous avons étudié l'influence de la taille de SB, et plus précisément la différence entre la taille des PCU et $PRCU$ et la taille du RCU sur la chance de la réalisation de la composition. Pour notre évaluation, nous avons utilisé 200 services atomiques et 13 services composites nous avons considéré jusqu'au cinq niveaux d'imbrication des activités structurées ayant entre 2 et 13 processus. Pour chaque service, nous avons généré plusieurs variantes (qui sont structurellement différentes, mais supposées offrentes une même fonctionnalité) de la manière suivante :

- En variant la taille des PCU,
- En variant la taille des RCU,
- En variant la taille des $PRCU$,
- En variant l'ordre d'exécution des processus atomiques des UC,

En résumé, nous avons généré 10 services composites abstraits et 50 services offerts comme base de test.

En utilisant l'outil $Ws - BeC$, nous avons évalué la faisabilité de l'approche de sélection et de composition comportementale de services. Plus précisément, nous avons utilisé les données des essais présentés ci-dessus, puis nous avons généré les différentes CU des différentes SB de services offertes afin d'être combinées pour réaliser une RCU d'une RSB de la requête utilisateur. Par conséquent, les différentes combinaisons trouvées sont énumérées comme une réalisation d'une RCU donnée pour toute stratégie de composition adoptée.

Dans nos expériences, nous avons varié la taille de RCU d'une RSB, $RCU\,Depth$, de 2 à 64. Ainsi le nombre d'attributs de QoS est varié de 2 à 9, alors que leur valeurs

et leur poids ont été générés aléatoirement. L'étude de valeurs de ces paramètres n'est pas traitée pour cette expérimentation.

La Figure 5.5 montre la chance de réalisation d'une *RCU*, en d'autres termes le nombre de solutions trouvées pour réaliser une *RCU* donnée. Nous avons présenté les solutions trouvées séparément pour les trois stratégies adoptées selection (sélection), integration (intégration), et interleaving (chevauchement). Les expériences montrent que le nombre de combinaisons obtenues par chevauchement est toujours supérieur àux nombres de celles obtenues par les autres solutions, parce qu'elles sont des cas particulières de la combinaison par chevauchement. Théoriquement, le nombre de combinaisons par chevauchement de services est toujours supérieur à la somme des nombres de celles obtenues par sélection et par intégration.

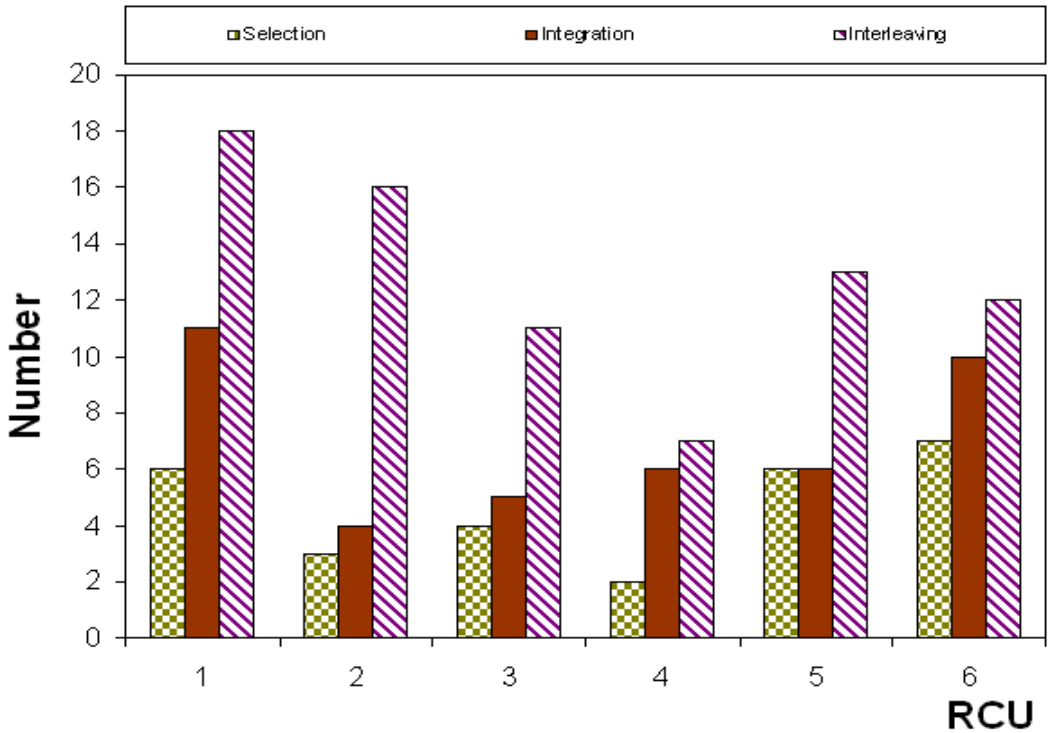


FIGURE 5.5 – La chance de réalisation d'une *RCU*

La Figure 5.6 montre l'influence de la taille des *UC* offertes et requises vis-à-vis au nombre de solutions trouvées pour les différentes stratégies de combinaisons adoptées. Nous avons choisit de présenter les stratégies adoptées ainsi que les tailles

des *CU* en un seul graphe afin de bien indiquer l'influence à la fois de la taille des *RCU* et *PCU* pour les différentes stratégies de composition. Dans ce cas, Les expériences montrent :

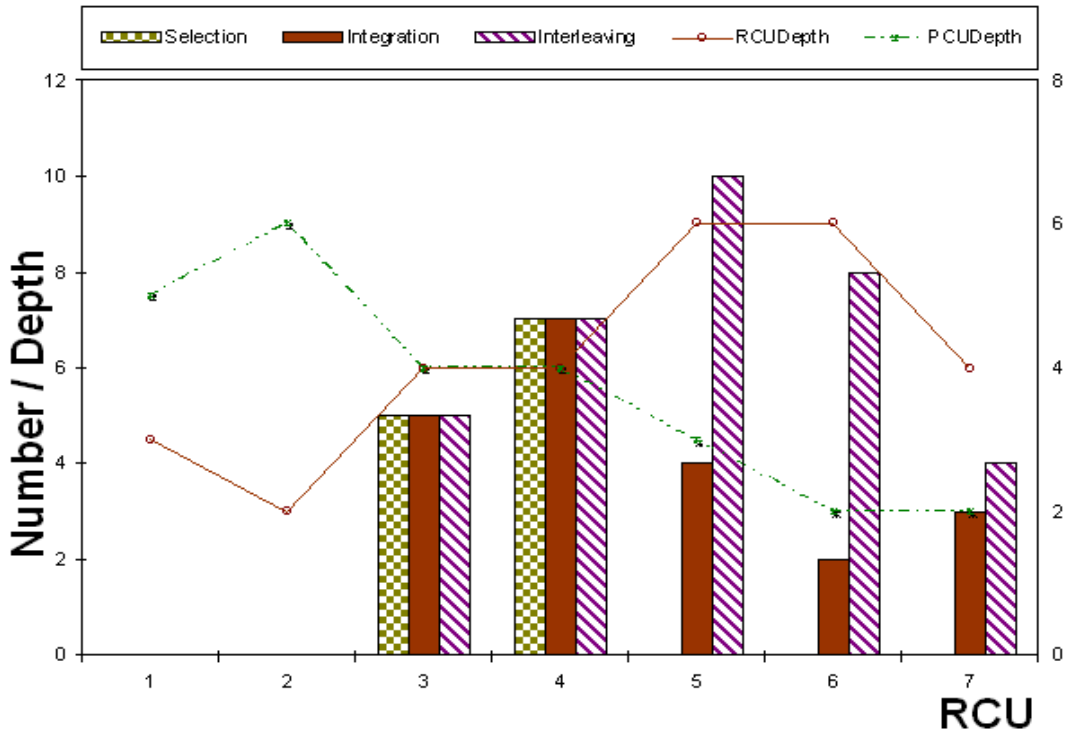


FIGURE 5.6 – L'influence de la taille des *CU*

- si la taille de *RCU* est inférieure à la taille de *PCU*, aucune solution ne peut être trouvée ; c'est-à-dire, la composition ne peut être trouvée, ni par la stratégie de sélection, ni par intégration et ni par chevauchement. Ce qui signifie que, notre solution de composition comportementale via la notion de *PCU* garantie les contraintes d'utilisation de services convenablement comme il est exigé par leur fournisseurs.
- si la taille de *RCU* est égal à la taille de *PCU*, le nombre de solutions est le même pour toutes les stratégies de combinaison. Dans ce cas, la stratégie de combinaison par sélection est retenue alors que celles par chevauchement et par intégration ne les sont pas. En effet, la solution de composition favorise la notion de sélection des *PCU* préconçues, offertes par un seul service, au lieu

de constituer de nouvelles UC.

