République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Mémoire de Magister

Présenté à l'Université de Guelma
Faculté des sciences et sciences de l'ingénierie

Département de : **Informatique**
Ecole **D**octorale **N**ationale **S**cience et **T**echnologie de l'**I**nformation et de la **C**ommunication
Spécialité : **Informatique**
Option : **S.I.C**

Présenté par : **Mr**: **KHEBIZI Ali**

Prise en Compte des Contraintes lors de la Découverte et de l'Orchestration des Services Web

JURY

Président :	Pr. Hamid Séridi	Université de Guelma
Rapporteur :	M.C. Hassina Séridi	Université de Annaba
Examinateurs :	M.C. A. Chaoui	Université de Constantine
	M.C. M.K Kholladi	Université de Constantine

2009

Remerciements

Je remercie chaleureusement mon encadreur Mme Séridi Hassina maître de conférences à l'université de Annaba pour ses orientations, ses conseils valeureux et sa disponibilité. Son soutien moral durant les moments difficiles m'a considérablement aidé pour l'élaboration de ce travail dans les délais.

Je remercie Mr Chaoui A. et Mr Kholladi M.K, Maîtres de conférence à l'université de Constantine d'avoir accepté d'examiner mon travail.

Je tiens à remercier, particulièrement, Pr. Séridi Hamid pour les efforts consentis pour l'ouverture et l'encadrement de l'école doctorale d'informatique et pour l'intérêt qu'il attache au développement de la formation post-graduée à l'université de Guelma.

Je suis très reconnaissant à Mr: Toumani Farouk Professeur à l'université de Clermont Ferrand -France- de m'avoir accueilli en stage dans le laboratoire LIMOS, où j'ai découvert le monde de la recherche scientifique, et de m'avoir éclairé et soutenu tout au long de l'élaboration de mon travail.

Je remercie, également, Professeur Houari Nourine de l'Université de Clermont Ferrand pour les riches discussions inachevées.

Je tiens également à remercier mon collègue Nouara Redouane pour sa collaboration et d'avoir accepté de s'aventurer avec moi dans le monde des services Web.

Mes remerciements les plus sincères à tous ceux qui m'ont soutenu et aidé durant ma formation post-graduée. Particulièrement, ma femme pour sa patience et sa compréhension et mes frères et sœurs pour leurs encouragements.

Je ne dois pas oublier de remercier tous mes amis et collègues pour leur soutien et compréhension.

Mes vifs remerciements à : mon cousin Mokhtar, Françoise et Niness.

Dédicace

A la mémoire de: ma mère, mon frère
Ramdane et Toufik

A mon père pour ses sacrifices

A Halima et nos deux perles:
Raji et Rami

A toute ma grande famille

« I have reasoned with my perception and we have no conception of the absolutely uncognizable »

C.S Peirce

Sommaire

Sommaire

Résumé

En offrant un nouveau paradigme pour les systèmes d'information distribué et hétérogènes, les services Web présentent de fortes potentialités pour l'intégration des applications intra et interentreprises. Mais, bien que beaucoup d'avancés ont été réalisées pour surmonter les problèmes d'interopérabilité, les services Web n'ont pas encore atteint la maturité technologique leur permettant de traiter de vrai processus métiers trans-organisationnels. Ce constat est dû, essentiellement, à des insuffisances dans la prise en compte et dans la modélisation des contraintes fonctionnelles et non fonctionnelles liées aux services Web.

Pour un déploiement effectif et une large adoption de cette technologie, les protocoles de services, bien qu'ils décrivent certaines caractéristiques fonctionnelles liées au comportement externe des services, nécessitent d'autres enrichissements.

Dans ce mémoire, nous traitons de la prise en compte des contraintes lors de la découverte et l'orchestration des services Web et nous nous intéresserons, tout particulièrement, à la modélisation des contraintes transactionnelles.

Dans un premier temps, nous proposerons une modélisation des effets transactionnels, puis nous présenterons un enrichissement de la description du comportement externe des services Web par la prise en compte des contraintes transactionnelles en exposant le nouveau modèle de protocoles de services à effets transactionnels.
Dans un deuxième temps, nous nous attacherons à étudier les conséquences conceptuelles, d'un tel enrichissement, sur l'analyse de compatibilité et d'équivalence des protocoles de service. La première analyse est relative à la découverte des services et la deuxième concerne le processus d'orchestration. En identifiera les classes d'équivalences des protocoles à effets transactionnels et en présentera les algorithmes adéquats. Divers opérateurs de manipulation des effets transactionnels et des protocoles de services seront présentés et formalisés, en vue de consolider l'assise théorique existante pour la gestion et la manipulation des protocoles de services à effets transactionnels.

Mots Clés : *Services Web, Protocoles de services, Contraintes transactionnelles, Compensation, Compatibilité de services, Equivalence de services.*

Abstract

By offering a new paradigm for distributed and heterogeneous information systems, Web services presents strong potential for intra and inter-entreprises applications integration. Although many advanced have been made to overcome problems interoperability, Web services have still immature technology enabling them to deal with real inter-organizational process. This is due mainly to shortfalls in addressing and modeling of functional and non-functional constraints related to Web services

For an effective deployment and wide adoption of this technology, service protocols although they describe some functional characteristics related to the service external behavior, require other enhancements.

In this work, we are dealing with the consideration of constraints during the discovery and orchestration of Web services. We will be interested, especially, to modeling transactional constraints.

As a first step, we propose a model for representing transactional effects, then we expose the external behavior description enrichement of Web services by taking into account the transactional constraints. After we introduce the new protocol services model with transactional effects.

In a second time, we will focus to consider the conceptual consequences of such enrichment on compatibilty and equivalence analysis of service protocols. The first analysis deal with service discovery and the second deal with their orchestration. We identify, therefore, equivalence class's protocols with transactional effects and we present the adequate algorithms.
Various operators handling protocols services will be presented and formalized in order to consolidate the existing theoretical management and handling service protocols with transactional effects.

Keywords : *Web Services, Web services Protocols, Transactional Constraints, Compensation, Compatibility analysis, Service Equivalency*

ملخص

من خلال تقديم نموذج جديد لنظم المعلومات الموزعة والغير متجانسة، تقدم **خدمات الواب** إمكانات كبيرة لتكامل التطبيقات داخل وبين المؤسسات.

لكنه على الرغم من أن العديد من الاحرازات حققت للتغلب على مشاكل التشغيل المتبادل، فإن **خدمات الواب** لم ترق بعد إلى مرحلة النضج التكنولوجي التي تسمح لها بإجراء تعاملات حقيقية بين التنظيمات. ويرجع ذلك أساسا إلى النقص في معالجة وفي تصميم القيود الوظيفية وغير الوظيفية ذات الصلة لخدمات الويب .

وعليه، و من أجل انتشار واسع واعتماد فعلي لتكنولوجيا **خدمات الواب**، فعلى الرغم من أن بروتوكولات الخدمات تصف بعض الخصائص الوظيفية المتعلقة بالسلوك الخارجي للخدمات ، فإنها تحتاج إلى المزيد من التحسينات.

في هذه المذكرة ،نهتم ونأخذ بعين الاعتبار القيود بصفة عامة وهذا خلال مراحل اكتشاف وتنسيق خدمات الواب. وسنركز دراستنا بصفة خاصة على **قيود المعاملات**.

في المرحلة الأولى ، فإننا سوف نقترح نموذج تصميمي لتمثيل آثار المعاملات ثم نقدم تحسين لوصف السلوك الخارجي **لخدمات الواب** مع مراعاة التفاعلات بين الخدمات ، و بعد ذلك نستعرض نموذج جديد لبروتوكولات الخدمات المحسن بآثار المعاملات.

في مرحلة ثانية ، نسعى جاهدين لاستكشاف الآثار التصميمية المترتبة على هذا التخصيب لبروتوكولات الخدمات فيما يخص تحليل التوافق والتكافؤ بين بروتوكولات الخدمات ذات آثار المعاملات. و يتعلق التحليل الأول باكتشاف الخدمات أما التحليل الثاني فهو مهم بالنسبة لصيرورة التنسيق. كما أننا سنبحث تحديد أصناف التكافؤ للبروتوكولات ذات آثار المعاملاتية ونستعرض بالتدقيق الخوارزميات المتعلقة بأصناف التكافؤ.

في الأخير نقترح و نشكل عدد من المعاملات لتسيير آثار المعاملات، وبالطبع التعامل مع بروتوكولات الخدمات وهذا من أجل توطيد الأسس النظرية الحالية لإدارة ومعالجة بروتوكولات الخدمات المحسنة بآثار المعاملات.

الكلمات الرئيسية : خدمات الواب ، بروتوكولات خدمات الواب، قيود المعاملات ، التعويض ، تحليل التوافق، تكافؤ الخدمات.

Table des Figures

Liste des Tableaux

Abréviations et Acronymes

2PC	Two Phases Commit
ACID	Atomique, Cohérente, Isolée et Durable
API	Application Programming Interface
B2B	Business To Business
BPEL4WS	Business Process Execution Language For Web Services
BPML	Business Process Modeling Language
BTP	Business Transaction Protocol
CGI	Common Gateway Interface
CORBA	Common Object Request Broker Architecture
DTD	Document Type Definition
EAI	Entreprises Applications Integration
ebXML	Electronic Business using XML
EDI	Electronic Data Interchange
EDIFACT	Electronic Data Interchange For Administration, Commerce and Transport
HTML	HyperText Marckup Language
HTTP	HyperText Transfer Protocol
HTTPS	Secure HyperText Transfer Protocol
IDL	Interface Description Language
JMS	Java Message Service
LAN	Local Area Network
MAC	Message Authentification Code
MOM	Message Oriented Middleware
OCL	Object Constraints Language
OWL-S	Ontology Web Language for Services
QoS	Quality of Service
RPC	Remote Procedure Call
S.I	Système d'Informations
SGML	Standard Generalized Mackup Language
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol

SSL	Secure Socket Layer
TCP	Transmission Control Protocol
THP	Tentative Hold Protocol
TLS	Transport Layer Secure
UDDI	Universal Description Discovery and Integration
URI	Uniform Ressource Identifier
W3C	Word Wide Web Consortium
WSCI	Web Service Choregraphy Interface
WSCL	Web Service Conversation Language
WSDL	Web Service Desciption Language
WSFL	Web Service Flow Language
WSLA	Web Service Level Agreement
WSMF	Web service Modeling Framwork
WSMO	Web Service Modeling Ontology
WSTx	Web services Transaction
xCBL	XML Common Business Library
XKMS	XML Key Management Specification
XML	Extensible Marckup Language
XSL	eXtensible Stylesheet Language
XSLT	eXtensible Stylesheet Language Transformations

Introduction Générale

Introduction Générale

L'avènement des réseaux informatiques et particulièrement de l'Internet a suscité beaucoup d'engouements chez les acteurs économiques pour l'échange de leurs données. En effet, les modes de communication ont évolué du simple transfert statique de fichiers, en passant par l'invocation de procédure à distance, puis vers l'interaction entre les composants logiciels formant, ainsi, des applications réparties et ouvrant de nouvelles perspectives pour les systèmes d'information distribués.

Les services Web, réalisation concrète des architectures SOA (Service Oriented Architecture), sont l'aboutissement historique de ce processus évolutionnaire. Ils offrent de nouvelles potentialités pour l'intégration des applications d'entreprises et une infrastructure favorable aux interactions dans un environnement hétérogène et versatile qui est le Web.

L'architecture et la technologie des services Web définissent un ensemble de spécifications pour la description, la publication et la découverte des services permettant de favoriser les interactions dans un environnement ouvert, distribué et versatile. Bien que beaucoup d'avancés aient été réalisées en ce sens, des problèmes résiduels sont encore posés et entravent le déploiement effectif et à grande échelle de ce nouveau paradigme.

Le but visé par ce mémoire est de se focaliser sur la description du comportement externe des services Web afin de déceler les éventuelles insuffisances liées à la prise en compte des contraintes fonctionnelles lors de la découverte et de la composition des services. Un intérêt particulier sera attaché aux contraintes transactionnelles.

Effectivement, la description actuelle des services Web, par leur interface de services, reste limitée et n'exprime pas toute la sémantique des interactions impliquées lors des invocations et de la composition des services. Les protocoles de services tentent de répondre à cette insuffisance. En effet, diverses caractéristiques décrivant le comportement externe des services Web; telles que les contraintes d'ordre et les contraintes temporelles ont été prises en compte dans la modélisation des protocoles de services [15], [16]. Par ailleurs, l'infrastructure actuelle des services Web (SOAP, WSDL et UDDI) ait été enrichie par des spécifications; telles que les Framework: WS-Coordination [5] et WS-Transaction [6], nécessaires à la gestion des transactions au niveau du Middleware.

En parallèle à ces avancées, la prise en compte et la gestion des contraintes transactionnelles, au niveau des protocoles de services, bien qu'elles revêtent un aspect critique, n'ont pas bénéficié de tout l'intérêt qu'elles méritent, ni de la part des industriels ni de la part de la communauté des chercheurs du domaine.

Pour un déploiement effectif et pour une large adoption de la technologie des services Web, les protocoles de services, décrivant certaines caractéristiques liées au comportement externe des services, nécessitent d'autres enrichissements.

Dans ce mémoire nous proposons une modélisation des effets transactionnels, observés lors des interactions entres services, et nous rehaussons la description du comportement externe des services Web par la prise en compte des contraintes transactionnelles, indissociables des interactions entre services.

Dans le nouveau contexte des protocoles de services à effets transactionnels, l'analyse de compatibilité et d'équivalence des protocoles sera revue et réexaminée à la lumière des enrichissements proposés. Nous proposerons aussi les algorithmes de manipulation des protocoles à effets transactionnels ainsi que les opérateurs adéquats pour leur gestion.

Notre contribution est par conséquent à quatre niveaux:

1. **Analyse des contraintes fonctionnelles et des protocoles de services:** L'étude comparative sur la gestion des transactions dans les protocoles de services quant à la modélisation des effets des transactions, est considérée comme notre première contribution dans le sujet. Elle est basée sur des critères jugés utiles à notre travail, particulièrement, la prise en compte des contraintes transactionnelles au niveau protocole de service dans l'objectif d'une découverte plus pertinente des services et en vue d'une orchestration plus efficace lors de leur composition.
2. **Une modélisation des effets transactionnels:** Vu le déficit caractérisé dans la modélisation des contraintes transactionnelles, constaté lors de l'étude comparative des protocoles de services et des modèles de représentation des effets, nous avons proposé un modèle basé sur les requêtes de mise à jour de la base données pour leur représentation. Cette deuxième contribution répond d'une manière conséquente aux objectifs du raisonnement que nous voulons opérer sur les protocoles de service, à savoir l'analyse de compatibilité lors de la découverte et l'analyse d'équivalence lors de la composition des services.
3. **L'analyse de compatibilité et d'équivalence des protocoles de services:** Après rehaussement de la description du comportement externe des services Web, par l'intégration des contraintes transactionnelles, l'analyse de compatibilité et d'équivalence des services est revue et réétudiée à la lumière des enrichissements apportés. A ce niveau, des résultats très intéressants concernant la spécification de nouvelles classes d'équivalence ont été obtenus.
4. **Proposition d'algorithmes de manipulation des protocoles à effets transactionnels et des opérateurs de gestion des effets transactionnels:** En vue d'une spécification plus formelle des résultats obtenus, nous avons présenté les algorithmes associés aux différentes classes d'équivalence des protocoles de services et nous avons proposé un ensemble d'opérateurs de manipulation des effets transactionnels. Cette dernière contribution ouvre la voie à une réflexion plus large autour de la construction d'une algèbre de protocoles à effets transactionnels.

Organisation du document

Le document est structuré en deux parties:

Dans la première partie nous exposerons un état de l'art sur la prise en compte des contraintes dans les services Web. Elle est composée de trois (03) chapitres. Dans le premier chapitre nous introduisons le domaine des services Web en présentant leur technologie, leur architectures et les standards y afférents. En second chapitre nous parlerons de la coordination et de la composition dans les services Web, deux aspects inévitables dans les contextes

opérationnels des services Web. Au niveau de ce chapitre, nous aborderons en détails la gestion des transactions dans les processus de coordination et lors de la composition des services. Le troisième chapitre sera consacré à la modélisation du comportement externe des services Web par leur protocole de services et à la prise en compte des différents types de contraintes lors de la découverte et de la composition des services. L'intérêt de l'analyse et de la gestion des protocoles sera présenté à ce niveau.

Dans la seconde partie du document nous présenterons notre contribution. Elle est composée de deux chapitres. Le quatrième chapitre met en évidence le besoin d'un enrichissement de la modélisation du comportement externe des services Web par la prise en compte des contraintes transactionnelles. Nous y proposons un modèle de représentation des effets transactionnels que nous injecterons dans le modèle de base de protocoles de service. Le nouveau modèle de protocoles à effets transactionnels sera présenté à ce niveau. Nous terminerons au cinquième chapitre par l'analyse de compatibilité et d'équivalence des protocoles à effets transactionnels en présentant les algorithmes nécessaires la détermination des classes d'équivalence. Nous proposerons aussi un ensemble d'opérateurs de manipulation des effets transactionnels. Enfin, nous concluons ce travail et nous évoquerons nos perspectives de recherche.

Pour préserver l'essence du document et pour éviter son encombrement, nous avons jugé utile de présenter les spécifications WSDL, UDDI et SOAP, avec des exemples concrets, en annexes.

Partie I

Etat de l’art

Chapitre 1

Présentation des Services Web

Introduction

Les organisations (entreprises économiques, administrations…), quelques que soit leur dimension, sont contraintes d'échanger leur données avec les autres partenaires (réception d'un BDC, envoi de Facture…) et d'une manière générale avec leur environnement (Impôts, sécurité sociale…). L'avènement des réseaux informatiques et particulièrement de l'Internet a suscité beaucoup d'engouements chez les acteurs économiques pour l'échange de leurs données.

Dans un environnement concurrentiel caractérisé par les fusions, acquisitions et partenariat une communication efficace, rapide et fiable est capitale pour la survie des organisations. Elle constitue, en plus, un vecteur de création de valeurs ajoutées.

Ce processus de communication interentreprises a évolué du simple transfert statique de fichiers électroniques (**EDI**: Electronic Data Interchange, **EDIFACT**: Electronic Data Interchange For Administration, Commerce and Transport), en passant par l'invocation de procédure à distance (**RPC**: Remote Procedure Call, **TP Monitors**: Transaction Processing Monitors) puis vers l'interaction entre les composants logiciels pour former des applications réparties (**CORBA**: Common Object Request Broker Architecture, **DCOM**: Distibuted Component Object Model, **MOM**: Message Oriented Middlware).

Ce processus évolutionnaire, a été, historiquement, guidé par le paradigme du langage de programmation de chaque étape. La programmation procédurale a induit la communication de type RPC, l'invocation des objets distants relève de la programmation objet et l'échange de message est le fruit de la programmation par aspect.

En parallèle à cette évolution la réutilisation des composants logiciels, promesse utopique du génie-logiciel, et le besoin d'interopérabilité à travers des infrastructures hétérogènes (systèmes d'exploitation, plate formes et langages) deviennent de nouvelles contraintes pour l'intégration des applications d'entreprises dans un environnement ouvert et dynamique et dans un contexte de programmation répartie.

Les différentes tentatives vers l'intégration des applications d'entreprises ont constitué, en ce sens, des avancés considérables sans pour autant résoudre le problème d'une manière radicale. En effet, les middlewares conventionnels (RPC, CORBA, MOM) demeurent restreints aux partenaires définis préalablement, créant ainsi des relations de dépendances et tout changement de partenaire exige l'adoption de nouveaux middlewares. Par conséquent, leurs limites sont évidentes et restent loin des exigences d'un environnement, aussi ouvert et versatile, qui est le Web. Quand aux EAI (Entreprises Applications Integration), leur coût excessif est hors-de-portée des petites et moyennes entreprises; épine dorsale de l'économie contemporaine. En plus, pour les EAI des coûts de maintenance et de développement des nouveaux adaptateurs sont à prévoir pour toute nouvelle fonctionnalité.

Parallèlement à ces exigences technologiques, les phénomènes globaux ayant affecté la société humaine durant les deux dernières décennies caractérisés par: une modernisation accrue, des moyens de communication ultrasophistiqués, un mode de consommation des biens et services en croissante évolution et une forte pression de l'environnement ont poussé les entreprises à revoir la philosophie de conception de leur système d'information (**S.I**). En effet, ce dernier doit répondre aux nouvelles exigences stratégiques suivantes:

1. **Indépendance:** Le S.I doit offrir à l'entreprise un degré de liberté supérieure. Elle ne sera plus contrainte aux aspects technologiques de son S.I pour nouer ou résilier des relations économiques. Les choix économiques constitueront les seuls critères de partenariat.
2. **Virtualisation:** Les frontières des entreprises étant, de plus en plus mouvantes et floues, le S.I doit être *dématérialisé*. Il pourra être construit d'une manière ad-hoc à partir de briques logicielles patrimoniales et/ou disponibles sur le Web, sans avoir une existence physique propre.
3. **Performance:** Le S.I doit être performant pour répondre aux demandes des clients potentiels du Web (clients nomades) et faire face à une concurrence de plus en plus rude, où petites et grandes entreprises se disputent les parts du marché avec les mêmes chances.

L'accélération des phénomènes cités précédemment et l'explosion exponentielle du Web ont mis les entreprises face à une problématique réelle pour répondre aux nouvelles exigences. Elles doivent, une fois de plus, affronter le problème de l'***intégration des applications intra et inter entreprises***, mais avec des nouvelles données (Web versatile, clients nomades, précarité des structures organisationnelles…etc.)
Les services Web constituent, dans cette perspective, une voie prometteuse. C'est un pas en avant de l'histoire des S.I et de l'intégration des applications d'entreprises.

Est-ce que l'étape ultime ?

1.1 Principes Fondateurs des services Web

D'un point de vue architectural, l'environnement d'un service Web n'est autre qu'un système Client/serveur avec la particularité d'un ***médiateur explicite*** (service d'annuaire) pour l'ensemble des partenaires. Cet aménagement offre l'avantage d'une architecture N-tiers pour le même médiateur, où la couche gestion de ressources (Ressource Management Layer) peut-être, elle-même, et d'une manière récursive, un autre S.I. L'architecture, ainsi offerte, répond parfaitement aux exigences d'extensibilité et d'ouverture des S.I.
Sur le plan fonctionnel, un service Web est un composant logiciel (application) exposé sur le Web via son interface et accessible par un utilisateur ou un autre service Web à travers son URI (Uniform Ressource Identifier). Il offre, ainsi, des *points d'entrée au S.I* pour les usagers externes et même internes du réseau **LAN(**Local area Network) désireux d'engager des interactions avec l'organisation, sous conditions de respecter les spécifications (contrats) publiées. Les fonctionnalités du S.I sont exposées préalablement sur le Web, en vue de futures éventuelles sollicitations.
Sur le plan technique, les interactions engagées dans le cadre des services Web sont basées sur les échanges de *messages asynchrones* (non bloquants). Ces messages sont formatés dans un standard universellement adopté qui est: XML (Extensible Marckup Language). Il est orienté syntaxe, auto- descriptif et portable. Ce format offre un cadre adapté au traitement automatique des documents échangés et favorise l'interopérabilité.
Cependant, pour l'utilisateur final ce nouveau paradigme n'a d'utilité que s'il permet la prise en charge effective des activités réelles (achat, réservation, sinistre assurance, transaction boursière…etc.), transactions de type B2B (Business To Business) et d'une manière générale: la prise en charge des processus métiers (Business Process). En effet, si à différents niveaux

du processus d'intégration les problèmes liés au protocole de communication, aux formats et aux types de données ont pu être résolus, les problèmes relatifs à la sémantique des messages, à l'ordonnancement des tâches et au contrôle du processus demeurent résiduels. En effet, dans le contexte des services Web, un processus métier, peut être vu comme un Workflow trans-organisationnel où le traitement est distribué à travers le Web et engage, donc, plusieurs opérations complexes et/ou plusieurs services. Le principe de la composition des services répond à cette problématique.

