Université
Dr Tahar Moulay
Saïda - ALGERIE

Faculté de Technologie **Département d'Informatique**

MÈMOIRE

présentée et soutenue le : **2 Juin 2016**

pour l'obtention du

Licence de l'université Dr. Tahar Moulay – Saida

(spécialité Informatique)

par

Mahsar Fadila & **Mebarki Zohra**

Encadré par

Mekour Mansour, Maître Assistant à l'Université de Saida

Résumé

Les dernières décennies ont été marquées par le développement rapide des systèmes d'information distribués, et tout particulièrement par la vulgarisation de l'accès à Internet. Cette évolution a entraîné le développement de nouveaux paradigmes d'interaction entre applications tel que les « services web ». Un service web est un programme modulaire, indépendant et auto-descriptif, qui peut être publié, découvert et invoqué via Internet ou intranet. La composition des services web permet de créer des nouveaux services web (dits services web composés) par regroupement des services web existants. Ainsi, un service web composé peut être vu comme une application distribuée qui possède des caractéristiques spécifiques. Ces caractéristiques influencent les aspects transactionnels dans ce domaine et engendrent le besoin de solutions flexibles et adaptables pour la composition des services web avec propriétés transactionnelles. 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 transactionnelles à cette composition. Notre travail permet de composer des services et de prendre en compte les propriétés transactionnelles.

Mots-clés:

Service web, Composition des service web, transaction ,Propriétés transactionnelles de services web .

Abstract

The last decades have been marked by the rapid development of distributed information systems, especially through extension of the Internet. This has led to the development of new paradigms for interaction between applications such as "web services". A web service is a modular program, independent and self-descriptive, which can be published, discovered and invoked over the Internet or intranet. The composition of web services can create new web services (compounds called web services) by consolidating existing web services. Thus, a compound web service can be seen as a distributed application that has specific characteristics. These characteristics influence the transactional aspects in this area and create the need for flexible and adaptable solutions for the composition of web services with transactional properties.

The study presented in this document allowed us to identify the one hand problems, the composition of web services, and on the other hand the association of the transactional properties of this composition. Our work for dialing services and take into account transactional properties.

Keywords:

web service, Composition of web service, transaction,transactional properties of web services.

Remerciements

Nous tenons à remercier :

Allah le tous puissant.

Mr MEKOUR MANSOUR , notre encadreur, pour ses conseils,

sa disponibilité et son encouragement qui nous ont permis de

réaliser ce travail dans les meilleures conditions.

Je tiens à remercier également les membres du jury pour l'honneur qu'ils m'ont attribué en acceptant d'examiner et d'évaluer mon travail. J'espère que ce travail sera à la hauteur de leurs exigences scientifiques.

Aux enseignants de notre université et du département d'informatique .

Je dédie ce travail à :

Mes parents

Vous vous êtes dépensés pour moi sans compter.

En reconnaissance de tous les sacrifices consentis par tous et chacun pour me permettre d'atteindre cette étape de ma vie.

Avec toute ma tendresse.

A mes frère et mes soeurs.

A tous les membres de ma promotion.

A mes amis.

A tous mes ensiegnant.

DADAH SIDI MAHFUD

A mes parents ;

A mes soeurs et mes frères ;

A tous ceux qui me sont chers.

OULD BAKAY MOHAMED

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

Depuis l'invention du World Wide Web par Tim Berners-Lee et Robert Cailliau 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 composer 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 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, ou seulement les interfaces qu'ils fournissent sont connues.

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, surtout lorsque celle-ci est munie de propriétés transactionnelles.

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 de propriétés transactionnelles 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 de réussir, tout en satisfaisant au mieux les besoins des clients de la composition.

Organisation du mémoire :

Ce document est composé d'une introduction générale, de trois chapitres et d'une conclusion générale.

Chapitre 1 :

Le chapitre 1 présente un rapide historique de l'architecture SOA, reprenant et complétant les différents points abordés dans cette introduction, pour permettre d'aborder la présentation et les questions amenant aux services web. Dans le chapitre 1 nous introduisons des définitions générales du :« L'architecture SOA » et « Service web » ; mais également en présentant la définition d'un service web par le W3C. Une rapide présentation des différents langages et technologies de services web est réalisée.

Chapitre 2 :

Ce chapitre est le coeur de notre travail. Il se divise en deux parties : la premiere partie nous abordons le sujet de la composition des services web.ainsi que les langages et protocoles utilisés pour leur déploiement ;et la deuxième partie nous introduisons la notion de transaction et l'aspect transactionnele dans le domaine de services web.

Chapitre 3 :

Le chapitre 3 présente la plateforme de test que nous avons développé pour le test de la composition de services. Il expose aussi une étude de cas que nous avons réalisé avec cette plateforme.

Glossaire

AOS : Architecture Orientée Service
API : Application Programming Interface
B2B : Business to Business
B2C : Business to Consumer
BEEP : Blocks Extensible Exchange Protocol
BPEL4WS :Business Process Execution Language for Web Services
BPEL :Business Process Execution Language
CORBA : Common Object Request Broker Architecture
FTP : File Transfer Protocol
HTTP : HyperText Transfer Protocol
IBM : International Business Machines
IDL : Interactive Data Language
IP : Internet Protocol
MIME : Multipurpose Internet Mail Extersin
POP : Post Office Protocol
RPC : Remote Procedure Call
SI : Système Information
SMTP :Simple Mail Transfer Protocol
SOAP : Simple Object Access Protocol
TCP : Transmission Control Protocol
UDDI : Universal Description, Discovery and Integration
URL :Uniform Resource Locator
W3C : World Wide Web Consortium
WSDL :Web Service Description Language
WSCL :Web Service Conversation Language
WSFL :Web Service Flow Language
XML : eXtensible Markup Language

Chapitre 1

L'architecture SOA et les services web

Sommaire

1.1 Introduction

L'orientation service permet d'organiser des ressources informatiques distribuées dans une solution intégrée, en éliminant les informations isolées. Et conférant davantage de souplesse à l'entreprise. Elle organise les ressources informatiques en modules, créant des processus métier faiblement couplés qui intègrent des informations provenant de différents systèmes. L'un des facteurs de réussite d'une architecture orientée services est la création de processus métier un peu soumis aux contraints de l'infrastructure informatique sous-jacente, afin de fournir à l'entreprise toute la latitude dont elle a besoin. L'architecture orientée services permet de créer toute une nouvelle génération d'applications dynamiques (parfois nommées applications composites). Ces applications offerent aux utilisateurs des informations plus précises et plus complètes ainsi qu'une meilleur connaissance des processus, mais aussi la possibilité d'accéder à ces informations en utilisant la forme et la présentation les mieux adaptées, que ce soit par le biais du web, d'un client riche ou d'un dispositif mobile.

1.2 Architecture orientée services

L'architecture orientée service s'imposée aujourd'hui comme un thème majeur pour les systèmes d'information d'entreprise. Plus qu'une nouvelle technologie ou méthode, c'est la convergence de plusieurs approches existantes, et l'émergence d'un style d'architecture et de gouvernance de SI.

1.2.1 Historique

Au niveau industriel, l'évolution de la complexité des systèmes informatiques, de même que leurs coûts et leurs fonctionnalités se traduit par une évolution des architectures logiciels (et matériels), avec un impératif qui revient régulièrement : la réutilisation de l'existant. Ainsi, les industriels sont confrontés au problème de l'interopérabilité des données et des architectures à travers le temps. Il est effectivement impossible (impensable) de redévelopper une application complète, dont le premier développement a pris plusieurs années, à chaque changement de machine (bien souvent accompagné par un changement de logiciels).
L'évolution de l'architecture est la conséquence de principaux facteurs tels que :
- la prise de conscience que la méthode précédente était imparfaite (souvent dûe à

une évolution des besoins de l'utilisateur également) ;
- l'évolution de la capacité des machines (tant en mémoire qu'en puissance de calcul) se traduisant par l'évolution de leurs fonctionnalités ;
- mais également la complexité des systèmes informatiques et leurs coûts.
Nous allons étudier les trois évolutions majeures de l'architecture de programmation : la programmation modulaire, la programmation orientée objet et enfin la programmation orientée composant.

Architecture basée sur la programmation modulaire

Fin des années 70, début des années 80, la programmation structurée ou modulaire était très à la mode, en même temps que l'arrivée des langages C (langage utilisé pour l'écriture des systèmes Unix), Pascal (langage développé dans un but principal d'enseignement) ou encore Ada (langage utilisé dans le domaine militaire). Le programmeur utilise des structures de données permettant de regrouper les informations ayant une relation entre elles, de plus il regroupe également le code traitant ces informations ou données dans un même module.

Architecture basée sur la programmation orientée objet

Parallèlement à l'arrivée de la programmation structurée, la programmation objet est arrivée début des années 80. La programmation objet reprenait le concept du regroupement des données et des fonctions (appelées méthodes en terminologie objet), pour obtenir un seul module : une classe, qui une fois instanciée, devient un objet.

Architecture orientée composant

L'architecture orientée objet et ses objectifs initiaux ne permettent pas d'utiliser la puissance et les nouvelles méthodes issues de l'évolution des architectures (grilles de calcul, l'utilisation du pair à pair, architecture 3-tiers ou même n-tiers) tendant toujours vers plus d'uniformisation tout en augmentant la complexité des modèles, et ce dans le but de séparer la logique métier et la logique système.
Une nouvelle architecture est apparue progressivement répondant à ces besoins : l'architecture orientée composant.

Cette nouvelle architecture orientée composant prend en compte les nouvelles architectures système comme le P2P (peer to peer) permettant de considérer chaque " acteur " du système sur un pied d'égalité, contrairement au modèle client-serveur ou n-tiers ; ou encore l'architecture basée sur les grilles de calcul, permettant de répartir les calculs et donc les composants sur un grand nombre de noeuds ou processeurs.
De l'architecture orientée objet à l'architecture orientée service :
Les différentes architectures vues précédemment ont évolué, et ce, particulièrement

car les besoins ont, encore une fois, changé et la puissance des machines également. De même, l'arrivée du réseau Internet sur toutes les machines favorise l'utilisation de protocoles de communication solutionnant les problèmes d'interopérabilités entres les différentes applications. Ainsi, avoir une base de données locale au niveau d'un poste et consultable uniquement sur ce poste n'est plus envisageable :
Les informations doivent être consultables depuis n'importe quel ordinateur à l'intérieur, voir à l'extérieur de la société. Toutes ces raisons ont favorisé l'évolution de l'architecture orientée objet et/ou composant en architecture orientée service. Le concept d'architecture orientée service, décrit la première fois par le groupe américain de recherche en technologie Gartner Group en 1996, se présente comme étant la solution adéquate à la demande actuelle des entreprises. Pour la première fois, l'architecture orientée service fut définie comme " client-server software design approach in which an application consists of software services and software service consumers (also know as clients or service requesters). SOA differs from the more general client/server model in its definitive emphasis on loose coupling between software components, and in its use of separately standing interfaces ".
Le passage à l'architecture orientée services (SOA - Service Oriented Architecture) se fait sur le long terme. La conservation de l'existant et la migration petit à petit est donc indispensable.
La méthode la plus adaptée est l'encapsulation de l'existant (existant reposant généralement sur CORBA, J2EE, DCOM, etc.) en respectant les nouveaux standards des publications. Ainsi, les applications métiers développées jusque là continuent de fonctionner, et les données et leurs traitements sont disponibles sous une forme publiée à travers un réseau et un protocole de communication standardisé. [10]

Cette architecture est construite autour de la notion de service, qui est matérialisée par un composant logiciel assurant une fonctionnalité particulière et accessible via son interface. Un service est toujours accompagne d'une description fournissant aux applications les informations nécessaires à son utilisation. L'objectif de la notion de service est de promouvoir un accès simple et rapide aux fonctionnalités mises à disposition par les organisations. [4]

1.2.2 Définition d'architecture de service orientée (SOA)

On trouve plusieurs définitions de l'architecture SOA dans la littérature, le concept de SOA est le sujet de définitions très variées propose la définition suivante :

L'architecture SOA permet l'intégration d'applications et de ressources de manière flexible, en représentant chaque application ou ressource sous la forme d'un service exposant une interface standardisée, permettant un service d'échanger des informations structurées (messages, documents, objets métier), coordonnant et en organisant les services, afin d'assurer qu'ils puissent être invoqués, utilisés et changés efficacement.
D'autre définition :
" Une architecture orientée service est un style d'architecture logiciels multi-tiers qui aide les organisations à partager leurs logiques métier et leurs données entre plusieurs applications et plusieurs modèles d'usage." (donnée en 1996 par le groupe Gartner) [31], et d'autres proposées par différents experts du domaine peuvent être trouvées dans la littérature [6, 9, 11].
Ces définitions se placent selon différents points de vue qui vont plus ou moins dans le même sens . Nous pouvons affirmer d'une manière générale que l'architecture orientée services est un style architectural pour la conception, le développement, le déploiement et la gestion de systèmes logiciels distribués qui délivre des fonctionnalités d'application sous forme de services interopérables, soit à l'utilisateur final ou à d'autres services. Les systèmes qui sont construits en se basant sur les caractéristiques SOA sont appelés système orientés services.

1.2.3 Les caractéristiques d'architecture orientée services

Les caractéristiques principales d'une architecture orientée services sont :
Le couplage faible, l'indépendance par rapport aux aspects technologiques et l'extensibilité .[4, 29]
- La propriété de couplage faible implique qu'il est possible de fournir et d'utiliser des services tout en restant indépendants les uns des autres, et sans divulguer leur fonctionnement interne. Cette exigence est satisfaite par l'adoption de protocoles et langages standardises fournissant un accès uniforme aux services et à leurs descriptions.
- L'indépendance par rapport aux aspects technologiques : assurée par le fait que les contrats d'utilisation qu'exposent les services sont auto-descriptifs et indépendants de la plateforme technique utilisée par le fournisseur du service. Par exemple WSDL (Web Service Description Language) permet d'exposer les fonctionnalités des web services en assurant cette indépendance.
- Enfin, l'extensibilité est rendue possible par le fait que de nouveaux services peuvent

être découverts et invoques à l'exécution.
Les architectures orientées services se construisent autour de plusieurs protocoles et Langages, selon quatre couches de fonctionnalités, à savoir [4] :
- La couche publication, qui permet la centralisation, le stockage et la diffusion des descriptions de services.
- La couche description, qui regroupe les détails nécessaires à l'invocation des services dans un document.
- La couche message, qui assure la structuration et l'échange uniformes des messages.
- La couche transport, qui permet de véhiculer les messages à travers le réseau.

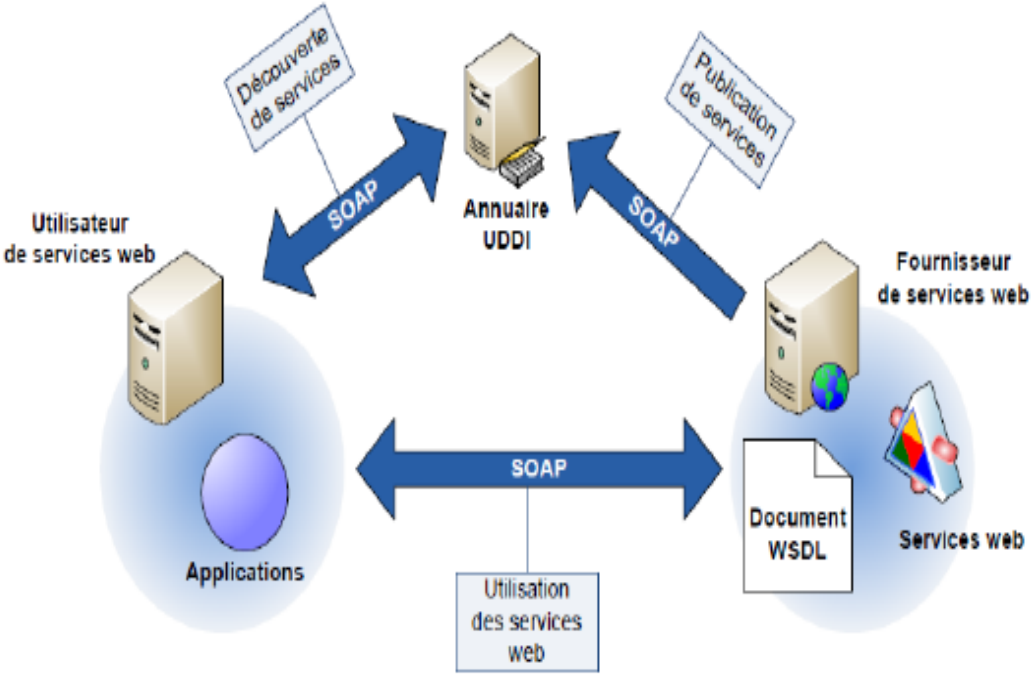

FIG. 1.1: Interactions dans une architecture orientée services.

1.2.4 Les rôles dans une architecture orientée services

Pour garantir une évolution et une intégration facile des applications hétérogènes au sien d'une architecture orientée services, l'approche à service prévoit un modèle d'interaction basé sur différents acteurs et des interactions entre eux. La figure1.1 montre que l'utilisation de ces services par ces acteurs suit un protocole spécifique qui est : la publication, la découvert, l'invocation. Les acteurs collaborant dans une SOA sont :

1. **Le fournisseur du service :** il désigne l'entité propriétaire du service. D'un point de vue technique, un fournisseur peut designer la plateforme d'accueil du service.
2. **Le client du service :** correspond au demandeur de service. D'un point de vue opérationnel, c'est le service client qui sollicite et invoque le service requis.
3. **L'annuaire des services :** est un registre qui donne la possibilité aux fournisseurs de service de publier des descriptions des services. Il permet aussi aux consommateurs de service de pouvoir consulter les différentes descriptions disponibles. En plus des spécifications de service, l'annuaire contient des références vers les fournisseurs de service. Ceci va permettre aux consommateurs de localiser les fournisseurs des spécifications de service qui correspondent à leurs besoins, l'annuaire sert donc d'acteur intermédiaire entre fournisseurs et consommateurs de service.

Les opérations possibles dans une architecture orientée services sont :

a) Publication : pour être accessible, une description de service doit être publiée de sorte qu'elle puisse être découverte et appelée par un consommateur de services.

b) Découverte : un demandeur de services localise un service via une requête, l'annuaire de services portant sur les critères du service recherché.

c) Invocation : après recherche de la description de service, le consommateur de services procède par appel du service selon les informations relatives à la description du service.

1.2.5 Concepts de l'architecture orientée services

L'architecture orientée services est axée autour de trois concepts fondamentaux qui sont :[25]

Service :

C'est-à-dire une fonction que l'on peut interroger à l'aide d'une requête et qui fournit une ou plusieurs réponses.

Interopérabilité :

Il permet la propagation des fonctionnalités des services web via des systèmes hétérogènes.

Couplage faible :

Couplage faible est le concept qui assure l'évolutive, la flexibilité et la tolérance. Son but est de minimiser les dépendances. Quand on minimise les dépendances on va aussi minimiser l'influence sur les autres systèmes.