- si la taille de RCU est supérieure à la taille de PCU, aucune solution n'est trouvée par la stratégie de sélection. Le nombre de solutions obtenues par chevauchement est toujours supérieure ou égale au nombre de solutions obtenues par intégration. Ce qui signifie qu'il y à des situations où des compositions des PCU peuvent ne pas être trouvées par la stratégie de composition par intégration, alors qu'elles peuvent être trouvées par celle de chevauchement. Ainsi, la réalisation d'une RCU par intégration est un cas particulier de celle par chevauchement.

En outre, nous pouvons remarquer que le nombre de solutions obtenues par intégration / chevauchement augmente suite à l'augmentation de la différence entre la taille d'une RCU et celle d'une PCU. La Figure 5.7 montre le nombre de PCU impliquées pour la réalisation d'une RCU. En d'autres termes le nombre de PCUs impliquées par un processus de composition flexible par le biais des stratégies de

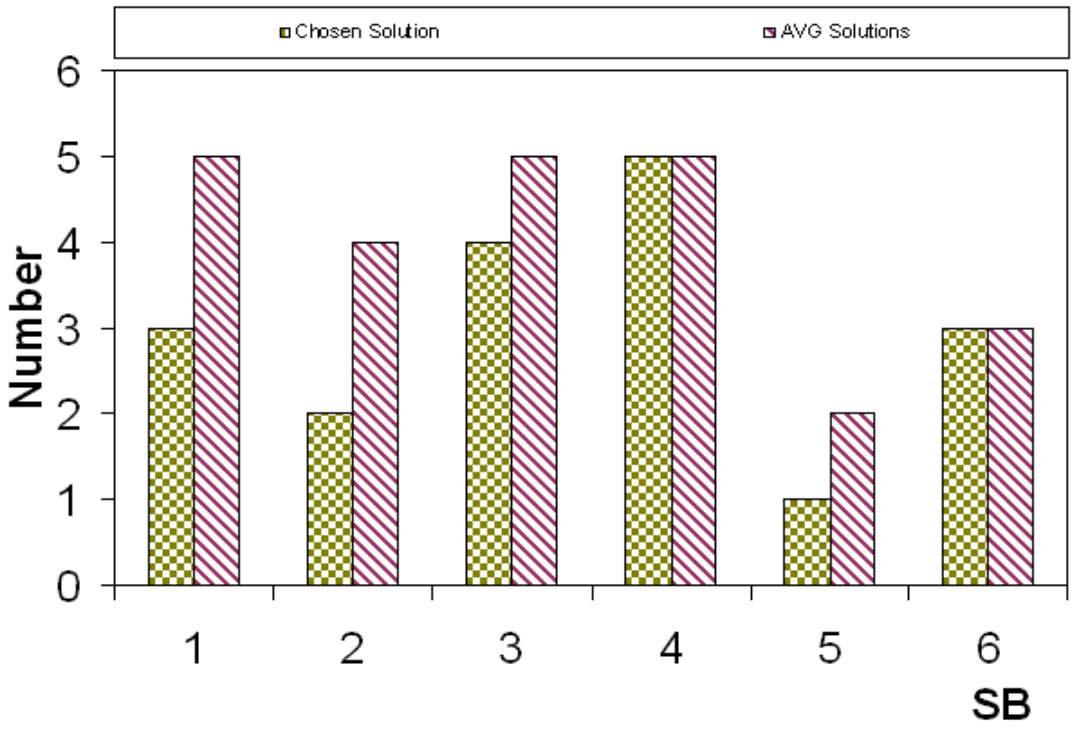


FIGURE 5.7 – Nombre de $PCUs$ impliquées par un SB

sélection, d'intégration et/ou de chevauchement pour la constitution d'une RCU. Nous avons présenté le nombre de solutions trouvées par notre outil Ws-BeC et la moyenne de nombres de solutions trouvées par les trois stratégies de composition comportementales : selection (sélection), integration (intégration), et interleaving (chevauchement). Les expériences montrent que le nombre de PCUs choisit, par l'outil Ws-BeC, est toujours inférieur à la moyen des nombres de PCUs impliquées par les trois stratégies de combinaisons. Ce qui signifie que notre algorithme de composition comportementale minimise toujours le nombre de PCUs et également des PSBs à intégrer et/ou chevaucher. La Figure 5.8 montre le nombre de services impliqués pour la réalisation d'un RSB. En d'autres termes, le nombre de services impliqués par le mécanisme de combinaison comportementale pour la réalisation nécessaire et suffisante de la composition (service composite abstrait) représentée par l'un des RSBs alternatifs. Nous avons présenté le nombre de services choisit par notre outil et la moyenne de nombres de services impliqués par les trois stratégies

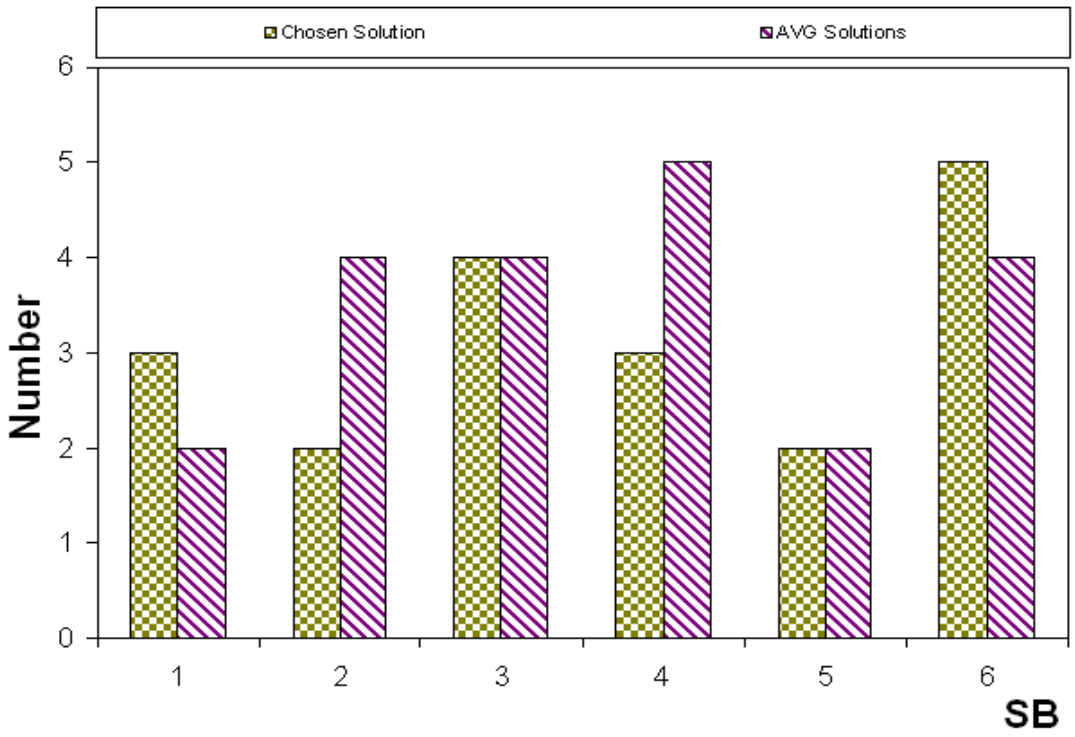


FIGURE 5.8 – Nombre de services impliqués par un SB

de composition comportementale pour la réalisation d'un *RSB* alternatif. Les expériences montrent que le nombre de services choisit n'est pas toujours inférieur à la moyen des nombres de services impliqués par les trois stratégies de combinaisons. Ce qui signifie que notre algorithme de composition comportementale favorise en premier lieu de minimiser le nombre de *PSB*s à combiner, et en deuxième lieu le nombre de services à composer. Par exemple s'il s'agit de deux combinaisons disponibles, l'une considère deux *PCU*s de services différents et l'autres considère trois *PCU*s de même service, alors notre solution va choisir la première combinaison parce qu'elle est la plus optimale du point de vu de *PCU* à combiner, alors que la deuxième est la plus optimale selon la perspective du nombre de services à impliquer. c'est pour cette raison, l'expérimentation obtenue par l'utilisation de l'outil Ws-BeC montre que notre solution ne minimise pas toujours le nombre de services à composer.