Néanmoins, une telle composition engage des interactions multiples entre les participants (applications, clients, services). Il devient, alors, inévitable de déterminer un cadre de « Bonne conduite » pour les conversations déclenchées. L'ordre des opérations impliquées et la spécification des dépendances fonctionnelles (opérations conditionnelles, opérations parallèles…etc.) doivent être précisés par chaque service web. Ainsi, il devient possible d'exécuter les processus métiers par articulation de divers services qui subissant, périodiquement, des phénomènes de «naissance et de mort» et contrairement aux processus classiques, qui sont fixés préalablement, cette caractéristique s'adapte parfaitement à la nature versatile du Web.

L'environnement des services web est dynamique, ouvert, hétérogène et multilinguistique donc des standards sont nécessaires à chaque niveau des échanges (description, communication, publication, conversation…etc.).

1.2 Vers une vision opérationnelle des services Web

Pour mieux appréhender le fonctionnement des services Web, nous aborderons leur dynamique avec un exemple réel.

Supposons que deux partenaires (un producteur et un sous-traitant) veulent engager des transactions (achat, ventes, livraisons…). Pour atteindre leurs objectifs, ces deux partenaires doivent agir par niveau de préoccupation.

1. Préalablement, ils doivent disposer d'une infrastructure matérielle de communication (Réseau) et adopter un même protocole de communication (HTTP, SMTP), au dessus du réseau de transport (TCP). En plus tous les messages échangés seront formatés dans une syntaxe commune: XML.
2. Une fois le cheminement des données garanti, un mécanisme permettant l'interaction entre les deux partenaires doit être mis en place. Le protocole SOAP (Simple Object Access Protocol) répond à cette préoccupation. Les partenaires peuvent échanger les messages au format XML pour invoquer un service ou transmettre une réponse au dessus du protocole de communication choisi.
3. L'étape suivante consiste à pouvoir décrire les interfaces des services en question (les opérations et leur signature). Le standard WSDL (Web Service Description Language) assure ce rôle et permet de décrire les opérations offertes par le service, spécifie comment les invoquer ainsi que leur localisation. Au format XML, il accomplie les mêmes fonctions que l'ancien Standard IDL (Interface Description Language).
4. Finalement, une fois que les services sont décrits et peuvent être invoqués, il ne reste plus qu'à les publier dans un répertoire de services afin qu'ils puissent être découverts et invoqués par les autres partenaires. Prévu pour un usage à grande échelle (Web), l'UDDI

(Universal Description Discovery and Integration) garantie une telle fonction de publication universelle.

5. L'invocation du service par son interface est assujettie à la sémantique des données échangées (exemples: Prix HT ou en TTC) et aux contraintes régissant les interactions dans lesquelles le service est censé s'engager (exemple: Envoie d'une facture après confirmation de la commande). Par conséquent, des standards pour la prise en compte des aspects comportementaux et de la sémantique sont inévitables pour un déploiement réel des services Web. La discussion précédente peut être synthétisée dans le tableau suivant:

Préoccupation	Niveau d'abstraction	Problèmes à Résoudre	Standard
Automatisation, Services intelligible	Sémantique	Quelle sémantique des données échangées ? Comment automatiser les services Web ?	OWL-S, WSMO, WSMF
Conversation et Coordination	Flux de services, Processus métiers	Comment exécuter les processus métiers ? Composer les services ?	BPEL
Publication et Découverte	Localisation des services	Comment les autres prennent connaissance de l'existence du service ?	UDDI
Description	Implémentation	Que fait le service Web, comment l'invoquer? Où le trouver ?	WSDL
Interaction	Echanges de messages et Communication	Comment faire interagir les partenaires ? Quel format de Message ?	SOAP
Transport	Géographique	Comment échanger les données ?	TCP, HTTP, SMTP

XML

Tableau 1.1: Les niveaux d'abstraction dans les services Web.

Ce nouveau modèle de collaboration offre aux entreprises un socle pour l'interopérabilité et un cadre favorable pour l'intégration de leur S.I. Basé fondamentalement sur le couplage faible entre les composants du S.I, il permet à l'entreprise d'une part, d'externaliser ses fonctions (par leur publication) et d'autre part de pouvoir s'abonner aux services externes des partenaires souhaités (invocation), ouvrant ainsi la voie à la communication inter-applications. De ce point de vue, l'intégration des différents systèmes patrimoniaux qui se sont accumulés au fils des années se fera par à une simple harmonisation.

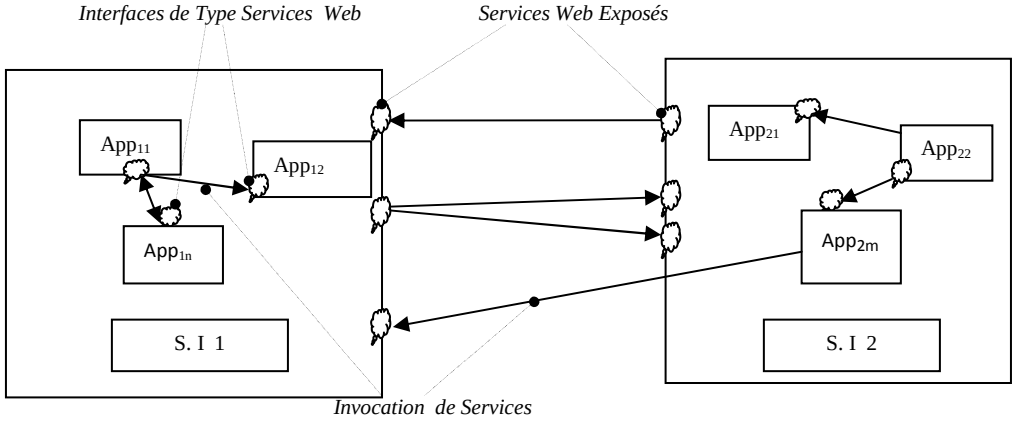


Figure 1.1: Intégration basée sur le Modèle de Service des applications Intra et Interentreprises

L'interaction basée sur le modèle de services Web devient plus aisée et évolutive. Les entreprises développeront des services Web autour des applications existantes, sans remettre en cause le capital applicatif accumulé au fil des années. De même, l'intégration interne, se fera à travers les interfaces homogènes de type services Web.

1.3 Les services Web: Technologie, Architecture et fonctionnement

Dans les sections précédentes, nous avons mis en relief les avantages des services Web en tant que solution ultime pour les problèmes d'intégration intra et extra entreprises. L'accent a été mis sur les nouvelles exigences des entreprises dans un environnement ouvert et dynamique. Les services Web, dépassement historique et sans rupture, des différentes technologies classiques d'intégration apportent une solution satisfaisante à ces exigences.
Dans la suite de ce chapitre, nous définissons d'une manière détaillée les services Web avec présentation de leurs objectifs, puis nous présenterons leur architecture et leur fonctionnement.

1.3.1 Qu'est-ce qu'un service Web ?

De façon très simpliste, nous pouvons décrire un service Web, comme une application accessible à d'autres applications à travers le Web. Cependant, cette définition ouverte, sous entend que tout objet possédant un URL (**CGI**: Common Gateway Interface, **API**: Application Programming Interface) est un service Web. [4]
Une définition plus précise est fournie par le W3C.
« Un service web est un système logiciel destiné à supporter l'interaction ordinateur-ordinateur sur le réseau. Il a une interface décrite en un format traitable par l'ordinateur (Particulièrement WSDL). Autres systèmes réagissent réciproquement avec le service web d'une façon prescrite par sa description en utilisant des messages SOAP, typiquement transmis avec le protocole HTTP et une sérialisation XML, en conjonction avec d'autres standards relatifs au Web». [2]
En d'autres termes, un service web désigne essentiellement une application (un programme) mise à disposition sur Internet par un fournisseur de service, et accessible via son interface par les clients à travers des protocoles Internet standard. Ces fonctionnalités peuvent être ***découvertes, invoquées et composées*** pour fournir une solution ou une réponse à un problème ou à une requête d'un utilisateur qui y accède via l'ubiquité des protocoles Web.
C'est un composant logiciel représentant une fonction applicative (ou un service applicatif). Il peut être accessible depuis une autre application (un client, un serveur ou un autre service Web) à travers le réseau Internet. Ce service applicatif peut être implémenté comme une application autonome ou comme un ensemble d'applications. Il s'agit d'une technologie permettant à des applications de dialoguer à distance via Internet, et ceci indépendamment des plates-formes et des langages sur lesquelles elles reposent. Pour ce faire, les services Web s'appuient sur un ensemble de protocoles standardisant les modes d'invocation mutuels de composants applicatifs [3].
Cette dernière définition met en évidence l'aspect d'interopérabilité en sus de la nécessité des protocoles standard pour les échanges.
Les définitions précédentes permettent de dégager au moins trois principes fondamentaux:

- Les services Web interagissent à travers l'échange des messages encodés en XML ; un standard largement répandu.

- Des protocoles universels constituent la superstructure permettant la publication, la découverte et l'invocation des services Web.
- Les interactions dans lesquelles les services Web peuvent s'engager sont décrites au sein d'interfaces.

La définition W3C restreint la portée des interfaces des services Web aux aspects fonctionnels et structurels, en utilisant le langage WSDL. Cependant, WSDL ne permet de décrire que des noms d'opérations et des types de messages. Cette insuffisance, ainsi que les solutions apportées seront abordées dans le chapitre 3.

En partant des définitions précédentes, il est à signaler que des confusions relatives au concept service Web sont souvent constatées. En effet, les pages Web, mêmes dynamiques, offrant des services aux clients via Internet ne constituent pas des services Web sans qu'elles ne soient développées et accessibles à travers les standards précités [4].

1.3.2 Objectifs visés par les Services Web

L'objectif ultime de l'approche service Web est de transformer le Web en un dispositif distribué de calcul où les programmes (services) peuvent interagir de manière intelligente en étant capables de se découvrir automatiquement, de négocier entre eux et de se composer en des services plus complexes [1].
En effet, les services Web ont pour objectifs de:

- **Faciliter l''interopérabilité**

Les services Web favorisent les interactions entre des environnements hétérogènes et diversifiés (systèmes d'exploitation, plates formes et langages). Ils deviennent, alors, un moyen technique intéressant pour interconnecter des modules s'exécutant sur des environnements autonomes et hétérogènes. Particulièrement, les échanges entre partenaires dans un scénario B2B sont plus souples et fluides.

- **Favoriser la réutilisabilité**

Les services Web, en tant que briques de bases du S.I de l'entreprise peuvent être réutilisés dans différentes applications internes et auxquels, les partenaires peuvent s'abonner pour réaliser des fonctions spécifiques. Dans ce contexte, le niveau de granularité (faible, moyen, fort) du service est un facteur déterminant pour la réutilisabilité.

- **Faciliter l'intégration d'applications**

Les services web sont homogènes (mêmes interfaces en WSDL) et faiblement couplés. Ils réduisent par conséquent les difficultés liées à l'intégration des applications intra et interentreprises.

- **Permettre l'automatisation des processus métiers**

Les services Web permettent d'intégrer, gérer et automatiser rapidement les processus métier intra et interentreprises en composant des services élémentaires et en échangeant des informations au format XML.
Ces objectifs convergent parfaitement avec la vision SOA (Service Oriented Architecture). En fait, les services Web sont une réalisation des SOA.

Pour l'entreprise, disposer des services Web c'est avant tout ouvrir son système d'informations à d'autres usages, d'autres besoins et d'autres clients extérieurs à l'entreprise [3].
Les applications possibles des services Web sont nombreuses et déjà opérationnelles. Des activités telles que fournir des cours de bourse, accès aux informations météorologiques, domaine financier et d'une manière général le commerce électronique se prêtent bien au développement de services Web. Les applications implémentant les fonctionnalités de chaque partenaire sont définies via un mécanisme standardisé permettant de les décrire, de les localiser en lignes, et de les faire communiquer les unes avec les autres.

1.3.3 Architecture et fonctionnement des Services Web

a. Architecture de base des services Web

L'architecture externe des services Web s'articule autour d'un médiateur central qui garantie la publication des services proposés par des fournisseurs au profit des futurs clients.

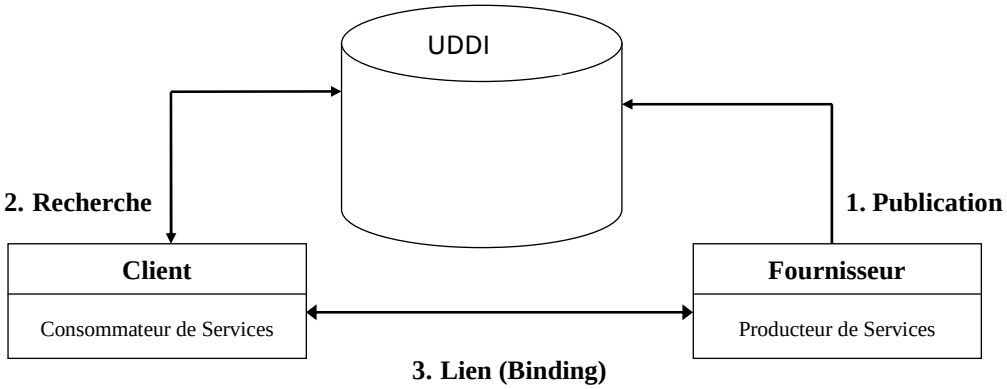


Figure 1.2: Architecture des Services Web

Les trois opérations de base de cette architecture sont la publication de la description, la découverte du service suite au déclenchement de la recherche et l'interaction entre le client et le fournisseur (Binding).
Nous allons décrire avec plus de détails chacune de ces opérations.

1. **La Publication:** Les services web sont centrés autour du concept de service. La première opération, consiste à le décrire. La sémantique des différentes opérations, leur signature (entrées/sortie) ainsi que l'ordre dans lequel ces opérations doivent être invoquées sont spécifiés en détails. Vu que les services sont dénués de contexte implicite à cause de l'absence d'un médiateur central (contrairement aux middlewares conventionnels), il est impératif de spécifier son adresse (URI) et son protocole (e.g. HTTP). Une fois ces informations formalisées dans le langage WSDL, le fichier de description doit être publié dans l'annuaire de services pour d'éventuels recherches.
2. **La Recherche**: L'annuaire de services contenant les descriptions des services (Fichiers WSDL), antérieurement enregistrés. Il permet aux utilisateurs de les chercher et de les localiser sur la base d'un ensemble de critères.
3. **Lien (binding):** Une fois le service localisé, le Binding consiste en l'invocation du service par le client et la réponse du fournisseur à cette requête dans un mécanisme d'interaction.

Par ailleurs, l'architecture fait ressortir les rôles et les interactions engagés dans un environnement service Web.

Les rôles:

Les trois rôles relatifs aux services Web sont:

- Le Fournisseur de service: C'est le propriétaire du service. Il publie le service au niveau de l'annuaire de service en déposant le fichier de description dans un format préalablement défini et contenant les informations commerciales, les services offerts ainsi que les informations nécessaires à l'invocation du service. D'un point de vue technique, il est constitué par la plate-forme d'accueil du service.
- Le Client ou Demandeur: C'est le consommateur de service. Il recherche le service désiré dans l'annuaire, sélectionne un service et l'invoque à travers la description publiée en interagissant directement avec le fournisseur. D'un point de vue technique, il est constitué par une application (ou un utilisateur) ou même un autre service Web.
- L'Annuaire de service: Correspond à un registre de descriptions de services offrant des facilités de publication de services à l'intention des fournisseurs et des facilités de recherche de services à l'intention des clients.

b. Fonctionnement des services Web

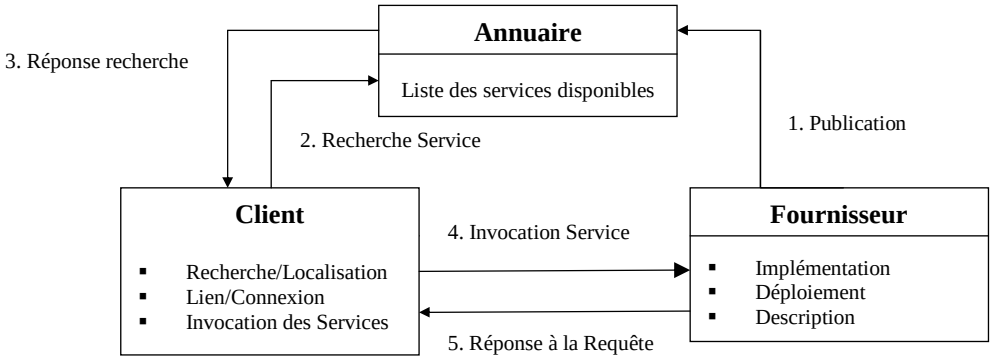


Figure 1.3: Fonctionnement des Services Web

Les interactions dans lesquelles les trois acteurs précédents peuvent s'engager sont décrites par les cinq opérations suivantes [4]:

1. **Publication:** Après avoir décrit le service dans le format WSDL, le fournisseur doit le publier dans l'annuaire de services (UDDI) afin de permettre aux futurs utilisateurs de prendre connaissance de sa disponibilité. Cette opération consiste en la fourniture des informations concernant le fournisseur (Information commerciales), relatives au service lui-même (les fonctions assurées) ainsi que les informations techniques (méthode d'invocation, adresse…etc).
2. **Recherche:** Le client interroge l'annuaire (UDDI) par l'envoi d'une requête encapsulée dans une enveloppe SOAP afin de rechercher le(s) service (s) répondant à ses besoins. Dans le cas de plusieurs réponses des techniques de sélection de services sont offertes (premier trouvé, aléatoirement) ou sur la base de paramètres techniques telle que la Qualité de service (QoS). Ce dernier type de sélection sera détaillé dans le chapitre 3 (Contraintes non fonctionnelles).

3. **Réponse à la Recherche:** Le résultat de la recherche dans l'annuaire est un fichier WSDL qui sera transmis, dans un message SOAP, au client demandeur. Ce fichier contient la description du service et les informations techniques nécessaires pour son invocation.
4. **Invocation:** En possession du fichier WSDL, le client va interagir directement avec le fournisseur du service. Il envoi un message SOAP, contenant en plus du (des) nom(s) de(s) l'opération(s) à exécuter et de ses paramètres, l'adresse (URI) et le type de protocole de communication (HTTP). Techniquement, cette invocation est basée sur un appel de procédure à distance (RPC).
5. **Réponse à l'Invocation:** Le fournisseur réagit à la réception du message d'invocation de la part du client en exécutant le stub serveur et en envoyant au client la réponse résultant de l'exécution de la procédure. Cette réponse est transmise sous la forme d'un message SOAP.

1.3.4 Pile conceptuelle des Services Web

Le tableau **1.1** fait apparaître, clairement, les sous problèmes (préoccupations) que les services Web tentent de résoudre pour aborder le problème de l'interopérabilité. En effet, à chaque niveau est associée une couche conceptuelle formant ainsi une architecture en pile. Chaque couche s'appuie sur un standard particulier. La couche de base est la couche de transport au dessus de laquelle se trouvent les trois couches (message: SOAP, description: WSDL, publication et recherche: UDDI) formant ainsi l'***infrastructure de base***. Cette dernière peut être augmentée par d'autres couches pour la prise en charge des fonctions suivantes:

- Le traitement des flux de service afin de réaliser l'exécution des processus métiers.
- La prise en compte de la sémantique des informations et des messages échangés.
- La prise en compte des aspects non fonctionnels tels que: la sécurité, les transactions et l'administration des services Web.

Ces fonctions étendent l'infrastructure de base par des couches verticales qui sont prises en compte à chaque niveau de la pile conceptuelle. (**Figure 1.4**)

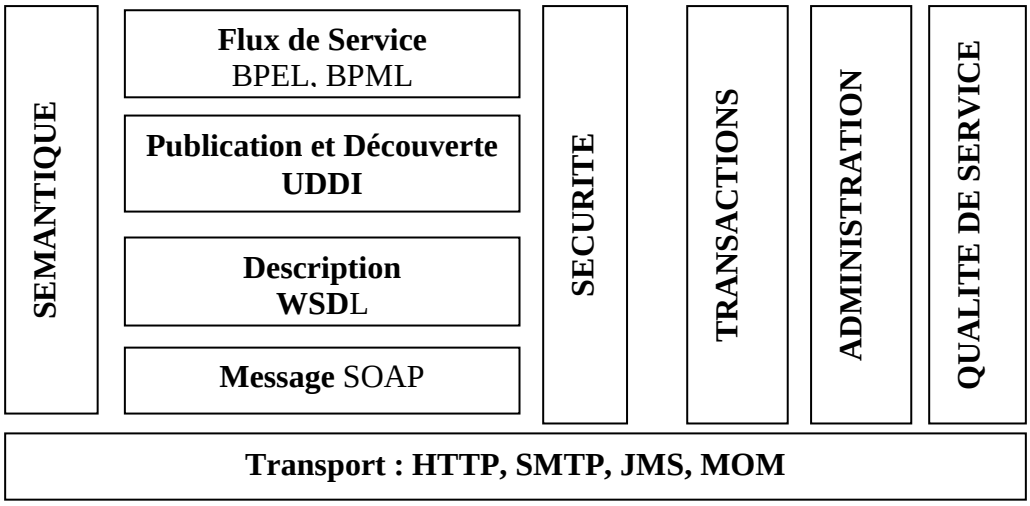


Figure 1.4: Pile Conceptuelle étendue des Services Web, d'après [1]

Cette infrastructure étendue, en plus d'être un cadre conceptuel offrant une réponse adéquate à la problématique de l'interopérabilité et par conséquent un *catalyseur* pour l'intégration, favorise et impulse cette dernière en prenant en considération la sémantique des échanges.

Par ailleurs, en prenant en compte les aspects non fonctionnels, inhérents au déploiement réel des services Web dans un contexte économique, où des questions telles que la sécurité ou la qualité de services sont incontournables, cette infrastructure offre un cadre conceptuel complet pour leur déploiement effectif.

1.4 Technologies Associées aux Services Web: Les standards

L'exploitation par les entreprises des potentialités offertes par Internet dans la perspective de l'intégration de leurs applications a induit des problèmes complètement différents de ceux posés par les middlewares conventionnels. En effet, le problème ne se limite plus à celui de la communication inter-systèmes hétérogènes, mais de nouvelles contraintes conceptuelles et des implications architecturales résultent de l'utilisation de l'Internet comme canal de communication par les systèmes à intégrer.

Le premier effort permettant un déploiement réel des services Web consiste en la standardisation des langages et protocoles utilisés, comme il a été fait pour le Web lui-même (HTTP, HTML, Browser, Serveur Web…). La nécessité de continuer cet effort, permettra non seulement le fonctionnement, mais aussi la prolifération des services Web. Par conséquent, leurs spécifications doivent demeurer aussi *génériques* que possible pour permettre leur exploitation par divers partenaires. A cet effet, des standards universels, à chaque niveau de la pile conceptuelle (**Figure 1.4**) sont inévitables.

Dans cette partie, nous allons voir les standards relatifs aux services Web, en commençant par XML, comme langage de base commun. Pour chaque standard on donnera sa définition et son objectif. La description détaillée de la structure de chaque standard avec un exemple illustratif sera donnée en annexes.

1.4.1 XML: Extensible Marckup Language

XML est un langage de balisage à mi-chemin entre la complexité exagéré de SGML (Standard Generalized Mackup Language) et le balisage figé de HTML. C'est un format de représentation de données et d'échange de documents sur Internet basé sur la syntaxe au lieu de la sémantique. Sa simplicité, sa portabilité sur différentes plates formes, son auto-description ainsi que son extensibilité ont conduit à sa grande popularité et son adoption comme standard d'échange à travers le Web, plus particulièrement, comme langage de base commun fondateur des services Web.