1.2.6 Approches de SOA

Deux approches sont définies pour implémenter l'architecture SOA tel qu'il est présenté dans la figure1.2 :

Bottom-Up :

Cette approche est inventée par Krafzig and Slama [5]. Son principe est de commencer par un petit groupe et on jouter jusqu'à on obtient une grande entreprise.

Top-down :

Cette approche aussi est inventée par Krafzig and Slama. On peut dire que c'est l'adverse de Bottom-Up. Top-down est la décomposition d'un système ou problème jusqu'à l'obtient d'un petit service de base. [25]

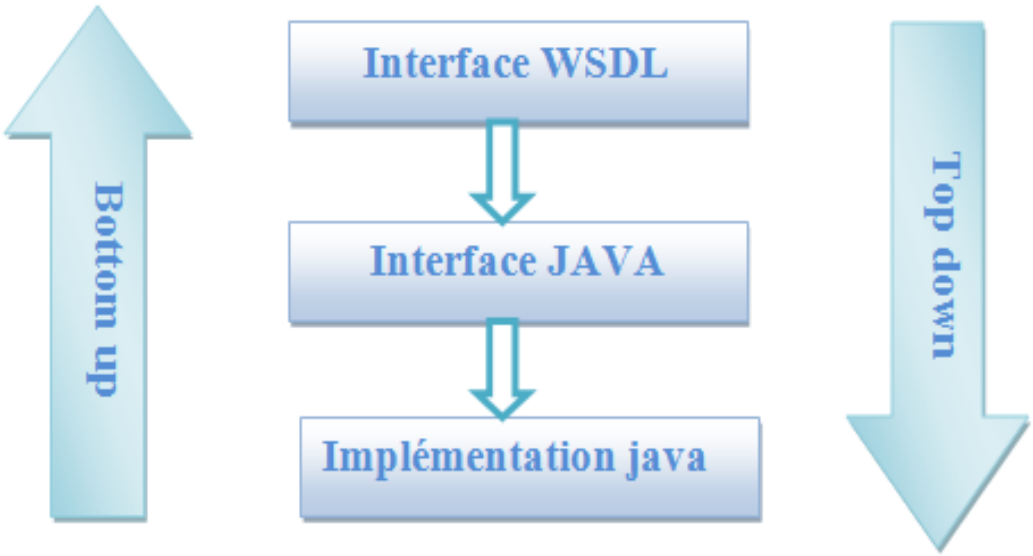

FIG. 1.2: Approches de développement SOA.

1.2.7 Les avantages et inconvénients d'architecture orientée services

1. **Les avantages :**
   - Obligation d'avoir une modélisation poussée.
   - Possibilité de découpler les accès aux traitements.
   - Localisation et interfaçage transparents (ouverture accrue).
   - Possibilité de mise en place facilitée à partir d'une application objet existante.
   - Réduction des coûts en phase de maintenance et d'évolution.
   - Facilité d'amélioration des performances pour des applications importantes (répartition des traitements facilitée).

2. **Les inconvénients :**
   - Coûts de conception et de développement initiaux plus conséquents.
   - Nécessité d'appréhender de nouvelles technologies.
   - Existant non SOA dans les entreprises.
   - Performances réduites pour des traitements simples (couche supplémentaire).

1.3 Services web

Afin de mettre en oeuvre une SOA, il est essentiel d'avoir une compréhension claire de ce qu'est un service web.

1.3.1 Origines

Les services web sont considères comme étant l'évolution naturelle du web. Ils s'inscrivent dans la continuité d'initiatives telle que CORBA (Common Object Request Architecture de l'OMG) en apportant toutefois une réponse plus simple, s'appuyant sur des technologies et standards reconnus et maintenant acceptés de tous.
On peut distingue trois phases de développement dans l'histoire du web :
Le web du document : le phénomène Internet original, utilisé principalement par des organisations et des particuliers pour publier des informations sur leur travail, leurs produits, etc..

Le web applicatif : le progrès grâce auquel les entreprises ont commencé à utiliser le web à des fins commerciales.
Le web des services : est la phase émergente, dans laquelle les serveurs d'application précédents communiquent désormais entre eux. Cette évolution a été poussée par le désir de pouvoir réaliser des échanges interentreprises dans un environnement automatisé et ouvert tel qu'Internet. L'échange de données informatisées entre deux applications nécessite une normalisation des messages échangés.
Une approche web service du système d'information vise à transformer chaque composant, base de données, applicatif métier, application de bureautique, en noeud s'exposant sur des standards de l'Internet, pour soit consommer des web services, soit pour en fournir.
Ainsi, on passe d'une interopérabilité applicative à une interopérabilité entre services.
Cette approche propose une API (Application Programming Interface) universelle qui ne requiert pas d'autres protocoles que ceux d'Internet : HTTP (HyperText Transfer Protocol) [48] sur TCP/IP (Transmission Control Protocol/ Internet Protocol) principalement. De plus, ils normalisent l'appel, l'échange et l'organisation de services applicatifs.

1.3.2 Définition des web services

Les services web représentent un domaine de recherche jeune, plusieurs définitions des services web ont été mises en avant par différents auteurs. Ci-dessous, nous citons quelques définitions généralement acceptées et fournies :
Selon IBM :
« web services are a new breed of web applications. They are self-contained, self- describing, modular applications that can be published, located, and invoked across the web. web services perform functions that can be anything from simple requests to complicated business processes ».
Cette définition affirme que les services web sont accessibles par d'autres à travers le web, en utilisant des protocoles et des formats standards.
Selon le W3C :
« A web service is a software system identified by a URI and designed to support interoperable machine-to-machine interaction over a network. It has an interface defined and described in a machine-processable format (wsdl) . Its

definition can be discovered by other software systems. Other systems may then interact with the web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other web-related standards».
Le consortium W3C (http ://www.w3.org/2002/ws/) définit un service web comme étant une application ou un composant logiciel qui vérifie les propriétés suivantes :
- Il est identifié par un URI.
- Ses interfaces et ses liens (binding) peuvent être décrits en XML.
- Sa définition peut être découverte par d'autres services web.
- Il peut interagir directement avec d'autres services web à travers le langage XML et en utilisant les protocoles d'Internet.

1.3.3 Les caractéristiques des services web

Un service web possède les caractéristiques suivantes :
- Il est accessible via le réseau ;
- Il dispose d'une interface publique (ensemble d'opérations) décrite en XML ;
- Ses descriptions (fonctionnalités, comment l'invoquer et où le trouver ?) sont stockées dans un annuaire ;
- Il communique en utilisant des messages XML, ces messages sont transportés par des protocoles Internet (généralement HTTP, mais rien n'empêche d'utiliser d'autres protocoles de transfert tels : SMTP, FTP, BEEP...) ;
- L'intégration d'application en implémentant des services web produit des systèmes faiblement couplés, le demandeur du service ne connaît pas forcément le fournisseur.
Ce dernier peut disparaître sans perturber l'application cliente qui trouvera un autre fournisseur en cherchant dans l'annuaire.

1.3.4 Les applications des services web :

L'application des services web est multiple, autant dans les domaines du B2C, B2B que pour des domaines de gestion, par exemple gestion de stock, gestion commerciale, etc..
B2C (Business to Consumer) : Qualifie une application, un site Internet destiné au grand public.
B2B (Business to Business) : Qualifie une application, un site Internet

destiné au commerce de professionnel à professionnel. [1]

1.3.5 L'intérêt des services web :

Les services web fournissent un lien entre applications. Ainsi, des applications utilisant des technologies différentes peuvent envoyer et recevoir des données au travers de protocoles compréhensibles par tout le monde. Les services web sont normalisés car ils utilisent les standards XML et HTTP pour transférer des données et ils sont compatibles avec de nombreux autres environnements de développement. Ils sont donc indépendants des plates-formes. C'est dans ce contexte qu'un intérêt très particulier a été attribué à la conception des services web puisqu'ils permettent aux entreprises d'offrir des applications accessibles à distance par d'autres entreprises. Cela s'explique par le fait que les services web n'imposent pas de modèles de programmation spécifiques. En d'autres termes, les services web ne sont pas concernés par la façon dont les messages sont produits ou consommés par des programmes. Cela permet aux vendeurs d'outils de développement d'offrir différentes méthodes et interfaces de programmation au-dessus de n'importe quel langage de programmation, sans être contraints par des standards comme c'est le cas de la plateforme CORBA qui définit des ponts spécifiques entre le langage de définition IDL et différents langages de programmation. Ainsi, les fournisseurs d'outils de développement peuvent facilement différencier leurs produits avec ceux de leurs concurrents en offrant différents niveaux de sophistication.
Les services web représentent donc la façon la plus efficace de partager des méthodes et des fonctionnalités. De plus, ils réduisent le temps de réalisation en permettant de tirer directement parti de services existants.

1.3.6 Langages et protocoles associés aux services web :

L'originalité de l'infrastructure des services web consiste à mettre en place ces services en se basant exclusivement sur les protocoles les plus répandus d'Internet. Pour garantir l'interopérabilité des trois opérations précédentes (publication,recherche et lien), l'infrastructure services web s'est concrétisé autour d'un ensemble de spécifications considérées comme des standards pour chaque type d'interactions :
Un protocole abstrait de description et de structuration des messages, SOAP1[35],

une spécification XML qui permet la publication et la localisation des services dans les annuaires, UDDI2 [39] et un format de description des services web publiées dans les annuaires, WSDL3.[40]

Le langage XML (eXtensible Markup Language)

Le langage XML standardisé par le W3C en 1998 est aujourd'hui largement reconnu et utilisé par de nombreuses entreprises comme format universel d'échange de données. XML est un métalangage de représentation de données. Il définit un ensemble de règles de formatage pour composer des données valides.
XML constitue la technologie de base des architectures web services ; c'est un facteur important pour contourner les barrières techniques. XML est un standard qui permet de décrire des documents structurés transportables sur les protocoles d'Internet. En effet, il apporte à l'architecture des services web l'extensibilité et la neutralité vis à vis des plateformes et des langages de développement [23]. De plus, grâce à la structuration, XML permet la distinction entre les données des applications et les données des protocoles.
La technologie des services web a été conçue pour fonctionner dans des environnements totalement hétérogènes. Cependant, l'interopérabilité entre les systèmes hétérogènes demande des mécanismes puissants de correspondance et de gestion des types de données des messages entre les différents participants (clients et fournisseurs). C'est une tâche où les schémas de type de données XML s'avèrent bien adaptés. C'est pour cette raison que la technologie des services web est essentiellement basée sur XML ainsi que les différentes spécifications qui tournent autour (les espaces de nom, les schémas XML, et les schémas de Type). [38]

La communication : SOAP (Simple Object Acess Protocol)

Les communications entre les différentes entités impliquées dans le dialogue avec le service web se font par l'intermédiaire du protocole SOAP (Simple Object Access Protocol). Ce protocole est normalisé par le W3C.

Principe :

SOAP, est un standard du Consortium W3C définissant un protocole de transmission de messages permettant la normalisation des échanges de données. Il

présente un ensemble de règles pour structurer des messages, qui peuvent être utilisées dans de simples transmissions unidirectionnelles, mais il est particulièrement utile pour exécuter des dialogues requête-réponse RPC (Remote Procedure Call) en utilisant HTTP comme protocole de communication, mais aussi les protocoles SMTP (Simple Mail Transport Protocol) et POP (Post Office Protocol) [24]. SOAP assure l'interopérabilité entre composants tout en restant indépendant des systèmes d'exploitation et des langages de programmation, donc, théoriquement, les clients et serveurs de ces dialogues peuvent fonctionner sur n'importe quelle plateforme et être écrits dans n'importe quel langage à partir du moment où ils peuvent formuler et comprendre des messages SOAP. Il représente donc un composant de base pour développer des applications distribuées, qui exploitent des fonctionnalités publiées comme services par des intranets ou Internet. SOAP utilise principalement les deux standards HTTP et XML :
HTTP : comme protocole de transport des messages SOAP. Il constitue un bon moyen de transport en raison de sa popularité sur le web.
XML : pour structurer les requêtes et les réponses, indiquer les paramètres des méthodes, les valeurs de retours, et les éventuelles erreurs de traitements.

Structure du message SOAP :

Les messages échangés lors de l'utilisation du protocole SOAP sont basés sur le langage XML. Ils sont composés de deux parties, l'en-tête de protocole de transport et l'enveloppe SOAP.(la Figure1.3).

(a) **L'en-tête du protocole de transport :** qui dépend de protocole de transport utilisé, par exemple si le protocole HTTP est utilisé, l'en-tête contient :
- La version de protocole HTTP utilisée.
- La date de génération de message SOAP.
- Le type d'encodage du contenu (généralement de type XML).

(b) **L'enveloppe SOAP :** la partie principale d'un message SOAP est l'enveloppe (symbolisée par la balise enveloppe), cette dernière est subdivisée en deux sous-parties : la partie en-tête (Header) et la partie corps du message (Body).

SOAP Header : l'en-tête de message SOAP, est le premier fils de l'élément enveloppe. Même s'il peut être vide, il doit impérativement être écrit. Cet élément apporte des données supplémentaires à SOAP. Ces données peuvent être des informations d'authentification, de gestion de transactions, de paiement, etc..

SOAP Body : l'élément Body contient l'information destinée au receveur. Il faut que cet élément encadre une balise contenant le nom de la méthode invoquée (pour une requête), ou le nom de la méthode suivie de Response (pour la réponse). Cette balise doit aussi contenir l'espace de noms correspondant au nom de service.

Un message SOAP peut aussi contenir un ou plusieurs attachements (document, images, etc.) à transmettre avec le message. Ces données ne sont pas représentables en XML.

De ce fait, SOAP utilise un mécanisme d'inclusion appelé MIME (Multipurpose Internet Mail Extensions), cette méthode est répandue pour transmettre des documents autres que du texte dans des courriers électroniques.

FIG. 1.3: Structure du message SOAP.

Couche de Description de Service WSDL (Web Service Description Language) :

WSDL [48] a été crée dans le but de fournir une description unifiée des services web. Il se présente comme un standard actuel dans ce domaine, et il est de plus normalisé par le W3C. Son objectif principal est de séparer la description abstraite du service de son implémentation. Le langage de description WSDL se comporte donc comme un langage permettant de décrire l'interface visible (ou publiée) du service web.

Il décrit à l'aide du langage de balises XML les différents éléments de service. WSDL se compose en deux parties. Une premiere partie définit de manière abstraite les éléments, les opérations et les types de données, tandis qu'une seconde partie précise de manière concrète les adresses physiques de ces opérations ainsi que le mapping des données avec les protocoles de transport. Cette distinction est utile car elle permet de concevoir le service indépendamment de l'environnement de déploiement.

Plus précisement, le WSDL définit les services web à travers six éléments :

- Types : décrivent sous la forme d'une spécification XML Schéma les types des données échangées entre le client et le fournisseur de services.

- Message : définit les informations échangées lors d'une requête ou d'une réponse. Un message a un nom et potentiellement des parts qui font référence à des paramètres et des valeurs de retour.

- PortType : combine plusieurs messages pour former une opération. Il y a quatre types d'opérations : one-way, request-response, solicit-response et notification.

- Binding : spécifie le protocole de communication (le plus souvent HTTP, mais aussi SMTP, FTP, etc..) et le format d'encodage des données (encodage RPC, Literal Document, etc..) pour les opérations et messages définis par un type de port donné. Il est possible grâce à des extensions internes de WSDL de définir des binding SOAP.

- Port : est un point d'accés au service identifié de manière unique par la combinaison d'un binding et d'une adresse Internet.

- Service : regroupe un ensemble de ports, chaque port offre une alternative (différents protocoles, etc..) pour accéder au service.[50]

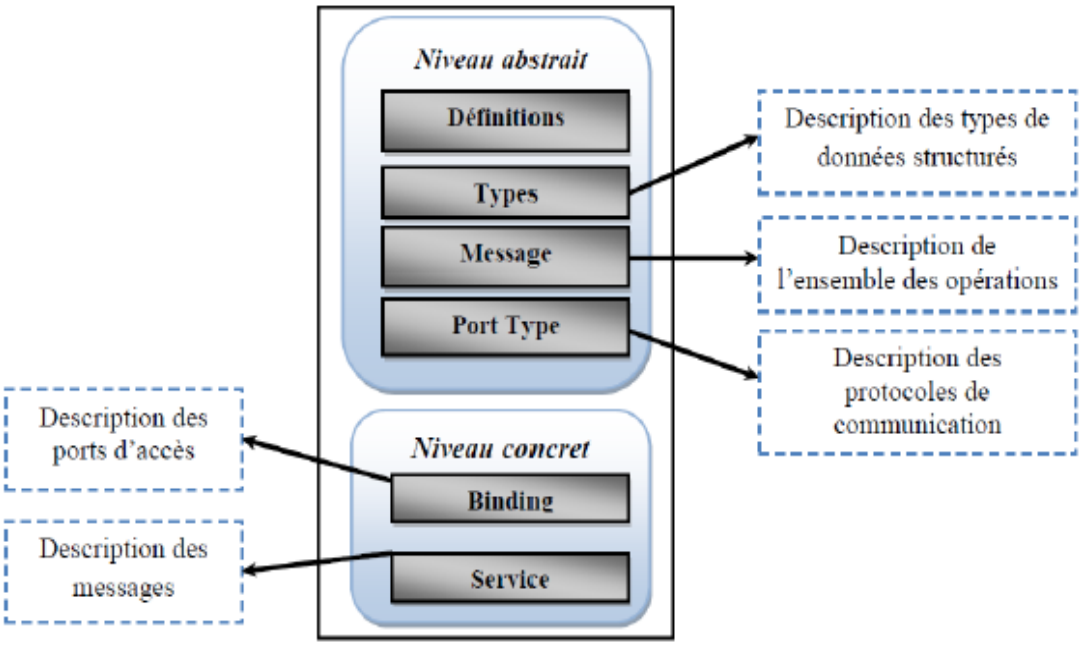

FIG. 1.4: Structure de document WSDL.

Comme nous remarquons, la portée de WSDL est limitée aux descriptions structurelles, WSDL ne renseigne donc pas sur le comportement du service web décrit. Plusieurs initiatives de recherche, de développement et de standardisation sont en cours afin de définir des langages de description de services prenant en compte les aspects comportementaux, et d'utiliser ces descriptions comportementales lors du développement et de l'exécution des services. Le langage de description de processus métiers exécutables pour services web (Web Services Business Process Execution Language, BPEL) est utilisé dans ce contexte.

Aussi, WSDL ne permet pas de décrire des aspects non-fonctionnels des services tels que leur capacité à garantir une certaine qualité de service par rapport à des préoccupations telles que la sécurité, la fiabilité, la journalisation des accès ou la gestion de transactions. Le langage «WS-Policy » est présenté comme solution à ce problème, il est basé sur le concept de politiques d'usage. Une politique d'usage est une énonciation explicite des possibilités et des restrictions d'usage d'un service web.