La Figure 5.9 montre le temps d'exécution consommé pour la sélection d'un service individuel parmi un ensemble de 480 services candidats. Nous avons présenté

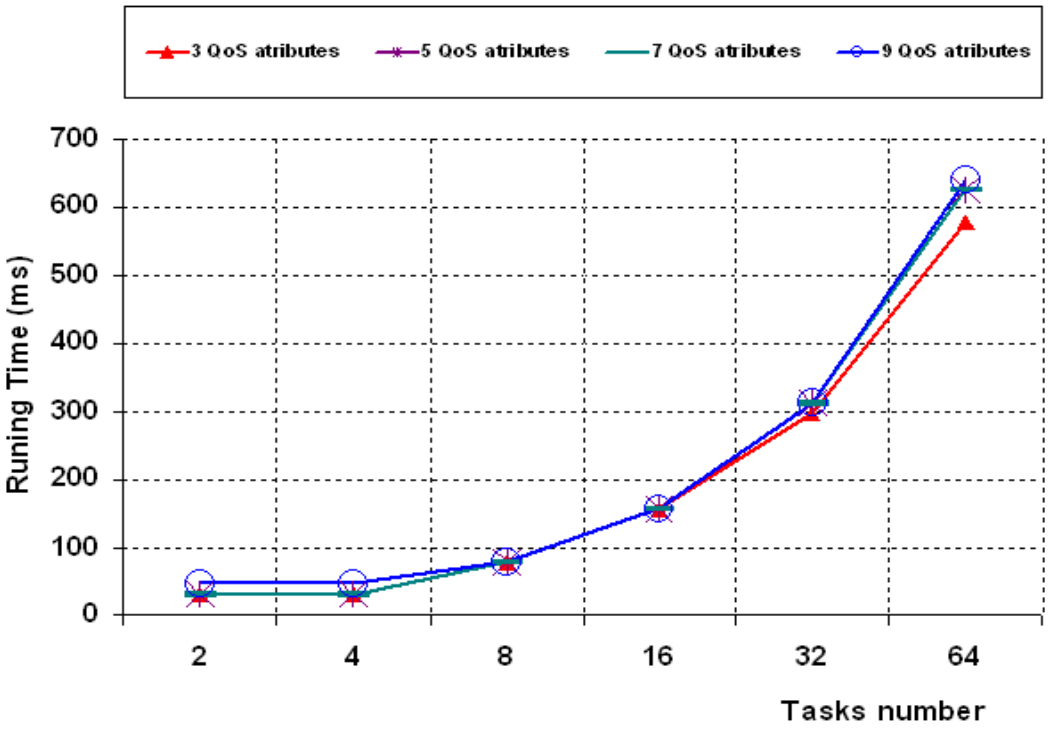


FIGURE 5.9 – Temps d'exécution pour 480 services candidats par classe

le temps d'exécution en indiquant à la fois la taille, le nombre de processus atomiques (services abstraites), et le nombre d'attributs de *QoS* impliqués par une *RCU*. Les expériences montrent que le temps d'exécution est presque le même pour une même *RCU* quel que soit le nombre d'attributs impliqués par cette *RCU*. Ainsi, La Figure montre que le temps d'exécution est d'environ 0.65 s (650 ms) pour la sélection des services d'une *RCU* de taille 64, en considérant un ensemble de services candidats de taille 480 services par classe (30720 services candidats). Alors, l'approche de sélection locale n'est pas influencée par l'augmentation du nombre d'attributs de *QoS* et/ou du nombre de tâches. Donc, c'est un avantage primordial s'il s'agit d'un domaine d'application réel puisque, pour ce genre de domaines, le nombre de tâches et le nombre d'attributs ne dépasse généralement pas 10 et 7 respectivement.

La Figure 5.10 montre le temps d'exécution écoulé pour la sélection des services individuels pour la réalisation d'une *RCU* de taille 32 (c'est-à-dire 32 processus atomiques). Nous avons présenté le temps d'exécution en indiquant à la fois le nombre

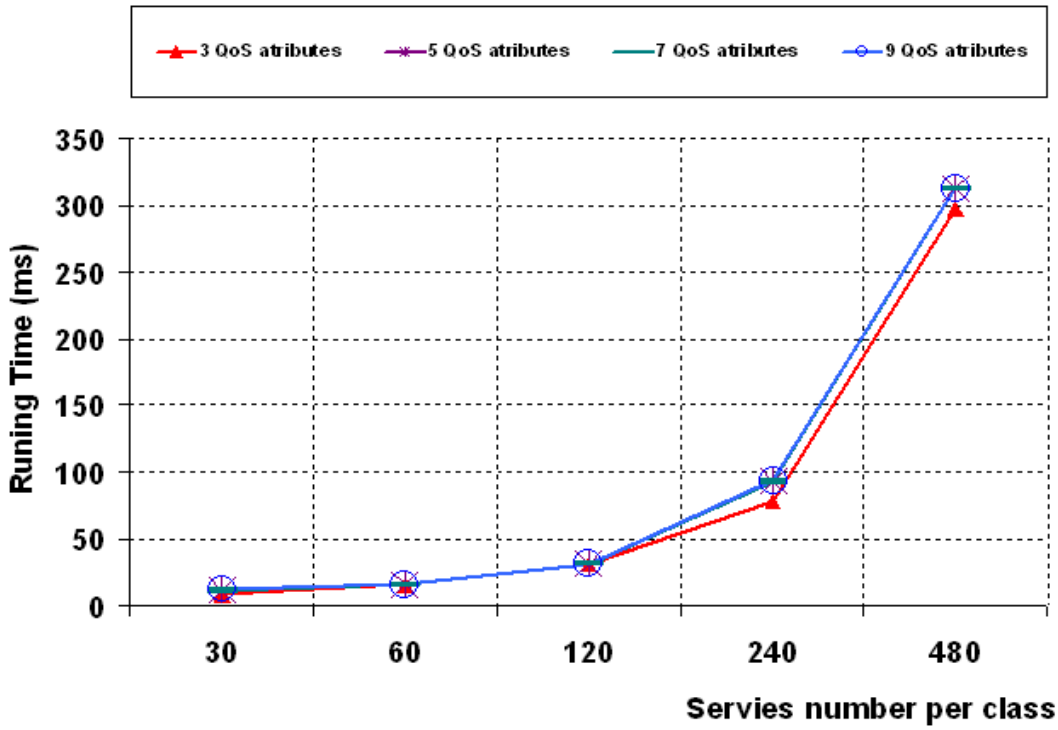


FIGURE 5.10 – Temps d'exécution pour une *RCU* de 32 tâches

de services, individuels, candidats par classe ainsi que le nombre d'attributs de *QoS* impliqués. Les expériences montrent que, pour le même nombre de services candidats par classe, le temps d'exécution est presque le même quel que soit le nombre d'attributs impliqués. La Figure montre que, pour une *RCU* de taille 32 et un ensemble de services de taille 480 services par classe (15360 services candidats), le temps d'exécution pour la sélection des services les plus appropriés est 0.3 s (300 ms). Ce qui signifie que l'algorithme de sélection adopté est très efficace du point de vue temps d'exécution malgré l'augmentation de nombre de services candidats impliqués par classe.

Par conséquence, Les résultats de l'expérience montrent que l'algorithme fonctionne d'une manière très efficace dans toutes les situations pour la sélection, dans un délai raisonnable, des services jugés comme les plus appropriés. L'avantage d'utiliser cet algorithme est plus évident s'il s'agit d'un domaine d'application réel où le nombre de tâches et d'attributs de *QoS* par service est très faible.

5.5 Conclusion

Dans ce chapitre, nous avons présenté le cadre applicatif de nos travaux. Nous avons décrit la mise en oeuvre des différents modules de l'architecture proposée, en précisant le rôle de chaque élément. Ensuite, une présentation été faite des outils utilisés et du prototype que nous avons développé comme support à notre approche. Enfin 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 démonter que l'approche proposée peut contribuer à l'assistance de l'utilisateur pour la spécification de ses préférences non fonctionnelle et également la sélection et la composition appropriée de services. Le prototype que nous décrivons dans ce chapitre a été développé en utilisant les technologies web standards. La mise en oeuvre du prototype a soulevé les questions ci-dessous :

- Quelle stratégie à définir afin de déterminer combien d'attributs de *QoS* il est nécessaire de l'impliquer ?