Exemple de document XML représentant les données d'un compte bancaire:

```
< ?xml version="1.0" encoding= "UTF-8" ?  >
  <Compte>
     <montant> 1000 </montant>
     <années>   20 </années>
     <taux-intérêt> 7 </taux-intérêt>
     <Taxe-annuel> 200< Taxe-annuel>
  </Compte>
```

Un document XML est un arbre composé d'un ensemble d'éléments structuré sous forme de balises. Cette structuration hiérarchisée ouvre la voie au traitement automatique du document. Par ailleurs, avoir une structure clairement définie et extensible favorise la prolifération d'outils de traitements des documents XML pour l'extraction de leur structure aussi bien que de leur contenu. Le test de validité d'un document par rapport à une structure préalablement définie (DTD ou XMLschéma) est alors possible. Il existe une panoplie d'outils permettant l'exploitation des documents XML (Xpath, Xquery pour interroger des documents, XSL, XSLT pour transformer des documents,…).

Un ensemble de standards basé sur XML et traitants des différents aspects inhérents au déploiement des services Web s'est développé et se développe encore. Ces spécifications émanant de constructeurs et consortium divers permettent de construire des services Web, d'assurer l'interopérabilité entre composants applicatifs et de garantir des conditions opérationnelles d'exploitation réelles de manière à:

- Assurer le fonctionnement et le déploiement des services Web: SOAP, WSDL, UDDI…
- Effectuer l'échange de données dans un contexte de commerce interentreprises: xCBL, RosettaNet, …Etc.
- Assurer l'intégration de processus collaboratifs sur le Web (Workflow): BPML, XLANG, BPEL …
- Fournir des mécanismes de sécurité: XKMS…

Dans ce qui suit, nous détaillerons les 03 standards de base: SAOP, WSDL et UDDI

1.4.2 SOAP: Simple Object Access Protocol

Pour assurer l'interopérabilité entre composants, tout en restant indépendant des plates formes et langages, le W3C a proposé le standard SOAP qui définit un protocole assurant des appels de procédures distantes (RPC) basé sur XML et HTTP. (D'autres protocoles comme SMTP peuvent être utilisés, mais HTTP est le plus populaire).

Un message SOAP est un document XML encapsulé dans une enveloppe permettant d'organiser les informations d'une manière qu'elles puissent être échangées entre partenaires.

SOAP a pour objectif de [4]:

- Offrir un format de message pour les communications entre partenaires, décrivant comment les informations à échanger sont structurées dans un document XML.
- Définir les conventions nécessaires à l'appel de procédures à distance pour invoquer un service par un client, en envoyant un message SOAP, et comment le service répond par un autre message SOAP.
- Décrire comment un message SOAP doit être transporté au dessus de HTTP ou SMTP.

Il s'agit, donc, d'un mécanisme fondamental pour assurer l'interaction et le dialogue entre applications distribuées dans un environnement services Web, du moment que clients et fournisseurs peuvent formuler, transmettre et comprendre les messages SOAP échangés.

La description détaillée de la structure d'un message SOAP ainsi que des exemples de messages sont présentés dans l'Annexe 1.

1.4.3 WSDL: Web Service Description Language

Similaire à l'ancien IDL (Interface Description Language) des middlewares conventionnels, WSDL décrit les services Web et, particulièrement, leur interface dans le format XML. En

plus de la spécification des opérations offertes par le service, WSDL décrit les mécanismes pour l'accès aux services Web (Protocole de communication) et sa localisation (URI), afin de préciser où envoyer les messages SOAP.
Une nette séparation entre l'interface de service et les mécanismes d'accès (Binding) permet à différents services d'implémenter la même interface, mais fournissent le service à des adresses différentes et qui peuvent être invoqués avec des protocoles différents [4].
En résumé les objectifs de WSDL sont: (d'après [4])

- Décrire les interfaces des services en précisant les opérations offertes et leur signature (Paramètres d'Entrée/Sorties et types). Ces interfaces constituent les contrats que les clients doivent respecter pour pouvoir interagir avec le service.
- Préciser le (s) protocole (s) d'accès au service (HTTP, SMTP…).
- Fournir la localisation du service où il peut être invoqué (URI).
- Définir une description de la sémantique du service par la fourniture des informations permettant, éventuellement, aux développeurs de traiter la sémantique des services.

Cependant, cet objectif reste loin d'être réalisé avec la spécification actuelle de WSDL (Version 1.2) et les développeurs font recours à d'autres mécanismes en dehors de WSDL pour rattacher une sémantique aux services.

Il est clair que les définitions WSDL facilitent le déploiement des services web en les rendant "*auto descriptifs*". En effet, elles leur permettent de décrire, de manière abstraite et indépendante du langage de programmation, ce qu'ils font, comment ils le font et comment les clients peuvent les exploiter.
Une description détaillée de la structure d'une interface WSDL ainsi que des exemples se trouvent en annexe 2.

Cependant, WSDL ne permet de décrire que les aspects fonctionnels grâce aux noms d'opérations et des types de messages. Par contre les aspects non fonctionnels tels que la qualité de service, inévitable pour le processus de découverte, ou les protocoles de coordination, garant d'une conversation correcte entre services en vue d'éventuelles compositions, donc à l'exécution des processus métiers, ne sont pas pris en charge par WSDL. Par ailleurs, La sécurité, l'administration et les transactions, aspects fondamentaux pour le déploiement réel des services Web dans un environnement économique, n'ont pas été prises en considération par les spécifications WSDL.

1.4.4 UDDI: Universal Description, Discovery and Integration

Contrairement aux standards précédents proposés par le W3C, UDDI (Universal Description, Discovery and Integration) est né de la collaboration entre plusieurs entreprises dont IBM, Microsoft et Ariba. Actuellement, il est géré par OASIS (une organisation indépendante regroupant plus de 300 compagnies du monde informatique).
Assurant le rôle d'un médiateur centralisé pour l'ensemble des partenaires désirant faire des échanges à l'échelle du Web, l'annuaire de services répond parfaitement aux besoins des fournisseurs des services pour la publication de leurs services et aux besoins des clients recherchant des services sur le Web.

Les objectifs de l'Annuaire UDDI sont: (d'après [4])

- Offrir un cadre pour la description et la découverte des services Web, en fournissant des structures de données et des API (Application Programming Interfaces), pour la publication des descriptions des services dans le registre et pour la découverte de ces publications pour les clients.
- Soutenir les développeurs dans la recherche des informations concernant le service afin qu'ils puissent réaliser des stubs clients pour interagir avec ces services.
- Permettre les liens (Binding) dynamiques en offrant aux clients la possibilité d'interroger le registre UDDI et récupérer les références des services (description WSDL..) qui les intéressent.

L'annuaire UDDI est consultable de différentes manières:

a-Pages blanches (BusinessEntity)

Décrites sous la forme d'un schéma XML, elles contiennent les éléments relatifs à l'entreprise qui propose le service (nom, coordonnées, secteur d'activité, l'adresse du site Web…).

b-Pages jaunes (BusinessService)

C'est un ensemble de services proposés, répondant à un besoin métier spécifique. Les pages jaunes contiennent des informations relatives au métier de l'entreprise et à la description des services Web qu'elle propose (nom du service, description, code…). Une entreprise peut avoir plusieurs métiers et, par conséquent, plusieurs services Web pour ses différentes fonctionnalités.

c-Pages vertes (bindingTemplate)

Contiennent les informations techniques sur un service Web, et les références aux **tModels** (spécification des interfaces des services Web)

L'Annexe 3 donne une description détaillée de la structure du registre UDDI, de son fonctionnement ainsi que les différentes API.

1.5 Synthèse et Perspectives pour les Services web

Les services Web sont une technologie importante pour la coopération interentreprises. Ils s'appuient sur des standards d'Internet ce qui leurs assurent un déploiement à grande échelle rapide et facile. En Effet, contrairement aux middlewares classiques, leur environnement n'exige pas d'être sur le même réseau LAN et la communication se fait à travers le réseau Internet, en s'appuyant sur ses protocoles (HTTP, TCP/IP …).

En plus, les services Web sont basés sur des standards au format universel XML (SOAP, WSDL, UDDI), ce qui leur confère une simplicité exceptionnelle et favorise leur vulgarisation. Ces standards sont indépendants de la plate-forme de développement et du langage avec lesquels les services ont été développés, ce qui assure une grande interopérabilité des services.

Néanmoins, à l'état actuel, l'utilisation des services web est restreinte et les objectifs visés par cette technologie émergente ne sont pas atteints d'une façon conséquente. Cette situation est due, principalement, à la jeunesse du domaine, qui n'a pas encore atteint sa maturité, et au processus historique d'évolution de cette technologie qui vise, d'un côté à prendre en charge les aspects relatifs à d'intégration des applications d'entreprises (EAI), en considérant les services Web comme une évolution des middlewares et des EAI, et de l'autre côté à offrir un cadre basé sur XML pour l'intégration et l'interopérabilité à travers le Web.

Concilier ces deux visions complémentaires pour les entreprises exige des efforts de recherches plus approfondis pour faire face aux enjeux architecturaux, structurels et fonctionnels, sans pour autant remettre en cause les fondements théoriques et conceptuels formalisés par les SOA (Service Oriented Architecture).
En effet, les axes de progressions possibles pour des services Web peuvent se résumer en:

- Au niveau de la recherche dans l'annuaire UDDI: des travaux sont en cours afin de pouvoir exécuter des requêtes complexes, car pour l'instant la recherche ne fonctionne qu'avec des requêtes dites simples.
- Les contraintes relatives à la qualité de service ne sont pas prises en charge lors de la découverte des services. Les travaux de recherche dans ce sens convergent globalement vers un remaniement du registre UDDI pour qu'il puisse traiter les informations relatives à la qualité de service.
- Au niveau de la sécurité: au départ SOAP a beaucoup été critiqué pour son manque de sécurité. WS-Security est une extension de SOAP qui permet d'implémenter l'intégrité et la confidentialité. D'autres extensions ont été rajoutées à SOAP pour combler ses défaillances (WS-Adressing, WS-Routing, WS-Policy….). Cependant, cette prolifération des standards devient une entrave au déploiement des services Web, car beaucoup de standards anéantissent les standards.
- Au niveau de la composition des services: Peu de spécifications existent et l'exploitation effective des services Web devient celle de l'exécution d'un Workflow trans-organisationnel qui se traduit par la spécification des protocoles de coordination et des mécanismes nécessaires à l'exécution d'un tel Workflow.
- Intégration de la sémantique dans les services Web dans une perspective d'un Web sémantique pour les services. Cette tendance, très ambitieuse, fait la jonction entre le Web sémantique et les services Web pour créer des Services Web Sémantiques où les ontologies joueront un rôle fondamental [1].
- Les interactions engagées dans un environnement services Web sont basées sur les descriptions offertes par les interfaces de services WSDL. Or, la structure actuelle du fichier WSDL est insuffisante pour représenter tous les aspects des interactions.

1.6 Conclusion

Aux termes de ce chapitre, nous avons introduit le domaine des services Web en présentant leur contexte historique et leurs apports pour les systèmes d'informations des organisations. Les abstractions de haut niveau relatives aux différentes préoccupations, que tente de prendre en charge cette technologie, ont été identifiées à travers un scénario réel.
La technologie des services Web a été abordée d'une manière détaillée et les différents standards associés aient été décrits succinctement.
Par ailleurs, nous avons dors et déjà, repéré quelques insuffisances inhérentes à l'état actuel d'évolution de la technologie des services Web et nous avons essayé de synthétiser les perspectives de recherche dans ce domaine.
Nous aborderons dans le chapitre suivant la composition et la coordination dans les services Web.

Chapitre 2

Techniques de Coordination et de Composition dans les Services Web

Introduction

Dans le chapitre précédent, nous avons introduit le domaine des services Web en tant que technologie d'intégration des systèmes d'information hétérogènes et distribués. L'accent a été mis sur les acquis qu'offre l'approche services Web pour les entreprises évoluant dans un contexte économique ouvert et mouvant sous l'effet des nouveaux mode de communication et, particulièrement, suite à l'avènement et à la généralisation de l'exploitation de l'Internet. Les enjeux imposés, aux différents acteurs économiques, par cette nouvelle technologie ont été abordés. L'infrastructure de base des services Web et les standards y afférents ont été présentés en détails.

Cependant, dans les applications réelles les clients des services Web n'invoquent pas de simples opérations indépendantes ni des services Web isolés. Pour atteindre leurs objectifs et réaliser leurs activités métiers, les clients déclenchent, plutôt, plusieurs opérations successives, voir plusieurs services appartenant à divers fournisseurs, rendant, ainsi, les interactions client/fournisseurs plus complexes et impliquant plusieurs services Web.
Dans ce chapitre, adapté en grande partie de [4], nous aborderons les techniques offertes dans un environnement service Web, pour la coordination d'un ensemble d'activités, et pour garantir une interaction correcte entre un client et le (es) fournisseur(s) de services. Par la suite, nous examinerons la composition des services Web, comme technique pour la réalisation d'activité plus complexes et répondant à des besoins métiers réels. En présentera, également, les différents standards offerts dans ce contexte.

2.1 Coordination dans les services Web

2.1.1 Nécessité de la coordination dans les services Web

Exécuter un service Web peut impliquer l'invocation de plusieurs opérations et dans un ordre bien déterminé. Les interactions entre le client et le service sont, donc, formées par un ensemble d'invocation d'opérations. On utilise le terme conversation pour se référer à des séquences d'opérations qui surgissent entre un client et un fournisseur suite à l'invocation d'un service Web. Quant au protocole de coordination c'est la spécification de l'ensemble des conversations correctes et acceptées par un service donné. [4]
Il s'en suit que le passage d'un mode d'invocation simple et indépendant à un mode d'invocation, qui implique plusieurs opérations ordonnées, exige des remaniements et un renforcement de l'infrastructure de base, afin de la rendre apte à supporter de telles contraintes et de tels échanges. Les nouvelles exigences structurelles, auxquelles il faut faire face, sont d'ordre interne: *du point de vue implémentation* et d'ordre externe: *du point de vue interactions*.

- **Du point de vue implémentation:** Le client doit être capable d'exécuter des procédures complexes conformes à la réalisation des différentes opérations et dans l'ordre correct. Cette exécution multiple doit, en plus, garder le contexte d'exécution et faire passer les données d'une opération à l'autre. La logique interne du client est, évidemment, plus complexe. Elle peut être implémentée en utilisant les langages de programmation conventionnels, mais les techniques de **composition des services** sont plus avantageuses.

- **<u>Du point de vue interaction:</u>** L'exécution de la séquence d'opérations par le client doit obéir aux contraintes d'ordre imposées par le fournisseur à travers son ***protocole de coordination***, qu'il publiera dans l'annuaire. En effet, un service Web autorise l'invocation de ses opérations, uniquement, dans un ordre défini préalablement. Si les contraintes d'ordre ne sont pas respectées, le service Web ne peut pas traiter le message et retourne une erreur.

2.1.2 Modélisation de la conversation entre un client et un fournisseur

Les interactions clients/fournisseurs expriment les échanges engagés. Elles nécessitent d'être représentées avec des outils expressifs ou «*modèles*» qui offrent des abstractions de la conversation réelle. Le modèle de conversation client/fournisseur permettra, ainsi, au fournisseur de service Web de spécifier l'ensemble des conversations correctes que le client doit respecter, pour pouvoir interagir avec son service exposé. La modélisation a pour objectifs de décrire les opérations possibles et les enchaînements permis ainsi que les séquences de conversations possibles.

Etant donné l'ensemble des opérations fournies par l'interface du service Web, la description des conversations avec des modèles précisera l'ensemble des conversations correctes, en spécifiant quelle opération de l'interface invoquer, tout en se basant sur l'état actuel de la conversation. Par conséquent, la spécification de la conversation devient le protocole de coordination pour ce service Web. [4]

Ainsi, le fournisseur de service peut spécifier comment interagir avec son service, en complétant les informations déjà publiées dans la description WSDL.

2.1.3 Modélisation de la conversation à travers plusieurs services Web

Dans la pratique, les conversations engagées peuvent s'étendre à plusieurs services Web. Dans ce contexte, le client interagit avec plusieurs fournisseurs de services, qui peuvent à leur tour invoquer d'autres services Web (composition). C'est le cas d'un fournisseur de produits qui sous-traite la fourniture de produits auprès d'une centrale d'achat, en cas de non disponibilité des produits commandés au niveau de ses magasins. Le problème devient, alors, celui de la conversation entre plusieurs services Web, du fait que chaque intervenant impose des contraintes, en tant que fournisseur et subit, en même temps, d'autres contraintes en tant que client.

Pour répondre à cette problématique de modélisation de la conversation, il existe plusieurs modèles concis, fidèles et expressifs permettant de décrire les protocoles de coordination. Malgré leur diversité, la philosophie générale demeure la même. En effet, la spécification du protocole est basée sur la démarche suivante:

- L'identification des **rôles** (client, fournisseur, …)
- L'identification des **messages** échangés entre les différents partenaires.
- La spécification des **contraintes** d'ordre sur les messages échangés.

La spécification des contraintes d'ordre peut être représentée par le triplet:

< *Déclencheur, Opération, Fournisseur* > exprimant que la transition d'un état à un autre se manifeste si le «*Déclencheur*» invoque «l'*Opération* » proposée par le «*Fournisseur* ». Il est clair que la notion de rôle est primordiale dans un contexte multifournisseur, afin de situer la transition. Le cas contraire, il devient ambigu de localiser la transition du fait que plusieurs fournisseurs proposent les mêmes opérations qui peuvent être invoquées par divers clients.

Nous reviendrons sur les modèles de représentation des protocoles de coordination dans le prochain chapitre, où la modélisation du comportement externe des services Web sera abordée en détails.

2.1.4 Classification des Protocoles des services Web

Les protocoles des services Web peuvent être classés en deux catégories:

a) Les protocoles verticaux ou protocoles métiers (Business Protocols)
Ils sont spécifiques à un secteur d'activité particulier et ne sont applicables que dans ce domaine. Ils décrivent, en détails, toutes les informations nécessaires à l'accomplissement des transactions métiers concrètes. En effet, les documents impliqués dans les transactions, leur format, leur contenu et les opérations concernées par la réception ou l'envoi de ces documents sont présentés en détails. La plupart de ces protocoles sont proposés pour supporter les transactions B2B, comme le format EDI qui est actuellement étendu pour supporter le format XML et les concepts de services Web. Les protocoles métiers les plus connus sont: xCBL (XML Common Business Library), ResettaNet et ebXML (Electronic Business using XML). Il est à signaler que le protocole d'un service quelconque (réservation, achat,…etc) est aussi un protocole vertical.

b) Les protocoles horizontaux ou Protocoles des Middlewares
Ils définissent une infrastructure commune et indépendante du domaine d'application (Figure 2.1). Leur but est de doter les échanges entre services Web d'un haut niveau d'abstraction et de doter les développeurs d'un niveau de transparence, afin de les décharger des détails relatifs aux échanges et aux transactions. WS-Coordination et WS-Transaction sont des exemples de ce type de protocoles.

Les protocoles verticaux et horizontaux sont souvent combinés. Les protocoles horizontaux sont exécutés par le middleware pour fournir les propriétés nécessaires à l'exécution des protocoles verticaux. Dans un scénario typique, le service Web implémente et exécute le protocole vertical. Pour garantir les propriétés aux différents messages de la conversation, le middleware exécute les protocoles horizontaux. Par exemple, le middleware garantira la livraison des messages et la gestion des transactions pour un protocole de service pour la réservation de places dans un vol d'une compagnie aérienne donnée.

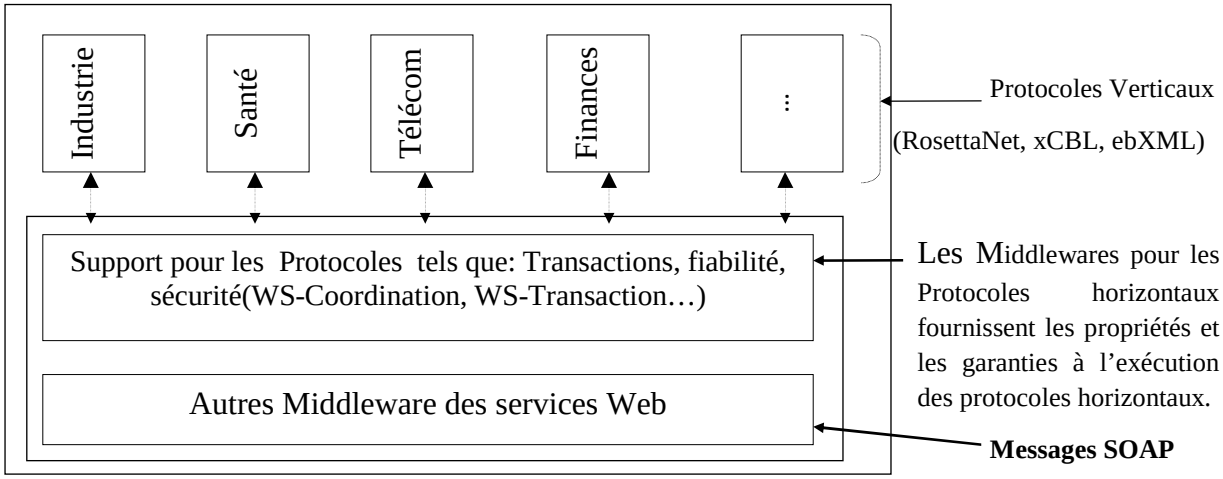


Figure 2.1: Protocoles horizontaux et verticaux pour les services Web, d'après [4]

2.1.5 Infrastructure pour les protocoles de coordination

L'infrastructure logicielle a pour fonction de prendre en charge l'automatisation de certaines fonctions, tout en facilitant le déroulement de la conversation. Les problèmes liés au routage des messages, dans le cas d'une conversation multipartenaires, et à la vérification de la conformité de la conversation avec le protocole de coordination sont pris en charge par cette infrastructure. Par ailleurs, certaines fonctions de traitement des messages (réception, interprétation et envoi) peuvent être prises en charge par l'infrastructure sans l'intervention du service Web.

Les deux outils mis à disposition, au niveau de l'infrastructure, sont les ***contrôleurs de conversation*** et ***les gestionnaires génériques de protocoles***.

a) Contrôleurs de conversation

Les contrôleurs de conversation sont des outils qui facilitent l'exécution des conversations. Ils fournissent deux sortes de fonctionnalités: le cheminement de la conversation (*conversation routing*) et la vérification de la conformité du protocole (*protocol campliance vérification*).

Le cheminement de la conversation concerne la gestion de l'expédition du message à l'objet interne approprié. Cette fonction est inévitable dans un contexte d'exécution multiple d'un même service Web par différents clients, car le service Web qui reçoit plusieurs messages doit déterminer à quelle conversation appartient ce message.

Pour détailler cette problématique, considérant un service Web qui est engagé dans plusieurs exécutions différentes d'un protocole de coordination P (les instances d'exécution sont nommées P1, P2, P3, P4 et P5 dans la figure 2.2). Comme les clients invoquent les opérations fournies par le même port, alors ils envoient leurs messages au même service Web. Quant le service Web reçoit le message, il doit déterminer à quelle conversation appartient le message.

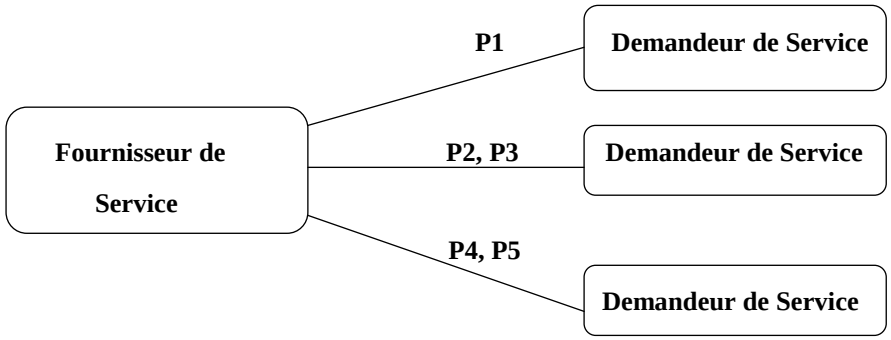


Figure 2.2: Exécution multiple d'un service Web conformément à un protocole de coordination, d'après [4]

D'un point de vue implémentation, le service Web doit gérer toutes les conversations, garder la trace de leur état (variables) et dresser toute la logique nécessaire pour déterminer à quelle conversation appartient chaque message; chose qui est très complexe et fastidieuse. Une autre technique, pour pallier à ce problème consiste à créer un objet pour chaque nouvelle conversation et laisser à l'infrastructure (contrôleur de conversation) le soin de prendre en charge le routage des messages aux objets appropriés, comme illustré dans la Figure 2.3.