UDDI (Universel Description, Discovery and Integration) :

UDDI est un standard défini par OASIS. Il définit la structure d'un annuaire de services web, et la structure de gestion de services (publication, localisation, découverte) sous forme de répertoire, il permet de stocker les informations nécessaires pour retrouver et accéder à un service, telles que les informations techniques et l'adresse des services web, le nom de la personne/société qui gère un service donné, la description des fonctionnalités [8].

Un service d'annuaire UDDI est un service web qui gère les méta-données des services, l'information sur les fournisseurs de services et les implémentations des services. Afin de trouver un service web, il est possible d'utiliser un annuaire UDDI en précisant des exigences concernant service requis. On cherche le service par son nom et/ou par des mots clés.

Consultation de l'annuaire :

L'annuaire UDDI se concentre sur le processus de découverte de l'architecture orientée services (SOA), et utilise des technologies standards telles que XML, SOAP et WSDL qui permettent de simplifier la collaboration entre partenaires dans le cadre des échanges commerciaux. L'accès au référentiel s'effectue de différentes manières. Les informations sur un service publié dans un annuaire UDDI se présentent sous trois facettes comme illustre dans la figure1.5 [19] :

Les pages blanches comprennent la liste des entreprises ainsi que des informations associées à ces dernières (coordonnées, description de l'entreprise, identifiants...).

Les pages jaunes recensent les services web de chacune des entreprises sous le standard WSDL.

Les pages vertes fournissent des informations techniques précises sur les services fournis.

Le modèle d'information UDDI est composé de cinq types d'entités :

BusinessEntity : cette entité décrit l'entreprise (également appelée fournisseur) qui permet d'accéder aux services web qu'elle publie : cette entité permet de regrouper toutes les informations concernant le nom, les contacts de l'entreprise. Chaque enregistrement est associé à une clé unique appelée UID (Universal Unique Identifier). Ce sont les éléments accessibles par l'annuaire pages blanches.

BusinessService : cette entité représente une classification des services. Elle est

contenue dans l'entité BusinessEntity décrite précédemment, et elle contient une clé unique identifiant un service particulier. Ce sont les éléments accessibles par l'annuaire pages jaunes.

BindingTemplate : cette entité est utilisée pour les détails techniques des services web. Elle contient des informations sur le point d'accès du service (l'adresse du service). Ce sont les éléments accessibles par l'annuaire pages vertes.

TModel : cette entité permet de stocker les informations spécifiques aux services, comme le comportement, les conventions de typages et les types eux-mêmes utilisés par les services. Elle regroupe donc les informations contenues dans les fichiers WSDL.

Publisher Assertion : ensemble d'informations contractuelles entre partenaires dans le cadre d'échanges commerciaux.

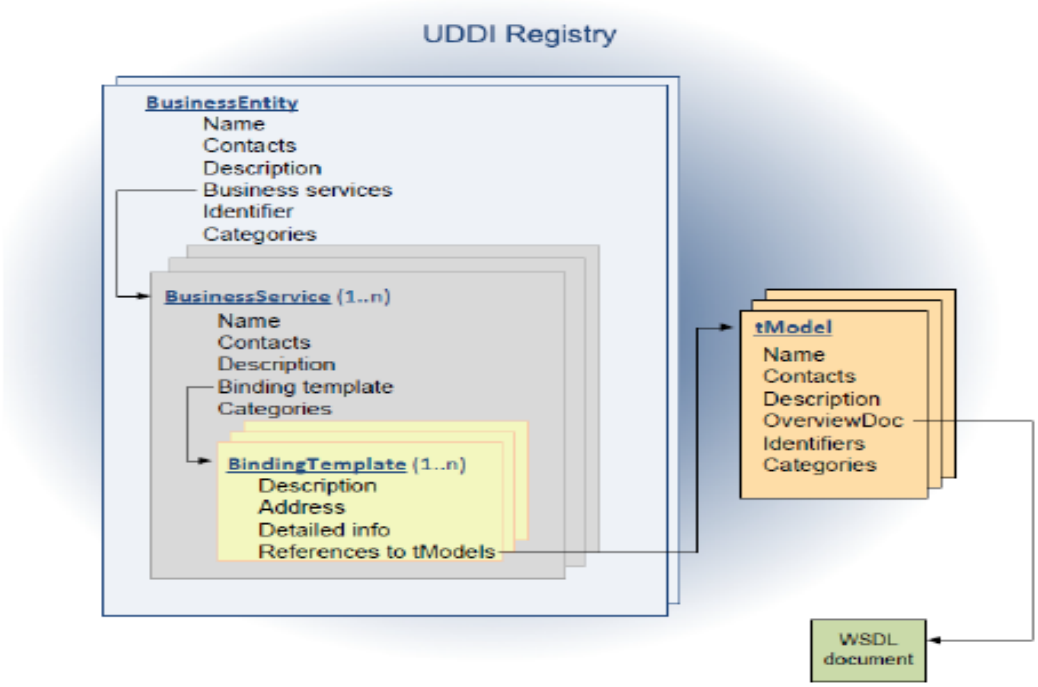

FIG. 1.5: Entités composant un annuaire UDDI.

1.3.7 Les avantages et les inconvénients des services web

1. **Les avantages :**
   Parmi les avantages des services web on trouve :
   - Les services web fournissent l'interopérabilité entre divers logiciels fonctionnant sur diverses plates-formes.
   - Les services web utilisent des standards et protocoles ouverts.
   - Les protocoles et les formats de données sont au format texte dans la mesure du possible, facilitant ainsi la compréhension du fonctionnement global des échanges.
   - Basé sur le protocole HTTP, les services web peuvent fonctionner au travers de nombreux pare-feux sans nécessiter des changements sur les règles de filtrage.
   - Les outils de développement, s'appuyant sur ces standards, permettent la création de nouveaux programmes utilisant les services web existants.

2. **Les inconvénients :**
   Parmi les inconvénients des services web on trouve :
   - La multiplication de la masse d'information véhiculée.
   - Le surcharge de traitement.
   - La complexité d'implémentation.

1.4 Conclusion :

Les services web assurent, à travers l'Internet, l'interaction entre les applications, les ordinateurs et les processus métier en permettant d'accéder à partir d'un seul site web à plusieurs services distants. Ce nouveau modèle de programmation et de déploiement d'application assure l'interaction de services web basant sur les standards suivants : HTTP, SOAP, WSDL, et UDDI.

Dans ce chapitre, nous avons détaillé comment les services web permettent à des applications de dialoguer via Internet par échange de messages, et ceci indépendamment des plates formes et les langages sur les quelles sont implémentées.

Nous avons également exploré les différentes technologies, qui permettent la mise en oeuvre des services web.

Nous continuerons avec les services web dans le chapitre suivant, et nous allons présenter en détail un concept important qui est : la composition des services web, en fournissant la description, le fonctionnement et les différents langages(spécifications et standards)de définition de ce concept et l'aspect transactionnel au sein de la composition.

Chapitre 2

La composition des services web avec propriétés transactionnelles

Sommaire

2.1 Introduction

Les services web offrent une architecture par composants permettant à des applications d'offrir leurs fonctionnalités sous formes de services à travers des protocoles Internet universels. Ceci marque une évolution significative dans l'histoire d'Internet qui jusque là a été destiné à jouer le rôle d'un vecteur d'échange de données. Avec les services web, Internet se transforme en une plate-forme de composants auto descriptifs, facilement intégrables et faiblement couplés [13]. Comme toute innovation, l'apparition des services web donne lieu à un ensemble d'opportunités et de nouvelles applications. Dans ce travail une motivation anime notre intérêt pour les services web à savoir la composition des services web et l'aspect transactionnel. Le problème de la composition de services web se réfère à la construction des nouveaux services web «services web composites» à partir des services existants. Dans cette partie nous présentons d'abord le besoin de composition des services web, puis nous détaillons le concept de transaction, et l'aspect transactionnel dans la composition mais avant ça nous introduisons l'archietcture étendu de service web.

2.2 Architecture étendue

Actuellement ; SOAP, WSDL et UDDI sont les trois standards qui constituent l'architecture des services web. Ensemble ; ils résolvent les problèmes de l'hétérogénéité des systèmes pour l'intégration d'applications en ligne [41] cependant, une application B2B nécessite d'invoquer un ensemble de services dans un ordre précis et selon une logique bien définie. Ou SOAP, WSDL et UDDI ne s'intéressent pas à ce problème et se situe plutôt au niveau transport et données. Toutefois, il existe d'autres aspects essentiels à mettre en oeuvre avant de parler d'automatisation de processus de découverte et d'intégration des services web. A ce propos, plusieurs technologies ont été proposées. Ces technologies s'intéressent à différents problèmes et à différents niveaux, des propriétés non fonctionnelles (comme la sécurité et la fiabilité) au niveau transport à la qualité de services au niveau procédé.
L'architecture étendue (avancée) est constituée de plusieurs couches se superposant les unes sur les autres, d'où le nom de la pile des web services. La figure 2.1 décrit un exemple d'une telle pile.

La pile est constituée de plusieurs couches, chaque couche s'appuyant sur un standard particulier. On retrouve, au dessus de la couche transport, les trois couches formant l'infrastructure de base décrit précédemment. Ces couches s'appuient sur

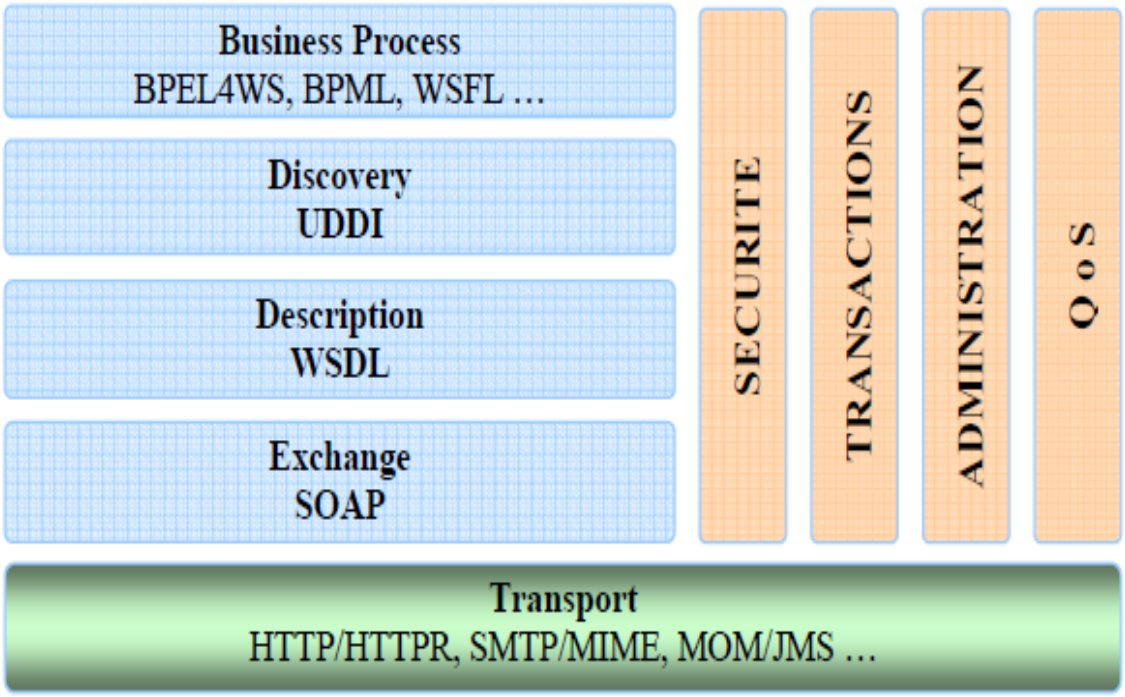

FIG. 2.1: La pile des services web.

les standards émergents SOAP, WSDL et UDDI. Comme mentionné précédemment, l'infrastructure de base définit les fondements techniques permettant de rendre les business processes accessibles à l'intérieur d'une entreprise et au-delà même de frontières d'une entreprise. Dans ce contexte deux types de couches permettant de la compléter :
(i) les couches dites transversales [20](eg : sécurité, administration, transaction et qualité des services (QoS)) rendent viable l'utilisation effective des services web dans le monde industriel.
(ii) une couche Business processus permet l'utilisation effective des services web dans le domaine e-bussiness.

2.3 Composition des services web

La composition de services web est la plus importante fonctionnalité assurée par une architecture SOA. Celle-ci offre un environnement homogène pour la composition dans la mesure où toutes les parties de la composition sont des services idéalement décrits de la même façon et communiquant par les mêmes standards

d'échange de messages. Dans cette section nous allons présenter le concept de composition de services web ainsi que les deux approches de composition l'orchestration et la chorégraphie.

2.3.1 Définition de composition de services web :

Une composition de services web est constituée de plusieurs services qui interagissent les uns avec les autres [2], afin d'offrir de nouvelles fonctionnalités qu'un seul service ne pourrait pas les offrir. La composition permet de combiner des services pour former un nouveau service dit composé ou composite. L'exécution d'un service composé implique des interactions avec des services partenaires en faisant appel à leurs fonctionnalités. Le but de la composition est avant tout la réutilisation de services(simples ou composés) et de préférence sans aucune modification de ces derniers.

Orchestration :

L'orchestration décrit l'interaction des services au niveau de messages, incluant le logique métier et l'ordre d'exécution des interactions. Les services web n'ont pas de connaissance (et n'ont pas besoin de l'avoir) d'être mêlées dans une composition et d'être partie d'un processus métier. Seulement le coordinateur de l'orchestration a besoin de cette connaissance.

La figure suivante montre le workflow dans l'orchestration des services web. Un coordinateur prend le control de tous les services web impliqués et coordonne l'exécution des différentes opérations des services web qui participent dans le processus.

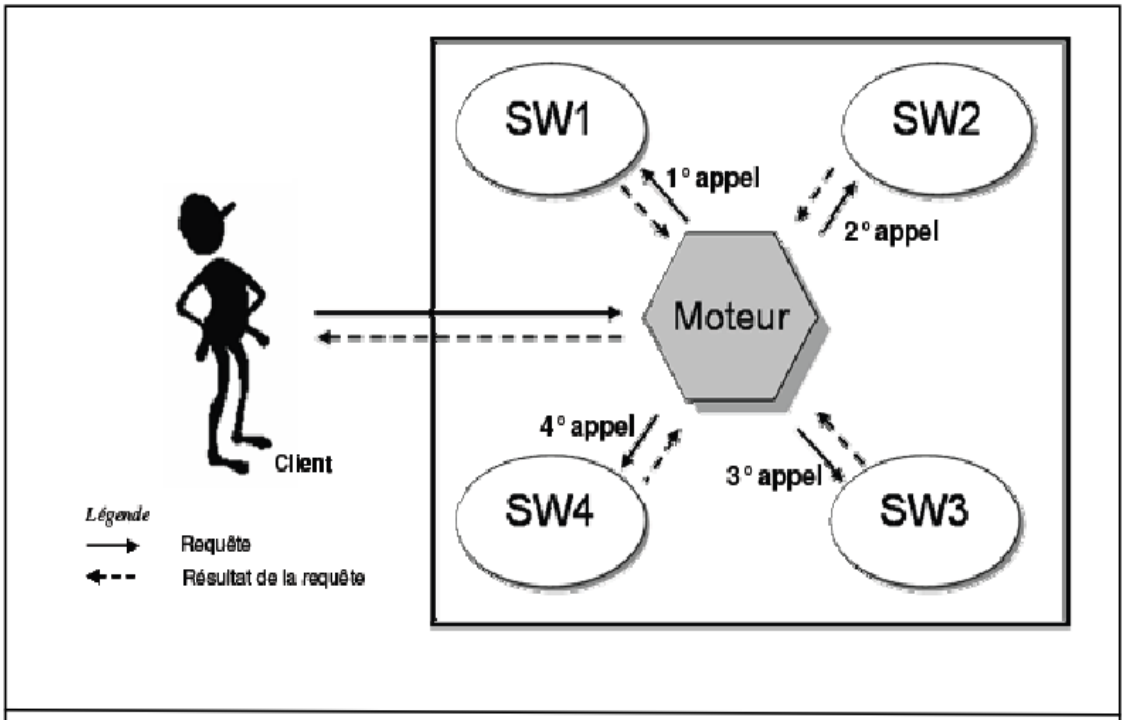

FIG. 2.2: Vue générale de l'orchestration [7].

La chorégraphie :

Dans [32], l'auteur a mentionné : « La chorégraphie permet de tracer la séquence de messages échangés dans un contexte de composition de services web. Elle est typiquement liée à la description de conversations existantes entre les services tout en impliquant plusieurs parties, incluant les clients, les fournisseurs et les partenaires».

Contrairement à l'orchestration, la chorégraphie offre une vision décentralisée de la composition (elle ne repose pas sur un procédé central pour gérer la composition). Chaque service web mêlé dans la chorégraphie connaît exactement quand ses opérations doivent être exécutées et avec qui l'interaction doit avoir lieu [26]. Le principe de la chorégraphie est illustré par la figure suivante :

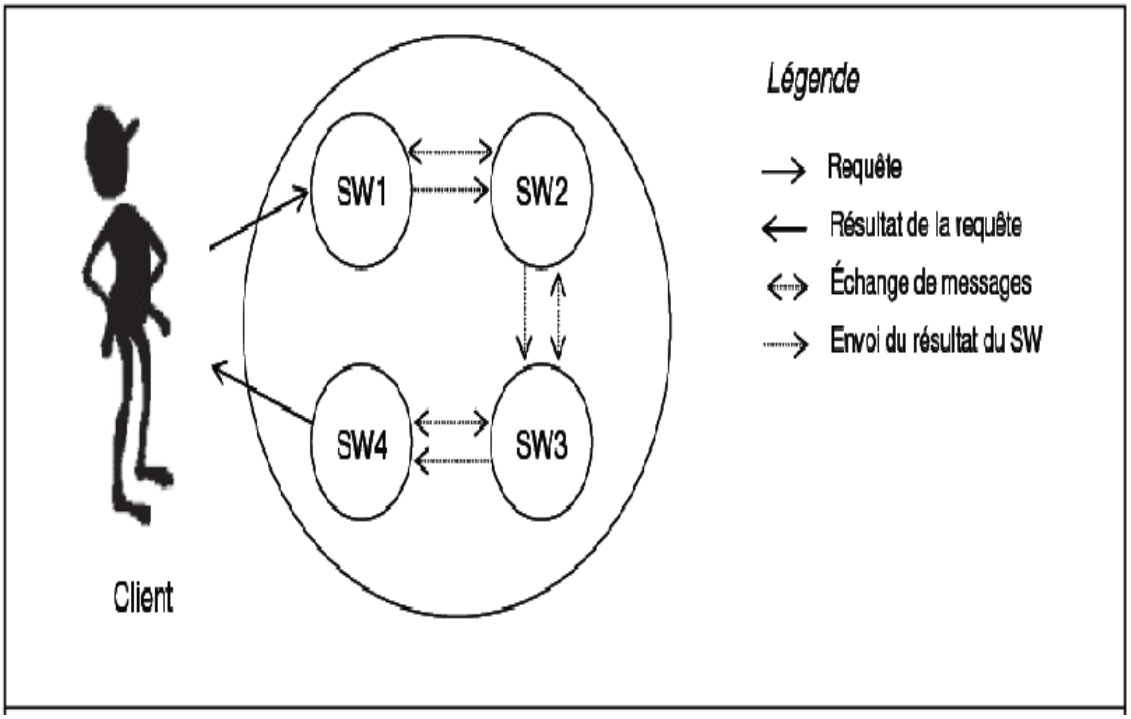

FIG. 2.3: Vue générale de la chorégraphie [7].