  Dans la version actuelle du prototype, nous avons adopté une approche de sélection locale de services individuels où les attributs sont impliqués de manière exhaustive, ce qui, dans le cas d'adoption d'une approche de sélection globale ou hybride, peut conduire à une explosion combinatoire. Il est donc nécessaire d'effectuer des expérimentations afin de mesurer le coût véritable induit, puis si nécessaire d'étudier des stratégies afin de diminuer ce coût.
- Les fournisseurs accepteront-ils d'exposer l'aspect comportemental de leurs services web à des fins collaboratifs ?
- Des mesures permettant d'évaluer le sur-coût occasionné par l'exécution de la fonction de sélection s'avèrent nécessaires. Il est important de revoir le modèle de qualité dans le but d'introduire le traitement multicritère soumis par une approche de sélection globale ou hybride.

Le prototype doit être complété par l'amélioration des modules de description et de composition dédiés aux phases de conception et d'exécution proposée dans le framework Ws-BeC.

Dans le chapitre ci-après nous discutons des conclusions et des perspectives générales liées à l'approche elle-même.

Chapitre 6

Conclusion Générale

Sommaire

Tout au long de ce document nous avons présenté le travail dont est issu le framework Ws-BeC. Nous présentons maintenant un bilan et nous dressons plusieurs perspectives envisageables tant sur le plan théorique que sur le plan pratique.

6.1 Synthèse

Dans cette thèse, nous avons proposé une solution pour la sélection et la composition extra-fonctionnelle de services basée non seulement sur des attributs de *QoS* mais aussi sur la spécification comportementale des services composants. L'approche

utilise des techniques d'appariement qui opèrent sur des modèles comportementals de services permettant les correspondances partielles ainsi que la composition appropriée des services disponibls au moment d'exécution. Par conséquent, même s'il n'existe pas un service satisfaisant exactement les besoins des utilisateurs, les plus similaires seront sélectionnés et proposés pour une combinaison par intégration et/ou de chevauchement qui est envisagée afin d'augmenter leur chance de réalisation. Pour ce faire, nous avons formalisé le problème de composition comportementale de services en étendant les grammaires formelles et nous avons proposé des algorithmes à cette fin. En utilisant notre formalisme de représentation de processus de services SPF, nous avons adapté la description comportementale de service afin de simplifier le processus de composition et de garantir la combinaison approprié de service en satisfaisant à la fois les préférences des utilisateurs ainsi que les contraintes des fournisseurs.

Nous avons illustré notre approche de sélection et de composition comportementale, en examinant l'utilisation de techniques d'appariement qualitatif (avec des attributs de QoS) et comportemental pour la sélection et la composition flexible (par intégration et chevauchement) via un scenario d'organisation de voyage.

Puisque nos contributions traitent l'aspect comportementale des services, les algorithmes de reconfiguration, de matchmaking, de selection et de combinaisons proposés qu'ont été appliqués sur des processus métiers de services OWL-S (processus model) pouvant être appliqués aux autres processus métiers de services écrits en un autre langage de composition de services. En outre, notre travail a été centrée sur la flexibilité de la composition comportementale de services par le biais des stratégies de sélection, d'intégration et/ou de chevauchement comportementale et non pas sur le contrôle de la composition et la substitution de services. Les travaux futurs peuvent traiter ce genre d'activités en envisagent la notion de substitution si nécessaire, après avoir contrôlé, non seulement de services individuels mais plutôt

tout un fragment d'un processus de services.

Enfin, nous avons développé le framework $Ws-BeC$ pour mettre en oeuvre notre approche de sélection et de composition de services par le biais d'appariement comportemental à des services OWL-S. Le framework $Ws-BeC$ nous a permis d'évaluer l'efficacité de la solution proposée de personnalisation de composition de services. La validation de notre approche est double. Tout d'abord, nous avons analysé la qualité du processus de combinaisons pour différentes stratégies de composition (sélection, intégration et chevauchement). En second lieu, nous avons testé le temps d'exécution de la méthode de sélection des services individuels en utilisant une approche de sélection locale.

Bien que l'outil d'évaluation permet de créer un classement de services, pour cette évaluation expérimentale, nous avons utilisé uniquement les fonctionnalités de l'outil qui nous permettons d'analyser des fragments du processus de service (SB/CU), en combinant ceux jugés comme plus adéquates. Par conséquent, nous avons évalué la qualité de composition en comparant le résultat obtenu par le prototype des différentes stratégies adoptées. De cette façon, l'outil d'évaluation de la faisabilité de notre méthode de composition comportementale nous a permis de spécifier des exigences comportementale et non-fonctionnelles ,QoS, pour guider le processus de composition selon les préférences des utilisateurs en garantissant les contraintes des fournisseurs. Etant donné que l'outil d'évaluation permet de créer un classement de service, dans nos travaux futurs nous envisageons de l'utiliser pour améliorer l'approche proposée en permettant la substitution des sous processus de services.

En résumé, les principales contributions de cette thèse décrites dans les sous-sections suivantes :

6.1.1 Analyse détaillée de composition de services

Les principaux résultats de cette analyse sont les suivants :

- Un examen des techniques de description et de publication de services en tenant en compte à la fois l'aspect sémantique ainsi que l'aspect non fonctionnel et comportementale ;
- Une étude comparative sur des modèles de composition de services web ;
- Une analyse des langages pour la composition de services selon les différents points de vue pour la composition de services web ;
- Une description des représentations formelles de services ;
- Une analyse des techniques utilisées pour l'appariement de services, qui ont été clarifier selon trois catégories : appariement de service à base des interfaces, de la sémantique et du comportement ;
- Une analyse des techniques courantes pour la composition dynamiques de services, qui ont été clarifier selon deux catégories : vision atomique et vision composite en tenant en compte la notion de sélection et de combinaison de services beaucoup plus complexes.

6.1.2 Techniques et Algorithmes proposés pour la composition de services

Dans cette thèse, une solution pour la composition de service basée à la fois sur les propriétés de QoS, ainsi que sur la spécification comportementale a été développée. En utilisant le formalisme SPF pour la représentation de comportement de services, nous avons :

- Adopté une méthode de sélection locale pour proposer un algorithme pour la sélection de services individuels afin de permettre une concrétisation d'un RCU d'un service abstrait.
- Proposé des algorithmes pour l'appariement et la composition comportementale des processus de services afin de favoriser l'accomplissement de composition et également d'augmenter la chance de réaliser des préférences d'utilisa-

teurs en tenant en compte les contraintes de fournisseurs.

6.1.3 Application de la composition aux services OWL-S

Compte tenu de l'importance des processus de services, dans cette thèse nous avons discuté notre approche pour la composition comportementale de services, en examinant l'utilisation des techniques d'appariement, de sélection et de composition dans le cadre des spécifications comportementales OWL-S. Le processus de composition est composé des étapes suivantes : premièrement, les documents OWL-S à comparer et composer sont transformés en SPFs. Ensuite, la reconfiguration et la génération des différentes SB sont générés, ainsi que les algorithmes d'appariement, et de sélection et composition sont appliqués (en tenant compte des mécanismes d'intégration et de chevauchement des PSB et de décomposition des RSB en cours de la réalisation de la composition). Ensuite, la composition réalisé est transforme en document OWL-S comme un fichier d'orchestration prés d'être exécutés.

6.1.4 Prototype pour la composition personalisée de services web

Nous avons développé un prototype appelé Ws-BeC, qui implémente l'approche proposée. Le framework permet l'exécution des algorithmes pour la sélection et la composition des services. Afin de valider notre approche, le prototype a été testé avec plusieurs scénarios d'application : la sélection et la composition comportementale des services OWL-S pour l'assouplissement de la combinaison et la sélection des services individuels pour tester le temps d'exécution écoulé pour la réalisation d'un RSB.

Dans la section suivante, nous discutons des perspectives envisagées.

6.2 Perspectives

Les travaux futurs peuvent être abordés en envisageant les points suivantes : (i) un référentiel pour la composition de services à la fois avec des attributs QoS et les propriétés comportementales, (ii) Techniques d'indexation pour les différentes combinaisons des processus de services (iii) Classement des combinaisons de processus de services.

6.2.1 Annuaire pour l'appariement comportemental

L'appariement des processus de services doit être implémenté d'une manière très efficace pour soutenir la découverte de service avec un référentiel de services à grand échelle en adressant le problème de comparaison de processus de services pour la sélection et la composition de services. Ainsi, un référentiel est nécessaire pour stocker les processus de services. Par conséquent, notre perspective consiste à adapter notre algorithme d'appariement comportemental de services pour ce genre de référentiels.