Techniquement, le cheminement des messages est réalisé, en incluant un identifiant unique dans l'entête de tous les messages échangés. Cet identifiant est généré par le contrôleur de conversation, à chaque fois qu'un message déclenchant une nouvelle conversation est reçu. Par la suite, l'identifiant est inséré dans tous les messages échangés concernant cette conversation.

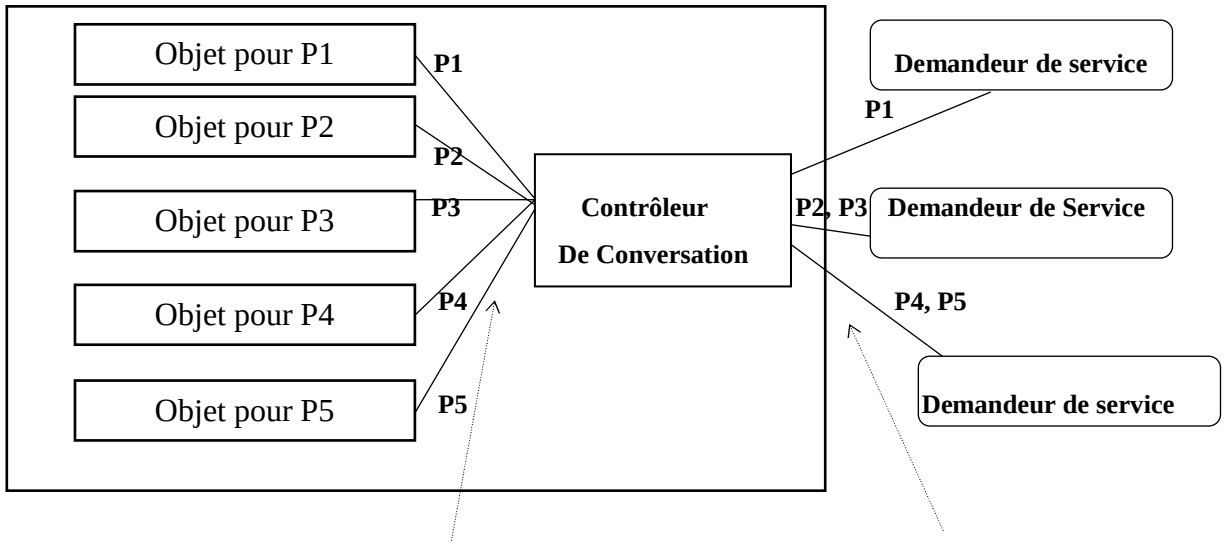


Figure 2.3: Aiguillage des messages par le contrôleur de conversation, d'après [4]

La deuxième fonction du contrôleur de conversation est la vérification de la conformité du protocole de coordination. Elle consiste à vérifier si une conversation engagée est conforme à la spécification du protocole. Le contrôleur vérifie si tous les messages respectent les contraintes imposées par le protocole, sinon il renvoie une erreur. Cela peut se produire, par exemple si un message est reçu avant l'autre, sans respecter l'ordre. Dans ce cas une erreur est renvoyée au client.

b) Gestionnaire génériques de protocoles

Les contrôleurs de conversation constituent une infrastructure générique qui peut supporter toute sorte de protocoles. En plus de ces composants génériques, les middlewares des services Web peuvent contenir des modules qui implémentent des protocoles de coordination spécifiques. Ces modules sont les *gestionnaires de protocoles* (*protocols handlers*). Ils comprennent une logique spécifique qui génère des messages conformément aux règles définies par le protocole. Ils sont, en principe, applicables pour tous les protocoles, mais c'est sur les protocoles horizontaux qu'ils sont souvent présents. Leur objectif est de renforcer les propriétés génériques des contrôleurs de conversations, comme les propriétés transactionnelles.

Le gestionnaire de protocole peut supporter l'exécution du protocole sous deux formes:

1. Le gestionnaire de protocole reçoit, interprète et envoi les messages de protocoles automatiquement, sans intervention du service Web. Dans ce cas, l'implémentation du protocole est transparente pour le service Web.

2. Le gestionnaire de protocole et le service Web partagent le poids de l'implémentation du protocole.

c) Besoin de standardisation pour les protocoles de coordination

Pour la prise en charge des questions relatifs au routage des messages, à la vérification de la conformité du protocole et à leur implémentation, l'infrastructure des services Web nécessite l'adoption d'un certains nombre de standards. Les efforts de standardisation doivent répondre aux préoccupations suivantes:

1. Déterminer une façon de générer et de transporter l'identifiant unique de la conversation dans l'entête des messages SOAP. Un premier effort remarquable, dans ce sens, est consenti dans le standard ebXML.
2. Définir un cadre d'application (Framework) et un ensemble de protocoles (qu'on appelle méta-protocoles) dont le but de prendre en charge les aspects liés à l'exécution et à la coordination entre les services Web, afin que ceux-ci puissent communiquer de la façon la plus simple et la plus efficace.
3. Nécessité de disposer de protocoles horizontaux standardisés, et profiter des propriétés additionnelles offertes par le middleware.
4. Nécessité d'un langage standard pour décrire les protocoles de conversation afin que le contrôleur de conversation puisse interpréter leurs spécifications et vérifier leur conformité.

Les fonctions précédentes, ainsi que d'autres, manquent dans les versions actuelles de la technologie des services Web (SAOP, WSDL et UDDI). Elles ne sont satisfaites que suite au recours à d'autres spécifications (WS-transaction, WS-Coordination, BPEL…etc.)

2.1.6 Quelques standards pour les protocoles de coordination

Dans cette section, nous présentons des exemples concrets de standards pour les protocoles de coordination. On se limitera à deux protocoles horizontaux et à deux autres protocoles verticaux.

a) Protocoles Horizontaux

Ils offrent des supports aux middlewares pour traiter les protocoles de coordination.

✓ **WS-Coordination**

Proposé par IBM, Microsoft et BEA en Aout 2002. Son principal objectif est de créer un cadre d'application (Framework) qui coordonne les actions des applications distribuées. A cet égard, il est prévu comme méta-spécification qui régira les caractéristiques implémentant les formes concrètes de coordination. Cet objectif est réalisé grâce à la standardisation des aspects suivant:

- Une méthode pour transmettre un identifiant unique: WS-Coordination définit une structure de données appelée *Contexte de coordination*.
- Une méthode pour informer le gestionnaire de protocole sur le port du service Web: WS-Coordination définit une interface d'enregistrement.
- Une méthode pour informer le gérant du protocole sur le rôle qu'il doit avoir dans la conversation, pour se faire WS-Coordination définit une interface d'activation.

Les entités de base du cadre d'application WS-coordination sont *les coordinateurs* et *les participants* qui peuvent interagir selon deux types d'architecture. Tous les participants peuvent interagir avec le même coordinateur (coordination centrale), ou chaque participant peut contacter son propre coordinateur (coordination distribuée).

Les spécifications de WS-Coordination décrivent un framework pour la coordination de services (ou coordinateur) qui consiste aux composants de services suivants [5]:

- **Service d'activation:** Doté d'une opération qui permet à une application de créer un nouveau contexte de coordination ou instance.
- **Service d'enregistrement:** Doté d'une opération qui permet à un service de s'enregistrer pour un protocole de coordination.
- **Type de coordination spécifique:** Ensemble de protocoles de coordination. Le coordinateur et le participant échangent des messages qui sont spécifiques au protocole de coordination.

La Figure 2.4 mis en évidence les services proposés et les messages utilisés.

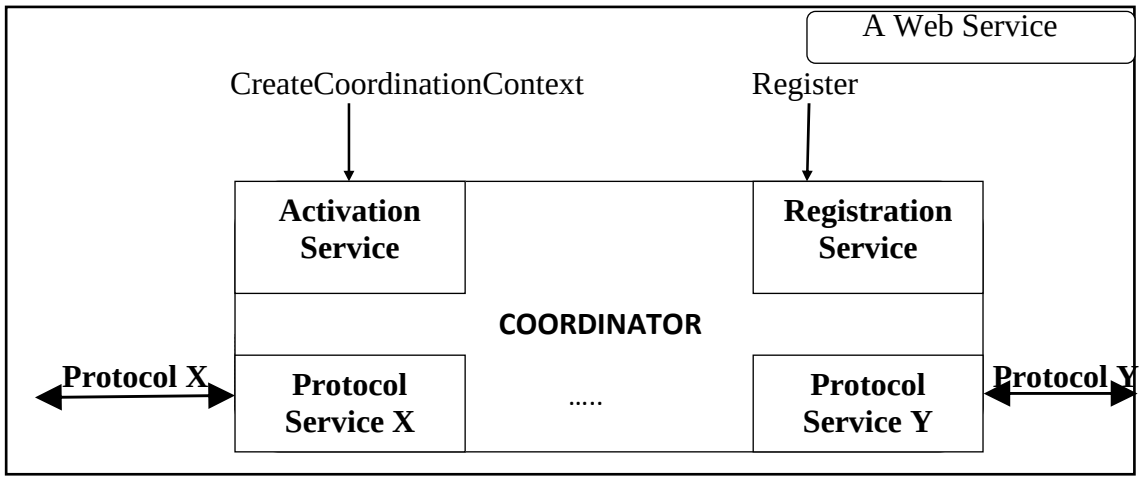


Figure 2.4: Composants et Opérations de WS-Coordination, d'après [5]

Trois types de messages sont identifiés dans le cadre de la coordination centralisée:

- Messages Opérationnels: échangés entre les services Web.
- Message WS-Coordination: entre les services Web et le coordinateur pour l'activation ou l'enregistrement.
- Message spécifique de protocoles: entre le service Web et le coordinateur en dehors du protocole X.

Malgré ses fonctionnalités, et les supports qu'il offre aux participants pour coordonner leurs activités (extension SOAP, méta-protocole, composants et interfaces), WS-Coordination n'est pas un langage pour décrire des protocoles de coordination.

✓ **WS-Transaction**

L'automatisation des interactions au sein des services Web requiert plusieurs protocoles. En effet, les principaux protocoles nécessaires pour la plupart des intégrations d'application sont ceux qui servent de support aux transactions.

Dans le contexte des services Web, de tels protocoles sont définis par WS-Transaction qui est un ensemble de spécifications construites au dessus du cadre d'application WS-Coordination. Il a été proposé en Août 2002 par IBM, Microsoft et BEA [6].

Les transactions dans les Services Web

Les transactions implémentées à travers les services Web sont souvent longues à s'exécuter. En effet, une différence importante entre les middlewares conventionnels et les services Web est liée au manque de ressources fixes. Dans une base de données transactionnelle il y a des définitions claires des termes «*Ressource*», «*Lock*», «*Commit*», «*RollBack*» et il existe, également, une structure claire des données, avec un ensemble d'opérations fixes (*Création, Insertion, Modification, Suppression*). Par contre, dans services Web il n'y a pas de telles caractéristiques. Les opérations WSDL peuvent représenter n'importe quoi, de l'insertion dans une base de données à l'envoi d'une lettre au client. Dans ce contexte, un *RollBack* sur une transaction est complètement différent d'une transaction à une autre.

L'idée de base adoptée par les services Web pour gérer les transactions consiste au fait que chaque participant du service Web peut mettre à jour son propre stockage persistant après chaque étape de la transaction. Si la transaction échoue pour une raison quelconque, le service Web exécute une opération de ***compensation*** qui annule les effets de l'exécution de la transaction. Pour répondre à ce besoin WS-Transaction spécifie un protocole standard pour les longues transactions appelées activités métiers (Business activities) [7] et un autre pour les transactions atomiques à courte durée (Atomic transaction) [8]. Les transactions et la compensation seront revues en détails dans le prochain chapitre.

b) Protocoles Verticaux

Ils sont spécifiques à un secteur d'activité donné et ne sont applicables que dans ce domaine. Nous présenterons comme exemples ebXML et xCBL.

- ✓ **ebXML: Electronic Business Using eXtensible Marckup Language**

Né sous l'égide de l'ONU et OASIS (Organisation for the Advanced of Structural Information standard), le standard ebXML a pour but de définir une infrastructure globale pour le commerce électronique basée sur XML et Internet, afin de pallier aux insuffisances des échanges avec le format EDI. ebXML ne s'occupe pas des documents échangés dans les transactions commerciales, mais de la formalisation du processus de transaction, c'est-à-dire du processus métier.
ebXML fournit une architecture qui décrit les formats de messages et les composants génériques utilisés dans les processus. Il propose également une structure d'annuaires répartis, recouvrant une partie de la spécification UDDI. [9]
ebXML n'est pas un langage standard métier. Il est, plutôt, un standard d'infrastructure ou de middleware. Il offre une spécification pour définir des collaborations métier. La spécification ebXML décrit ce qu'il faut faire exactement (document, processus et moyen de transport). Deux partenaires peuvent décider de collaborer s'ils peuvent échanger les mêmes documents, les mêmes processus et utiliser les mêmes mécanismes de transport. Ceci est exprimé sous la forme de CPA (Collaboration Protocol Agreement). En cours d'exécution les messages ebXML sont échangés selon les spécifications de ebXML Messaging Service.

Les divers services offerts par l'infrastructure ebXML sont utilisables soit pendant la phase de spécification/conception des services Web, soit en cours d'exécution. L'architecture technique ebXML fournit un framework de collaboration électronique de processus métiers. Elle permet aux métiers de collaborer afin de se découvrir mutuellement, négocier les accords de collaboration et exécuter les processus métier. L'interopérabilité de la communication des données est assurée dans cette architecture par un mécanisme de transport de messages standard avec une interface bien définie des règles de packaging et un modèle de livraison prévisible.
ebXML se présente aujourd'hui comme un ensemble de spécifications XML, de processus associés et de comportements définis pour fournir une infrastructure de collaboration B2B et d'intégration entre partenaires commerciaux.
Pour plus d'informations sur ebXML, consulter le site http://www.ebxml.org/

- ✓ **xCBL: XML Common Business Library**

La spécification XML Common Business Library (xCBL) [10], proposée par CommerceOne combine une version standardisée de document XML valable dans l'EDI et des protocoles métiers prédéfinis. Il prescrit, non seulement, quels documents doivent être échangés en tant qu'éléments des protocoles de gestion de commandes, mais également les contenus et la sémantique de ses messages (comme EDI mais en utilisant XML). Les spécifications incluent des rôles (client, fournisseur), déterminent qui lance quel échange et quel document est impliqué, et quand un document peut être utilisé. L'approche de xCBL est de se concentrer sur les échanges qui sont largement répandus et de standardiser ces échanges.
Pour plus d'informations vous pouvez consulter le site http://www.xcbl.org/

2.1.7 Synthèse sur la coordination dans les services Web

La coordination des services étend leurs fonctionnalités au-delà de leur simple interopérabilité. Les extensions conceptuelles rajoutent un niveau d'abstraction relatif à la coordination et aux échanges entre divers services. Les protocoles verticaux sont spécifiques à un secteur d'activité particulier, alors que les protocoles horizontaux fournissent aux services Web une infrastructure pour interagir ensemble dans un contexte de coordination afin de réaliser des processus métiers.

De part leur nature, les protocoles de coordination requièrent des langages pour leur définition et une infrastructure pour leur exécution. Un langage normalisé de protocole permet au fournisseur de décrire les interactions qu'il supporte à travers son service Web, et par conséquent, permet aux clients d'identifier comment les applications clientes doivent être développées pour qu'elles puissent interagir avec le service sélectionné.

La standardisation est un enjeu majeur dans le domaine de la coordination. Les efforts consentis par ebXML, xCBL ou RosettaNet ont anticipé l'apparition de SOAP, UDDI et WSDL. Les spécifications comme WS-Coordination et WS-Transaction sont plus récentes et sont construites au dessus de SOAP, WSDL et UDDI. Elles consolident l'infrastructure de base des services Web pour prendre en charge les problèmes de coordinations engendrés par des exécutions multiples des services Web.

2.2 Composition des services Web

Les services Web commencent à se proliférer sur le Web et sont déjà présents dans plusieurs secteurs d'activité. En effet, beaucoup d'entreprises exposent leurs services Web aux partenaires et clients. Cependant, chaque service Web, pris individuellement, ne fournit qu'une fonctionnalité limitée. Alors que pour réaliser leurs activités réelles, les clients n'invoquent pas de simples opérations. Ils réalisent, plutôt, des activités complexes ou «processus métiers». Il devient, alors, opportun de rechercher l'ensemble des services Web disponibles, de sélectionner les plus satisfaisants et de les combiner pour répondre à la requête du client. L'application cliente obtenue, peut à son tour être exposée comme service Web. C'est le principe de *la composition des services*.
Un service Web implémenté par la combinaison de fonctionnalités offertes par d'autres services Web est appelé *service composé* et le processus de développement d'un service composé s'appelle *la composition de services*.
Les services Web ont une chance de faire de la composition une proposition fiable, pour l'intégration des applications interentreprises, en surmontant les problèmes rencontrés par les EAI et les Workflows. [4]
Le processus de développement de services Web composites est une tâche assez complexe, car elle engendre des problèmes liés à la sélection des services impliqués dans la composition et à l'ordre des opérations à invoquer. D'autres difficultés sont dues à la gestion et au contrôle du flux de données à travers les services à composer, dues à la gestion des transactions et à la prise en charge des situations d'exception.

Dans cette section, nous étudierons les abstractions, les outils et l'infrastructure offerts par les middlewares des services Web pour décharger les développeurs de certaines tâches liées à l'implémentation, afin qu'ils puissent aisément définir et exécuter des services Web composites, tout en se focalisant sur la logique métier.

2.2.1 Les bases de la composition des services

Les services sont les briques de base de l'environnement service Web. Ils sont faiblement couplés facilitant, ainsi, leur réutilisation dans différentes applications et leur agrégation pour la construction d'applications (services) plus complexes. La composition de services permet d'obtenir de nouveaux services (du point de vue de l'utilisateur), ayant une valeur ajoutée, en combinant divers services existants.

L'un des aspects intéressants de la composition c'est la ***récursivité***. Etant donné que l'application cliente peut à son tour être exposée comme service Web, alors elle peut être utilisée comme composant pour d'autres applications de niveau de composition supérieur. La récursivité permet de définir des applications, de plus en plus, complexes en combinant des services existant à des niveaux d'abstraction supérieurs. Elle permet de réduire la complexité des systèmes, car les services complexes sont construits progressivement par composition des services simples.
Contrairement à la composition traditionnelle, la composition des services n'est pas basée sur l'intégration en dur (physique) de tous les composants. En effet, les services Web ne sont pas des bibliothèques de librairies qu'il faut compiler et lier comme partie de l'application.

Dans un contexte de composition de services, le client est engagé dans différentes conversations avec plusieurs services Web. Ces différentes conversations doivent être guidées par les différents protocoles. En même temps, chaque service Web invoqué ne se rend pas compte que le client est en train d'invoquer d'autres services Web. Cette situation engendre des problèmes de coordination, car des règles de synchronisation claires, des opérations invoquées, doivent être présentes et respectées. Ces règles forment le protocole de conversation de la composition. Le problème de la composition est indissociable de celui de la coordination.
Différents aspects sont liés à la composition de services. Dans ce qui suit nous aborderons: les scénarios de compositions (orchestration et chorégraphie), les types de composition (du point de vue automatisation) ainsi que la substitution et la compatibilité des services.

a) Scénarios de composition

La composition des services Web peut se faire de deux manières: Orchestration et Chorégraphie.

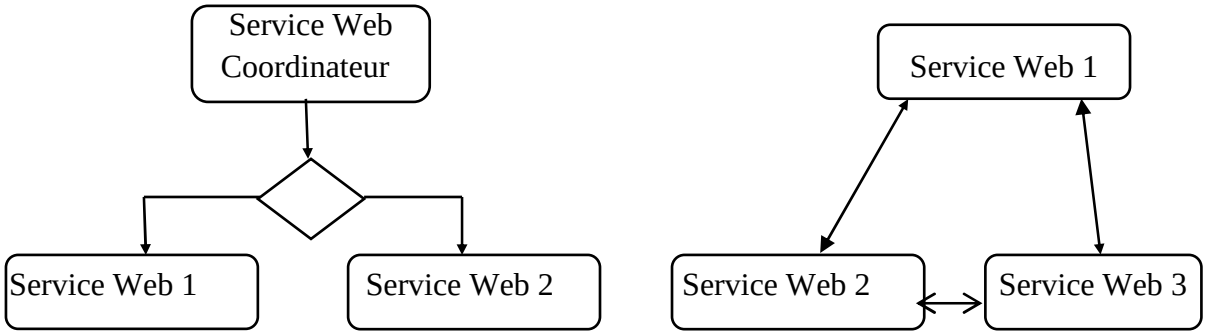
- **Orchestration de services**

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

La Figure 2.5 (a) montre le workflow relatif à l'orchestration des services Web. Un coordinateur prend le control de tous les services impliqués et coordonne l'exécution des différentes opérations participantes.

- **Chorégraphie de services**

Contrairement à la l'orchestration, la chorégraphie ne nécessite pas de coordinateur central. Chaque service Web impliqué dans la composition connaît exactement quand ses opérations doivent être exécutées et avec qui l'interaction doit avoir lieu. C'est un effort de collaboration dans lequel chaque participant au processus décrit l'itération le concernant et trace la séquence des messages impliquant plusieurs services Web.
La Figure 2.5 (b) montre la collaboration de plusieurs services dans la chorégraphie.



(a) Orchestration **(b) Chorégraphie**

Figure 2.5 Orchestration et chorégraphie des services Web, d'après [4]

Dans la perspective de la composition des services Web, l'orchestration est une articulation plus flexible que la chorégraphie, pour les raisons suivantes:

- Le coordinateur du processus métier est identifié.
- Les services Web peuvent être incorporés sans les considérations liées à leur utilisation dans une éventuelle composition. Ils ne se rendent pas compte de leur appartenance à un processus métier.

b) Substitution et équivalence de services

L'implémentation des services Web par composition d'applications (autres services) implique la possibilité de pouvoir remplacer, à tout moment, un service Web par un autre qui lui est fonctionnellement équivalent. Le service substituant est *équivalent* au service substitué s'il produit les mêmes sorties, le même comportement et la même sémantique des attributs. Ce besoin de substitution est motivé dans les cas où un service est défaillant, s'il existe un service web offrant des qualités de services meilleures où lorsqu'une nouvelle version de service Web est offerte par le fournisseur.

Deux services Web sont substituables si:

- Le service remplaçant fournie au moins les mêmes fonctionnalités que le service à remplacer.
- Les paramètres de qualité de services des deux services sont similaires.

c) Types de compositions des services

Du point de vu degré d'automatisation, la composition est classée dans l'une des trois catégories.

- **Composition Manuelle**

  Cette classe suppose que le développeur génère la composition à la main via un éditeur de texte par la sélection et la fusion des services à incorporer. Elle s'adapte aisément aux besoins de l'utilisateur, car il peut tout définir à son goût depuis le début. Seulement, il ne dispose d'aucun outil dédié et il est obligé de maîtriser les techniques de programmation de bas niveau.

- **Composition Semi-automatique**

  Les techniques de composition semi-automatique font des suggestions sémantiques aux utilisateurs pour les aider à sélectionner les services impliqués dans le processus de composition. L'utilisateur maintient certains contrôles sur le processus de composition, sans avoir à maîtriser les techniques de programmation de bas niveau. Il dispose d'outils graphiques de modélisation et de conception pour définir ses propres processus.

- **Composition automatique**

  La composition totalement automatisée prend en charge tout le processus de composition et le réalise automatiquement, sans qu'aucune intervention de l'utilisateur ne soit requise. Le service Web composite est une boite noire inaccessible et dont la flexibilité est minimisée.