2.3.2 Comparaison :

Quoi qu'il en soit, il y a une différence importante entre les deux conceptions. Dans une orchestration, le processus est toujours contrôlé du point de vue de l'une des parties. Une orchestration exprime une organisation de workflow spécifique. Cela signifie que l'organisation possède et contrôle la logique d'une orchestration, même si cette logique implique une interaction avec les partenaires commerciaux externes. La chorégraphie est plus axée sur la collaboration : toutes les parties impliquées dans le processus doivent décrire complètement le rôle qu'elles jouent dans le processus.

Elle n'est pas nécessairement la propriété d'une entité unique. Elle agit comme un modèle communautaire d'échanges utilisé à des fins de collaboration par les services de fournisseur de différentes entités. Une orchestration de services web est vue comme un orchestre, où un processus particulier joue le rôle de chef d'orchestre. Celui-ci coordonne l'exécution des autres services. Il s'agit d'un point de vue individualiste : les services, à l'exception de l'orchestrateur, n'ont pas besoin de savoir qu'ils font partie d'une plus grande partie. Il y a un seul document de haut niveau représentant les étapes du processus et ce document est seulement connu et traité par l'orchestrateur. Dans une chorégraphie, en revanche, les services sont vus comme des danseurs qui savent exactement quoi faire et de quelle manière interagir avec les

autres parties. Il s'agit d'une approche collaborative et chacun des participants a besoin d'un document dans lequel l'interaction est décrite. Ces documents mettent l'accent sur l'échange de message.

2.3.3 Langages de composition des services web

Ils existent plusieurs langages de définition pour la composition de services web. Dans cette section nous allons présenter brièvement quelques uns d'entre eux , notamment les plus connus et acceptés afin de faire le choix le plus convenant pour notre projet.

XLANG : XML Business Process Language

Le langage XLANG [22] est une extension de la spécification WSDL. Elle fournit en même temps un modèle pour une orchestration des services et des contrats de collaboration entre celles-ci.

Les actions sont les constituants de base d'une définition de processus de XLANG. Le fichier de description de service XLANG contient donc la description WSDL, et y ajoute les deux autres genres d'action : arrêts (date-limite et durée) et exceptions.

WSCI : Web Service Choreography Interface

WSCI [43] est un langage reposant sur XML. Il propose de se focaliser sur la représentation des services web en tant qu'interfaces décrivant le flux de messages échangés (la chorégraphie de messages).

Il propose ainsi de décrire le comportement externe observable du service. Pour cela, WSCI propose d'exprimer les dépendances logiques et temporelles entre les messages échangés à l'aide de contrôles de séquences, corrélation, gestion de fautes et transactions. On remarque que WSDL et ses définitions abstraites sont réutilisées afin de pouvoir également décrire par la suite les modalités de concrétisation des éléments manipulés pour modéliser un service.

WSCL : Web Service Conversation Language

WSCL [44] propose de décrire à l'aide de documents XML les services web en mettant l'accent sur les conversations de ceux-ci. En outre, les messages à échanger sont pris en compte. WSCL a été pensé pour s'employer conjointement avec WSDL.

Les définitions WSDL peuvent être manipulées par WSCL pour décrire les opérations possibles ainsi que leur chorégraphie. En retour, WSDL fournit les concrétisations vers des définitions de messages et des détails techniques pour les éléments manipulés par WSDL.

Le Langage BPEL : Business Process Execution Language

BPEL [13] ou BPEL4WS est un langage issu de la fusion de deux langages prédécesseurs :

WSFL (Web Services Flow Language) et XLANG (Web Services for Business Process Design). WS-BPEL 2.0 est devenu un standard OASIS en 2007. BPEL permet de modéliser les processus métiers en termes d'orchestration, en décrivant le flux de données (les variables) et le flux de contrôle (les activités simples et composées, les partenaires, etc.), son but est de créer une fonctionnalité complexe qui réutilise les services existants. Il permet aussi de donner une vue centralisée de l'exécution de la composition. Nous distinguons deux types de processus BPEL (abstrait et exécutable).

Processus abstrait :

Ce processus spécifie les messages échangés entre les différents services web composants sans indiquer le comportement de chacun d'eux.

Processus exécutable :

Ce processus permet de spécifier l'ordre d'exécution des activités, les partenaires concernés, les messages échangés entre ces partenaires, et les mécanismes de gestion des erreurs potentielles. ce processus peut être exécuté au moyen d'un engin d'orchestration.

Comme montré dans la figure 2.4, la spécification BPEL offre plusieurs éléments décrits comme suit :

<Process> : L'élément < process > représente la racine du fichier BPEL. Il contient l'attribut " name " qui permet de définir le nom du processus.

<PartnerLinks> : Il permet de spécifier les différents partenaires participant dans la composition. Il est composé d'un ou plusieurs éléments <PartnerLink>. Ce dernier contient les attributs suivants :

- name : nom donné au PartnerLink.

FIG. 2.4: Structure d'un fichier BPEL.

- myRole : spécifie le rôle du processus BPEL.

- partnerRole :spécifie le rôle du partenaire ou du client.

- partnerLinkType :représente le type de PartnerLink défini dans la description WSDL.

Si l'attribut < myRole > est uniquement utilisé (sans <partnerRole >), cela signifie que seules les interactions vers le processus sont autorisées. Dans le cas opposé (si l'attribut <partnerRole > est uniquement utilisé sans <myRole >), seules les interactions vers les partenaires et les clients sont autorisées. Il est à noter que les deux attributs peuvent être utilisés en même temps.

<Variables> : Il permet de définir les différentes variables utilisées dans le processus BPEL. Ces dernières servent à stocker des données ou des messages échangés afin de les réutiliser ultérieurement.

<CorrelationSets> : Une orchestration peut être exécutée plusieurs fois. Chaque exécution est nommée instance. Pour acheminer les données à une instance particulière, nous utilisons l'élément <correlationSets>.

<FaultHandlers> : permet de gérer les exceptions au niveau du <catch>.

<EventHandlers> : Il représente les événements pouvant survenir au cours de l'exécution. Il permet aussi d'associer un traitement à chacun de ces évène-

ments.

Les activités :

- <**receive**> :une attente bloquante d'un message entrant.

- <**reply**> : Elle retourne un message suite à une réception d'un autre message (à l'aide de la primitive <receive>).

La combinaison receive-reply utilise une opération request-response d'un port-Type du processus.

- <**invoke**> : Elle permet l'appel d'un partenaire (service) en se basant sur une opération de type one way ou request-response.

- <**Assign**> :Cette activité permet la mise à jour des variables.

- <**Throw**> : Elle permet d'indiquer les erreurs et les exceptions survenues lors de l'exécution du processus et de les envoyer au catch de fault handler.

- <**Wait**> : Elle permet de bloquer l'exécution du processus pour une période donnée.

- <**Empty**> : C'est l'opération rien-faire, elle est utile pour la synchronisation.

Activités de base peuvent être combinées pour définir un algorithme complexe spécifiant les étapes par lesquelles passe le processus en utilisant les activités structurées telles que :

- <**Sequence**> : Elle définit une suite d'activités exécutées par ordre d'apparition dans la séquence.

- <**flow**> : Elle définit des activités exécutées en parallèle.

- <**if**> : Elle définit un branchement conditionnel.

- <**while**> : Elle modélise une boucle conditionnelle.

- <**Pick**> : Elle bloque une activité jusqu'à la réception d'un message ou l'expiration d'une période de temps.

De façon générale, un processus BPEL commence par une activité <receive> et se termine par une activité <reply>. Il est interprété par un moteur d'orchestration tels que : Oracle BPEL Process Manager, TIBCO Business Works, etc..

```
<process name="myMeteo">
    <partners>
        <partner name="geoIPService"
                 serviceLinkType="geoIPLink"
                 myRole="geoIP"
                 partneRole="geoIPContext"/>
        <partner name="globalWeather"
                 serviceLinkType="globalWeatherLink"
                 myRole="weather"
                 partnerRole="globalWeatherService"/>
    </partners>
    <containers>
        <container name="getLocalisation" messageType="getGeoIPContextSoapIn"/>
        <container name="getWeatherContainer" messageType="getWeatherSoapIn"/>
    </containers>
    <flow>
        <invoke partner="geoIPService"
                portType="GeoIPServiceSoap"
                operation="GetGeoIPContext"
                inputContainer="getLocalisation" />
        <invoke partner="globalWeather"
                portType="GlobalWeatherSoap"
                operation="GetWeather"
                inputContainer="getWeatherContainer" />
    </flow>
</process>
```

FIG. 2.5: Définition du processus exécutable de la demande du service de météorologie myMeteo à l'aide de BPEL4WS.

2.3.4 Concepts de base du traitement transactionnel

Notion de Transaction :

Une transaction peut être considérée comme une unité de traitement cohérente et fiable. Une transaction prend un état d'une base de données, effectue une ou des actions sur elle et génère un autre état de celle-ci. Les actions effectuées sont des opérations de lecture ou d'écriture sur les données de la base. Par conséquent, une transaction peut être définie comme étant une séquence d'opérations de lecture et d'écriture sur une base de données, qui termine en étant soit validée soit abandonnée. Si la base de donnée est cohérente au début de la transaction, alors elle doit rester cohérente à la fin de l'exécution de la transaction bien que cette dernière peut s'exécuter de manière concurrente avec d'autres ou qu'une panne survienne lors de son exécution. Une base de données est dite cohérente si elle est correcte du point de vue de l'utilisateur, c'est à dire qu'elle maintient les invariants de la base ou les contraintes d'intégrité. La notion de cohérence recouvre plusieurs dimensions comme décrit dans [21]. Du point de vue des demandes d'accès, il s'agit de gérer l'exécution concurrente de plusieurs transactions sans que les mises à jour d'une transaction

ne soient visibles avant sa validation, on parle de cohérence transactionnelle ou isolation. Du point de vue des données répliquées, il consiste à garantir que toutes les copies d'une même donnée soient identiques, on parle de cohérence mutuelle. La cohérence transactionnelle est assurée à travers quatre propriétés, résumées sous le vocable ACID :

Atomicité : toutes les opérations de la transaction sont exécutées ou aucune ne l'est. C'est la loi du tout ou rien. L'atomicité peut être compromise par une panne de programme, du système ou du matériel et plus généralement par tout évènement susceptible d'interrompre la transaction.

Cohérence : La cohérence signifie que la transaction doit être correcte du point de vue de l'utilisateur, c'est-à-dire maintenir les invariants de la base ou contraintes d'intégrité.

Une transaction cohérente transforme une base de données cohérente en un base de données cohérente. En cas de non succès, l'état cohérent initial des données doit être restauré.

Isolation : elle assure qu'une transaction voit toujours un état cohérent de la base de données.

Pour ce faire, les modifications effectuées par une transaction ne peuvent être visibles aux transactions concurrentes qu'après leur validation. En outre, une transaction a une opération marquant son début (begin transaction) et une autre indiquant sa fin (end transaction). Si la transaction s'est bien déroulée, la transaction est terminée par une validation (commit). Dans le cas contraire, la transaction est annulée (rollback, abort).

Durabilité :une fois que la transaction est validée, ses modifications sont persistantes et ne peuvent être défaites. En cas de panne de disque, la durabilité peut être compromise.

Les propriétés ACID sont très difficiles à maintenir car elles représentent un frein aux performances du système. Par exemple, l'atomicité cause un sérieux problème quand l'environnement est réparti sur un système à large échelle puisque toutes les sites participant à une transaction doivent valider localement avant que la transaction ne soit validée globalement. Autrement dit, le maintien de la cohérence exige que toutes les sites participants soient mises à jour au sein de la même transaction, ce qui ralentit la validation. Pour des besoins de performances, certaines propriétés ne sont pas parfois garanties dans l'optique d'améliorer les performances du système. En effet, les propriétés C et I peuvent être relâchées au profit d'un degré de concurrence plus élevé et donc d'un débit transactionnel plus important.

2.3.5 Mécanismes garantissant les propriétés d'ACIDité

1 Contrôle de concurrence :

Le contrôle de concurrence est chargé d'empêcher les transactions d'utiliser de manière intempestive les données modifiées par les transactions concurrentes non encore validées. On distingue deux grandes familles de contrôle de concurrence : continues et par certification. Les méthodes continues, ou pessimistes, effectuent un contrôle lors de chaque accès à une donnée partagée afin de détecter les conflits a priori, tandis que les méthodes par certification, ou optimistes, attendent la fin de la transaction pour vérifier a posteriori si des conflits se sont produits. Dans les deux familles de méthodes, si les conflits entraînent que les transactions en cause ne peuvent pas être sérialisées alors il faut choisir une ou plusieurs victimes qui seront rejetées.

Les effets engendrés sur la base par les victimes sont annulés avant que celles-ci ne puissent tenter de s'exécuter de nouveau. Si le but essentiel du contrôle de concurrence est de garantir l'isolation, les rejets qu'il génère sont gérés par la composante suivante qui assure l'atomicité.

2 Validation Et Reprise aprés panne : Il s'agit de garantir les propriétés d'atomicité et de permanence en présence de défaillances telles que celles qui provoquent l'arrêt de la transaction avant sa terminaison (Violation de la propriété de l'atomicité) et celle qui entraîne une disparition de l'information en mémoire secondaire (Violation de la propriété de permanence).

La mise en oeuvre du contrôle de l'atomicité repose sur deux principes de base la redondance et l'exécution de la transaction en deux étapes [27] :

La redondance : un journal est prévu en mémoire permanente permet l'enregistrement des informations nécessaires pour défaire (undo) et refaire (redo) les actions d'une transaction. Les deux formes de journaux les plus répondu sont le Journal avant (before journal) : qui stocke les valeurs des objets avant leur modification par une transaction, il est utilisé pour défaire les actions d'une transaction et le journal-après(after-journal) qui stocke les valeurs des objets modifiés par une transaction, il est utilisé pour refaire les actions d'une transaction.

L'exécution de la transaction se fait en deux étapes : calcul et validation. Pendant l'étape de calcul qui correspond à l'exécution de la transaction, les modifications sur les objets sont enregistrées dans le (les) journal (aux), et pendant l'étape de validation (commitment) le système consolide les actions qui ont été exécutées sur les objets pendant la phase de calcul. La validation consiste à propager les mo-

difications enregistrées dans l'espace de travail de la transaction vers la mémoire secondaire. Les journaux permettent de fiabiliser cette opération. Lors d'un redémarrage après une défaillance, le système transactionnel parcourt le journal pour analyser l'état des transactions interrompues au moment de l'incident.

Pour chaque transaction, il exécute une procédure selon l'état enregistré dans le journal :

- **Etat actif :** défaire les actions de la transaction.
- **Etat validé :** refaire la procédure de validation.
- **Etat invalidé :** refaire la procédure d'invalidation.
- **Etat terminé ou abandonné :** pas d'action.

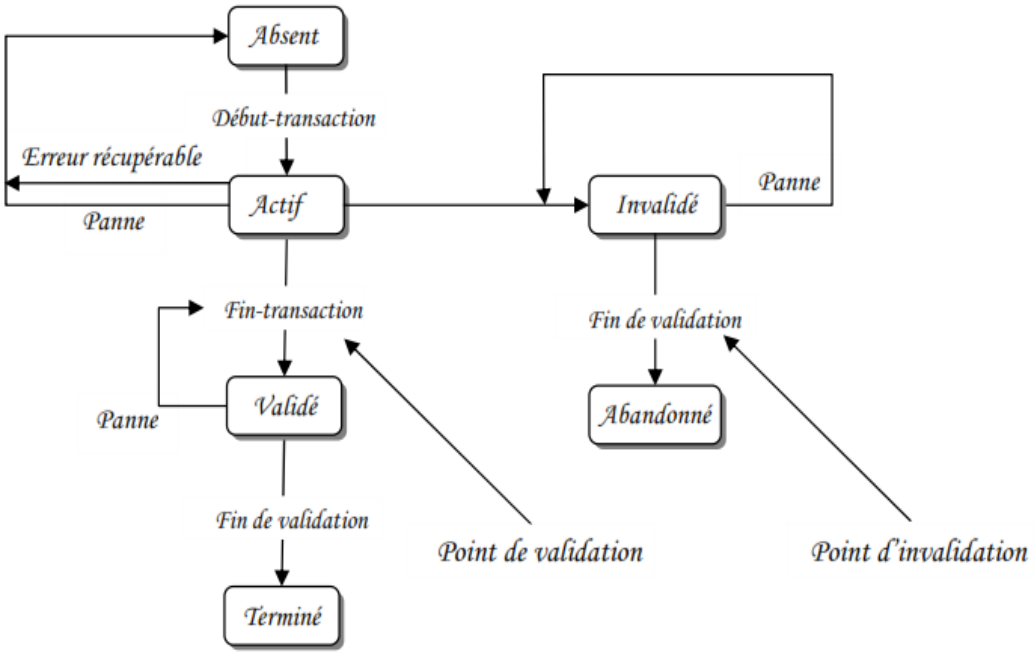

FIG. 2.6: Digramme d'état d'une transaction.

2.3.6 La validation de transactions dans les systèmes distribués :

Revenons sur la terminaison d'une transaction. Si la transaction est répartie, il n'est pas certain qu'elle puisse être validée, malgré le fait qu'elle s'est déroulée normalement jusqu'au point de terminaison. Par exemple, elle a pu modifier des objets se trouvant sur un site qui est ensuite tombé en panne. Si les effets de la

transaction sont perdus sur le site défaillant, la transaction ne peut pas être validée.

Autrement dit, la validation d'une transaction répartie nécessite un protocole permettant d'établir un consensus entre les participants de la transaction. La mise en oeuvre d'un tel protocole est compliquée par le fait qu'il doit être résistant aux pannes (par exemple, il doit prendre en compte les pannes de communication). Par ailleurs, il est souhaitable qu'il puisse permettre aux participants d'abandonner la transaction à tout moment (par exemple, pour débloquer une ressource afin qu'une autre transaction puisse y accéder). En fait, ceci n'est pas toujours possible, chaque protocole ayant "une fenêtre de vulnérabilité" dans laquelle les participants n'ont pas le droit d'abandonner la transaction unilatéralement. Un participant entre dans la fenêtre de vulnérabilité lorsqu'il est prêt à valider ; après avoir communiqué cette décision aux autres participants, il reste bloqué jusqu'à l'obtention d'un consensus sur le mode de terminaison (par validation ou par abandon). Dans la suite nous présentons le protocole de validation à deux phases [16] qui est le plus répandu, car assez général et performant.