6.2.2 Techniques d'indexation pour l'appariement de processus de services

L'adoption des techniques d'indexation est nécessaire pour des référentiels de services à grand échelle puisque le nombre de processus métiers à comparer est beaucoup plus grand. Ainsi, le parcours entier des registres de services pour calculer l'appariement est très coûteux en temps et en espaces. Puisque notre approche de sélection et de composition de service est basée sur l'appariement comportementale, étant donné que de nombreuses approches ont été proposées dont la littérature pour l'indexation des données complexes (comme les graphes par exemple), un travail futur peut être réalisé pour déterminer quel est l'index le plus approprié pour

l'utiliser ou l'adapter avec nos descriptions comportementales proposées afin de suggérer un algorithme d'appariement pour la découverte, la sélection et la composition comportementale.

6.2.3 Classification de processus de services

L'appariement et la combinaison des processus de services est l'opération fondamentale pour la recherche et la collaboration, étant donnée un besoin donné, des partenaires prometteurs pour certains genre d'objectifs. Il s'agit d'un processus commun à plusieurs scénarios dans l'ère d'Internet, s'étendant de e-commerce (commerce électronique) aux services web, au grid computing, à la gestion des ressources humaines, aux services de rencontres (dating and meeting services), a l'informatique pair-à-pair.

Comme ce processus peut conduire à diverses correspondances possibles, le classement devient centrale, pour fournir une liste de partenaires potentiels ordonnée selon certains critères. Ainsi, notre travail futur est de développer une technique de classement de services par domaine spécifique, en se basant sur la notion d'appariement de processus de services qui prennent en compte d'autres paramètres comme le temps d'exécution, la disponibilité de service, etc. Tout ce la pour améliorer de plus en plus le processus de composition personnalisée de services web.

6.2.4 Validation expérimentale

Nous avons développé un prototype pour la validation expérimentalement de notre proposition de composition de services. Tandis qu'il manque la validation avec une véritable application industrielle.

D'autre part, notre modèle prône pour la non-composition préalable entre les services participant d'une composition. Il reste à prouver, dans le cadre d'une application réelle que les fournisseurs de services acceptent d'exposer les propriétés

comportementales de leurs services.

Comme il est indiqué par la figure 6.1, les différentes perspectives d'amélioration de notre framework $Ws - BeC$ de sélection et de composition sont, indiquées

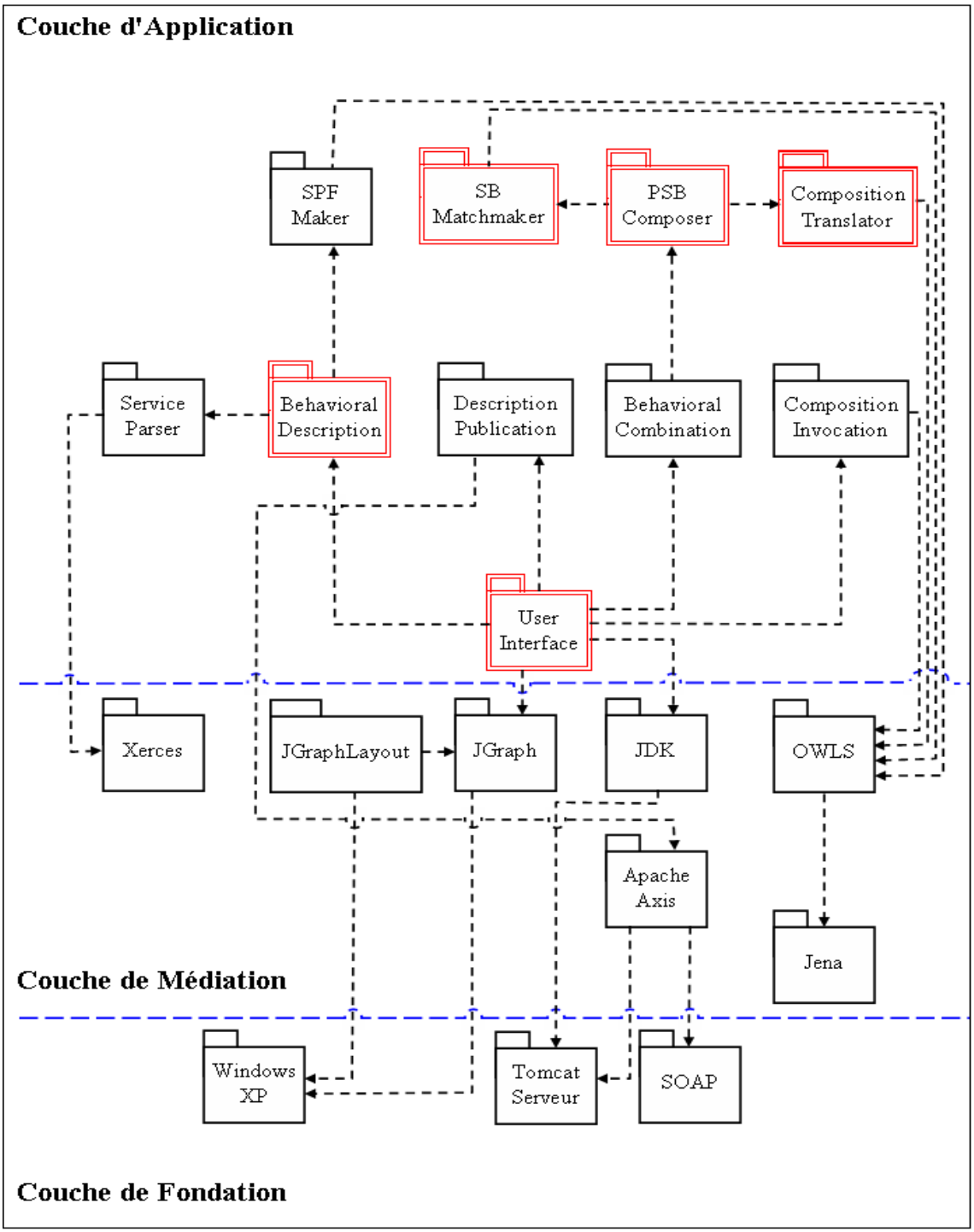


FIGURE 6.1 – Perspectives d'Amélioration du Framework $Ws - BeC$

par les packages de couleur rouge : Le package *PSB Composer*, *SB Matchmaker*, *Composition Translator*, *Behavioral description* et le package *User Interface*.

Publications

Les publications issues de la thèse.

1. M. Mekour and S. M. Benslimane, "SP4PS : service process rewriting for efficient and proper web services composition". *International Journal of Web Engineering and Technology*, 8 (4) (2013) 327-346.
2. M. Mekour and S. M. Benslimane, "BRC : Behavior Reconfiguration and Combination to Enhance the Dynamic Semantic Web Services Composition". *Seventh International Conference on Next Generation Web Services Practices*, (2011) 410-415.
3. M. Mekour and S. M. Benslimane, "BH : Behavioral Handling to Enhance Powerfully and Usefully the Dynamic Semantic Web Services Composition". *The First International Conference on Model and Data Engineering*, (2011) 50-61.

Bibliographie

[Aalst., 1999] W. van der Aalst. Interorganizational workflows : An approach based on message sequence charts and petri nets. 1999.

[Aalst., 2002] W. Van der Aalst and K. V. Hee. Workflow management- models, methods, and systems. In MIT Press, 2002.

[Abhijit et al., 2004] Abhijit Patil, Swapna Oundhakar, A. Sheth, and Kunal Verna. Meteors web service annotation framework. In Proc. of WWW Conference, 2004.

[Andrews et al., 2003] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S.Weerawarana. Business process execution language for web services, version 1.1. In Standards proposal by BEA Systems, International Business Machines Corporation, and Microsoft Corporation, 2003.

[Arkin et al., 2002] A. Arkin and S. et al. Askary. Web service choreography interface (wsci) 1.0. W3C - Web Services Choreography Working Group, 2002.

[Aversano et al., 2004] Aversano, L., Canfora, G. and Ciampi, A. (2004) An algorithm for web service discovery through their composition, Second International Conference on Web Services, pp.332-339.

[Banerji et al., 2002] A. Banerji, C. Bartolini, D. Beringer, V. Chopella, K. Govindarajan, A. Karp, H. Kuno, M. Lemon, G. Pogossiants, S. Sharma, and S. Williams. Web services conversation language (wscl) 1.0. In W3C, 2002.