2.2.2 Relation Composition - Coordination

La composition et la coordination des services Web sont deux mécanismes liés au contexte réel de multitude de services et de clients invoquant des services isolés ou composites.

Dans ce qui suit, nous allons éclaircir les confusions possibles autour de ces deux concepts.

La composition de services concerne l'implémentation interne des opérations dans le service Web. La spécification de la composition est réalisée par une entreprise et gardée privée car l'entreprise ne veut pas dévoiler la façon dont elle implémente les opérations pour développer une activité donnée. La spécification de la composition est destinée à l'exploitation par le middleware des services Web qui automatise l'exécution des opérations offertes par les services Web constitutifs, et ce conformément au schéma de composition. Etant relative à l'implémentation, le fait que le service soit simple ou composé est non pertinent pour le client.

Par contre, les protocoles de coordination sont des documents publics décrits dans des langages standards. Il n'y a aucun intérêt de garder leur spécification privée. Au contraire, les protocoles de coordination doivent être annoncés au niveau des registres des services Web, afin de permettre aux clients de les consulter et d'en tenir compte lors de la découverte, lors de l'exécution et durant la composition.

En résumé, la composition et la coordination diffèrent dans leur portée. La composition est une technique d'implémentation interne, contrairement à la coordination qui traite des interactions externes.

2.2.3 Modélisation de la composition de services

Vu la complexité du processus de composition des services web, il est nécessaire de passer par une modélisation qui traitera des différents aspects afférents à ce processus. L'utilisation des outils, techniques et concepts du domaine de la planification de tâches, tels que les workflows est intéressante dans cette perspective. Elle est justifiée par le fait qu'un service Web composé peut être vu comme un Workflow trans-organisationnel, où le traitement est distribué à travers le Web et engage, donc, plusieurs opérations complexes et/ou plusieurs services. En effet, le terme *définition de processus* (Process definition) est utilisé pour se référer à un schéma de composition, le terme *exemple de processus* (Process example) désigne une exécution individuelle et spécifique (ou instance) d'une définition de processus, tandis que celui de schéma d'*orchestration* (Orchestration schema) (ou simplement orchestration) se rapporte à la partie du schéma de composition qui spécifie l'ordre dans lequel les différentes services composants doivent être invoqués.

Pour caractériser le modèle de composition de services, les aspects suivants sont à définir:

- **Le modèle de composant:** définit la nature de l'élément à composer (Messages XML, normes de services Web: SOAP, WSDL).
- **Modèle d'orchestration**: définit les abstractions et les langages utilisés pour spécifier l'ordre dans lequel les services seront invoqués et composés comme un tout. De nombreuses possibilités existent avec des variations dans le formalisme utilisé: Diagrammes d'activités, Réseau de petri, π-calculs, Hiérarchie d'activités et orchestration basée sur des règles.
- **Modèle de données et d'accès aux données:** Il définit les types de données manipulées (spécifiques à l'application, Données de contrôle de flux) et comment ces données sont échangées entre les différents composants (*Blackboard, explicit data flow*).

- **Modèle de sélection de service:** Définit comment les liens (binding) sont établis (statique, dynamique par référence, dynamique par consultation ou sélection dynamique d'opérations) et comment un service spécifique est sélectionné comme composant.
- **Les transactions:** L'approche utilisée par les services Web pour traiter les transactions consiste à permettre la définition de *régions atomiques* dans le schéma d'orchestration. La région atomique regroupe un ensemble d'activités vérifiant la propriété du «tout ou rien», comme le montre la figure 2.6 et où les principes ACID (Atomique, Cohérente, Isolée et Durable) des transactions traditionnelles sont relaxés.
  Dans le contexte des services Web, certains processus sont trop-long (long live running), donc les ressources ne peuvent être bloquées trop longtemps. La définition d'*une activité de compensation* spécifique à chaque région atomique permettra la libération de la ressource à la fin de la région. En plus, chaque activité est liée à une activité de compensation qui s'exécutera en cas de problèmes.

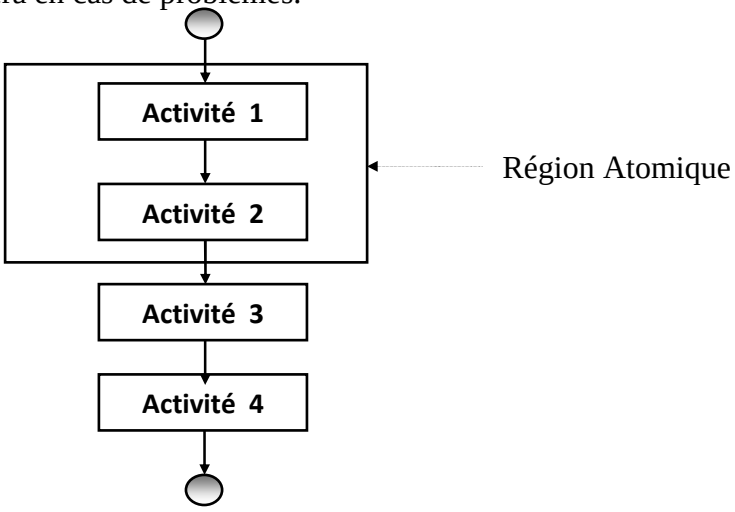


Figure 2.6: Exemple Simple de région atomique, d'après [4]

- **Le traitement des exceptions:** définit comment les situations exceptionnelles qui peuvent surgir lors de l'exécution d'un service sont pris en charge sans que le service composé ne soit abandonné. Les exceptions peuvent avoir différentes causes: Erreur dans le système, erreur lors de l'invocation d'application et des situations peu fréquentes, qui malgré la sémantique clair du service composé, n'ont pas été prises en charge.
  Les transactions sont une première solution possible de gestion des exceptions, mais d'autres techniques de modélisation et de traitement des exceptions existent telles que: approche à base de flux, approche Try-catch-throw et l'approche à base de règles.

2.2.4 Les standards de la composition des services Web

Dans cette partie, nous présenterons la panoplie des standards ayant émergé dans le domaine de la composition des services Web.

a) BPML (Business Process Modeling Language)

C'est un métalangage de modélisation des processus collaboratifs qu'engage l'entreprise avec ses partenaires sur Internet. Il prend en compte aussi les aspects les plus complexes de la gestion et de la coordination de ces processus, en abordant les transactions, la sécurité et la liaison dynamique avec les services Web en cours d'exécution. C'est une spécification neutre

des processus métier, visant l'automatisation des échanges entre services Web et applications Web. XML est y utilisé pour le typage des données (XML schéma) et pour le corps exécutable du processus.

En effet, un processus BMPL est un enchainement d'activités simples ou complexes et de processus incluant une interaction entre participants, dans le but de réaliser un objectif métier. Le processus de l'entreprise est perçu comme une collaboration entre participants qui échangent des messages XML. Dans ce contexte:

- Un processus est un ensemble d'activité ;
- Une activité est un ensemble de tâches ;
- Une tâche est une opération élémentaire.

Ce standard est géré par le BPMI (Business Process Management Initiative: http://www.BPMI.org/), une organisation qui regroupe plus d'une centaine de sociétés comme: Sun, HP, SAP, Siemens…, et qui c'est fixée comme objectif la formalisation des processus métier des entreprises en utilisant BPML.

b) WSCI (Web Service Choregraphy Interface)

C'est un langage reposant sur XML et qui décrit le flux de messages échangés par un service Web participant à une chorégraphie. Il décrit, ainsi, le comportement externe observable du service par le biais d'interfaces. Pour cela, WSCI propose d'exprimer les dépendances logiques et temporelles entre messages échangés à l'aide de contrôle de séquences, corrélations, gestion de fautes et transactions. Il fonctionne en jonction avec le format WSDL et ses définitions abstraites (opérations, port types). De cette façon il pourra interagir avec un autre service qui exprime les mêmes caractéristiques en WSDL.
En résumé, WSCI décrit l'échange collectif de messages entre les services Web en interaction. Il fournit une vue globale orientée messages de l'ensemble des interactions et ne traite pas l'implémentation interne du processus qui guide les échanges, mais se limite aux interfaces.
Pour plus de détails consultez: http://www.w3.org/TR/wsci/

c) WSCL (Web Service Conversation Language)

Proposé par le consortium W3C (http://www.w3.org/TR/wscl10/), WSCL propose de décrire les services Web à l'aide de documents XML en mettant l'accent sur leurs conversations. En outre, les messages échangés sont pris en compte. WSCL a été pensé pour être employé conjointement avec WSDL.
Les descriptions WSDL peuvent être manipulées par WSCL pour décrire les opérations possibles et leur chorégraphie. En retour, WSDL fournit les concrétisations des définitions de messages et les détails techniques pour les éléments manipulés par WSDL.

d) XLANG

XLANG de Microsoft, permet de décrire formellement les processus d'entreprises comme des interactions entre participants. La description complète d'un processus consiste aussi bien à définir le comportement de chaque participant, qu'à décrire la manière dont l'ensemble des participants interagit pour produire le processus complet. La description du processus

consiste, alors, à décrire les messages échangés. La façon dont chaque participant implémente sa composante du processus n'est pas prise en compte par le protocole XLANG. [12]
L'objectif de XLANG est de spécifier seulement les comportements que chaque participant veut exposer à ses partenaires afin de faciliter l'intégration de services. XLANG est le format proposé par Microsoft, pour représenter en XML, l'orchestration des activités qui constituent un processus métier.
XLANG couvre essentiellement les fonctions suivantes: [12]

- L'orchestration des flots de contrôle séquentiels et parallèles ;
- La gestion des transactions longues;
- La gestion des corrélations des messages entre eux;
- La gestion des défaillances et des erreurs ;
- La découverte dynamique de services;
- La gestion de contrats multipartis.

XLANG s'appuie sur WSDL en réutilisant un certain nombre de concepts. Il reprend, en particulier, la description WSDL d'un service en termes de groupe de ports et de liaisons à des protocoles de transport. Chaque port étant constitué, à son tour, d'opérations caractérisées par un échange de messages.

e) WSFL (Web Service Flow Language)

C'est une approche IBM pour décrire les flux métiers. C'est un langage XML permettant de décrire des orchestrations de services Web. WSFL définit deux types de composition de services:

- Le premier type décrit les processus d'entreprises comme des chorégraphies de service Web à l'instar de XLANG. Il explicite des flux de contrôle et de données entre les services Web constituant le processus.
- Le second type définit les processus métier en explicitant les relations producteur/consommateur de messages entre les différents services Web constituant le processus global.

WSFL aborde les workflows par une démarche à deux niveaux:

- Il se base sur une approche basée sur un modèle graphique direct pour définir et exécuter les processus métiers.
- Il définit une interface publique qui permet aux processus métiers de s'annoncer en tant que services Web.

Un des avantages de WSFL est la possibilité de créer des modèles récursifs: une composition de services Web peut être considérée comme un service Web qui est alors utilisable dans une autre composition.

f) BPEL4WS (Business Process Execution Language For Web Services)

Nouvelle spécification pour décrire les interactions entre les services Web qui composent un processus métier. Elle s'inspire des deux langages de processus qui sont XLANG de Microsoft et WSFL d'IBM. BPEL s'est imposé comme standard de base pour la composition des services Web.

BPEL (Acronyme utilisé au lieu de BPEL4WS) est un langage qui supporte la spécification des schémas de composition et des protocoles de coordination. Il profite de la similarité des techniques de modélisation de ces deux aspects pour fournir un cadre applicatif unique pour définir les deux.
Le schéma de composition exprimé en BPEL est une véritable spécification d'un processus exécutable qui définit la logique d'implémentation d'un service composite. Par ailleurs, le protocole de coordination est une perspective orientée service. Cette dualité descriptive de BPEL, permet de définir la séquence de messages échangés par un service (messages envoyés et reçus) et prend en compte les contraintes d'ordre entre les opérations d'envoi et de réception.
En d'autres termes, BPEL peut être utilisé pour définir le comportement externe d'un service (à travers un processus abstrait: *BPEL Abstract*) aussi bien que pour spécifier l'implémentation interne (à travers un processus exécutable).
Le processus BPEL spécifie l'ordre exact de l'invocation des services Web participant dans la composition. L'invocation peut se faire soit en parallèle soit séquentiellement. L'expression d'un comportement conditionnel est offerte. Par exemple, lorsque l'invocation d'un service peut être dépendante du résultat d'une invocation antérieure. La construction des boucles, la déclaration de variables et l'affectation de valeur…etc, sont aussi possibles. De plus, il est possible de combiner toutes ces constructions et de définir des processus métiers complexes d'une manière algorithmique. [10]

BPEL suppose que les interfaces des services Web en interaction sont définies en WSDL et que les services échangent les messages WSDL.

2.2.5 Synthèse sur la Composition des services Web

Dans le monde réel, pour atteindre leurs objectifs opérationnels, les clients réalisent des processus métiers complexes, engendrant l'exécution de plusieurs services Web. L'objectif de la composition de services est de répondre, justement, à cette exigence en offrant des fonctionnalités à valeur ajoutée par articulations de fonctionnalités déjà offertes par d'autres services Web.

Etant donnée une spécification de haut niveau des objectifs d'une activité (processus métier particulier), la composition de services implique la capacité de sélectionner, d'articuler et de faire inter-opérer des services existants, pour réaliser cette activité. Contrairement aux Business Process «traditionnels» qui sont exécutés d'une manière prévisible et répétitive dans un environnement statique, les services Web composés s'exécutent dans un environnement ouvert et versatile, où les services offerts sont diversifiés et évolutifs et où la forte compétition engendrée par la multitude de fournisseurs oblige les entreprises à mieux adapter leurs services pour répondre aux besoins croissants des clients et à moindre coût.

Cependant, pour permettre une telle composition, des abstractions, des outils et des environnements sont nécessaires. Ils permettent de sélectionner les services impliqués dans la composition, de spécifier l'ordre des opérations à invoquer, de gérer et de contrôler le flux de données à travers les services à composer, ainsi que la gestion des transactions et des situations d'exception. La composition est une technique d'implémentation qui reste

indissociable des protocoles de coordination décrivant le modèle de conversation imposé par le fournisseur et qu'il faut toujours respecter.

De nombreux langages ont émergés dans la perspective de la description et/ou de l'exécution de tels processus métiers complexes. BPEL est la convergence des efforts de standardisation de la part des différents acteurs du domaine. Il offre une infrastructure adéquate pour la prise en charge de l'ensemble des aspects liés à la composition et à la coordination des services Web. Cependant, en tant que langage procédural, BPEL reste inefficace dans la prise en compte des contraintes liées à la versatilité du Web, à la qualité de service et à la sécurité. Il n'offre pas de possibilités réelles d'adaptabilité des services composites à leur environnement.

Des travaux concernant cette problématique commencent seulement à émerger. Ils s'intéressent à une modélisation abstraite des services et à la définition d'un cadre formel pour les composer. En effet, des travaux de la *communauté Web sémantique* commencent à explorer des approches combinant des outils d'annotation de services et de planification de manière à pouvoir composer automatiquement des services, en vue d'atteindre des fonctionnalités prédéfinie. D'autres travaux en cours émanent de la *communauté des* systèmes *multi-agents.*

Dans ce contexte, il est impératif de focaliser les efforts sur un modèle formel solide qui constituera un socle pour la composition et la coordination des services, comme l'a été le modèle relationnel dans le domaine des Bases de Données.

2.3 Conclusion

Au niveau de ce chapitre, nous avons jugé utile d'aborder le contexte opérationnel d'un service Web. Les services Web ont été étudiés dans leur dynamique d'interaction avec d'autres services et dans un contexte d'utilisation multiple. En effets, les techniques de coordination, les techniques de composition et l'infrastructure étendue aux protocoles de coordination ont été présentées.

Un intérêt particulier a été attaché à l'étude de la gestion des transactions dans les processus de coordination et lors de la composition des services.

Divers standards de protocoles de coordination (verticaux ou horizontaux) et la panoplie des standards de composition ont été exposés.

Malgré la diversité des langages visant la description et/ou l'exécution des processus métiers complexes, ces langages restent limités quant à l'expression de certains types de contraintes.

Nous estimons, dore et déjà, que pour répondre à la problématique de la prise en comptes des contraintes, lors de la découverte et de la composition des services, les efforts de réflexion doivent se focaliser d'avantage, sur les aspects conceptuels et théoriques. Dans le prochain chapitre, nous entamerons, justement, l'aspect modélisation du comportement externe des services Web.

Chapitre 3

Modélisation du Comportement Externe des Services Web

Introduction

L'infrastructure actuelle des services Web n'offre pas une description assez riche pour permettre une interaction effective entre partenaires, et, un déploiement réel des services Web. En effet, l'interface de services décrite, via WSDL, reste limitée aux aspects fonctionnels et n'offre qu'une base d'interopérabilité. Cependant, dans un environnement réel, les clients n'invoquent pas de simples opérations ni des services isolés. Pour réaliser leurs activités, ou processus métiers (Business Process), ils invoquent un ensemble d'opérations appartenant souvent à des services différents. Ils composent des services.

Il devient, ainsi, impératif de prendre en considération les aspects comportementaux des services Web durant toutes les phases du cycle de vie: Développement, découverte, sélection et interaction. Cette action ne peut être menée à bien que par une modélisation des business process via leurs protocoles.

Dans ce chapitre, nous aborderons dans un premier temps, la notion de ***protocole de service*** (***Business Protocol***) tout en présentant les formalismes existants pour leur représentation. Dans un deuxième temps, nous présenterons un enrichissement des modèles de protocoles par l'introduction d'autres aspects fonctionnels non pris en compte dans le modèle de base, telles que: les contraintes temporelles, transactionnelles et non fonctionnelles telles que: Qualité de service, sécurité, administration. Nous terminerons le chapitre par une analyse des protocoles tout en montrant l'intérêt d'une telle analyse et le besoin d'opérateurs algébriques qui permettront la manipulation des protocoles.

3.1 Notion de Protocole de service

Exécuter un service Web peut impliquer l'invocation de plusieurs opérations et dans un ordre bien déterminé. Par exemple, un client qui veut faire une réservation d'un voyage auprès d'une compagnie aérienne doit procéder par étape: il doit d'abord s'identifier, demander les vols disponibles, choisir un vol, demander les prix, puis, suivant la réponse reçue procéder à la réservation et, enfin soumettre le paiement. Toutes ces opérations sont nécessaires pour la réalisation de la réservation (service) et doivent s'exécuter dans l'ordre précité.

Il est rare que les opérations à exécuter par un service puissent être invoquées aléatoirement. Alors, *le protocole* décrit la séquence des messages échangés et supportés par le service. Le protocole exprime, donc, des contraintes d'ordre sur les opérations.

Comme, les interactions entre le client et le service sont formées par un ensemble d'invocations d'opérations, le terme *conversation* est utilisé pour se référer à des séquences d'opérations qui surgissent entre un client et un fournisseur suite à l'invocation d'un service Web. Pour garantir une conversation correcte et acceptée par un service Web, le fournisseur de service définit préalablement son protocole, auquel les clients potentiels doivent se conformer. Il s'agit du *protocole de coordination* du fournisseur:

« Une fois le protocole de coordination spécifié par le fournisseur du service, il est impératif de le publier dans l'annuaire de service afin que les clients potentiels se rendent compte de cette information et peuvent, par conséquent, développer les stubs clients pouvant interagir correctement avec le service Web ». [4]

Du côté du client, la logique métier interne est aussi soumise à des contraintes d'ordre sur les opérations qu'il veut réaliser. Le client a son propre protocole métier (business Protocol).

Il est évident, que « si deux ou plusieurs services expriment le besoin d'interopérer, leurs protocoles doivent être compatibles» [13].

La figure 3.1 illustre la conversation du processus de réservation précédent entre un client possédant une logique métier interne (Business Protocol) et un fournisseur imposant un protocole de coordination.

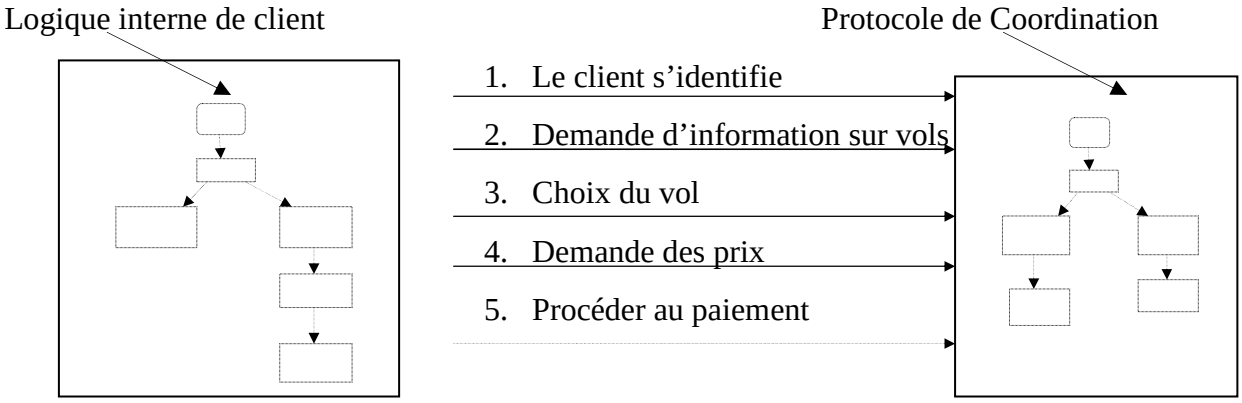


Figure 3.1: Conversation Client-fournisseur conformément au protocole de coordination

3.2 Modélisation des protocoles

Les protocoles nécessitent d'être représentés avec des outils expressifs ou "***modèles***" offrant des abstractions de la conversation correcte que le client doit respecter pour pouvoir interagir avec le service exposé par le fournisseur.

Différents modèles ont été proposés dans la littérature pour représenter les protocoles des services Web. Nous dresserons dans cette section un panorama des modèles existants et nous conclurons par une évaluation comparative basée sur les critères d'expressivité, de manipulation et de richesse sémantique.

3.2.1 Modélisation des protocoles par des automates à états finis déterministes

Un protocole peut être modélisé avec un automate d'état finis déterministe (state machine) où les arcs étiquetés correspondent aux opérations. Les nœuds ou états de l'automate, sont les phases du processus métier. En effet, « les états représentent les différentes phases par lesquels transite le service durant son interaction avec le fournisseur et les transitions sont déclenchées par les messages envoyés par le client au fournisseur ou vice versa. Un message correspond à l'invocation d'une opération d'un service ou à sa réponse » [13]. Dans [13], les messages échangés sont dotés d'une polarité (+,-) pour exprimer s'ils sont en entrée ou en sortie.

La Figure 3.2 donne un exemple de modélisation du protocole du service réservation de places avec des automates d'états finis déterministes.

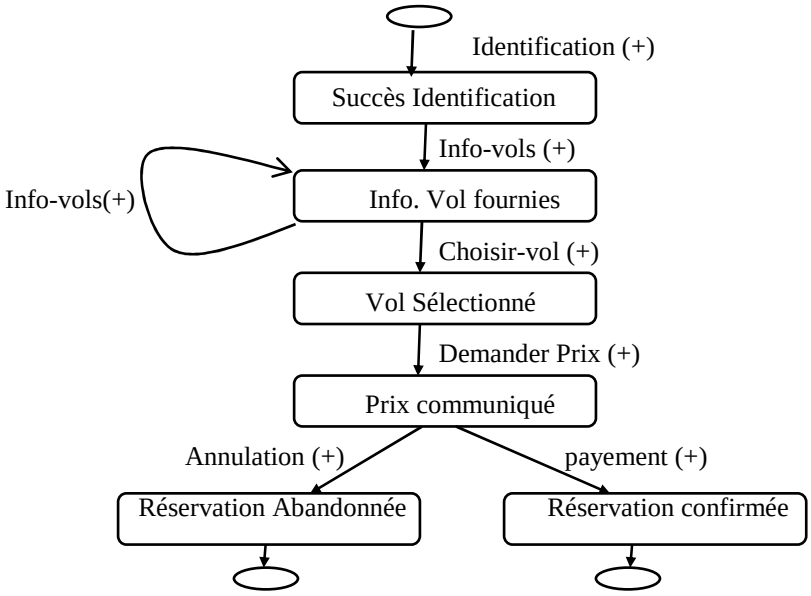


Figure 3.2: Modélisation d'un protocole à l'aide des automates d'états finis.