Le protocole de validation à deux phases : Considérons le modèle centralisé (l'un des participants est le coordinateur et les autres sont les subordonnés).

1. Dans la 1-ère phase, le coordinateur demande aux participants s'ils peuvent valider la transaction.

- Si un des participants répond NON, la transaction est abandonnée.

- Un participant répond OUI, s'il a réussi à préparer la transaction pour la validation (par exemple, il a sauvegardé dans le journal les enregistrements permettant de refaire localement la transaction). De plus, en répondant OUI, le participant renonce au droit d'annuler la transaction unilatéralement (il entre dans la fenêtre de vulnérabilité) et il attend la décision du coordinateur.

2. Le coordinateur décide de valider la transaction si tous les participants ont répondu par OUI. Dans ce cas, il doit enregistrer dans le journal le fait que la transaction peut être considérée comme validée. À partir de cet instant, rien ne peut empêcher la validation de la transaction.

Dans la deuxième phase, le coordinateur fait connaître sa décision aux participants des autres sites. Si la décision est celle de valider, les participants valident localement les effets de la transaction. Dans le cas contraire, les participants qui ont répondu OUI abandonnent localement la transaction.

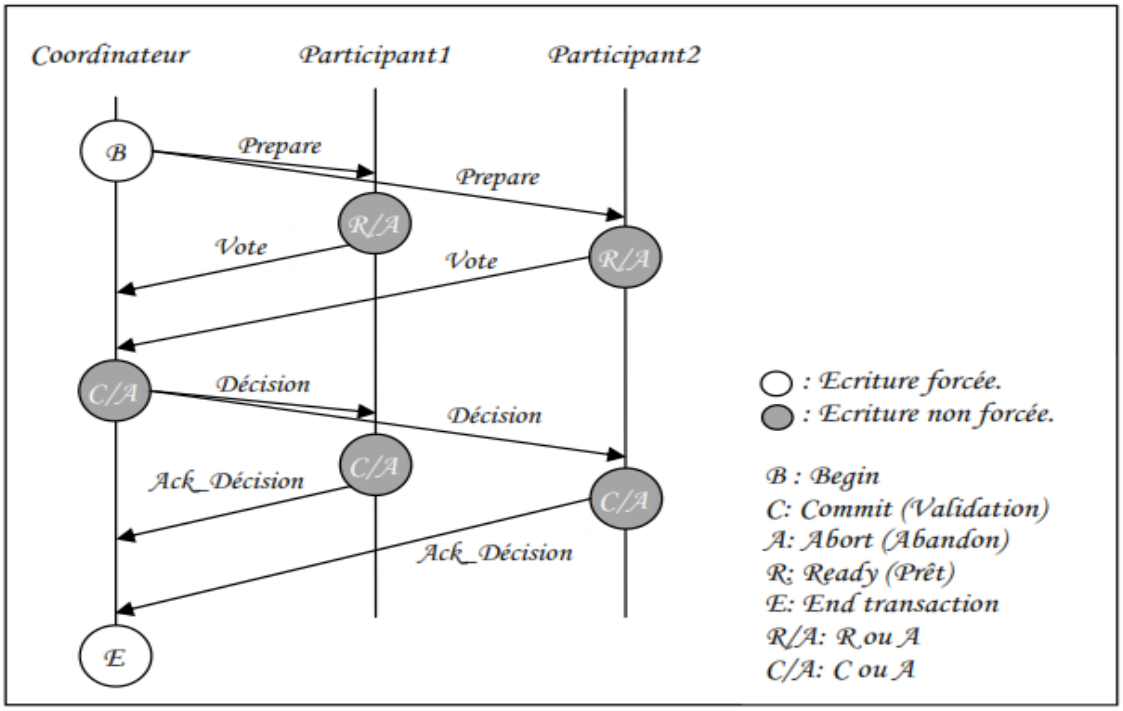

FIG. 2.7: Le protocole 2PC de base.

2.4 Aspects transactionnels dans la composition de services web

2.4.1 Les services web et les aspects transactionnels

Les services web sont des programmes modulaires, indépendants et auto descriptifs, qui peuvent être découverts et invoqués via Internet ou un intranet. Les services web présentent une technologie qui permet des interactions entre applications qu'elles soient intra ou interentreprises. Leurs particularités par rapport aux autres technologies de l'informatique répartie résident dans le fait qu'ils offrent un modèle de composants à couplage faible en utilisant la technologie Internet comme infrastructure pour la communication. La technologie des services web utilise des langages et des protocoles qui reposent sur XML (WSDL : pour décrire les services web, SOAP : pour structurer les messages). [4, 36]

Un des concepts intéressants qu'offre la technologie des services web est la possibilité de définir un nouveau service à valeur ajouté (dit service web composé) par composition de services web existants. Par exemple, pour partir en vacances, le client fait appel au service composé " organiser un voyage ", ce dernier regroupe des ser-

vices web appartenant à des organisations différentes pour réaliser les fonctionnalités demandées : (acheter un billet de train, réserver une chambre d'hôtel ,etc..).

1 Classification transactionnelle des services web élémentaires :

Un service web munie de propriétés transactionnelles ou simplement un service web transactionnel, est un service web qui permet l'acquisition de ressources par un client (par exemple l'achat d'un billet de train, la réservation d'une chambre d'hôtel). Cette acquisition ne peut s'opérer que par le biais d'une transaction entre le client et le fournisseur du service. Ainsi, le fournisseur doit doter ses services web d'opérations pour permettre la cession de la ressource au client. Il est possible de caractériser un service web selon les propriétés des transactions qu'il est capable d'exécuter [14, 15]. Plusieurs classifications transactionnelles ont été proposées :

Dans [14, 15], une classification basée sur le degré d'atomicité du service web a été proposée. Par conséquence, un service web peut être : atomique, quasi-atomique ou non-atomique :

un service est dit d'exécution atomique (c'est-à-dire avec la sémantique du tout ou rien) lorsqu'il offre les opérations suivantes : obtenir (si possible) la réservation d'une ressource, annuler une réservation demandée, valider définitivement l'acquisition d'une ressource réservée ; un service quasi-atomique offre une opération pour acquérir (si possible) définitivement une ressource, il offre aussi une opération de compensation ; un service est dit non-atomique lorsqu'il fournit seulement une opération pour acquérir définitivement une ressource. Il n'y a pas d'opération de réservation ni de compensation.

Dans [33, 34], les auteurs se sont inspirés du modèle de transaction flexible d'où les définitions suivantes :

-Un service S est dit compensable " Compensatable " s'il offre des mécanismes de compensation pour annuler sémantiquement son travail.

-Un service S est dit rejouable " Retriable " s'il se termine toujours avec succès après un nombre fini d'activation.

-Un service S est dit pivot " Pivot " si une fois qu'il se termine avec succès, ses effets ne peuvent pas être compensés.

Un service web peut être : compensable, rejouable, pivot, et peut même combiner deux propriétés telles que (rejouable et pivot) ou (rejouable et compensable). Dans [30], les auteurs ont considéré qu'une opération offerte par un service web peut être soit : compensable, quasi-compensable, rejouable ou pivot. Une opération quasi-compensable est une opération qui s'exécute sous un système qui supporte les mécanismes de validation à deux phases 2PC.

2.4.2 Notions transactionnelles dans la composition de services web

Un service web composé peut être modélisé par un graphe des tâches. Un service web composé peut être validé même si certaines de ses tâches échouent, ces tâches sont dites non vitales, les autres sont dites vitales. Un service web composé ne peut être validé que si toutes ses tâches vitales ont été validées.

Une tâche peut avoir des tâches alternatives ou contingentes ; dans ce cas elle est dite remplaçable. Une tâche alternative est exécutée si la tâche à laquelle elle est associée échoue. La vitalité et la "remplaçabilité " des tâches sont des propriétés transactionnelles qui doivent être exprimées au niveau du service web composé. Remarquons que les propriétés transactionnelles d'un service web composés peuvent être changées durant l'exécution. Par exemple si une tâche vitale échoue, l'utilisateur peut rendre cette tâche non vitale empêchant ainsi l'annulation de la composition.

La sélection des services web participant à une composition consiste à choisir des services web qui répondent aux tâches qui composent un service web composé. Les services web participants sont faiblement couplés, indépendants les uns des autres, et ont des propriétés transactionnelles hétérogènes. Un service web composé doit supporter les propriétés des services web participants à sa composition. En effet, il arrive très fréquemment que plusieurs services web répondent à une même tâche. Ces services sont dits des services alternatifs. Si un service web participant échoue, il est possible de le remplacer par l'un des ses services alternatifs, empêchant ainsi l'abandon de la composition.

Le domaine des services web est dynamique et évolutif ; de nouveaux services peuvent y être ajoutés, les services existants sont constamment modifiés, momentanément suspendus, ou finalement supprimés. Une réponse à cette volatilité des services web peut être le choix dynamiquement (au moment de l'exécution de la composition) des services participants (ou des services web alternatifs), cela permet d'augmenter les chances de validation de la composition. Il faut noter que les interactions entre les services web ont des durées variables, elles peuvent être de courtes ou de longues durées.

La volatilité des services web, la variabilité et le changement dynamique des propriétés transactionnelles des services web composés rendent difficile, voir impossible, la prévision statique dès la conception de tous les scénarios pouvant se présenter du-

rant la composition d'un service web composé ; d'où le besoin de techniques dynamiquement adaptables. Les caractéristiques citées ci-dessous montrent le besoin de solutions flexibles et adaptables pour la composition de services web avec propriétés transactionnelles. En effet, plusieurs solutions ont été proposées pour résoudre le problème de la composition des services web avec propriétés transactionnelles. Ces solutions s'inscrivent soit dans le cadre des protocoles spécifiques ou dans le cadre des propositions académiques.

2.4.3 Protocoles spécifiques

Les efforts de standardisation ont abouti à un certains nombre de protocoles que nous citons dans ce qui suit :

1. **" BTP " Business Transactions Protocol [18] :**

   BTP s'appuie sur une variante du protocole de validation à deux phases pour garantir la propriété d'atomicité. Il introduit deux modèles de transactions :

   (i) Atom : pour garantir une atomicité stricte.

   (ii) Cohésion : qui permet de relâcher la propriété d'atomicité.

   BTP souffre d'un inconvénient majeur : il ne fait pas la distinction entre le protocole de transaction, et les aspects fonctionnels. Aussi, il exige que tous les services web participants supportent les mécanismes de validation à deux phases, cette condition rend la solution très restrictive.

2. **" WSTF " Web Services Transaction Framework [28] :** La spécification WSTF sépare le protocole chargé de la coordination " Web Services Coordination : WS-C " du protocole chargé de la transaction " Web Services Transaction WS-T ".

   WS-T introduit deux modèles de transactions :

   (i) Web Services Atomic Transaction " WS-AT " : garantie l'atomicité stricte pour les transactions courtes.

   (ii) Web Services Business Activity " WS-BA " : permet le relâchement de la propriété d'atomicité.

   WS-T exige soit que tous les services web participants disposent des mécanismes de validation de deux phases, soit tous les services web participants exposent des opérations de compensation ce qui est très exigeant.

3. **" WS-CAF " Web Service Composite Application Framework [3, 15, 45] :**

WS-CAF est un canevas pour composer des services web de différentes origines (.Net,J2EE, CORBA, etc..). Il fournit les spécifications pour résoudre les problèmes liés à la fourniture d'une infrastructure commune où il est nécessaire de gérer les différents contextes liés à la diversité des données, comme c'est le cas des services web appartenant à différentes entreprises mais qui sont composés pour produire un résultat commun. WS-CAF comporte trois spécifications : un gestionnaire de contexte, un coordinateur générique et un support pour les modèles de transactions.

" WS-CTX " Web Service Context :

Le contexte présente des informations additionnelles qui permettent d'améliorer l'utilisation d'un service web. Les informations de contexte sont transférées dans les messages SOAP. WS-CTX comporte des mécanismes qui permettent de définir, de structurer et de partager le contexte [47].

" WS-CF " Web Services Coordination : définit " le coordinateur " qui est responsable d'augmenter et de diffuser le contexte. WS-CF supporte les trois services suivants [46] :

(i) Service d'activation : commence une nouvelle activité et spécifie les protocoles de coordination disponibles.

(ii) Service d'enregistrement : permet aux services web de s'enregistrer et de choisir le protocole de coordination.

(iii) Service de coordination : définit le comportement des modèles de coordination spécifique. Par exemple : ACID Transaction,..

" WS-TXM " Web Services Transaction Management : il permet de définir trois modèles de transaction [3, 45] :

- ACID Transaction (AT) : conçu pour gérer des transactions courtes selon la sémantique ACID.

- Long Running Action (LRA) : est une activité, ou groupe d'activités, qui ne garantissent pas obligatoirement les propriétés ACID. Tous les services web participants à une LRA doivent avoir des opérations de compensation (Compensators).

Lors de l'exécution d'une LRA si tous les services web participants valident les activités demandées, la LRA sera validée. Sinon, la LRA sera annulée et les participants qui ont déjà validé des activités demandées procèdent à des compensations.

- Business Process (BP) : ce modèle reflète le concept de " processus métier "

qui a pour rôle la réalisation d'une fonction spécifique de l'entreprise. Un BP regroupe un ensemble de transactions atomiques (AT) ou de Long Running Actions (LRA) selon les besoins de l'application.

2.4.4 Approches académiques

Parmi les solutions académiques proposées, nous avons étudié deux approches que nous avons jugé les plus pertinentes.

1. **" ABT " Web Services In an Agent Based Transaction Model [37] :**

   L'approche ABT décrite dans [37] définie un système multi-agents pour gérer les transactions pour les services web. Elle propose l'utilisation des quatre agents :

   (i) l'agent de coordination et de délégation.

   (ii) l'agent de ressource.

   (iii) l'agent d'encapsulation.

   (iv) l'agent de découverte et d'estimation.

   Cette approche permet de composer des services web et des ressources locales en :

   - respectant l'autonomie des services participants.

   - incluant des agents permettant d'encapsuler les services web. Ces derniers permettent de cacher l'hétérogénéité des services web participants.

   - incluant un agent de découvert et d'estimation évolutif.

   - utilisant une liste des participants alternatifs.

   Mais cette approche n'a pas bien étudié les problèmes transactionnels rencontrés lors de la composition de services web à savoir :

   - l'expression des propriétés transactionnelles au niveau de service web composé n'est pas étudiée.

   - les interactions entre les agents d'encapsulation et les services encapsulés.

2. **Reliable Web Services Compositions By Ensuring The Failure Atomicity [33] :**

   Cette étude propose un modèle de services web transactionnel qui a pour but d'assurer des exécutions correctes des services web composés. Ce modèle distingue en particulier entre le flot de contrôle et le flot transactionnel d'un service web composé transactionnel (SCT). Le flot de contrôle définit l'ordre

d'invocation des services composants. Le flot transactionnel définit les mécanismes de recouvrement en cas d'échecs.

Dans ce modèle, trois approches ont été développées pour assurer des compositions fiables de services web : la première repose sur la validation du modèle de composition du SCT conformément aux besoins des concepteurs. La deuxième procède par réingénierie du SCT. La troisième repose sur le concept de patron transactionnel. Bien que cette approche présente une solution pour assurer des compositions fiable des services web, elle ne permet pas d'exprimer les propriétés transactionnelles au niveau du service web composé.

2.5 Conclusion

La composition des services web permet de combiner des services web élémentaires afin d'obtenir des services plus élaborés. Elle décrit un ensemble d'interactions ou processus métier, faisant intervenir différents services web. La composition est décrite indépendamment de sa implémentation futur i.e. elle indique uniquement les types de services web nécessaires mais ne précise pas nominativement les services web qui serons utilisés. Par ailleurs, la composition peut comporter plusieurs niveaux en permettant à des services web élaborés d'être à leur tour combiné pour construire de nouveaux services.

Dans ce chapitre nous avons présenté quelques langages et spécifications qui permettent la composition des services, et nous avons détaillé le concept de transaction et l'effet de l'aspect transactionnel sur la composition des services web.

Chapitre 3

Implémentation

Sommaire

3.1 Introduction

Afin de pallier aux problèmes transactionnels rencontrés dans l'environnement des services web, de nombreux protocoles et approches académiques ont été proposés. Une des limitations principales de ces solutions est qu'elles n'offrent pas un bon niveau d'adaptabilité, en termes de support restreint du changement dynamique caractérisant cet environnement et aussi le changement dynamiques des besoins transactionnels des utilisateurs. Nous présentons notre implémentation ; nous nous concentrerons tout particulièrement sur la vérification des propriétés transactionnelles.

Dans ce chapitre nous décrivons un prototype implémentant l'environnement de recherche, de composition de services web. Nous décrivons en premier lieu, larchitecture proposée , ensuite nous présentons les corpus utilisés pour évaluer l'environnement, après nous montrons les expérimentations menées, et finalement nous discutons les résultats obtenus.

3.2 L'architecture proposée

L'architecture globale que nous proposons pour la composition des services web avec propriétés transactionnelles est illustrée par la figure suivante :

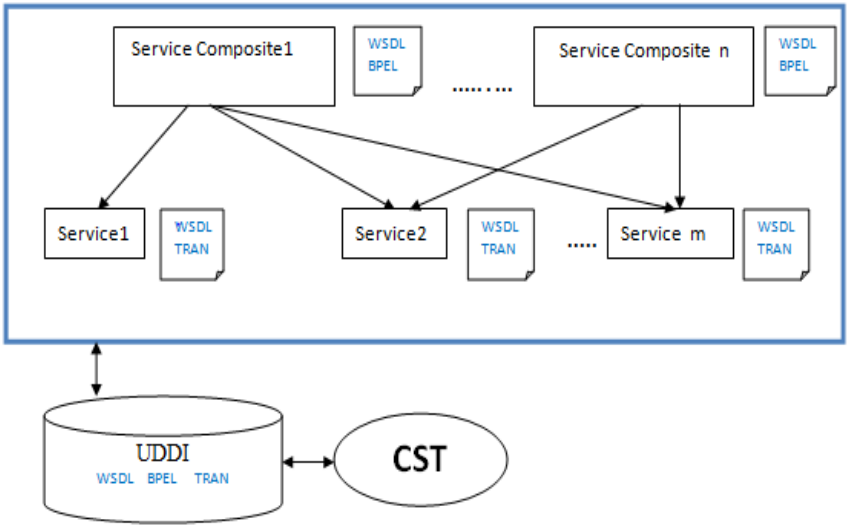

FIG. 3.1: Architecture proposée.

L'architecture est composée de deux parties :

La composition de services est gérée par le premier composant de l'architecture Dans laquelle, on distingue deux types de services :

Les services élémentaires : Ces services sont des services web élémentaires (service1.. n). Chaque service possède sa description WSDL et sa propriété transactionnelle.

Les services composés : (composite services) regroupent un ensemble de services. Chaque service composite est écrit en BPEL. Lors de l'étape de composition, plusieurs plans de composition peuvent être trouvés qui satisferaient les besoins de client, aprés de mettre en oeuvre la composition, on vérifie les propriétés transactionnelles donc on obtient un service composite transactionnel qui est noté CST. Ce dernier sera présenté dans le registre UDDI.