[Bansal et Vidal., 2003] S. S. Bansal and J. M. Vidal. Matchmaking of web services based on the DAML-S service model. In Proc. of AAMAS, pages 926-927, 2003.

[Barhamgi et al., 2010] M. Barhamgi, D. Benslimane, and B. Medjahed, "A Query Rewriting Approach for Web Service Composition". *IEEE Transactions on Services Computing*, 3(3) :2010, pp. 206-222.

[Barros et al., 2006] A. Barros, M. Dumas, and P. Oaks. Standards for web service choreography and orchestration : Status and perspectives. In Proc. of Business Process Management Workshops, 2006.

[Benatallah et al., 2005] B. Benatallah, R. Dijkman, M. Dumas, and Z. Maamar. Service Oriented Software System Engineering, chapter Service Composition : Concepts, Techniques, Tools and Trends. IDEA Group Publishing, 2005.

[Benatallah et al., 2004] B. Benatallah, F. Casati, and F. Toumani. Analysis and management of web services protocols. In Proc. of ER, 2004.

[Benatallah et al., 2003] B. Benatallah, F. Casati, F. Toumani, and R. Hamadi. Conceptual Modeling of Web Services Conversations. In CAiSE. Proc. of the 15th Intl. Conf. On Advanced Information Systems Engineering (CAiSE'03), no. 2681 in Lecture Notes in Computer Science, Klagenfurt/Velden, Austria., Springer, 16-18 June 2003.

[Benatallah et al., 2003] B. Benatallah, M.S. Hacid, C. Rey, and F. Toumani. Semantic reasoning for web services discovery. In Proc. of ESSW, 2003.

[Benatallah et al., 2002] B. Benatallah, M. Dumas, and Z. Maamar. Definition and Execution of Composite Web Services the SELF-SERV Projet. IEEE Computer Society Technical Committee on Data Engineering Bulletin, 25(4), 2002.

[Berardi et al., 2003] D. Berardi, F. ROSA, L. SANTIS, and M. MECELLA. Finite state automata as conceptual model for e-services. 2003.

[Bergstra et al., 2001] J.A. Bergstra, A. Ponse, and S.A (eds) Smolka. Handbook of process algebra. North Holland, Elsevier, 2001.

[Bernstein et al., 2002] A. Bernstein and M. Klein. Towards high-precision service retrieval. In Proc. of ISWC, 2002.

[Bordeaux et al., 2004] L. Bordeaux, G. Salan, D Berardi, and M. Mecella. When are two web services compatible ? In Proc. of TES, 2004.

[Brambilla et al., 2002] M. Brambilla, S. Ceri, S. Comai, P. Fraternali, and I. Manolescou. Model-driven Specification of Web Services Composition and Integration with Data-intensive Web Applications. IEEE Computer Society Technical Committee on Data Engineering Bulletin, November 2002.

[Bunke., 2000] H. Bunke. Graph matching : Theoretical foundations, algorithms, and applications. pages 82-88, 2000.

[Canfora et al., 2008] G. Canfora, D.I. Penta, M.R. Esposito, M.L. Villani, A framework for QoS-aware binding and re-binding of composite web services, Journal of Systems and Software, 81 (10) (2008) 1754-1769.

[Canfora et al., 2005] G. Canfora, M. D. Penta, R. Esposito, M. L. Villani, "An Approach for QoSaware Service Composition based on Genetic Algorithms".

Sixth Conference on Genetic and evolutionary computation, June 2005, pp. 1069-1075.

[Cao et al., 2012] B. Cao, J. Yin, S. Deng, D. Wang, and Z. Wu, Graph-based workflow recommendation : on improving business process modeling, in Proceedings of the 21st ACM international conference on Information and knowledge management, ser. CIKM '12, pp. 1527-1531, 2012.

[Cao et al., 2013] Bin Cao, Jianwei Yin, Ying Li, Shuiguang Deng. A Maximal Common Subgraph Based Method for Process Retrieval. IEEE 20th International Conference on Web Services, 316- 323, 2013.

[Cardoso et Sheth., 2003] J. Cardoso and A. Sheth. Semantic e-workflow composition. Journal of Intelligent Information Systems, 21 :191-225, 2003.

[Cassar et al., 2013] G. Cassar, P. Barnaghi, K. Moessner, Probabilistic matchmaking methods for automated service discovery, IEEE Trans. Serv. Comput. 99, 1-14, 2013.

[Chakraborty et al., 2005] D. Chakraborty, A. Joshi, T. Finin, and Y. Yesha, Service Composition for Mobile Environments. *Journal on Mobile Networking and Applications*, 10(4) :2005, pp. 435-451.

[Charlet et al., 2003] J. Charlet, P. Laublet, and C. Reynaud. Web sémantique Rapport final. Rapport de recherche, Action spécifique 32 CNRS / STIC, FRANCE, Octobre 2003.

[Chen et al., 2011] Li Chen, Zi-lin Song, Ying Zhang and Zhuang Miao. A Method of Web Service Matchmaking Based on Snippets. AISS : Advances in Information Sciences and Service Sciences ; pp. 250 -258, 2011.

[Chiat et al., 2004] L. C. Chiat, L. Huang, and J. Xie. Matchmaking for semantic web services. In Proc of Services Computing, 2004 IEEE International Conference on (SCC'04), 2004.

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

[Constantinescu et Faltings, 2004] Constantinescu, I. and Faltings, B. (2004) Large scale, type-compatible service composition, Second International Conference on Web Services, pp.506-513.

[David et al., 2004] M. David, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, M. Solanki, N. Srinivasan,

and K. Sycara. Bringing semantics to web services : The owl-s approach. In Proc of the First International Workshop on Semantic Web Services and Web Process Composition, 2004.

[Dong et al., 2004] L. Dong, A. Halevy, J. Madhavan, E. Nemes, , and J. Zhang. Similarity search for web services. In Proc. of VLDB, 2004.

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

[Dustdar et Schreiner., 2007] G. Gardarin, Xml, des bases de donnees aux services web, Dunod, 2007.

[ESOA., 2005] , Extending the service oriented architecture, Business Integration Journal (2005), 18-21.

[Emerich., 2000] W. Emerich. Software Engineering and Middleware : A roadmap. Proceedings of the conference on The future of Software Engineering, Limerick, Ireland, pages 117-129, 2000.

[Emmerich et Kaveh., 2001] W. Emmerich and N. Kaveh, Component technologies : Java beans, com, corba, rmi, ejb and the corba component model, SIGSOFT Software Engineering Notes 26 (2001), no. 5, 311-312.

[Fan et al., 2010] G. Fan, H. Yu, L. Chen, C. Yu, "An Approach to Analyzing User Preference based Dynamic Service Composition". *Journal of Software*, 5(9) :2010, pp. 982-989.

[Farrag et al., 2011] Farrag Tamer A, Saleh AI, Ali HA. ASWSC : automatic semantic web services classifier based on semantic relations. In : Proceedings of the IEEE seventh international conference on computer engineering and systems, 2011.

[Farrag et al., 2012] Farrag Tamer A, Saleh Ahmed I, Ali Hesham A. Towards SWSs discovery : mapping fromWSDL to OWL-S based on ontology search and standardization engine. IEEE transactions on knowledge and data engineering, 13 February 2012. IEEE computer Society Digital Library. IEEE Computer Society ; 25 (5) 2012.

[Gamha et al., 2008] Y. Gamha, N. Bennacer, and G. V. Naquet, "A framework for the semantic composition of web services handling user constraints". *Sixth IEEE International Conference on Web Services*, 2008, pp. 228-237.

[Ganapathy et al., 2010] G. Ganapathy, C. Surianarayanan, "Identification of Candidate Services for Optimization of Web Service Composition". *World Congress on Engineering*, 2010, pp. 448-453.

[Gerbe et al., 1998] O. Gerbe, R. Keller, , and G. Mineau. Conceptual graphs for representing business processes in corporate memories. 1998.

[Hamadi et al., 2003] R. Hamadi and B. Benatallah. A petri net-based model for web service composition. In Fourteenth Australasian Database Conference-ADC, pages 191-200, 2003.

[Hamdi et al., 2014] Hamdi Yahyaoui, Mohamed Almull, Hala S. Own. A novel non-functional matchmaking approach between fuzzy user queries and real world web services based on rough sets. Future Generation Computer Systems 35, pp 27-38, 2014.

[Harney et al., 2008] J. Harney, P. Doshi, "Selective Querying For Adapting Web Service Compositions Using the Value of Changed Information". *IEEE Transactions on Services Computing*, 1(3) :2008, pp. 169-185.