Il est à signaler que les automates sont déterministes en offrant, ainsi, une unicité de la transition d'un état à un autre suite à la réception d'un message. Le cas contraire, l'automate sera indécis et pourra transiter vers un état imprévisible pour le client. En plus, les transitions sont conditionnées par les messages seulement sans prise en compte des paramètres qu'ils contiennent. Ce modèle décrit les opérations possibles (identification, demander informations sur les vols, choix du vol, demander prix et payement ou annulation) et les séquences de conversations possibles.

Le protocole de réservation de place autorise deux séquences de messages possibles:

1/ Identification →Info-vols→choisir-vol→demander prix→payement.
2/ Identification →Infos-vols→choisir-vol→demander prix→annulation.

3.2.2 Modélisation des protocoles par des Réseaux de PETRI

Un protocole peut être représenté par un réseau de PETRI, où les nœuds correspondent aux phases du protocole et les transitions étiquetées sont les opérations WSDL. Certaines transitions sont dotées de prédicats booléens exprimant le déclenchement ou non de l'opération. Les réseaux de PETRI sont une combinaison des diagrammes d'états et des diagrammes d'activités.

« Très utilisés dans le domaine des communications, gestion des processus concurrents, la synchronisation et le partage de ressources, le modèle des réseaux de PETRI fournit les primitives nécessaires pour la spécification du processus de conversation » [14].

Un des avantages des réseaux de PETRI c'est qu'ils sont bien étudiés et ils sont dotés d'un formalisme très rigoureux avec une disponibilité d'outils de vérification (détection d'inter-blocage, conditions erronées).

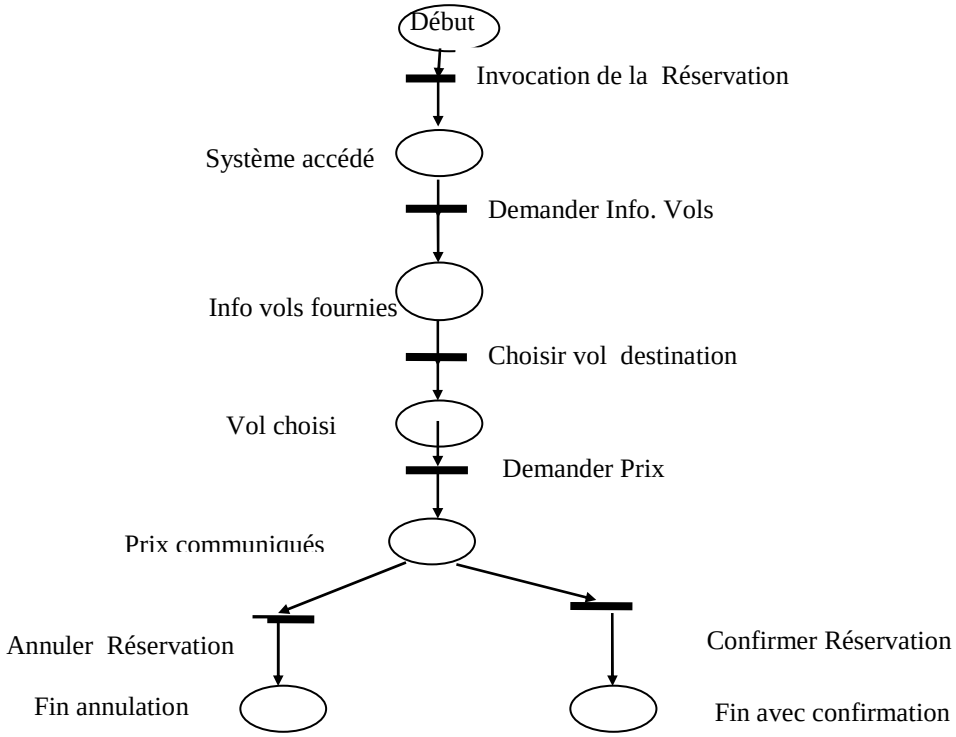


Figure 3.3: Modélisation d'un protocole à l'aide des Réseau de PETRI.

« Les réseaux de Pétri colorés rajoutent des primitives à la version de base pour permettre la définition des types de données et de leur manipulation » [14].

3.2.3 Modélisation des protocoles par des diagrammes de séquences

Ce modèle est basé sur les rôles (colonnes) et les messages (flèches étiquetées) comme ingrédients de base. Quant aux contraintes d'ordre, elles sont schématisées par la chronologie descendante qui exprime le temps.

En supposant que la compagnie tierce possède des vols et des places disponibles, le protocole du service réservation de places de l'exemple précédent est représenté dans la Figure 3.4 avec un diagramme de séquence.

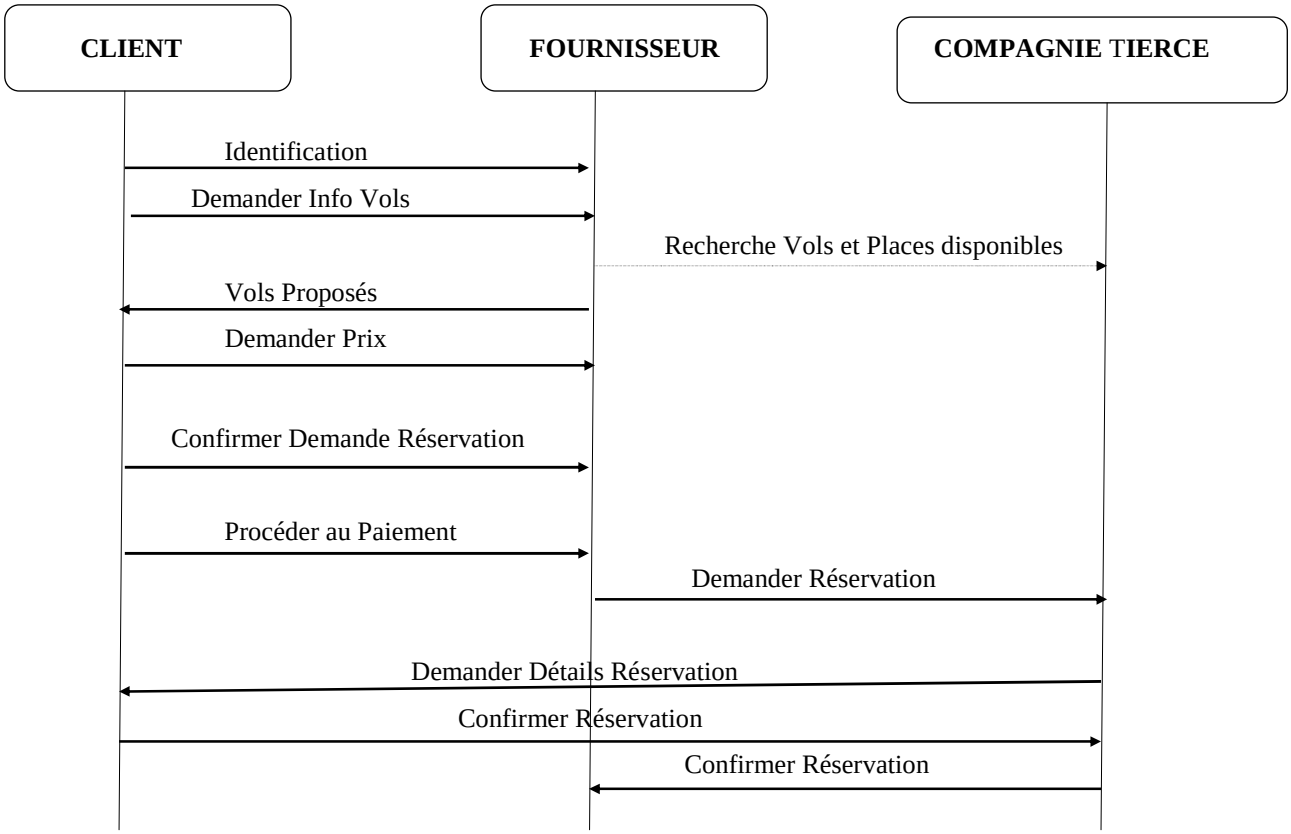


Figure 3.4: Protocole de Réservation modélisé par un Diagramme de Séquence

3.2.4 Modélisation des protocoles de services par les diagrammes d'activités

Le Diagramme de séquence de la figure 3.4 représente le protocole de réservation d'une manière simple et compréhensible. Cependant, il n'offre pas d'outils pour exprimer les séquences parallèles ou des situations d'exécution alternatives. Une solution possible pour traiter les situations alternatives consiste à construire autant de diagramme que de cas possibles. Seulement, elle n'est pratique que si le nombre de situations est limité. Le cas contraire le nombre de diagrammes devient encombrant et complique la compréhension du protocole de coordination.

Les diagrammes d'activités remédient aux insuffisances citées précédemment et permettent de modéliser les protocoles complexes. Les rôles sont représentés en colonnes. Les "activités" correspondent aux messages envoyés par un service Web jouant un rôle particulier vers un autre service Web jouant un rôle différent. Les tests conditionnels induisent des scénarios d'exécution alternatifs. Avec une telle approche, les exécutions alternatives et même parallèles peuvent être modélisées.

La figure 3.5 reprend la description du protocole de réservation de places et mis en évidence les deux séquences "scénarios" possibles en fonction de la disponibilité ou non de places auprès du fournisseur.

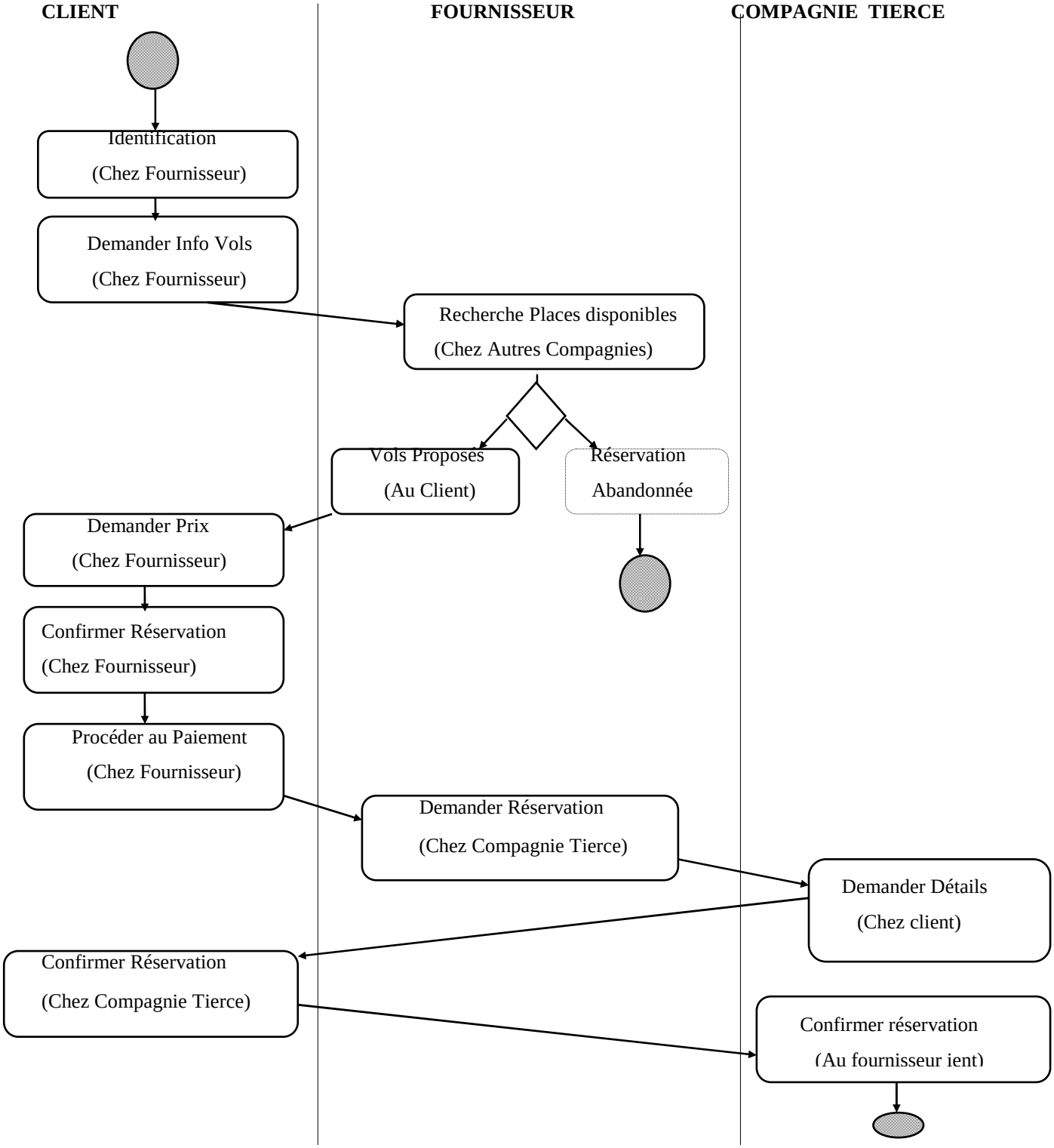


Figure 3.5: Protocole de Réservation modélisé par un Diagramme d'Activité

3.2.5 Le modèle BPEL: BPEL abstract

BPEL est un langage qui supporte la spécification des schémas de composition aussi bien que des protocoles métiers. Il profite de la similarité des techniques de modélisation de ces deux aspects pour fournir un cadre applicatif unique pour spécifier ces deux aspects.
La partie abstraite d'un programme BPEL (L'attribut « AbstactProcess =Yes » dans la définition du processus) permet de définir la séquence de messages échangés par un service

(messages envoyés et reçus) et prend en compte les contraintes d'ordre entre les opérations d'envoi et de réception. En d'autres termes, BPEL peut être utilisé pour définir le comportement externe d'un service (à travers un processus abstrait: **BBEL abstract**):
« Le processus BPEL spécifie l'ordre exact de l'invocation des services Web participant dans la composition. L'invocation peut se faire soit en parallèle soit séquentiellement. L'expression d'un comportement conditionnel est offerte, par exemple, lorsque l'invocation d'un service peut être dépendante du résultat d'une invocation antérieure. La construction des boucles, la déclaration de variables et l'affectation de valeur…etc, sont aussi possibles. De plus, il est possible de combiner toutes ces constructions et de définir des processus métiers complexes d'une manière algorithmique ». [10]
Le modèle d'orchestration de BPEL combine les approches diagrammes d'activités et l'hiérarchie d'activités pour exprimer les business protocols. Il permet de définir des activités structurées qui peuvent être un groupe d'activités de base ou structurées pour spécifier les contraintes d'ordre entre elles. Les activités structurées suivantes peuvent être définies:

- *Sequence:* Ensemble d'activités ordonné à exécuter séquentiellement;
- *Switch*: Exécution conditionnelle d'activités (branchement);
- *Pick:* Exécution alternative liée à l'apparition d'un évènement;
- *While:* Exécution répétitive jusqu'à une certaine condition;
- *Flow*: Exécution parallèle d'un ensemble d'activité.

En résumé, les parties *Abstract* d'un processus BPEL sont de vrais processus métiers qui ont la puissance d'expressivité pour spécifier les protocoles avec différents degrés de détails opérationnels.

3.2.6 Etude comparative des différents modèles de représentation des protocoles

Le tableau 3.1 dresse une synthèse comparative des modèles de protocoles présentés précédemment. Cette comparaison est basée sur un ensemble de critères tels que: la notion d'état, temps, mécanismes de transition (boucles, branchement…), la notion de rôle et la disponibilité des outils de vérification.

Modèles / Critères de Comparaison	Automates	Réseaux de Pétri	Diagramme de séquence	Diagrammes d'activité	Abstract BPEL
Popularité	Oui	Oui	Communauté UML	Communauté UML	Langage BPEL
Existence de Modèle Formel	Oui	Oui	Non	Non	Non
Notion d'état	Oui	Oui	Non	Non	Non
Exécutions Alternative	Oui	Non(*)	Non	Oui	Oui
Exécutions Parallèle	Non	Non	Non	Oui	Oui
Exécutions Itérative	Oui	Oui	Non	Non	Oui
Prise en compte des paramètres des Messages	Non	Oui (*)	Non	Non	Oui
Notion de Temps	Non	Oui (*)	Non	Non	Non
Notion de Rôle	Non	Non	Oui	Oui	Oui
Conversations Multiples	Non	Oui	Oui	Oui	Oui
Existence d'Outils de Vérification Automatiques	Oui	Oui	Non	Non	Oui

Tableau 3.1: Comparaison des modèles de représentation des protocoles

(*): Critère vérifié seulement dans la version étendue des réseaux de Petri colorés (CP-nets).

3.3 Enrichissement de la modélisation des protocoles de services

Malgré les apports offerts par les protocoles pour faciliter l'interaction avec les clients en spécifiant les conversations correctes que ces derniers doivent respecter, ils restent limités et n'expriment pas toutes les caractéristiques d'un service, ni les interactions impliquées dans un environnement services Web. En effet, les protocoles expriment seulement des contraintes sur l'ordre des messages. Pour un déploiement effectif et une large adoption de cette technologie, une extension du modèle de protocole de base, et son augmentation par l'ajout d'informations exprimant les propriétés réelles des services, est plus que nécessaire. Les protocoles doivent être enrichis pour prendre en compte d'autres types de contraintes qui caractérisent les services Web. D'après [15], cet enrichissement permettra la définition des propriétés des services qui serviront de supports aux:

- Humains dans la compréhension des propriétés;
- Machines clientes pour découvrir les services;
- Applications dans l'automatisation de certaines fonctions.

Cet enrichissement sera matérialisé par un modèle qui sera simple pour faciliter son utilisation et utile pour une prise en compte des besoins réels en termes d'expressivité des caractéristiques réelles des services Web.

3.3.1 Identification des abstractions liées aux conversations

Sans s'étaler dans la proposition d'une multitude de modèles qui ne seront compréhensibles ni exploitables que par une communauté spécifique, et vu la nature collaborative des services Web où les services seront exploités par des équipes diversifiées et des compagnies disparates, il sera plus intéressant d'analyser les protocoles et leurs besoins réels afin de déceler les abstractions de haut niveau, et enfin de proposer un méta-modèle qui servira de base pour représenter l'ensemble des caractéristiques des services.
Les abstractions identifiées dans [15], après l'analyse des portails commerciaux, se répartissent en deux classes:

- **Abstractions d'accomplissement: (Completion abstractions)**

Les modèles d'abstraction d'accomplissement décrivent les implications des transitions et leurs effets du point de vue du client (Si un client peut annuler une opération par exemple).
D'autres opérations de *compensation* prévues par le fournisseur de services, permettent d'annuler les effets des opérations invoquées par les clients. Par exemple, les opérations d'annulation d'une commande client ou d'une réservation.
Du côté fournisseur, exécuter des opérations de la part des clients et éventuellement leur compensation s'avère très coûteux en temps et en ressources. Par conséquent, les fournisseurs associent des coûts aux opérations d'annulation. Par exemple, l'annulation d'une réservation induit une retenue sur le montant du billet si la durée de validité du billet est expirée.
L'exécution d'autres types opérations, appelées: *opérations bloquantes de ressource* semble détenir les ressources du client pour une période importante.

- **Abstractions d'activations: (Activation abstractions)**

Les modèles d'abstraction d'activation décrivent le déclenchement d'une transition (par exemple, quand l'opération devrait se produire).

La plupart des transitions se déclenchent suite à une invocation par les clients. Cependant, certaines opérations implicites se déclenchent sans invocations de la part des clients. Un cas typique de ce type de transition c'est les opérations déclenchées par le fournisseur suite à l'expiration d'un délai ou d'une date. Dans d'autres conversations, les transitions exigent la vérification de certaines *conditions* (Exemple: délai particulier pour un type de clients). D'autres propriétés des conversations *à instances spécifiques* varient d'un client à un autre et suivant le type de services commandé. (Exemple: coût d'annulation nul pour les abonnées).

3.3.2 Modélisation des propriétés fonctionnelles des protocoles

Il est nécessaire alors, de définir un ensemble de propriétés qui seront utilisées par les développeurs pour capturer les abstractions d'accomplissement et d'activation citées précédemment.

D'après [15], au moins quatre propriétés sont identifiées.

- Propriété d'activation: Permet de décrire les conditions de déclenchement de la transition.
- Propriété transactionnelle: Spécifie l'effet de la transition sur l'état du client.
- Propriété de blocage: Pour le blocage ou la réservation des ressources.
- Propriété de description: Pour décrire les protocoles afin qu'ils soient compréhensibles par des humains.

Ces propriétés abstraites peuvent dépendre de certains paramètres; définissant ainsi des *contraintes fonctionnelles* sur les protocoles qu'il faut différencier des contraintes non fonctionnelles (voir section 4). On distingue trois types des contraintes fonctionnelles qui seront abordées en détails par la suite: Contraintes temporelles, contraintes transactionnelles et contraintes de pré et post--conditions.

a) Contraintes temporelles

Les protocoles de services expriment la séquence des opérations à exécuter, donc un contexte d'exécution relatif. Or, certaines opérations, de part leur nature intrinsèque, ne peuvent être exécutées que si certaines conditions liées aux temps sont vérifiées. A titre d'exemples:

- Une livraison d'une commande client ne peut se faire que deux jours après son paiement.
- L'annulation d'une réservation n'est possible que 24 Heures avant la date du vol.
- L'expiration du délai de paiement engendre l'annulation de la commande.
- La remise des articles vendus par un magasin n'est possible que pendant trois jours après livraison.

Les opérations suscitées se caractérisent par un contexte d'exécution plus général lié au protocole et non pas aux opérations. Parfois, le contexte est objectif et indépendant du protocole (une date et/ou heure atteintes).

D'après [16] et malgré que la majorité des transitions d'états d'un protocole se déclenche suite à des opérations d'invocation explicites, il existe des situations où les transitions sont dues à des *opérations implicites*. La grande majorité des opérations implicites est due à des effets temporels (expiration de délai).

Donc les contraintes temporelles expriment quand une opération peut ou doit être exécutée.

Pour modéliser ce type de contraintes temporelles dans [16], le concept de *transition temporel* est introduit (Timed Transition). « Une transition temporelle se déclenche automatiquement après qu'un intervalle de temps ait écoulé, puisque la transition est permise, ou quand une certaine date et/ou heures sont atteintes » [16].
Dans [16], pour modéliser ce type de contraintes temporelles, le modèle de protocole des automates d'états finis avec message polarisé (voir section 2.1) est enrichi en rajoutant le concept de temps. Ainsi, un message *m(p),* où *p* est la polarité, est étendu à *m(p,t)* où *t* est le temps qui s'écoule entre le début de l'exécution du service pour les message entrants et le temps nécessaire pour l'émission du message sortant, ou, plus formellement:
m((+),t): Message entrant devant être reçu à l'instant *t* depuis le début de la conversation.
m((-), t): message sortant devant être émis à l'instant *t* depuis le début de la conversation

Cet enrichissement des protocoles de bases donne des protocoles temporels (Timed Protocols) qui définissent des comportements visibles des protocoles de services, aussi bien du point de vue ordre de messages que de celui de la prise en compte des contraintes temporelles.