3.3 Présentation

La présentation de la mise en oeuvre de notre étude, se concentre, dans un premier temps, sur le cadre et les objectifs du développement d'une plateforme de services web, suivis, dans un deuxième temps, par la présentation de quelques outils technologiques utilisés permettant de déployer et d'exécuter des services web possédant une description comportementale.

3.3.1 Plateforme de services web :

Pour obtenir la plus grande portabilité, le langage Java a été retenu. De plus, de nombreux outils du domaine des services web étant développés principalement en Java, l'utilisation de Java nous permet de rester le plus proche de ceux-ci. Le tout, en utilisant au maximum les outils et bibliothèques libres de droit, permettant ainsi de s'affranchir des problèmes d'un outil non porté sur une plateforme donné ou dont le développement n'est plus assuré. Avant de vouloir aller plus loin dans la composition de services web avec vérification de certaines propriétés pour assurer que l'interaction se déroulera sans soucis, il est nécessaire de disposer d'un module permettant d'invoquer n'importe quel service web répondant à certains critères. Ce travail part d'un constat : l'invocation d'un service web basique disposant d'une description WSDL est réalisable relativement aisément avec les outils actuels. Par contre, l'invocation d'un service web complexe disposant d'une description comportementale avec un langage tel que BPEL n'est pas aussi facilement réalisable avec les

outils actuels. L'approche utilisée par les industriels consiste souvent à écrire du code guidant cette invocation pour un service web donné, ce code est alors trop proche de ce service web et n'est pas réutilisable pour un autre service web ou même une nouvelle version de celui-ci. Dans ce qui suit nous allons présenter plus de détails.

Langage de programmation :

Le langage utilisé pour la réalisation de notre outil est le langage orienté objet " Java " qui a été créé à la fin des années 80 lorsque bill Joy, cofondateur de Sun Microsystems en 1982, commença à imaginer un nouveau langage. Ceci incita un groupe de programmeurs de Sun Microsystems en l'occurrence : James Osling ,Patrick Naughton et MikeSheridan à développer un langage de programmation qu'ils ont baptisé OAK dans un premier stade pour prendre par la suite en janvier 1995 le nom JAVA qui veut dire café en argot American. Ce langage a été présenté officiellement le 23 mai 1995 au Sun World, après quoi plusieurs autres versions plus améliorées ont été réalisées par la maison Sun.La société Sun a été ensuite rachetée en 2009 par la société Oracle qui détient et maintient désormais Java. Notre choix s'est posé sur ce langage compte tenu des multiples avantages qu'il offre, entre autres :

- Java est un langage dynamique, simple et robuste.

- La sémantique du langage Java est indépendante de la plateforme. Donc un programme écrit en Java fonctionne de manière indépendante de l'architecture matérielle.

Les logiciels développés avec le langage Java sont facilement portables sur plusieurs systèmes d'exploitation tels que : UNIX, Microsoft Windows, Mac OS.

- Java reprend en grande partie la syntaxe du langage C++, très utilisé par les informaticiens. Néanmoins, Java a été épuré des concepts les plus subtils du langage C++ et à la fois les plus déroutants, tels que l'héritage multiple qui a été remplacé par les interfaces. Les concepteurs ont privilégié l'approche orientée objet de sorte qu'en Java, tout est objet à l'exception des types primitifs (nombres entiers, nombres à virgule flottante, etc.).

- Java a donné naissance à un ensemble d'environnements de développement (Eclipse, Netbeans, etc.), des machines virtuelles (MSJVM, JRE) applicatives multiplateformes (JVM), une plateforme java de base (java SE) avec des API pour la création des interfaces graphiques (AWT, Swing). La portabilité du code Java est assurée par la machine virtuelle.

Environnement de développement :

NetBeans 8.0 :

NetBeans est un environnement de développement intégré (IDE) pour Java, placé en open source par Sun. En plus de Java, NetBeans permet également de supporter différents autres langages, comme C, C++, XML et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditer en couleur, projets multi-langage, éditeur graphiques d'interfaces et de pages web).

NetBeans est disponible sous Windows, Linux et d'autres systèmes d'exploitation. Il est lui-même développé en Java, ce qui peut le rendre assez lent et gourmand en ressources mémoires.

3.3.2 Plateformes d'exécution côté serveur :

Nous n'allons pas faire ici l'inventaire de toutes les plateformes d'exécution de services web, mais seulement présenter une plateforme que nous avons sélectionnée. Les critères de sélections sont basés sur leur disponibilité, leur état d'avancement dans leur développement et la communauté gravitant au tour de ces produits. Notre environnement de test est basé sur le serveur ActiveBPEL pour des raisons pratiques concernant l'installation et la configuration sur la plateforme utilisées pour notre travail. Bien évidement, la plateforme du côté client ne se limite pas à ce serveur, mais fonctionne avec tous les serveurs respectant le standard BPEL.

ActiveBPEL :

Le serveur d'applications inclus dans les outils ActiveBPEL est un serveur sous licence libre. Il est agrémenté par des outils, développés par Active Endpoints, de conception et de développement de services BPEL qui eux sont payants. Il repose sur le serveur Tomcat d'Apache, libre également. L'ajout à Tomcat du serveur ActiveBPEL est relativement simple. De plus, une fois déployé, il possède une interface d'administration entièrement utilisable par une interface web. Cette interface permet le déploiement et la gestion des services sur le serveur.

Afin de mieux connaître cet outil que nous avons utilisé pour la génération des processus BPEL, nous introduisons dans ce qui suit les capacités qu'il offre. Il offre une palette avec toutes les opérations possibles (voir section 2.2.3). Il faut sélectionner l'opération adéquate, la glisser-déposer (drag-drop) jusqu'à la représentation graphique du processus et la mettre dans l'emplacement préféré.

3.4 Etude de cas : Agence de voyage :

L'objectif principal de ce scénario est de faciliter la compréhension et de montrer l'enchaînement d'événements de notre étude à travers un exemple simple et réaliste. Nous ne cherchons pas ici à montrer que notre étude peut être utilisée pour le développement de service composés complexes, bien que cela soit tout à fait possible.

Une agence de voyage " e-TravelAgency " fournit typiquement les services pour :

consultation, réservation, paiement et annulation de billet d'avion, de chambre d'hôtel et de locations de voiture. Afin de fournir ces services à ses clients, l'agence de voyage doit établir des liens avec d'autres entreprises : compagnies aériennes, compagnies de location de voitures et réseaux hôteliers.

Une institution financière (banque) est également nécessaire pour faciliter les transactions financières entre le client et l'agence de voyages, ou entre l'agence de voyage et les autres partenaires.

Afin d'illustrer les propriétés transactionnelles des services web composés, nous utilisons la représentation à l'aide du graphe de tâches présentée dans [17]. Un service web composé peut être modélisé par un graphe des tâches, où chaque tâche représente une fonctionnalité demandée par le client (figure 3.2).

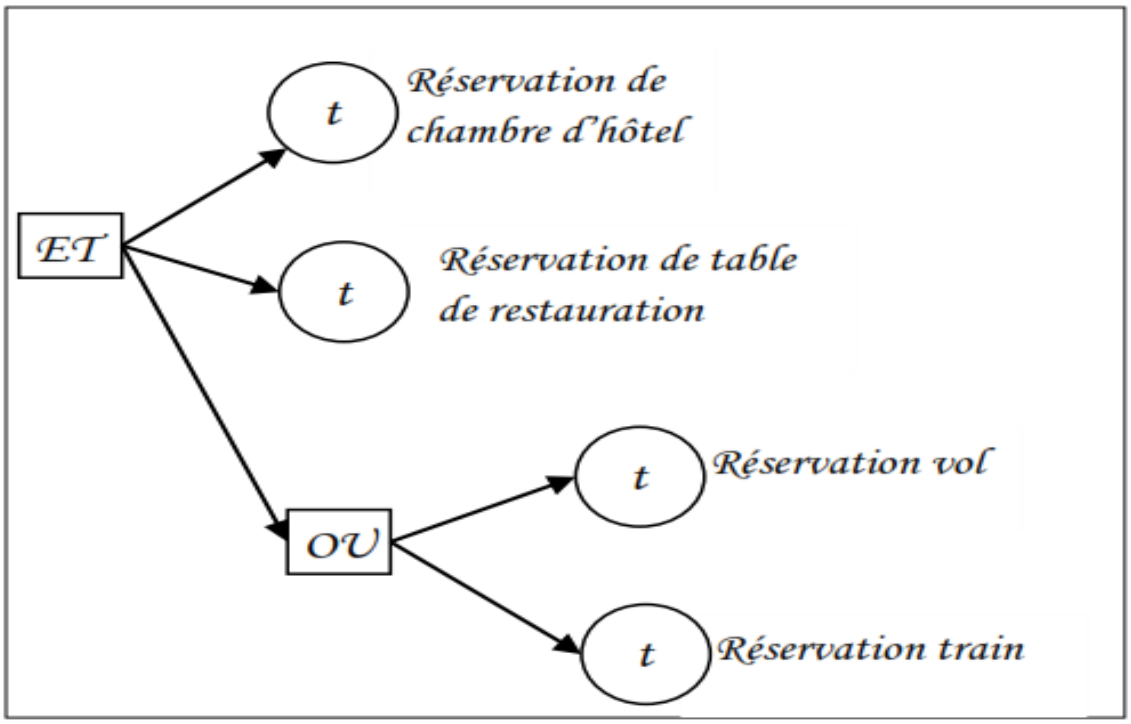

FIG. 3.2: Modélisation graphique Agence de voyage.

La figure 3.2 présente le service web composé SWC " Organiser un voyage " qui comporte quatre tâches t1, t2, t3 et t4 qui représentent respectivement une réservation de chambre de d'hôtel, une réservation de vol, une réservation de train et une réservation de table de restaurants.

Propriétés transactionnelles :

Prenons par exemple l'acquisition d'un billet d'avion. Si le service a un comportement atomique, il est possible de réserver le billet, en général pour un temps déterminé. Le fait de réserver le billet le rend indisponible aux autres clients, la ressource est donc verrouillée. C'est la garantie qu'au moment de valider la transaction la ressource sera toujours disponible. Avant que le délai ait expiré, le client peut soit annuler sa réservation, soit acheter le billet réservé. Dans le premier cas, la transaction est annulée et la ressource est rendue disponible à nouveau. Dans le second cas la transaction a réussi. L'expiration du délai provoque l'annulation de la transaction et libère la ressource. Si le service est quasi-atomique, le client peut acquérir le billet d'avion s'il est disponible, la compagnie peut cependant offrir des conditions particulières pour le report du voyage. L'opération de compensation consiste ici à modifier les dates du voyage par exemple, il s'agit de libérer la ressource acquise puis d'en acquérir une nouvelle. Finalement, dans le cas d'un service non-atomique le client peut acheter un billet d'avion mais, s'il change d'avis, il ne dispose d'aucune opération, ni d'annulation ni de compensation.

3.4.1 L'invocation de services dans une composition :

Les invocation des services dans une composition sont dans un ordre bien précis. On distingue les types d'exécution suivant :

- Exécution séquentielle : Dans une exécution séquentielle, un service est invoqué une fois que tous les web services précédents ont été exécutés.

FIG. 3.3: Exécution séquentielle.

- Exécution en parallèle : Dans ce cas, les web services s'exécutent en parallèle. Elle peut être représentée par l'opérateur AND, le web service 2 s'exécute en parallèle avec le web service 3.

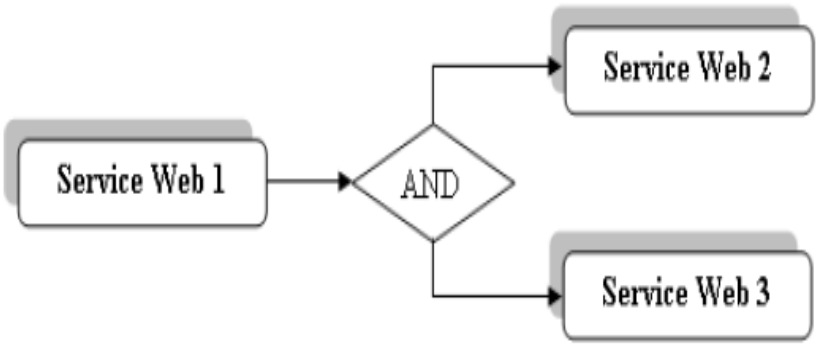

FIG. 3.4: Exécution parallèle.

- Exécution conditionnelle : Un chemin est choisi parmi plusieurs, ce choix est fait à l'aide d'une décision prise au moment de l'exécution. Elle peut être représentée par l'opérateur OR.

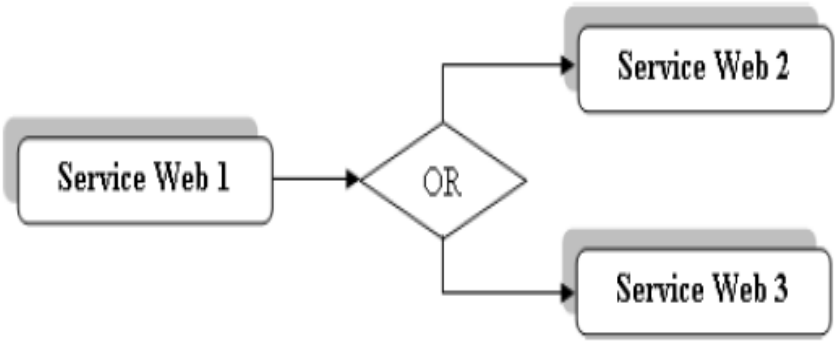

FIG. 3.5: Exécution conditionnelle.

- Exécution en boucle Un web service peut être invoqué plusieurs fois.

FIG. 3.6: Exécution en boucle.

3.4.2 Création de processus BPEL :

Notre processus BPEL Agence de voyage implémente trois services web : réservation de vol, réservation d'hôtel et location de voiture. Conformément à la terminologie BPEL, impliqués par PartnerLinks dans le processus BPEL. Les étapes suivantes sont prises pour créer le processus décrit utilisant ActiveBPEL Designer :

Etape 1 : Démarrage d'un nouveau processus.

Etape 2 : Ajout de références web.

Etape 3 : Planification et conception d'un processus :

Une fois les références web des PartnerLinks sont disponibles le processus BPEL peut être créé . Ce nouveau fichier .bpel aura au moins deux messages d'erreur. Ignorer ces erreurs pour le moment ; ouvrez le fichier de processus BPEL et commencer à spécifier les activités au processus BPEL. Dans ce qui suit, nous présentons toutes les activités nécessaires à la réalisation du notre processus BPEL.

Etape 4 : Ajout d'activités de processus et propriétés :

Utilisez l'Assistant *Opération* pour créer un BPEL activité avec le approprié WSDL la description. L'assistant vous aide ainsi à créer un nouveau fichier WSDL avec le partenaire type de lien, de type port, le fonctionnement et messages. Sinon, utilisez la palette d'outils du processus.

Comme premier pas sélectionnez une activité *Sequence* puisque tous les activités serons exécuté de manière séquentielle.

Sélectionnez une activité *Receive-Reply*, puisque les opérations de notre processus reçoit les entrées de la composition et répond à la fin d'exécution en récupérant les sorties engendrées par cette exécution. Terminer cet assistant, en utilisant les propriétés par défaut pour les options restantes.

La seconde activité dans le processus BPEL consiste à récupérer certaines données dans la demande de l'utilisateur (par exemple le nombre de place réservé dans un vol, le nombre de chambre, le nom de client, etc..). Pour ce faire, utilisez une activité *Invoke* BPEL, qui met en oeuvre un appel synchrone à une opération. Après cela, terminez l'assistant pour cette activité ; pour exécuter l'activité d'appel créé précédemment, un message doit être créé pour être utilisé dans l'opération réservation de vol. Utilisez donc une activité BPEL *Assign*.

Connecter les activités BPEL dans le processus à l'autre, selon l'ordre d'exécution ; pour compléter la spécification de processus BPEL, définir quelle activité instancie le processus BPEL. Dans cet exemple, ceci est la première activité de réception. Pour ce faire, sélectionnez l'activité *Recevoir*, allez à l'onglet de propriétés,

et définissez l'option *Instance Créer* Oui. Après tout ce que les messages d'erreur devraient avoir disparu et le processus BPEL est complet et prêt à être déployé et exécuté.

A la fin de ces étapes du processus BPEL doit ressembler à celui présenté dans la figure suivant :

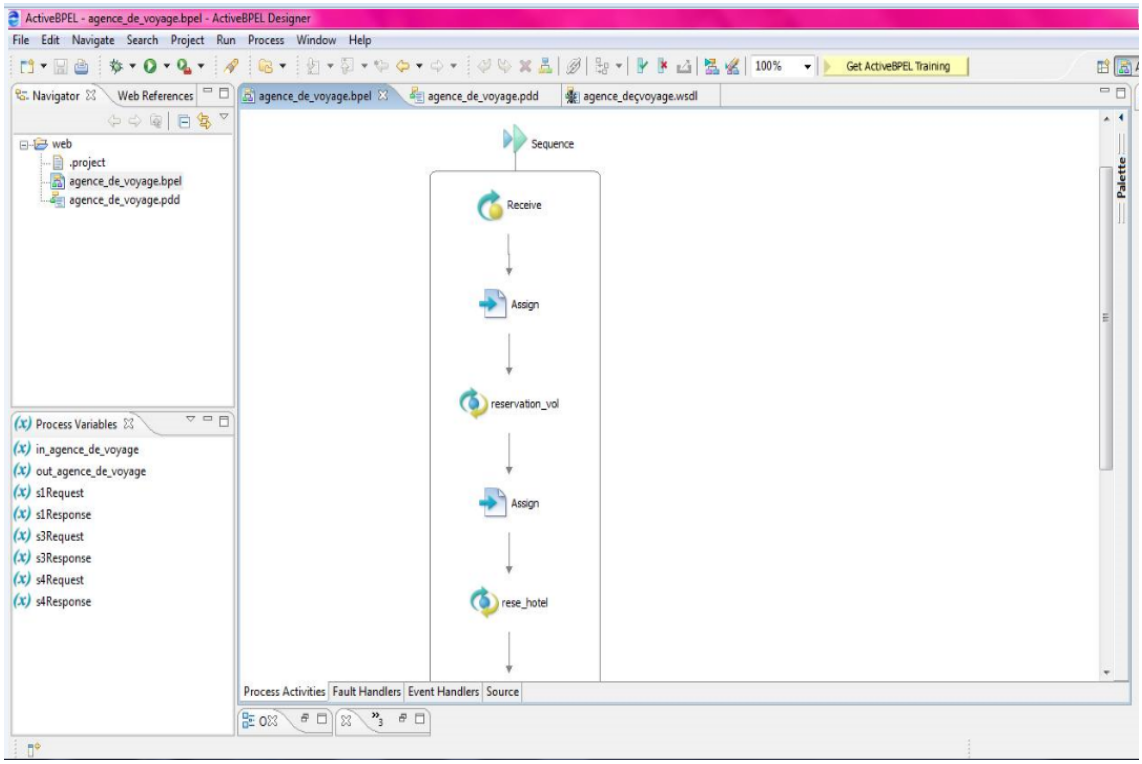

FIG. 3.7: Processus BPEL Complet (cas séquentielle).