[Hatzi et al., 2012] Hatzi, O., Vrakas, D., Nikolaidou, M., Bassiliades, N., Anagnostopoulos, D. and Vlahavas, I. (2012) An integrated approach to automated semantic web service composition through planning, IEEE Transactions on Services Computing, Vol. 5, No. 3, pp.319-332.

[Hoare., 1985] C. A. R. Hoare. Communicating sequential processes. Prentice Hall International, 1985.

[Hopcroft et al., 2001] R. Hopcroft, J. E.and Motwani and J. D. Ullman. Introduction to automata theory, languages, and computation. 2001.

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

[Hwang et al., 2007] S. Y. Hwang, E. P. Lim, C. H. Lee, and C. H. Chen,"On composing a reliable composite web service : a study of dynamic web service selection". *Fifth IEEE International Conference on Web Service*, 2007, pp. 184-189.

[Jamal et al., 2013] Jamal, B., Hamdi, Y., Melissa, K. and Zakaria, M. (2013) Symbolic model checking composite web services using operational and control behaviors, Expert Systems with Applications, Vol. 40, No. 2, pp.508-522.

[Kavantzas et Burdett., 2004] N. Kavantzas and D. Burdett. Ws choreography model overview. W3C - Web Services Choreography Working Group, 2004.

[Kavantzas et al., 2004] N. Kavantzas, D. Burdett, G. Ritzinger, T. Fletcher, and Y. Lafon. Web services choreography description language version 1.0. W3C - Web Services Choreography Working Group, 2004.

[Kawamura et al., 2003] T. Kawamura, J.A. De Blasio, T. Hasegawa, M. Paolucci, and K. Sycara. A preliminary report of a public experiment of a semantic service matchmaker combined with a uddi business registry. In Proc. of ICSOC, 2003.

[Kokash et al., 2006] N. Kokash, W. van den Heuvel, and V. D'Andrea. Leveraging web services discovery with customizable hybrid matching. In Proc. of ICSOC, pages 522-528, 2006.

[Kourtesis et Paraskakis., 2008] D. Kourtesis and I. Paraskakis. Combining sawsdl, owl-dl and uddi for semantically enhanced web service discovery. In Proceedings of the 5th European semantic web conference on The semantic web : research and applications, ESWC'08, pages 614-628, Berlin, Heidelberg, 2008. Springer-Verlag.

[Kritikos et al., 2013] K. Kritikos, D. Plexousakis, Novel optimal and scalable non-functional service matchmaking techniques, IEEE Trans. Serv. Comput. 99, 1-14, 2013.

[Kuang et al., 2006] Li Kuang, Ying Li, Shuiguang Deng, Jian Wu, Wei Shi, and Zhaohui Wu. Expressing service and query behavior using pi-calculus for matchmaking. In Proc of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, pages 629-632, 2006.

[Kwon et Lee, 2012] Kwon, J. and Lee, D. (2012) Non-redundant web services composition based on a two- phase algorithm, Data & Knowledge Engineering, Vol. 71, No. 1, pp.69-91.

[Lara et al., 2004] R. Lara, D. Roman, A. Polleres, and D. Fensel. A conceptual comparison of WSMO and OWL-S. In Proceedings of the European Conference on Web Services (ECOWS 2004), 11 2004.

[Lecue et al., 2009] F. Lecue, A. Deltiel, A. Leger, "Web Service Composition as a Composition of Valid and Robust Semantic Links". *International Journal of Cooperative Information Systems*, 18(1) :2009, pp.1-62.

[Li et al., 2012] Li, B., Xu, Y., Wu, J. and Zhu, J. (2012) A petri-net and QoS based model for automatic web service composition, Journal of Software, Vol. 7, No. 1, pp.149-155.

[Li et al., 2011] Li, X., Fan, Y., Sheng, Q.Z., Maamar, Z. and Zhu, H. (2011) A petri net approach to analyzing behavioral compatibility and similarity of web

services, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 41, No. 3, pp.510-521.

[Li et al., 2003] L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. In Proc of WWW 2003, pages 331-339, 2003.

[Iosif et al., 2010] E. Iosif, A. Potamianos, Unsupervised semantic similarity computation between terms usingweb documents, IEEE Trans. KnowledgeData Eng. 22 (11) (2010) 1637-1647.

[Luo et al.,] J.Z. Luo, J.Y Zhou, Z.A. Wu, "An adaptive algorithm for QoS-aware service composition in grid environments". *Service Oriented Computing and Applications*, 3(3) :2009, pp. 217-226.

[Mahleko et al., 2006] B. Mahleko and A. Wombacher. Indexing business processes based on annotated finite state automata. In Proc. of The IEEE International Conference onWeb Services (ICWS'06), pages 303-311, 2006.

[Majithia et al., 2004] S. Majithia, D. W. Walker, and W. A. Gray. "A framework for automated service composition in service-oriented architecture". *First European Semantic Web Symposium*, 2004, pp. 265-283.

[Martello et Toth., 1987] S. Martello and P. Toth, Algorithms for Knapsack problems, Discrete Math, 31 (1987) 70-79.

[Mazzara et al., 2006] M. Mazzara and R. Lucchi. A pi-calculus based semantics for WS-BPEL. Journal of Logic and Algebraic Programming, 2006.

[Mekour et Benslimane, 2013] M. Mekour and S. M. Benslimane, "SP4PS : service process rewriting for efficient and proper web services composition". *International Journal of Web Engineering and Technology*, 8 (4) (2013) 327-346 .

[Mekour et Benslimane, 2011a] M. Mekour and S. M. Benslimane, "BRC : Behavior Reconfiguration and Combination to Enhance the Dynamic Semantic Web Services Composition". *Seventh International Conference on Next Generation Web Services Practices*, (2011) 410-415.

[Mekour et Benslimane, 2011b] M. Mekour and S. M. Benslimane, "BH : Behavioral Handling to Enhance Powerfully and Usefully the Dynamic Semantic Web Services Composition". *The First International Conference on Model and Data Engineering*, (2011) 50-61.

[Mendling et al., 2005] J. Mendling and J. Ziemann. Transformation of bpel processes to epcs. In Proc. of the 4th GI Workshop on Event-Driven Process Chains (EPK2005), 2005.

[Milner., 1982] R. Milner. A calculus of communicating systems. Springer-Verlag NewYork, Inc, 1982.

[Milner., 1999] R.Milner. Communicating and mobile systems : the p-calculus. Cambridge University Press, 1999.

[Misra., 2010] Jayadev Misra. Orc tutorial lectures. Technical report, Department of Computer Science, University of Texas at Austin, 2010.

[OWL-S., 2004] OWL-S Coalition. Bringing semantics to web services : The OWL-S Approach. In Semantic Web Services and Web Process Composition Workshop, volume 3387 of Lecture Notes in Computer Science, pages 26-42, San Diego, CA, USA, July 2004.

[Orc, 2011] Orc. Orc reference manual v2.0.2. Technical report, The University of Texas at Austin, 2011.

[Ould et al., 2006] N. Ould, A.M'Bareck, and S. Tata. Bpel behavioral abstraction and matching. In Business Process Management Workshops, pages 495-506, 2006.

[Paolucci et al., 2002] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara. Semantic matching of web services capabilities. In Proc. of ISWC, 2002.

[Papazoglou et Georgakopoulos., 2003] M. Papazoglou and D. Georgakopoulos. Introduction, service-oriented computing. Commun. ACM, 46(10) :24-28, 2003.

[Papazoglou et Van., 2007] M. P. Papazoglou and W-J. Van Den Heuvel, Service oriented architectures : Approaches, technologies and research issues, IEEE Internet Computing 16 (2007), no. 3, 389-415.

[Papazoglou., 2003] M. P. Papazoglou, Service-oriented computing : Concepts, characteristics and directions, Proceedings of the 4th international conference on web information systems engineering (wise'03), 2003, pp. 3-12.

[Peterson., 1981] J. L. Peterson. Petri net theory and the modeling of systems. In Prentice-Hall, 1981.

[Piccinelli et al., 2001] G. Piccinelli, G. Di Vitantonio, and L. Mokrushin. Dynamic service aggregation in electronic marketplaces. Computer Networks, 2(37), 2001.

[Plebani et al., 2009] P. Plebani, B. Pernici, URBE : web service retrieval based on similarity evaluation, IEEE Trans. Knowledge Data Eng. 21 (11), 1629-1642, 2009.