Cette modélisation permet une analyse plus rigoureuse des protocoles et favorise leur gestion. En effet, la compatibilité de deux protocoles ne peut être considérée que dans son simple aspect structurel (ordre des messages), elle doit prendre en compte les contraintes temporelles par le biais de *trace d'exécution temporelle*.
Exemple de trace d'exécution temporelle: *m1((+),t1).m2((+),t2).m3((-),t3).m4((+),t2)*
En ce sens: « Une trace d'exécution de protocole est conforme à un protocole donné si l'ordre des messages de la trace d'exécution est conforme à celui du protocole en question. De plus, si chaque message qui apparaissant à un instant dans un événement de la trace (entrée/sortie) se trouve conforme aux contraintes temporelles imposées par le protocole » [16]
La prise en compte des contraintes temporelles dans la modélisation des protocoles impose une nouvelle définition de la compatibilité et de la remplaçabilité (replaceability) des protocoles de services en se basant sur les protocoles temporels (Timed-dependant compatibilty, timed dependant replaceability). En ce sens les protocoles seront analysés (compatibilité, équivalence, subsomption et remplaçabilité), non seulement, par rapport à l'ordre des messages, ils doivent obéir aux mêmes contraintes temporelles durant leurs échanges de messages. (voir [16] pour des définitions plus détaillées et formelles).

b) Contraintes transactionnelles

Dans un environnement service Web, et contrairement aux environnements base de données ou Workflow, les opérations sont de longues durées et peuvent bloquer (immobiliser) les ressources pour une durée importante, entravant ainsi, l'exécution du service Web, même pour d'autres clients. Pour pallier à ce problème, la sémantique transactionnelle dans les services Web est garantie par les concepts de *région atomique* et de *compensation*.

- **Définition de la Région atomique**

Une région atomique correspond à une ou plusieurs opérations vérifiant la propriété d'atomicité « tout ou rien ». Elle est décrite par son schéma d'orchestration. Soit que l'ensemble des opérations est exécuté en totalité, soit qu'aucune opération n'est exécutée. L'atomicité est respectée en exécutant le protocole 2PC (Two Phases Commit) pour les

services impliqués dans la transaction. Le protocole WS-Transaction permet de prendre en charge, au niveau du middleware, cet aspect transactionnel, sans que le développeur ne s'implique dans la programmation de ces aspects.

- **Définition de la compensation**

En cas d'échec d'une transaction, pour une opération simple, ou pour une région atomique, les opérations validées doivent être « annulées » pour préserver la cohérence de l'activité déclenchée. Cette « annulation » est assurée en exécutant d'autres opérations permettant de garantir la sémantique de l'échec de l'exécution. (non validation du processus).

Or, dans les services Web, l'échec d'une transaction ne consiste pas en une simple annulation des opérations exécutées et la restauration d'un état cohérent, semblable à celui pratiqué dans le contexte des bases de données. En effet, « du côté du fournisseur de service, exécuter des opérations, et par conséquent leur compensation peut être couteux en temps et en argent ainsi que d'autre ressources. Comme résultat, les fournisseurs affectent une partie du coût associé au client » [15].

- **Principe de la compensation**

En cas d'échec d'une transaction et pour annuler l'exécution partielle d'une région atomique ou d'une opération « Le middleware initie et lance l'exécution du **protocole de compensation** pour les activités validées dans la région atomique afin d'annuler leurs effets » [4].
La compensation est assurée par le middleware (Web services + WS-Transaction) d'une manière transparente en exécutant *un autre protocole défini par le développeur du service Web simple ou composé*.
La question, est alors, de spécifier comment les protocoles de services prennent en compte les contraintes transactionnelles afin de permettre une éventuelle compensation automatique. Autrement, comment modéliser les effets des transactions pour chercher des protocoles de compensation compatibles ?

Il est clair que les propriétés transactionnelles expriment des transitions qui diffèrent par leurs effets sur l'état du client. Dans [15], différentes abstractions qui fournissent l'expressivité nécessaire pour décrire le comportement transactionnel de la conversation du service ont été identifiées, à savoir:

1. **Transitions sans effet (Effectless Transitions):** La transition n'a pas d'effet sur le client. Dans ce cas, aucune opération n'est exécutée pour annuler les effets (pas de compensation).
2. **Transitions définitives (Definite Transition):** Des transitions ayant un effet transactionnel définitif et permanent et donc ne peuvent être compensées.
3. **Transitions compensables (Compensatable Transition**): ont un effet sur l'état du client mais peuvent être annulées en exécutant des transitions de compensation.
4. **Transitions de révélation d'informations (Credential-disclosure Transitions):** Ces transitions n'ont pas d'effet sur l'état du client d'un point de vu transactionnel, mais exige de celui-ci de fournir des informations personnelles. (mot de passe, adresse).
5. **Transitions de Reprise (Retriable transitions):** Offrent aux clients la possibilité de demander aux fournisseurs de ré-exécuter une transition complète sans invoquer une transition de compensation.

c) Contraintes liées aux Pré-conditions et Post-conditions

Les contraintes liées aux pré-conditions et post-conditions expriment les conditions que doit vérifier une opération pour son exécution. Chaque opération est décrite par l'ensemble des données échangées par l'opération, la fonctionnalité de l'opération ainsi que des pré et post-conditions.

« Les pré-conditions et post-conditions constituent, alors, la partie sémantique du contrat d'utilisation du service. Les pré-conditions permettent une meilleur fiabilité d'exécution en contrôlant les parcours algorithmiques et les post-conditions vérifient que les résultats produits par le service sont conformes aux attentes du client sinon ils sont suspects et non renvoyés (une erreur est générée)». [17]

Une pré-condition est une contrainte qui doit être vérifiée pour que l'opération soit lancée. Alors qu'une post-condition est une contrainte qui doit être exécutée à l'issue de l'accomplissement d'une opération:

« C'est une spécification informelle d'un ensemble de contraintes dans le contrat de service. Les pré-conditions permettent d'énoncer les règles à respecter pour le déclenchement de l'opération. Pour chaque pré-condition on précise également la post-condition qui définit les conditions d'émission du résultat de l'opération. La spécification des pré- et post-conditions constitue des éléments importants pour le contrat d'utilisation du service. » [17]

« Les pré- et post conditions sont représentées par des assertions. Une assertion est une formule logique qui renvoie une valeur booléenne (vrai, faux). C'est bien le cas pour les pré- et post-conditions qui sont effectivement des équations logiques qui agissent essentiellement sur les paramètres d'entrée du service et qui se comportent comme des « interrupteurs » binaires laissant passer ou non le flux d'entrée (pré-conditions) et de sortie (post-conditions) » [17].

- **Pré-conditions**

« Une pré-condition permet de spécifier une contrainte prédicative qui doit être vérifiée avant l'appel d'une opération » [18].

Elle renvoie une valeur booléenne « vrai » si elle est vérifiée, « faux » dans le cas contraire.

La règle s'applique sur les valeurs des paramètres du message d'entrée et d'autres données propres au contexte de l'opération. La somme des pré-conditions constitue un ensemble exhaustif des cas de déclenchement de l'opération.

Exemple de pré-condition: Carte de crédit valide pour pouvoir invoquer l'opération « Payement » d'un service Web e-commerce.

Les pré-conditions peuvent être liées à la sécurité (test de nombre de fois d'introduction du mot de passe) ou au processus métier lui-même (ouverture de compte bancaire pour un client plus de 16 ans).

- **Post-conditions**

« Une contrainte qui exprime une condition sur l'état final d'exécution de l'opération (ce qui doit être vrai dans l'état juste après que l'action soit terminée) » [18], elle est associée à une ou plusieurs pré-conditions. Elle décrit les propriétés que le résultat envoyé doit satisfaire (ex: une liste de valeurs possibles).

Vue par le consommateur, l'utilisation de l'opération est décrite par un ensemble de conditions d'usage. Si le consommateur respecte les pré-conditions, alors, le fournisseur

certifie que le résultat renvoyé par l'opération respectera des conditions de restitution ou post-conditions.
Il existe différents modèles pour la représentation formelle des contraintes pré- et post conditions, tels que le langage OCL (Object Constraints Language) ou la représentation via des éléments XML. Il est clair que les pré et post-conditions constituent des contraintes fonctionnelles qu'il faut prendre en considération lors de la découverte et de la composition des services Web.

3.4 Contraintes non fonctionnelles des services Web

Contrairement aux contraintes fonctionnelles exprimées dans la description du service (WSDL), les contraintes non fonctionnelles sont des conditions et des garanties auxquelles le client doit se conformer ou exiger durant les différentes phases du cycle de vie du service (découverte, sélection, invocation, composition). Ces contraintes sont publiées séparément dans les *tmodels*, ou, dans des extensions du registre UDDI. Pour d'autres, elles sont exprimées par des spécifications adéquates. On distingue deux types de contraintes non fonctionnelles: qualité de service, sécurité et confidentialité, qui seront détaillés dans cette section.

3.4.1 Contraintes de Qualité de Services (QoS)

Vu la disponibilité de plusieurs services proposés par différents fournisseurs et offrant des fonctionnalités équivalentes ou similaires, il sera intéressant pour le client de pouvoir choisir le service qui répond au mieux à ses exigences. Ainsi, en partant d'un ensemble de spécifications de critères de qualité de service (Quality of service: QoS), défini préalablement par le client, la question est de pouvoir découvrir et sélectionner le service qui satisfait au mieux ces spécifications. Comme critères de qualité de service, nous pouvons citer:

- **La performance:** Mesure la vitesse de réalisation du service. Elle est définie par le temps de réponse (délais écoulé entre l'invocation et l'accomplissement du service) et les sorties (nombre de requêtes réalisées durant une période);
- **Coût du service:** Correspond aux charges à payer par le client suite à l'invocation du service. Il peut être défini par rapport au nombre de services invoqués ou pour une période donnée.
- **La disponibilité du service:** C'est la probabilité qu'un service soit opérationnel et peut répondre aux requêtes des clients.
- **La fiabilité du service:** C'est la capacité d'un service à répondre aux requêtes des clients sous certaines conditions pour une période spécifique.
- **La capacité:** Nombre de requêtes concurrentes qu'un service peut supporter.

Les critères suscités ne sont pas fournis par les fournisseurs de services dans le registre de publication (UDDI), mais sont plutôt fournis dans les *tmodels* ou dans différentes variantes d'extensions de l'architecture de base des services Web.
L'état de l'art dans le domaine de la découverte des services sur la base de qualité de services, fait ressortir différents problèmes et propositions liés à la modélisation, la représentation et le stockage des informations afférentes à la qualité du service. En effet, la problématique de la recherche de service à base de qualité de service, peut être abordée de deux points de vues

différentes; découverte statique lors de l'implémentation et dynamique, lors de l'exécution du service.
Du fait de l'absence de modèle de spécifications des métriques de qualité de service, ces métriques ne sont pas supportées par la structure actuelle du registre UDDI. Les travaux de recherche convergent vers des remaniements et des extensions de ce registre. En effets, dans [19] et [20], l'auteur propose une catégorisation des *tmodels* de l'UDDI, pouvant être utilisée pour fournir des informations de qualité de service. Les métriques QoS sont représentées par des couples clé-valeur qui seront stockées dans les *tmodel*.
Dans [21], une extension du modèle de base est présentée. Le nouveau modèle propose un *certificateur de qualité* qui vérifie les paramètres QoS annoncés d'un service avant son enregistrement. Ainsi, le client peut vérifier ses critères QoS avec ceux annoncés dans le certificateur, et ce avant l'invocation du service (avant binding). Dans [22], une infrastructure à base d'agents logiciels est combinée avec une ontologie contenant les concepts de qualité de services, permet de sélectionner dynamiquement les services sur la base de leur qualité. Dans cette architecture, la qualité de service est déterminée d'une manière collaborative par les services participants et les agents logiciels. Les données de qualité de service sont collectées puis partagées entre les différents agents. Quant à l'ontologie de qualité de service, elle capture et définie les concepts génériques de qualité de service.
Les travaux les plus intéressants dans la découverte des services basée sur la qualité sont ceux qui prennent en compte les traces et les historiques d'invocation des services. En effet, divers *modèles de réputation* de services ont été proposés dans [23] et [24]. La réputation d'un service est un score public affecté suite à l'utilisation d'un service par les clients. Les scores sont collectés, traités et actualisés après chaque utilisation effective d'un service. Ces modèles offrent plus de possibilités aux clients pour sélectionner les services répondant au mieux aux besoins de qualité et de fiabilité.
Il est à signaler, que dans le cadre de la prise en compte de la qualité de service, lors de leur découverte et leur sélection, différents standards ont été proposés. Parmi ces standards, Web Service Level Agreement (WSLA) qui défini un langage de spécifications des caractéristiques d'un service en termes d'administration et de métriques de qualité de service. Avec ce langage, les clients peuvent exprimer leurs exigences de qualité et les fournisseurs expriment leurs garanties. Par ailleurs, les mesures à entreprendre en cas de déviation par rapport aux exigences sont envisagées. Quand à WS-Policy, c'est une grammaire flexible et extensible qui permet d'exprimer les possibilités, exigences et caractéristiques générales d'un service Web. Ce standard défini un cadre et un modèle pour l'expression de ces propriétés en tant que politiques. WS-Policy représente un ensemble de spécifications qui décrivent les possibilités et les contraintes des politiques de sécurité et de qualité de service.

3.4.2 Contraintes de Sécurité et de confidentialité

La sécurisation des systèmes distribués mettant en œuvre les services Web est inévitable. Non seulement dans le cadre des réseaux internes à une organisation, mais également, dans un cadre Internet totalement ouvert à l'extérieur [25]. En effet, du fait que l'environnement des services Web est ouvert, les échanges de messages entre partenaires à travers le réseau doivent être sécurisés, car, les données circulant sur le réseau peuvent être interceptées, consultées ou modifiées par des entités non autorisées. Cette situation est d'autant plus dangereuse pour les

services Web, parce qu'ils sont auto-descriptifs et leurs données sont échangées sous format XML auto-documentés. Dés lors, il devient primordial de prévoir des mécanismes de sécurité pour les services Web. Malgré l'existence de techniques de sécurité, au niveau connexion (HTTPS, firewalls) celles-ci restent limitées et ne répondent pas aux enjeux réels de sécurité. En effet, les messages échangés transitent sur des couches de transport différentes, circulent entre plusieurs entreprises, déclenchant des traitements métier et étant éventuellement modifiés lors de leur passage. Dans ce contexte, il est impératif de doter l'infrastructure des services Web de techniques de *sécurité au niveau message*, consistant à propager l'information de sécurité avec les messages. Ces informations deviennent des propriétés des messages. D'après [25], on distingue différents enjeux relatifs à la sécurité dans les services Web et pour lesquels des techniques sont proposées, à savoir:

- **Vérification de l'identité de l'émetteur ou du récepteur d'un message**

  Un des besoins de base de la sécurisation des services Web est la vérification par le client de l'identité du serveur, et, réciproquement la vérification par le fournisseur de l'identité du client (*Authentification*), afin de s'assurer que les échanges se font réellement avec l'entité souhaitée. Cette Vérification est réalisée par l'utilisation d'un protocole commun d'identification, par exemple, basé sur une technologie de certificat, de jetons...etc. Dans le cadre des services Web, comme dans celui des applications Web classiques, la nature précise de l'entité ayant invoqué le service (une machine, une application, un utilisateur, une organisation) est exploitée comme paramètre principale d'authentification.

- **Confidentialité des données échangées**

  Les données confidentielles circulant sur le réseau doivent être cryptées afin d'éviter leur interception par des entités malveillantes. Ce cryptage peut porter sur tout le message, ou posséder une granularité plus fine. La lecture des éléments cryptés n'est possible que pour les utilisateurs possédant les clefs de déchiffrement adéquates. L'utilisation du protocole HTTP au dessus du protocole SSL (Secure Socket Layer) ou TLS (Transport Layer Secure) permet de garantir la confidentialité des données. SSL ou TLS sont configurés pour crypter les messages SOAP échangés. D'autres technologies comme IPsec ou le standard WS-Security peuvent être utilisées.

- **Intégrité des données échangées**

  Il est important de s'assurer que les données reçues sont effectivement celles qui ont été envoyées et qu'elles n'ont pas été modifiées par des utilisateurs malveillants, à la volé, lors de leur passage par les divers nœuds du réseau. HTTP est utilisé au dessus de SSL ou TLS pour s'assurer de l'intégrité des données échangées lors de l'interaction services Web. Dans ce cas, la couche SSL/TLS utilise un algorithme MAC (Message Authentification Code) permettant de détecter les modifications des messages lors de leur transit sur le réseau.

- **Autorisation**

  L'invocation d'un service Web déclenche un traitement du côté serveur visant à obtenir et/ou modifier certaines données. Le serveur (fournisseur de service) applique, alors, des politiques d'autorisation visant à restreindre l'exécution du service demandé uniquement

aux utilisateurs autorisés et sous certaines conditions. Un moteur d'autorisation est configuré et activé chez le fournisseur pour obtenir les informations sur le contexte des utilisateurs et vérifier leur autorisation (identité du client, Adresse IP..etc.)

- **Protection contre les attaques**

  Des entités non autorisées peuvent engager des tentatives d'intrusion en profitant des failles dans la sécurité des services Web afin de prendre le contrôle du système en modifiant ou en contrôlant ses données. Il est important, par conséquent, de pouvoir repérer et neutraliser ces tentatives d'intrusion. Le protocole IPsec (IP Security) a été conçu pour éviter ces problèmes.

  Un autre type d'attaque, consiste à rendre inopérant un service en le saturant par un nombre important de demandes. Il est nécessaire de se protéger contre ce type d'attaques dites « Déni de Service » en limitant, par exemple, le nombre d'invocation provenant d'un même utilisateur.

- **Non répudiation**

  Lorsqu'un service Web est invoqué, il peut être nécessaire pour le fournisseur d'être en mesure d'empêcher le client de répudier dans le futur cette invocation, c'est dire de nier l'avoir effectuée. De même, lorsqu'un service Web émet une réponse vers un client, ce dernier peut souhaiter être en mesure de prouver que telle ou telle réponse a bien été émise par le service Web.

  La propriété de non répudiation repose sur la mise en œuvre de techniques de signature numérique sécurisée. Une telle signature est un élément ne pouvant provenir que de l'émetteur et indissociablement liée au message. Des standards de signature numérique existent (par exemple XML signature) et pourront être utilisés dans le cadre des services Web à des fins de non-répudiation.

3.4.3 Les standards de la sécurité des services Web

Différents spécifications ont émergés pour répondre aux problèmes de sécurité des services Web:

WS-Security: C'est une spécification visant à offrir une sécurité au niveau message dans le cadre des services Web lors des échanges de messages SOAP. Son objectif principal est de sécuriser les messages SOAP transitant par plusieurs partenaires [25] (Sécurité end-to-end).
Techniquement WS-Security décrit comment attacher des signatures et des éléments cryptés aux messages SOAP en proposant trois mécanismes:

- Propagation des assertions de sécurité (Permettant l'authentification et l'autorisation);
- Intégrité des messages;
- Confidentialité des messages.

Cependant, WS-security ne spécifie pas une solution de sécurité complète. Il constitue un socle sur lequel des spécifications de plus haut niveau pourront s'appuyer [25].

WS-Security-Policy: Permet de décrire les capacités et les contraintes des politiques de sécurité des différents nœuds prenant part dans les interactions services Web sécurisées (Algorithme de cryptage, règles de confidentialités…)

WS-Trust: Offre un cadre permettant d'établir des relations de confiance directes ou agrégées, permettant la mise en œuvre d'interaction services Web sécurisée.

WS-Privacy: Spécifie un modèle de description de politiques de confidentialité des informations. Un tel modèle permettra, par exemple, d'établir si la politique de confidentialité souhaitée par le client est compatible avec celle appliquée par le service Web.

WS-SecureConversation: Décrit comment un service Web et un client sont mutuellement authentifiés et comment établir les contextes de sécurité mutuels. (Clefs de session, clefs d'authentification de messages…)

WS-Federation: Décrit comment construire des zones de sécurité fédérées à l'aide des techniques décrites par WS-Security, WS-Policy, WS-Trust et WS-SecureConversation.

WS-Autorisation: Décrit comment les politiques d'accès aux services Web sont spécifiées et gérées. La manière dont les assertions de sécurité et comment elles seront interprétées par les moteurs d'autorisations sont décrites en détails.

3.4.4 Synthèse sur les contraintes non fonctionnelles

Les contraintes non-fonctionnelles de qualité de services et de sécurité sont une partie intégrante de l'environnement service Web. Elles constituent des garanties pour un déploiement effectif de cette technologie. La qualité de service répond à des besoins d'opérationnalité et d'efficacité dans un environnement concurrentiel caractérisé par des exigences des clients, de plus en plus accrues. Quant à la sécurité, elle permet de garantir la discrétion nécessaire lors des interactions entre clients et fournisseurs. Certaines techniques de l'infrastructure de sécurité des applications Web classiques peuvent être utilisées dans le cadre de la sécurité, mais elles restent insuffisantes.

L'augmentation de la pile des protocoles des services Web par de nouvelles couches verticales spécifiquement dédiées à la qualité de service et à la sécurité permettra un meilleur déploiement de cette technologie.

3.5 Intérêt de l'analyse des protocoles de services et leur gestion

Dans la section 2 de ce chapitre, nous avons mis en évidence le besoin d'une modélisation des protocoles de service par des modèles expressifs et concis qui fourniront des informations nécessaires aux développeurs pour mieux interagir avec les services. Dans cette section, inspirée de [15], nous reviendrons sur l'intérêt d'une telle modélisation et nous discuterons de la gestion des protocoles, particulièrement des notions de comptabilité, équivalence et de substitution, dans l'objectif de déceler si deux protocoles peuvent interagir correctement en se basant sur leur définition de protocole et si un service peut remplacer un autre. Nous commençons par exposer l'intérêt d'une telle gestion.

3.5.1 Intérêt de la spécification des protocoles et leur analyse

Additionnellement, au fait que la spécification des protocoles permet aux développeurs de construire des applications clientes qui interagissent correctement avec les clients, elle offre d'autres avantages applicatifs liés au développement, au déploiement, à l'exécution et à la gestion des services Web.

- **Génération automatique de code:** Le protocole peut supporter l'implémentation du service en permettant la génération automatique du squelette du programme.
- **Gestion automatique des exceptions:** Les protocoles peuvent offrir le développement d'outils génériques (protocoles indépendants) qui lisent leurs spécifications et vérifient en cours d'exécution, si l'interaction est conforme à la spécification et en envoyant une exception le cas contraire.
- **Analyse pendant le développement:** Durant la phase de développement, les protocoles du client et du fournisseur peuvent être analysés pour identifier les parties conformes des autres qui ne le sont pas. Cette analyse permet de déceler les conversations tolérées par les deux services, et par conséquent offrir la possibilité de faire les modifications adéquates afin d'accroître le degré de compatibilité avec le service désiré.
- **Lien statique et dynamique:** En comparant les définitions des protocoles du client et du fournisseur, le middleware des services Web peut restreindre la sélection des services retournés seulement à ceux qui sont compatibles avec celui du client. Les services non compatibles sont exclus de la sélection, car aucune interaction n'est possible.
- **Conformité:** L'analyse et la gestion des protocoles offrent des possibilités pour décider de la conformité d'un service avec certaines spécifications des standards et consortiums (protocoles verticaux spécifiques à certains domaines).
- **Gestion des versions:** L'environnement des services Web est dynamique et versatile. Des services apparaissent, d'autres subissent des modifications fréquentes et certains peuvent disparaître. L'analyse et la gestion des protocoles donnent la possibilité de localiser, statiquement ou dynamiquement, des protocoles alternatifs équivalents et éventuellement si une nouvelle version d'un protocole est compatible avec le client prévu.
- **Vérification de la conformité du protocole avec son implémentation:** La spécification du protocole peut être exploitée par des outils automatiques pour vérifier que l'implémentation est conforme au protocole. Cette action est réalisée en procédant, dans un premier temps, à la reconstitution du protocole à partir de son implémentation via un générateur de protocole, puis en vérifiant la conformité du protocole généré avec le protocole d'origine.
- **Requêtes à base de protocoles:** L'implémentation d'une base de protocole (Protocol repository) contenant un ensemble de protocoles d'un domaine spécifique peut être exploitée pour rechercher des protocoles particuliers. Une requête ayant comme paramètre le protocole à rechercher (ou certains éléments de sa spécification: messages, opérations) est introduite comme entré et après recherche dans l'annuaire, le (les) protocole (s) compatible (s) sera (seront) fourni(s) en réponse à la requête.