Etape 5 : Ajout de la rémunération et de corrélation (Ceci est en fait une étape vide).

Etape 6 : Simuler le processus.

Etape 7 : Déploiement du processus et Exécution du processus sur le serveur.

Une fois un processus BPEL est créé, il doit être déployé afin d'être exécuté. Pour déployer un processus BPEL dans le ActiveBPEL Engine,des fichiers de déploiement doivent être créés, qui sont utilisés par le ActiveBPEL Engine pour déployer et exécuter le processus BPEL. Nous utilisons le Designer ActiveBPEL pour créer de tels fichiers. Dans ce qui suit, nous discutons les mesures nécessaires pour effectuer le déploiement, et de donner quelques indications sur la façon d'exécuter le processus. Nous discutons également comment surveiller et déboguer un processus BPEL.

Afin de déployer un processus BPEL, prendre les mesures suivantes :

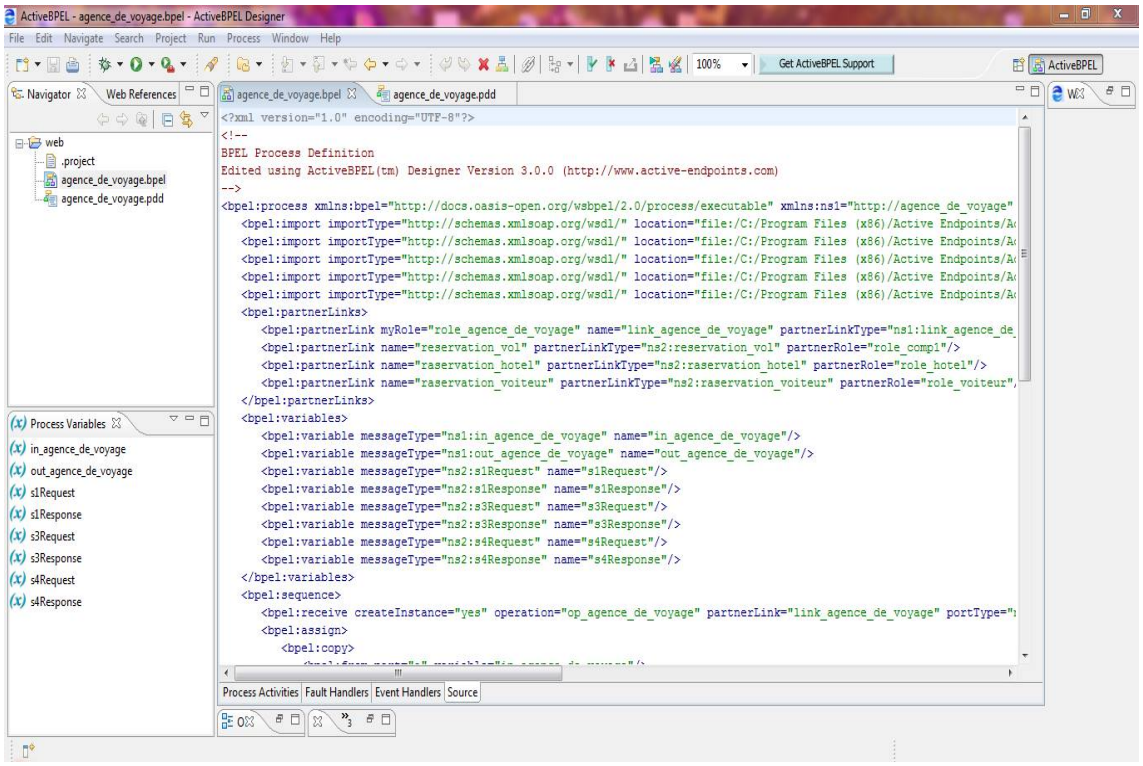

FIG. 3.8: Code source de fichier BPEL.

Dans le Concepteur de ActiveBPEL créer un nouveau descripteur de déploiement, sélectionnez le fichier BPEL du processus créé, utilisez l'option par défaut pour le déploiement, et dans le menu Liens partenaires sélectionner le type Endpoint : statique pour les liens partenaires, assurez-vous que l'adresse, le nom de service et les opérations sont correctement définies.

Après cela, générer une archive de processus d'affaires (.bpr) pour déployer dans le ActiveBPEL Engine. Dans le menu suivant, spécifier des informations sur la destination où le fichier .bpr doit être enregistré dans votre espace de travail (sélectionnez la destination d'exportation, le fichier bpr), et le répertoire de déploiement dans le serveur, cliquez sur Suivant et terminer l'assistant. En ce moment, le processus BPEL est déployé sur le serveur Tomcat, et prêt à être exécuté.

Pour exécuter le processus BPEL, Tomcat doit être en cours d'exécution. Une fois Tomcat est en cours d'exécution, ouvrez une fenêtre de navigateur et vérifier la console Engine ActiveBPEL, en ouvrant le lien http : // localhost : 8080 / BpelAdmin /.

Une interface avec le ActiveBPEL Engine devrait apparaître montrant des détails de configuration, l'état du déploiement, et l'état du processus. Cela permet de vérifier

si un processus BPEL est correctement déployé, et aussi pour surveiller les processus BPEL qui exécutent. Dans cette interface, nous pouvons vérifier le fichier processus de WSDL (agence de voyage .wsdl), et va être utilisé pour créer les services client web nécessaires au processus BPEL.

Une fois le processus BPEL est déployé, il peut être exécuté. Pour réaliser cette opération d'appel, nous supposons que le service web de propositions a accès au fichier WSDL du processus BPEL déployé pour créer un client pour accéder à cette opération de processus, ce fichier WSDL (figure 3.9). Nous utilisons l'assistant de service client web Eclipse pour générer le service web client pour le processus BPEL.

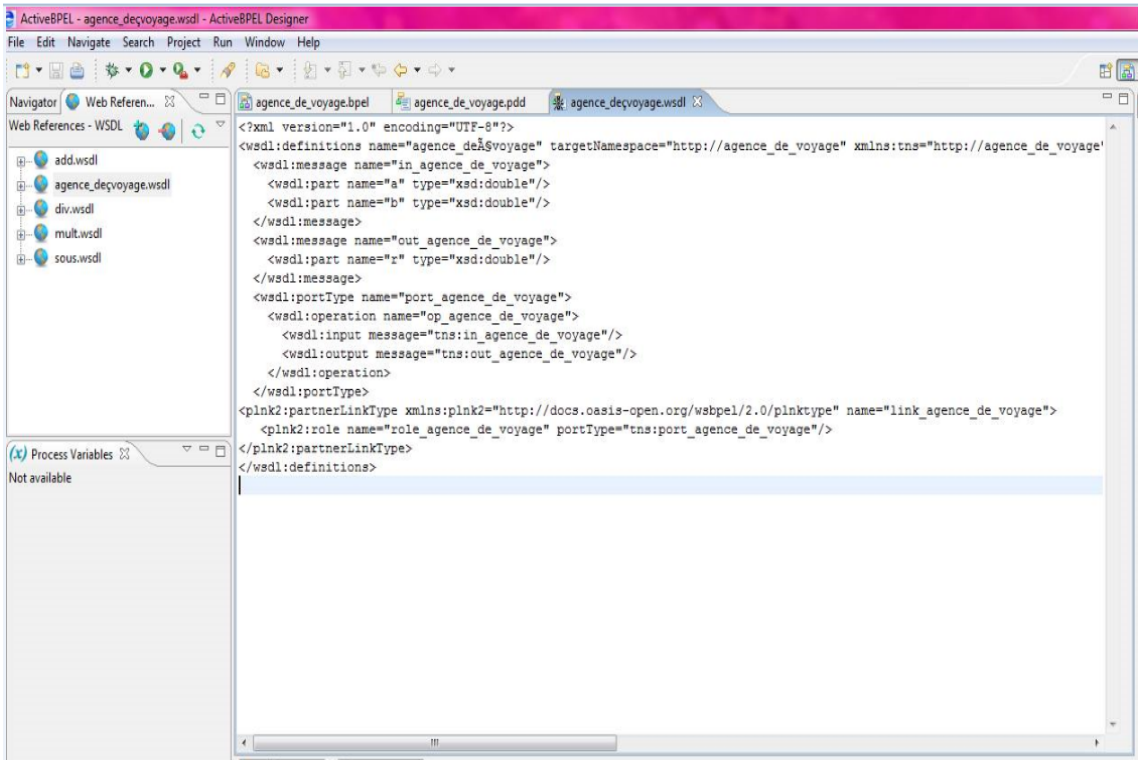

FIG. 3.9: Fichier WSDL de processus BPEL.

Une fois que nous avons un client pour le service de l'agence de voyage, nous sommes en mesure d'introduire toutes les données requises pour instancier le processus BPEL et de commencer à toutes les activités du processus BPEL. Avant cela, cependant, nous devons nous assurer que les services web de tous les PartnerLinks sont déployés quelque part, par exemple, dans un serveur d'application Tomcat. À ce stade, nous pouvons tester notre processus BPEL en démarrant Tomcat, le ActiveBPEL Engine.

3.5 Implémentation :

3.5.1 La transformation d'un processus BPEL en graphe :

Avant de faire la composition on passe par la transformation du processus de BPEL à un graphe orienté. Et pour faire on doit suivre les règles suivantes :

- les PartnerLinks (les services)sont transformmés aux sommets. D'où S : le nom du sommet et (x,y) : les coordonnés associés.

- les Links sont transformés aux arcs.

La première étape consiste à charger le fichier BPEL stocké sur la machine pour dessiner le graphe manuellement.

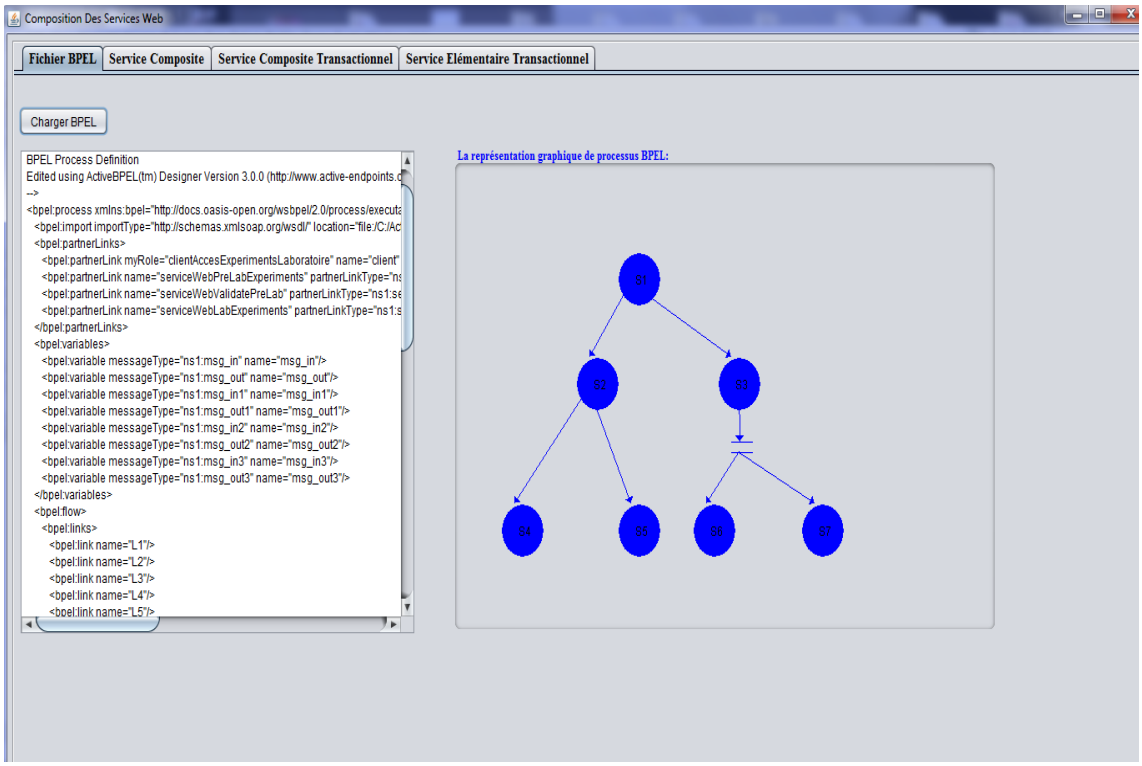

FIG. 3.10: Transformation d'un processus BPEL en graphe.

3.5.2 La composition des services web :

L'une des fonctionnalités principales de l'application est celle de la composition. Cette dernière prend en entrée un service ou un ensemble de services sur lesquels elle applique une opération ou une suite d'opérations pour retourner un service composé comme résultat.

Pour notre application nous réalisons une composition manuelle ; la composition manuelle des services web suppose que l'utilisateur génère la composition à la main via un éditeur de texte et sans l'aide d'outils dédiés. Pour accéder à cette opération, l'utilisateur doit choisir la fonctionnalité " Service Composite". Cette dernière permet de charger ou afficher le graphe de composition à réaliser et de tester la disponibilité de chacun de service participant à la composition en cliquant sur le bouton "Trouver les services" comme montré dans la figure suivante :

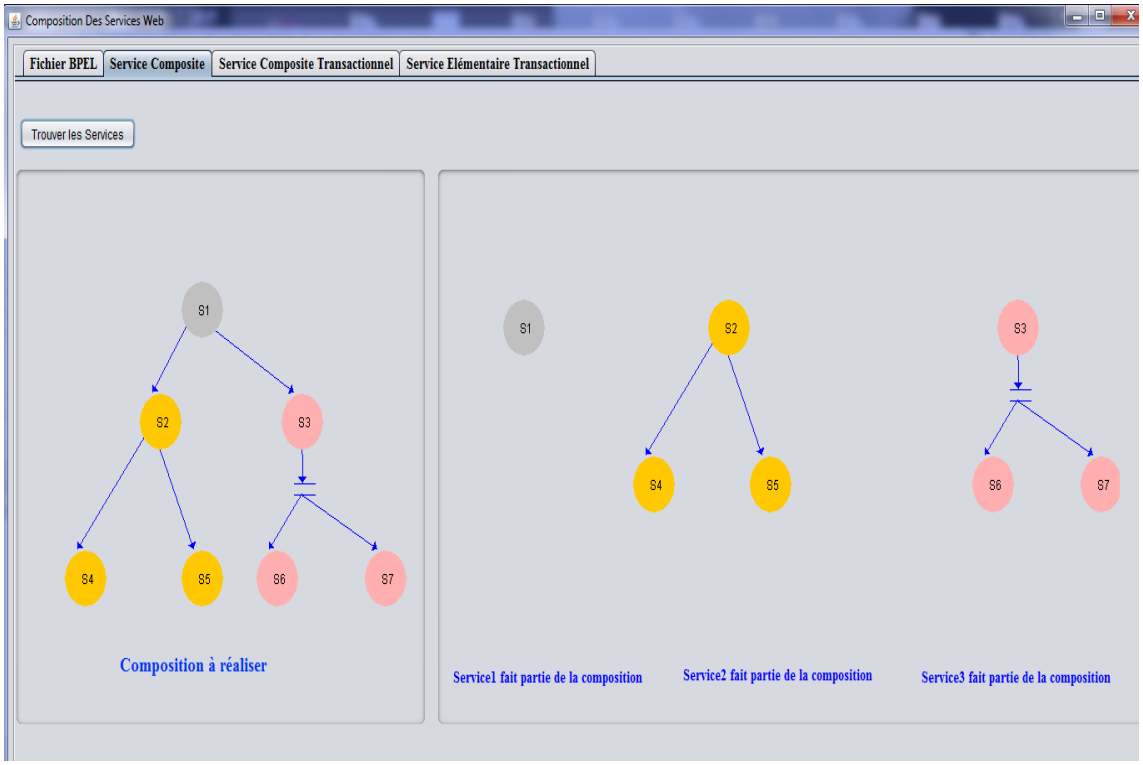

FIG. 3.11: Composition des services web.

3.5.3 La composition des services web avec les propriétés transactionnelle :

C'est l'interface la plus importante dans notre application elle se décompose de deux parties, la premiere contient la barre des boutons standard," Service Composite" pour afficher le service composite à réaliser,et "Vérifier les propriétés transactionnelles" pour vérifier la compatibilité des propriétés transactionnelles de chaque service elémentaire avec celle dans la composition.

La deuxième est un panel où on peut dessiner ou afficher le graphe et les résultats de la composition avec propriétés transactionnelles. Chaque service possède une propriété transactionnelle, on commence par le premier service participant à la composition jusqu'on arrive à la fin de la composition. Si les propriétés transactionnelles des services élémentaires sont compatibles avec celles dans le graphe de service composite ,celui-ci affiche le message de la figure (la composition est réalisée avec les propriétés transactionnelles).

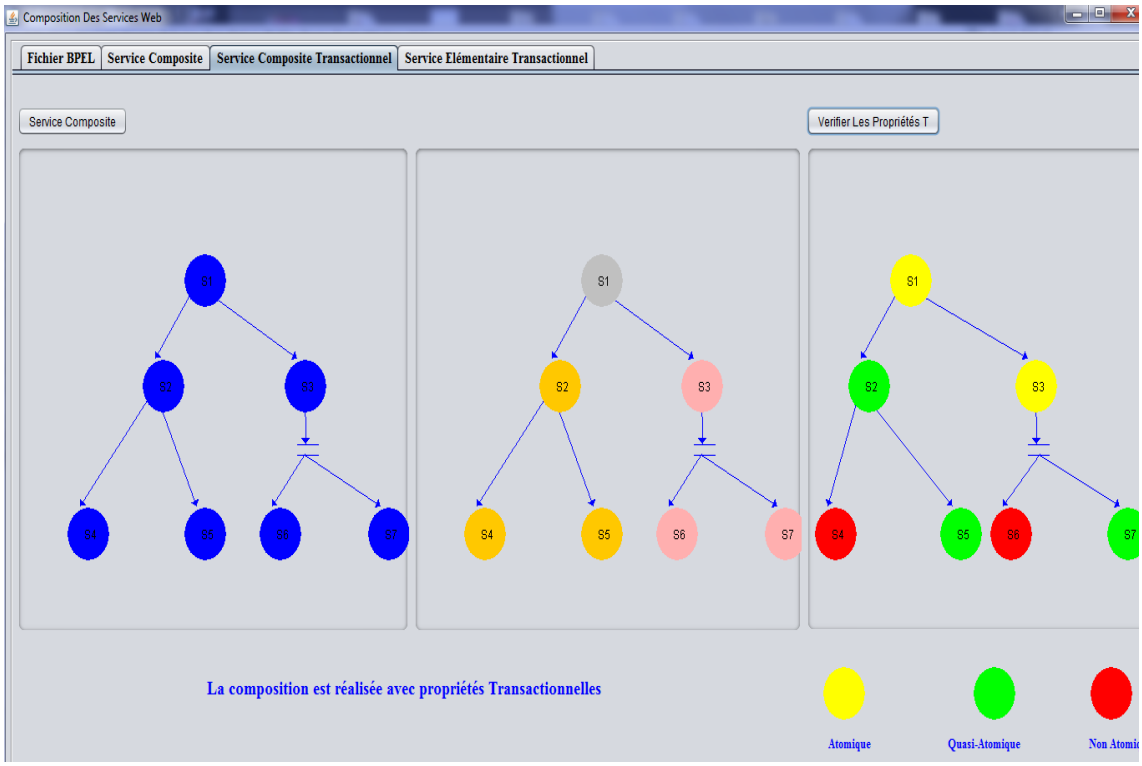

FIG. 3.12: La composition des services web avec les propriétés transactionnelles.