[Quan et al., 2012] Quan, Z., Sheng, M., Zakaria, Y., Lina, S., Claudia, S. and Scott, B. (2012) Behavior modeling and automated verification of web services, Information Sciences.

[Rathore et Suman, 2011] M. Rathore, and U. Suman, "A Quality of Service Broker Based Process Model for Dynamic Web Service Composition". *Journal of Computer Science*, 7 (8) (2011) 1267-1274.

[Saboua et Panb., 2007] M. Saboua and J. Panb. Towards semantically enhanced web service repositories. Web Semantics : Science, Services and Agents on the World Wide Web, 5 :142-150, 2007.

[Salaün et al., 2012] Salaun, G., Bultan, T. and Roohi, N. (2012) Realizability of choreographies using process algebra encodings, IEEE Transactions on Services Computing, Vol. 5, No. 3, pp.290-304.

[Sana et al., 2013] Sana Sellami, Omar Boucelma. Towards a Flexible Schema Matching Approach for Semantic Web Service Discovery. IEEE 20th International Conference on Web Services, 611- 612, 2013.

[Segev et al., 2009] A. Segev, E. Toch, Context based matching and ranking of web services for composition, IEEE Trans. Serv. Comput. 2 (3), 210-222, 2009.

[Schmidt et Reussner., 2002] H. Schmidt and R. Reussner. Generating adapters for concurrent component protocol synchronisation. In FMOODS '02 : Proceedings of the IFIP TC6/WG6.1, pages 213-229, 2002.

[Shen et al., 2005] Z. Shen and J. Su. Web services discovery based on behavior signatures. In Proc. of IEEE SCC, 2005.

[Sherry et Zhao, 2012] Sherry, X.S. and Zhao, J. (2012) A decomposition-based approach for service composition with global QoS guarantees, Information Sciences, Vol. 199, pp.138-153.

[Sivashanmugam et al., 2003] K. Sivashanmugam, K. Verma, A. Sheth, and J. Miller. Adding semantics to web services standards. In Proc of the the 1st International Conference on Web Services (ICWS'03), 2003.

[Stal., 2002] M. Stal, Web services : Beyond component-based computing, Communications of the ACM 45 (2002), no. 10, 71-76.

[Stroulia et al., 2005] E. Stroulia and Y. Wang. Structural and semantic matching for assessing web-service similarity. Int. J. Cooperative Inf. Syst., 14(4) :407-438, 2005.

[Tamer et al., 2013] Tamer A. Farrag, Ahmed I. Saleh b, H.A. Ali. Semantic web services matchmaking : Semantic distance-based approach. Computers and Electrical Engineering 39, 497-511, 2013.

[Tang and Zou, 2010] R. Tang, Y. Zou, "An Approach for Mining Web Service Composition Patterns from Execution Logs". *Twelfth IEEE International Symposium on Web Systems Evolution*, September 2010, pp. 53-62.

[Trastour et al., 2001] D. Trastour, C. Bartolini, and J. Gonzalez-Castillo. A semantic web approach to service description for matchmaking of services. In Proc. of SWWS, 2001.

[Wada et al., 2012] H. Wada, J. Suzuki, Y. Yamano, K. Oba, E^3 : A Multiobjective Optimization Framework for SLA-Aware Service Composition, IEEE Transactions On Services Computing, 5 (3) (2012) 358-372.

[Wang et al., 2010] X. Wang, Z. Wang, X. Xu, A. Liu, D. Chu, "A Service Composition Approach for the Fulfillment of Temporally Sequential Requirements". *IEEE Sixth World Congress on Services*, 2010, pp. 559-565.

[Weikum., 1999] G. Weikum. Towards guaranteed quality and dependability of information services. 1999.

[Wombacher et al., 2004] A. Wombacher, B. Mahleko, P. Fankhauser, and E. Neuhold. Matchmaking for business processes based on choreographies. In Proc. of EEE, 2004.

[Wu et al., 2005] J. Wu and Z. Wu. Similarity-based web service matching. In Proc. of IEEE SCC, 2005.

[Xia et al., 2012] Y. Xia, Y. Liu, J. Liu, and Q. Zhu, "Modeling and Performance Evaluation of BPEL Processes : A Stochastic-Petri-Net-Based Approach". *In IEEE Transactions on Systems*, Man, and Cybernetics, 42(2) :2012, pp. 503-510.

[Yu et Zhang., 2007] T. Yu, Y. Zhang, K.J. Lin, Efficient algorithms for web services selection with end-to-end QoS constraints, ACM Transactions on the Web, 1 (1) (2007) 1-26.

[Yu et Zhang., 2004] R. Dijkman and M. Dumas. Service-oriented Design : A Multiviewpoint Approach. Technical Report CTIT Technical Report Series No. 04-09, Centre for Telematics and Information Technology, University of Twente, The Netherlands, February 2004.

[Zeng et al., 2004] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. "QoS-aware middleware for web services composition". *IEEE Transactions on Software Engineering*, 30 (5) (2004) 311-327.

Résumé

Le paradigme AOS (architecture orientée service) est devenu un standard pour la conception et le développement des applications distribuées à base de services web. La composition de services implique la construction de services à valeur ajoutée très souvent par la découverte, l'intégration et l'exécution des services préexistants. Cela peut être effectué de manière à ce que des services préexistants soient orchestrés en un ou plusieurs nouveaux services qui répondent mieux à une application composite. Malgré tous les avantages qu'elles apportent en termes d'interopérabilité et de réutilisation, les solutions de développement associées au paradigme AOS sont destinées aux programmeurs et restent difficiles à comprendre par le monde de l'entreprise. Pour être en phase avec le monde de l'entreprise, les applications à base de services web doivent être décrites en termes de propriétés extra-fonctionnelles qu'elles permettent de satisfaire et non seulement en termes de fonctionnalités qu'elles permettent de réaliser. Ceci permet de minimiser la discordance conceptuelle entre les services logiciels et l'énoncé des exigences des utilisateurs.

L'étude présentée dans ce document nous a permis d'identifier les problèmes liés d'une part, à la composition de services web, et d'autre part à l'association des propriétés extra fonctionnelles à cette composition. Nous nous sommes intéressés à ces deux problématiques qui nous ont conduits à la conception du framework $Ws-BeC$ (*Web service-Behavioral composition*) pour la composition de services web en tenant compte des propriétés comportementales des services composites. Le modèle permet aux concepteurs de composer des services et de prendre en compte des contraintes liées à la portée du comportement, par le biais de restrictions et de préférences, ainsi que des QoS des services composants. Au moment de l'exécution − par le biais de sélection, d'intégration voire de chevauchement de processus de services − le framework choisit d'une manière très approprié et beaucoup plus efficace, parmi les services accessibles, ceux qui répondent mieux aux besoins de la composition tout en respectant les caractéristiques comportementales.

Mots-clés: service web, composition dynamique de services, QoS, comportement de service, représentation et réécriture de processus de service, matching de processus, sélection, intégration, chevalement de processus de service.

Abstract

The SOA (service-oriented architecture) paradigm has become the standard for the design and development of distributed Web service-based-applications. Services composition involves the development of customized services often by discovering, integrating, and executing existing services. This can be done in such a way that already existing services are orchestrated into one or more new services that fit better to the composite application. Despite all the advantages they offer in terms of interoperability and reuse, the development solutions associated with the SOA paradigm is intended for programmers and are difficult to understand by the business world. To be in tune with the business world, service-based applications should be described in terms of requirements they can meet and not only in terms of features can they achieve. This minimizes the discrepancy between the conceptual software services and the statement of user requirements.

The work presented in this document led us to identify two main issues : (i) web service composition and (ii) behavioral properties associated with this composition. As we are interested in both issues we have designed a framework called $Ws-BeC$ (*Web service-Behavioral composition*) which covers the design of compositions of Web services as well as their executions. The framework relies on a model for the composition of Web services associated with behavioral properties that takes into account the expression of the behavioral properties that composite services are required to fulfil. In addition, compositions can be parameterized so as to allow the end user to impose that certain quality of services constraints and preferences are satisfied by the executions of the composite service. At execution time − based on service selection, integration and even interleaving − the framework selects, in a very appropriate way and much more efficient, among accessible services, those which best meet the composition's needs. The framework automatically ensures that beahvioral properties associated with the composition are fulfilled by exploiting the behavioral properties of the underlying selected services.

Keywords: web service, dynamic composition of services, QoS, service behavior, representation and rewriting of service processes, processes matching, selection, integration, interleaving of service processes.