3.6 Expérimentations

Dans cette section nous décrivons les expériences menées pour analyser les performances de notre étude.

L'expérience qui nous avons effectuée consiste à mesurer le temps en rapport avec le nombre de composition (figure 3.13). Le service web que nous utilisons est un service web composite (une composition traditionnelle , et autre transactionnelle).

La figure montre les résultats de cette expérimentation. On remarque que le temps d'exécution est trés élevé dans le cas de la composition traditionnelle, puisque cette composition ne prend pas en compte l'aspect transactionnel c'est-à-dire que la composition est assurée mais n'est pas réaliste. Contrairement à l'autre cas(la composition avec propriétés transactionnelles) qui supporte les mécanismes de compensation, validation et annulation alors , cela peut être expliqué par la felxibilté et augmentation de la chance de composition avec succès.

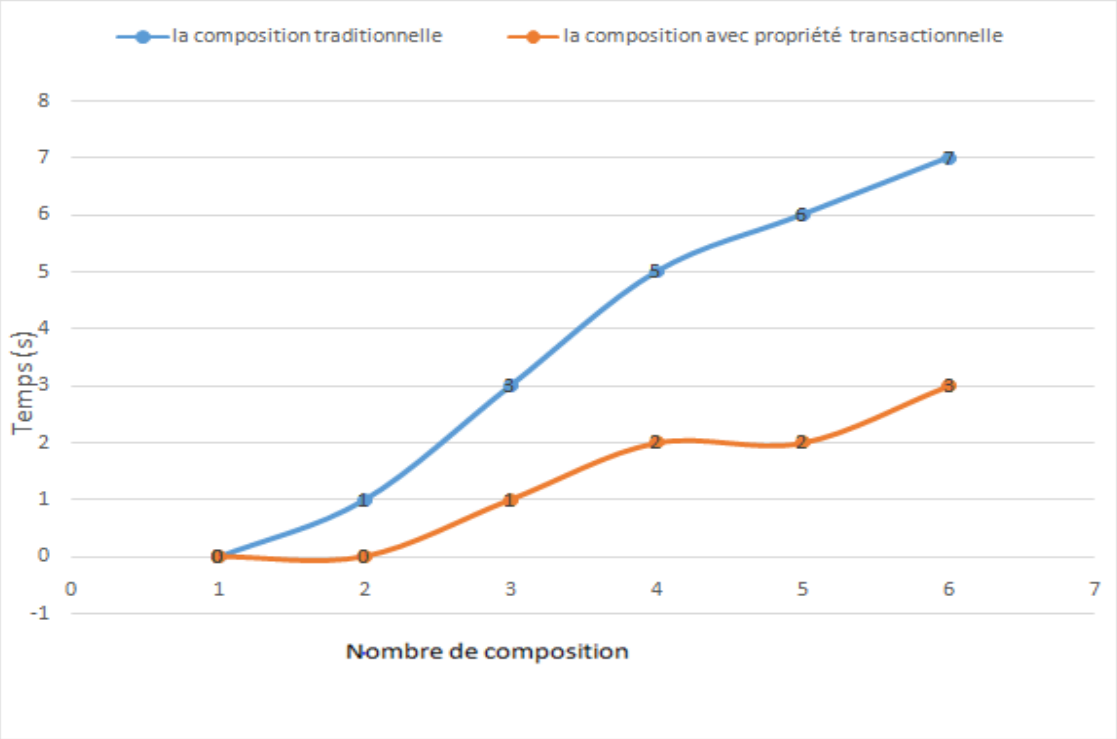

FIG. 3.13: Une comparaison entre la composition traditionnelle et transactionnelle.

3.7 Conclusion :

Dans la première partie de ce chapitre, nous avons présenté quelques détails relatifs à la réalisation de notre outil. Nous avons choisi le langage de programmation Java pour les raisons de compatibilités avec notre environnement de test, et aussi pour ses multiples caractéristiques de simplicité, robustesse et portabilité.

Nous avons proposé également une étude de cas où nous avons utilisé notre outil pour permettre de bien illustrer l'approche de composition et de vérification des propriétés transactionnelles des services web élémentaires et composites présentées dans le chapitre précédent.

Dans la deuxième partie nous avons présenté les principes de conception et d'implémentation des différentes parties de notre application. , nous présentons une expérimentation qui assure des exécutions correctes de services composés en intégrant de façon complémentaire l'aspect transactionnel, qui permet de définir la fiabilité des exécutions.

Conclusion Générale

Aujourd'hui, l'interopérabilité est devenue un domaine de recherche fondamental des systèmes d'information distribués et hétérogènes. Les services web sont considérés comme une solution potentielle aux problèmes d'interopérabilités. Ils définissent un nouveau paradigme de développement des interactions entre des applications distribuées de manière à ce qu'elles restent indépendantes des environnements et des plateformes d'exécution d'une part et aussi des choix des langages de développement et technologies d'implémentations utilisés d'une autre part. La composition de services web en particulier permet de combiner plusieurs fonctionnalités des services Web afin de répondre aux exigences qu'un seul service ne peut satisfaire.

Dans ce mémoire nous nous sommes intéressés à la composition de services web avec propriétés transactionnelles et plus spécifiquement à la propriété d'atomicité. Pour cela, nous avons réalisé une étude concernant les aspects transactionnels dans la composition des services web. Cette étude nous a permis de concevoir un service web composé comme une application distribuée qui doit offrir à ses utilisateurs la possibilité d'exprimer leurs besoins transactionnels (à savoir la vitalité et la possibilité de remplacement des tâches) en prenant en compte que ces besoins peuvent changer dynamiquement lors de l'exécution de la composition. Un service web composé doit aussi supporter les propriétés transactionnelles des services web participants à sa composition. Ces derniers étant accessibles dans un domaine dynamique et évolutif, dont les interactions ont des durées de vie variables. Ces caractéristiques montrent le besoin de flexibilité et d'adaptabilité.

Finalement, on a réussi à implémenter cette application d'une manière simple et compréhensible, même si elle est loin d'être utilisable sur le plan pratique mais elle reste une bonne expérience, cette dernière est un bon complément de notre formation de base, elle nous a permis d'enrichir nos connaissances théoriques et pratiques, et constitue la base de départ pour des futurs travaux. Nos perspectives étant de continuer dans le domaine des web services, de bâtir une base solide pour pouvoir développer des applications plus consistantes, et plus complètes.

Bibliographie

[1] Amara Mohamed *«Interopérabilité des services Web hétérogènes, Mémoire de projet de fin d'étude»*, Université de Tlemcen 2009 .

[2] A.Ait-Bachir,Archi Med *«un canevas pour la détection et la résolution des incompatibilités des conversations entre services Web»*, Thèse de doctorat, Université Joseph Fourier-Grenoble1, 2008.

[3] B.A.Schmit,S.Dustdar *«Towards Transactional Web services»* Proceedings of the 2005 Seventh IEEE International Conference on commerce Technology Workshops (CECW'05).

[4] C.Ba *«TComposition des services Web avec PEWS : approche par la théorie des traces »*Thèse de doctorat en informatique, Université Francois Rabelais Tours, 24 Novembre 2008.

[5] Carl Jones *«Do More Wirh SOA Integration : Best of Packt Integrate, automate and regulate your business processes with the best of packt's SOA books»*, packt publishing,December 2011, BIRMINGAM.

[6] C.Marin *«une approche orientée domaine pour la composition de services.»*, Thèse de doctorat Université Joseph Fourier 2008.

[7] C. Lopez-velasco *«Sélection et composition de services Web pour la génération d'applications adaptées au contexte d'utilisation»*,thèse de doctorat, université JOSEPH FOUR IER, 2008.

[8] Clement L,Hately A,Von Riegen C,Rogers T *«UDDI.3.0.2 , OSASIS Specification.»*, http ://uddi.org/pubs/uddi-v3, 2004.

[9] Endrie Mark,Jenny Ang,Ali Arsajani,Sook Philippe comotr,Pal Krogdahl,Minluo,Tony Newling-2004 *«Pattern :Service oriented architecture and web services»*,IBM INTERNTIONAL TECHNICAL SUPPORT ORGANIZATION ISBN073845371X.

[10] Frédéric Peschanski , Jean-Pierre Briot *«Architectures de composants répartis.»* In Mourad Oussalah, editor, Composants

[11] F.Porraz, Diapason *«approche formelle et centrée architecture pour la composition évolutive de service web.»*,Thèse de doctorat, LISTIC, Université de Savoie , France 2007.

[12] F.Curbera, A.Nagy, , S.Weerawarana *«Web Service : Why and How.»*,Dans workshop on Object-Oriented Web Services(in OOPSLA), Aout 2001. http ://www.research.ibm.com/people/b/bth/OOWS2001.html.

[13] F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte and S.Weerawarana, *«Business Process Execution Language for Web Service (BPEL4WS) 1.0.»*,Published on the World Wide Web by BEA Corp, IBM Corp and Microsoft Corp, Aug 2002.

[14] H.Duarte , M-C.Fauvet, M.Dumas, B.Benatallah *«vers un modèle de composition de services Web avec proprieties transactionnelles.»*,Revue Ingénierie des systemes d'informations Numéro spécial Service Web, Vol 10,no3/2005, pp.9-28,2005.

[15] H.DUARTE-AMAYA *«TCOWS : canevas pour la composition de services Web avec propriétés transactionnelles .»*, Thèse de doctorat en informatique, Université JOSEPH FOURIER France, 13 Novembre 2007.

[16] J.N.Gray *«Notes on database operating systems -Operating Systems An advanced Course .»*, lecture Notes in Computer Science 60 ; édité par R.Bayer, R.M. Graham et G.Seegmueller,pp.393-481,Springer-verlag, Berlin, 1978.

[17] J.EL Haddad, O.Spanjaord *«Composition de services Web et équité vis-à-vis des utilisateurs finaux.»*, 10éme Congrès de la société Française de recherche opérationnelle et d'aide à la décision, ROADER 2009 Nancy 10-12-Février 2009.

[18] J.Daniel*«Services Web - Concepts, techniques et outils .»*,Editions Vuibert Informatique, Paris 2003. ISBN 2-7117-4813-8.

[19] K.Hubert, M.Valérie *«LES WEB SERVICES .»*, Edition DUNOD, 2003. Concepts,techniques et outils.»,Vuiber, 2005. Chapitre9.(in french).

[20] K.D.Gottschalk, S.Graham , H.Kreger , J.Snell *«Introduction To Web Service Architecture .»*, IBM Systems Journal L1(2) :170-177,2002.

[21] K.Ramamrithan , P.K.Chrysanthis *«A taxonomy of correction criteria in database applications .»*, The VLDB Journal, 5(1) : 085-097,1996.

[22] Leymann F *«web service Flow language .»*, 1.0.TBM Report [en ligne],2001. http ://www-306.ibm.com/software/solutions/webservices/pdf/wsfl.pdf.

[23] M.Gharzouli *«Composition des web services sémantiques dans les systèmes peer-to-peer (in french) .»*, PhD thesis, Departement of computer science, University of constantine2,septembre 2011.

[24] Mitra N, Lafon Y *«SOAP.»*, Version 1.2 Part 0 : primer (Second Edition), W3C , http ://www.w3.org/TR/soap12-Part0/, 2003.

[25] Nicolai M.Josuttis *«SOA in Pratice .»*, O'Reilly Medias August 2007.

[26] N. Arenaza *«Composition semi-automatique des web services (in french) .»*, Master project, Federal Polytechnic School of Lausane, Switzerland, Fev 2006.

[27] N.Nouali-Taboudjemat *«Modèles et techniques adaptable pour les environnements mobiles.»*, Thèse de doctorat en informatique ;USTHB, Alger N09/2007-E/IN, 17 Novembre 2007.

[28] OASIS Standard *«Web Services Coordination (WS-Coordination) Version 1.2.»*,02 fevier 2009. Disponible sur (Derniere visite : Mars 2011) : http ://docs.oasis-open.org/ws-tx/wstx-wscoor-1.2-spec-os.pdf .

[29] Projet RNTL FAROS *«Etat de l'art sur la constraculalisation et la composition.»*, Aout 2006, Version : 1.0a, http ://www.lifl-fr/faros. (Derniere vvisite juillet 2009.

[30] P.F.Pires , M.R.Benevides, M.Mattoso *«Building Reliable Web Services Composition.»*, In Web, Web-Services and Database Systems, LNCS 2593,59-72, Springer,2003.

[31] RW.Schult,YV.Natis *«Service oriented architectures .»*, part1 et 2 http ://www.granter.com ;1996.

[32] S.Sanlaville *« Environnement de procédé extensible pour l'orchestration : Application aux services web (in French)»*, PhD thesis Departement of computer science ,Univesity of JOSEPH FOURIER, Grenoble, France ,December 2005.

[33] S.Bhiri *« Approche Transactionnelle pour assurer des compositions fiables de services web.»*, Thèse de doctorat en informatique, Université Henri Poincaré-Nancy 1, octobre 2005.

[34] S.Bhiri, O.Perrin, C.Godart *« Ensuring Required Failure Atomicity of Composite Web Services .»*, in Proceedings of WWW, 138-147, ACM Press, 2005.

[35] SOAP *«Simple Object Acess Protocol (SOAP)1.1 .»*,rapport may 2000,world wide web consortium,http ://www.w3.org/TR/soap.

[36] T.HANH *« Coordination adaptative de services à base de contrats.»*, Thèse de doctorat en informatique, Université JOSEPH FOURIER France, 2009. Institut Polytechnique de Grenoble.

[37] T.Jin,S.Goschnick *«Utilizing Web Services in an Agentbased Transaction Model (ABT). Proceedings, Workshop on Web Services and Agent-based Engineering.»*,at the AAMAS-2003 conference, Melbourne, Australia.

[38] T.Mellit *« Interopérabilité des services web complexes, application aux system multi-agents (in french).»*, PhD thesis Departement of computer science, University of paris IX Dauphine,2004.

[39] UDDI *«Universal Description Discovery and Integration .»*,rapport may 2000,OSASIS UDDI specification technical committee,http ://www.osasisopen.org/cover/uddi.html.

[40] WSDL *«Web Service Description Language (WSDL)1.1 .»*,rapport may 2000,world wide web consortium,http ://www.w3.org/TR/wsdl.

[41] Worx Author Team *« Service Web XML Professionnel.»*, Wrax, Paris,December 2001.

[42] Web Service Composition Standars *«Web Service Composition Standars .»*,http ://Lsdis.cs.uga.edu/proj/meteor/mwscf/standards.html.

[43] Web Service Cheography Interface WSCI *«Web Service Cheography Interface WSCI .»*,http ://www.w3.org/TR/wsci http ://www.isima.fr/ponge/dea/rapport-dea.pdf.

[44] Web Service Conversation Language WSCL *«Web Service Conversation Language WSCL.»*,http ://www.w3.org/TR/wsci http ://www.isima.fr/ponge/dea/rapport-dea.pdf.

[45] Web Services Composite Application Framework *«Web Services Composite Application Framework (WS-CAF) Ver1.0»*,July 28, 2003. Disponible sur (Dernière visite : Mars 2011) : http ://www.oasis-open.org/committees/download.php/4343/WS-CAF Primer.pdf.

[46] Web Services Coordination Framework Specification *«Web Services Coordination Framework (WS-CF) Ver1.0.»*,28 Juillet 2003.

[47] Mark Little, Eric Newcomer, Greg Pavlik *«Web Services Context Specification (WS-Context) Version 1.0.»*,20 Janvier 2006. Disponible sur(Dernière visite : Mars 2011) : http ://docs.oasis-open.org/ws-tx/wstxwsat-1.1-spec-os.pdf

[48] W3C *«Web Services Description Language (WSDL) 1.1 .»*,W3C Note 15 March 2001. Disponible sur (Dernière visite : Mars 2011) :http ://www.w3.org/TR/2001/NOTE-wsdl-2001-03-15 .

[49] WS-C1.2,OASIS Standard *«Web Services Coordination (WS-Coordination) Version 1.2.»*,02 fevier 2009. Disponible sur (Derniere visite : Mars 2011) : http ://docs.oasis-open.org/ws-tx/wstx-wscoor-1.2-spec-os.pdf.

[50] Z.I.Kazi Aoul *«une architecture orientée service pour la fourniture de documents multimédia composés adaptables.»*,Thèse de doctorat en informatique,Ecole Nationale Superieure des Télécommunications PARIS,18 Janvier 2008 .

Résumé

Les dernières décennies ont été marquées par le développement rapide des systèmes d'information distribués, et tout particulièrement par la vulgarisation de l'accès à Internet. Cette évolution a entraîné le développement de nouveaux paradigmes d'interaction entre applications tel que les « services web ». Un service web est un programme modulaire, indépendant et auto-descriptif, qui peut être publié, découvert et invoqué via Internet ou intranet. La composition des services web permet de créer des nouveaux services web (dits services web composés) par regroupement des services web existants. Ainsi, un service web composé peut être vu comme une application distribuée qui possède des caractéristiques spécifiques. Ces caractéristiques influencent les aspects transactionnels dans ce domaine et engendrent le besoin de solutions flexibles et adaptables pour la composition des services web avec propriétés transactionnelles. 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 transactionnelles à cette composition. Notre travail permet de composer des services et de prendre en compte les propriétés transactionnelles.

Mots-clés:

Service web, Composition des service web, transaction ,Propriétés transactionnelles de services web .

Abstract

The last decades have been marked by the rapid development of distributed information systems, especially through extension of the Internet. This has led to the development of new paradigms for interaction between applications such as "web services". A web service is a modular program, independent and self-descriptive, which can be published, discovered and invoked over the Internet or intranet. The composition of web services can create new web services (compounds called web services) by consolidating existing web services. Thus, a compound web service can be seen as a distributed application that has specific characteristics. These characteristics influence the transactional aspects in this area and create the need for flexible and adaptable solutions for the composition of web services with transactional properties.

The study presented in this document allowed us to identify the one hand problems, the composition of web services, and on the other hand the association of the transactional properties of this composition. Our work for dialing services and take into account transactional properties.

Keywords:

web service, Composition of web service, transaction,transactional properties of web services.