3.5.2 Analyse des protocoles de services

Pour évaluer les points communs et les différences entre deux protocoles, afin de déterminer s'ils peuvent engager des conversations correctes, deux classes d'analyse, ainsi que des opérateurs génériques de manipulation des protocoles sont introduits dans [15]. L'objectif est de pouvoir comparer et manipuler les définitions des protocoles.

a) Analyse de compatibilité de protocoles (Compatibility)

L'analyse de compatibilité des protocoles permet de caractériser si deux services peuvent interagir sur la base de leur spécification de protocoles. L'interaction peut être partielle ou totale. Dans ce contexte, deux types de compatibilité sont distingués.

- **Compatibilité partielle:** (ou simplement compatibilité): Un protocole P_1 est partiellement compatible avec un autre protocole P_2 (notée: *P-comp*(P_1, P_2)), s'il existe des exécutions de P_1 qui peuvent inter-opérer avec P_2. Donc, s'il existe, au moins une conversation qui peut être engagée entre le service supportant le protocole P_1 et le service supportant P_2.
- **Compatibilité totale:** Un protocole P_1 est totalement compatible avec un autre protocole P_2 (notée: F-*comp*(P_1, P_2)), si toutes les exécutions de P_1 peuvent interopérer avec P_2. Donc, n'importe quelle conversation générée par P_1 est comprise *par* P_2.

La relation de compatibilité partielle est symétrique, contrairement à la compatibilité totale qui est non symétrique.

Le concept de compatibilité est très important pour les développeurs; d'un côté, les services ayant des protocoles incompatibles (totalement ou partiellement) ne sont pas pris en considération dans la phase de sélection, en plus, les développeurs doivent déterminer les conversations permises dans le cas de compatibilité partielle.

b) Analyse de Remplaçabilité des protocoles (Replaceabilty)

L'analyse de remplaçabilité des services permet de vérifier si deux services sont fonctionnellement équivalents ; s'ils supportent la même séquence de conversation. Elle est utile dans le cas de défaillance d'un service afin de chercher un autre service équivalent pour le remplacer. Elle peut servir lors de la recherche d'un service ayant une qualité de service meilleure mais avec les mêmes fonctionnalités, ainsi que pour tester si une nouvelle version proposée est toujours équivalente ou si un nouveau service peut supporter les conversations exigées par des standards. Différentes classes de remplaçabilité ont été identifiées dans [15], à savoir:

- **Equivalence de protocoles**: Deux protocoles métiers P_1 et P_2 sont équivalents (notée: Equiv(P_1, P_2)), s'il supportent le même ensemble de conversations. Donc, s'ils peuvent être utilisés interchangeablement dans n'importe quel contexte et ce changement est transparent au client.
- **Subsomption de protocoles**: Un protocole P_2 peut être subsumé par un autre protocole P_1 (notée: Subs$(P2, P_1)$), si P_1 supporte au moins toutes les conversations

que P_2 supporte. Dans ce cas, le protocole P_1 peut être utilisé d'une manière transparente à la place de P_2, mais la réciproque n'est pas vraie.

- **Remplaçabilité avec respect d'un protocole client**: Un protocole P_1 peut remplacer un autre protocole P_2 avec respect du protocole client P_C (notée: $Repl_{[Pc]}(P_1, P_2)$), si P_1 se comporte comme P_2 lors de son interaction avec le protocole client spécifique P_C.
- **Remplaçabilité avec respect d'un rôle d'interaction**: Soit P_R un protocole métier. Un protocole P_1 peut remplacer un autre protocole P_2 avec respect au rôle du protocole P_R (notée: $Repl\text{-}Role_{[PR]}(P_2, P_1)$), si P_1 se comporte comme P_2 quant P_2 se comporte Comme P_R.

Cette dernière classe de remplaçabilité permet d'identifier des exécutions du protocole P_2 qui peuvent être remplacées par le protocole P_1, même lorsque P_1 et P_2 ne sont pas comparables par rapport aux classes citées précédemment.

Comme pour l'analyse de compatibilité, l'analyse de rempalçabilité met en exergue le besoin d'un ensemble d'opérateurs qui permettront, non seulement, de tester l'équivalence, la subsomption ou les différents types de remplaçabilté, mais permettront aussi de dire si deux protocoles sont non équivalents ou s'ils ne peuvent être subsumés.

3.5.3 Gestion des protocoles et Algèbre de protocoles

Pour permettre la gestion et la manipulation des protocoles, différents opérateurs ont été proposés dans [15]. Le but de ces opérateurs est de déceler les ressemblances et les différences entre protocoles en entrée, offrant ainsi un mécanisme de calcul des interactions possibles entre deux protocoles.

- **Composition compatible:** Cet opérateur permet de caractériser les interactions possibles entre deux protocoles d'un fournisseur et d'un client (notée $||^C$), il a en entrée deux protocoles P_1 et P_2 et en sortie le protocole $P=P_1||^C P_2$ qui décrit l'ensemble des conversations possibles. Le protocole en sortie décrit l'arbre des interactions complètes (d'un état initial à un état final) entre les protocoles en entrée.

  A signaler que la polarité du protocole résultat est nulle pour tous les messages, car les messages en entrée d'un protocole sont consommés en sortie par l'autre protocole. Les états du protocole résultat sont construits par combinaison des états des protocoles en entrée.
- **Intersection:** Consiste à rechercher les aspects communs entre deux protocoles. L'opérateur d'intersection permet de calculer le plus grand ensemble de parties commune entre deux protocoles (notée $||^I$), il a en entrée deux protocoles P_1 et P_2 et en sortie le protocole $P=P_1||^I P_2$ qui décrit l'ensemble des exécutions complètes qui sont communes entre les deux protocoles en entrée. Le protocole résultant est composé des messages appartenant aux deux protocoles et ayant le même nom et la même polarité.
- **Différence:** Permet de déceler les différences entre protocoles, cet opérateur (notée $||^D$) permet de dresser les parties qui appartiennent au premier protocole et pas au deuxième. Il a en entrée deux protocoles P_1 et P_2 et en sortie le protocole $P=P_1||^D P_2$ qui décrit l'ensemble des exécutions complètes qui appartiennent à P_1 mais pas à P_2. Le

protocole résultant est composé des chemins complets d'exécution de P_1 qui ne sont as conforme avec P_2.

- **Projection:** La projection d'un protocole compatible ($P_1 \|^C P_2$) par rapport à un protocole participant P_1 (notée $[P_1 \|^C P_2]_{P1}$), Permet d'identifier la partie du protocole P_1 qui peut interagir correctement avec P_2.

  Alors que la composition compatible permet de caractériser les interactions possibles entre deux protocoles chacun définissant le comportement d'un service jouant un rôle dans la collaboration, la projection permet d'extraire le protocole qui définie le rôle que le service joue dans la collaboration.

3.6 Conclusion

Dans ce chapitre, nous avons mis en relief le besoin d'un enrichissement de la description des services Web par des protocoles qui décrivent leurs comportements externes. Différents modèles de représentations ont été exposés et comparés d'un point de vue expressivité et opérationnalité. Les différents types de contraintes fonctionnelles et non fonctionnelles ont été étudiés et un intérêt particulier a été attaché aux contraintes transactionnelles. Nous avons met en évidence le besoin de rehausser la description des protocoles de services par la prise en compte des diverses contraintes. Pour atteindre cet objectif, différentes abstractions de haut niveau ont été identifiées. Une analyse des protocoles a été présentée et son utilité a été justifiée, ouvrant ainsi la voie à une gestion des protocoles par des opérateurs génériques en vue de leur manipulation et d'une calculabilité des protocoles qui consolidera la technologie existante par des modèles formels.

Partie II

Contribution

Chapitre 4

Enrichissement de la Description du Comportement Externe des Services Web par les Contraintes Transactionnelles

Introduction

L'infrastructure des services Web offre un modèle et un environnement de programmation distribués. Ce modèle donne la possibilité aux entreprises à une intégration intra et interentreprises des applications exposées comme services. Cependant, ces entreprises éprouvent un besoin réel pour assurer une gestion fiable et efficace des transactions lors de l'interaction avec les services dispersés dans l'environnement ouvert. En effet, différentes causes telles que les pannes matérielles, pannes de ressources (services ou données non existants) et panne de programmes (comportement imprévisible), peuvent entraver l'achèvement d'une interaction avec un service Web donné. Parfois, les protocoles client sont incompatibles avec le modèle d'exécution prévu. Dans d'autres situations les transactions sont simplement abandonnées par les clients.

Bien que l'infrastructure actuelle des services Web (SOAP, WSDL, UDDI) ait été enrichie par des spécifications telles que les Framework: WS-Coordination [5] et WS-Transaction [6], nécessaires à la gestion des transactions au niveau du Middleware, la gestion des transactions au niveau des protocoles de services revêt un aspect critique qui nécessite une attention particulière.

Les transactions dans les services Web différent par leurs effets et leurs processus d'annulation de celles liées aux bases de données, à cause de leur coût excessif et leur durée relativement longue. Dès lors, les fournisseurs des services procèdent, plutôt à une *compensation des transactions* (voir Chapitre 3, section 3.3.2.b). Cette compensation est assurée par le middleware (Web services+WS-transaction) d'une manière transparente en exécutant *un autre protocole prédéfini par le développeur du service Web.*

Le problème consiste, alors, à aborder une réflexion sur les questions suivantes: Comment décrire et modéliser les effets des transactions ? Comment modéliser les protocoles de services en prenant en compte les contraintes transactionnelles et leurs effets afin de permettre une éventuelle compensation en cas d'annulation ? Quels opérateurs de manipulation de protocoles sont utiles pour gérer les protocoles et leurs protocoles de compensation ? Quel sera l'impact de l'injection des contraintes liées aux transactions sur la gestion de la compatibilité et de l'équivalence des services Web ?

A notre connaissance, peu de travaux existent dans ce contexte et cette problématique exige un effort de recherche approfondi. Les modèles et les solutions à proposer contribueront à la bonne maitrise de la particularité des transactions, au niveau du protocole de service dans les services Web. Ils consolideront, par les éventuelles implémentations, à une large adoption de cette technologie.

4.1 Les motivations

Dans cette section, nous allons exposer les motivations qui ont incité ce travail de recherche.

4.1.1 Vérification de la compatibilité des protocoles de services

La compatibilité de deux protocoles permet de tester s'ils peuvent interagir correctement et de pouvoir engager des conversations entre eux. La définition de la compatibilité (partielle ou totale) telle que décrite dans [15] restreint le critère de test de compatibilité à l'ordre des

opérations et à la polarité des messages. Elle n'est abordée qu'en termes de messages échangés. Bien que, dans [16] cette définition ait été étendue pour prendre en compte les contraintes temporelles liées aux messages (délai, date, heure), la compatibilité de deux protocoles ne peut être abordée sans la prise en considération des effets des opérations. La Figure 4.1, où les protocoles sont modélisés avec des automates d'états finis, illustre un scénario de test de compatibilité.

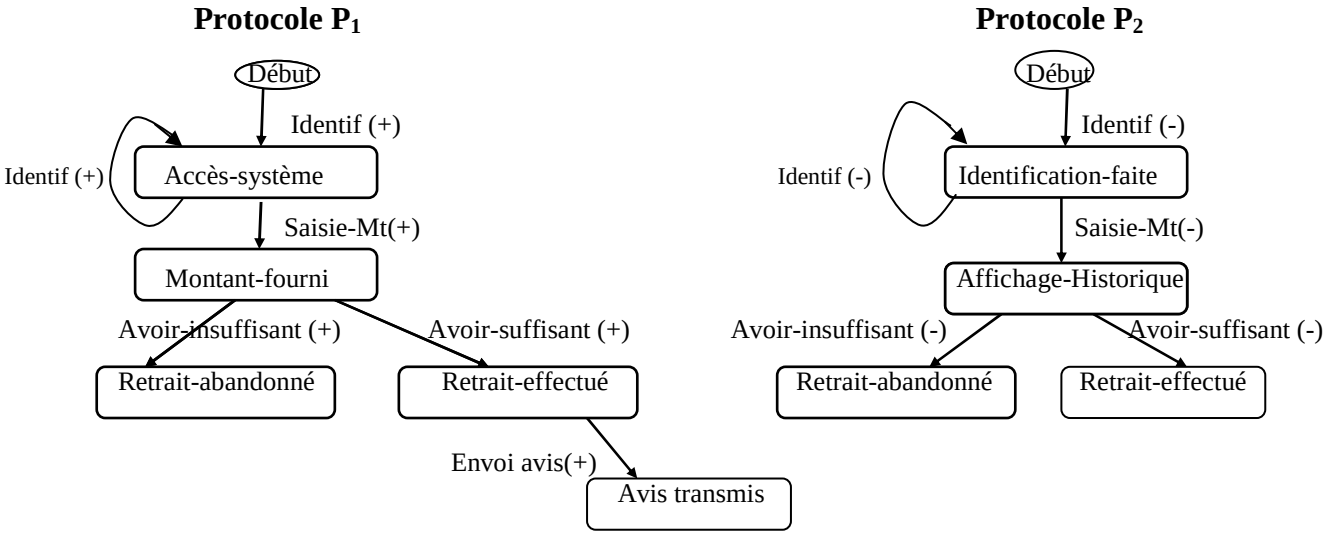


Figure 4.1 Deux protocoles partiellement compatibles sans considération des contraintes transactionnelles

Le chemin d'exécution (***Début.Identif.Identif.saisie-Mt.Avoir-insuffisant)*** est une conversation correcte et supportée par les deux protocoles P_1 et P_2, car la séquence de messages (***Identif(+).Identif(+).saisie-Mt(+).Avoir-insuffisant(+))*** de P_1 est supportée par le protocole P_2. Donc les deux protocoles sont partiellement compatibles.

Cependant, cette compatibilité, même partielle, n'est pas réelle du fait que P_1 et P_2, lors de leurs transitions d'un état à un autre, invoquent des *opérations différentes par les effets qu'elles engendrent*. Par exemple, l'opération déclenchée par le message ***Avoir-insuffisant(+)*** de P_1 peut avoir comme effets: Compte client crédité, compte fournisseur débité, envoi avis, Bonus fidélisation. Alors que l'opération correspondante de P_2 peut avoir d'autres effets: Envoie historique des trois derniers retraits et solde, compte fournisseur débité, Bonus fidélisation. De ce fait, les protocoles P_1 et P_2 auront des protocoles de compensation différents en cas d'annulation des réservations.

Les conséquences d'un tel constat sont très importantes d'un point de vue:

- **Modélisation:** Les effets des transactions sont des caractéristiques intrinsèques des protocoles de services. Ils doivent être représentés d'une manière à permettre leur analyse et leur traitement.
- **Gestion des transactions et de la compensation:** Comment garantir que les protocoles de compensation sont conséquents avec les protocoles abandonnés. Ils doivent défaire sémantiquement les effets observés du coté client.
- **Gestion des protocoles:** La manipulation des protocoles (compatibilité, substitution, équivalence…) doit tenir compte des aspects transactionnels liés aux opérations et à leurs effets.

4.1.2 Inférer les propriétés transactionnelles pour les scénarios existants

Les services Web existants sont implantés via le langage BPEL qui constitue un standard de facto; suite à sont adoption par les industriels du domaine et les consortiums. Ce langage offre un moyen pour décrire le service en tant que programme exécutable aussi bien que pour décrire son comportement (Partie abstact).
Vu la disponibilité de programmes BPEL, il sera opportun de pouvoir extraire les propriétés transactionnelles de ces programmes en vue de leur analyse et de leur manipulation. En effets, les entreprises ayant déjà investi dans la technologie des services Web sont propriétaires d'un patrimoine considérable de programmes BPEL, qu'il faut faire évoluer par l'inférence et l'intégration des propriétés transactionnelles.
Dans cette perspective, un programme BPEL est analysé afin d'extraire ses propriétés transactionnelles. Les éléments « s*cope* » définissant les activités d'un programme, peuvent être isolés (Attribut *isolated*= « Yes ») pour contrôler l'accès concurrent aux ressource partagées donc, ils expriment une transaction. Par ailleurs, les éléments de gestion des erreurs et des transactions d'un programme BPEL tels que:

<compensate> ,<compensate scopes> et <compensation Handler>

Constituent des blocs d'activités liées aux propriétés transactionnelles qu'il est opportun de récupérer dans la perspective de leur modélisation et de leur traitement.
L'objectif est de pouvoir extraire toutes les propriétés liées aux transactions à partir des programmes existants, de pouvoir les modéliser, de modéliser aussi leurs protocoles de compensation et vérifier leur consistance en termes d'effets avec le protocole à compenser. Cette action, offrira un cadre favorable à la vérification de la cohérence d'un protocole avec son protocole de compensation (ou d'autres protocoles) et permettra une gestion (compatibilité, équivalence, substitution…) des protocoles métiers avec prise en compte de leurs contraintes transactionnelles.

4.1.3 Vérifier la consistance d'un protocole existant (outils de conception)

En cas d'échec d'une transaction, il faut exécuter une autre activité de compensation ou ***protocole de compensation*** afin d'annuler les effets engendrés.
Le protocole de compensation est différent de la simple annulation des effets générés par le protocole déclencheur ; c'est un autre protocole métier. En effet, certains fournisseurs affectent une partie de la charge liée aux transactions, qui sont de longue durée et coûteuses en ressources, aux clients. A titre d'exemple, une compagnie aérienne pénalise les clients qui annulent leur réservation en retenant un taux du montant du billet qui dépend de la classe du voyage et de la date d'annulation.
Le problème consiste, alors, à vérifier si le protocole de compensation est consistant avec le protocole déclencheur ? Autrement, est-ce que la compensation garantie une « inversion » des effets ou « *Une annulation sémantique »* du protocole déclencheur ?
La modélisation des effets des transactions, ainsi que la manipulation des protocoles de services avec prise en compte des contraintes transactionnelles établira une assise théorique solide pour vérifier la consistance d'un protocole de compensation avec son protocole déclencheur. Les environnements logiciels adéquats offriront aux développeurs de services

les outils de conception nécessaires pour les décharger de certaines tâches et tests inutiles liés à la vérification de la consistance des protocoles.

4.1.4 Vérification de la conformité d'un programme avec un évènement externe

L'extraction de la spécification du protocole à partir de son implémentation dans un langage (BPEL), peut être exploitée par des outils automatiques pour vérifier la conformité des messages en entrée avec le protocole de service extrait à partir du programme. Il s'agira de tester qu'une ou plusieurs activités décrites dans le programme sont conformes aux messages reçus.

Dans un premier temps, la partie abstract d'un programme BPEL, décrivant le comportement externe du service et qui représente la chorégraphie du service, est traduite en protocole qui sera modélisé par des automates à états finis intégrant les propriétés transactionnelles. En second lieu, les messages reçus correspondent aux transitions entre les états de l'automate sont représentés avec le même formalisme. L'intérêt est de pouvoir tester, si une ou plusieurs parties du protocole, spécifiées préalablement, sont conformes aux messages reçus et quels sont les effets observés du côté client ? Quels seront les effets sur le client après exécution du protocole de compensation ?
La représentation des propriétés transactionnelles des protocoles et la proposition d'opérateurs de manipulation prenant en compte ces propriétés, fourniront une infrastructure pour vérifier la conformité des protocoles avec les messages reçus et faciliteront par conséquent le processus de développement.

4.2 Motivations par quelques scénarios réels

Dans cette section nous présentons deux scénarios réels motivant notre travail et nous discuterons de l'intérêt de la modélisation des effets transactionnels et de leurs effets de compensations. Les scénarios sont relatifs au protocole de réservation de places au niveau d'une compagnie aérienne décrit par sont protocole de service suivant:

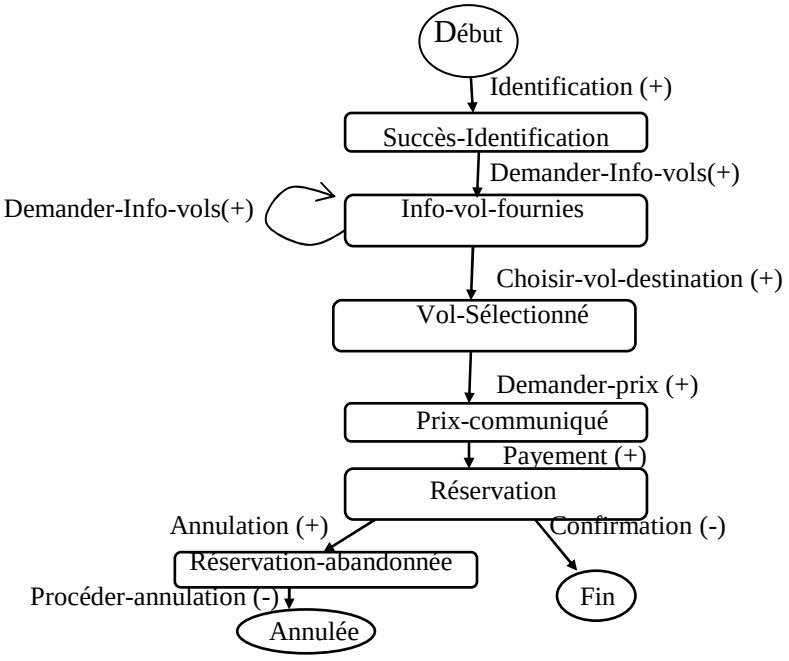


Figure 4.2: Protocole (P) de réservation de places.

Conformément au modèle de représentation des protocoles par des automates à états finis déterministes, les états représentent les phases par lesquels transite le protocole et les messages correspondent aux opérations. On observe que l'automate est une alternance de message et d'opérations où les messages sont dotés de polarité. La polarité (+) indique que le message est en entrée et (-) pour les messages en sortie suivant le modèle proposé dans [13].

Les chemins d'exécutions complètes (d'un état initial à état final) possibles du protocole de réservation sont:

> **Chemin a**: *Début.* ***Identification(+).*** *Succès-Identification.* ***Demander-info-vols(+).*** *Info-vol-fournies.* ***Demander-info-vols(+). Choisir-vol-destination(+).*** *Vol-sélectionné.* ***Demander-prix(+).*** *Prix-communiqué.* ***Payement(+).*** *Réservation.* ***Confirmation(-).*** *Fin*

Exprimant une réservation complète avec confirmation et payement par le client.

> **Chemin b**: *Début.* ***Identification(+).*** *Succès-Identification.* ***Demander-info-vols(+).*** *Info-vol-fournies.* ***Demander-info-vols(+). Choisir-vol-destination(+).*** *Vol-sélectionné.* ***Demander-prix(+).*** *Prix-communiqué.* ***Payement(+).*** *Réservation.* ***Annulation(+).****Réservation-abandonnée.* ***Procéder-annulation(-).*** *Annulée.*

Exprimant l'annulation de la réservation par le client.

A signaler que les transitions ***Confirmation()*** de l'état ***Réservation*** vers l'état ***Fin*** ainsi que ***procéder-annulation*** de l'état ***Réservation-abandonnée*** vers l'état ***Annulée*** sont des transactions *implicites* et sont déclenchées par le fournisseur sans l'attente de messages. Dans ce cas les messages sont dotés de polarités (-) exprimant le faits qu'ils sont en sortie (Produits par le fournisseur).

Scénario 1: Modéliser les effets transactionnels pour une éventuelle compensation.

Dans le cas ou le client annule sa réservation (chemin d'exécution (b)), le fournisseur doit lancer une activité de compensation. Il ne s'agit pas d'une simple annulation de la transaction. En effet, ce dernier va affecter des charges liées à l'immobilisation des ressources par le client. Ces charges vont dépendre de la date d'annulation, de la classe de réservation…etc. En d'autres termes, le fournisseur va exécuter un autre protocole pour compenser le protocole de réservation. Soit P_C le protocole de compensation de ***P,*** il décrit la logique métier du fournisseur dans le cas d'une annulation par le client.

Pour élaborer son protocole de compensation P_C, le fournisseur doit tout d'abord, identifier les effets de ce protocole sur le client, donc les conséquences ou les changements qu'apportera l'opération d'annulation de la réservation sur l'état du client.

A titre d'exemple le protocole de compensation peut être déclenché par le fournisseur suite aux évènements:
- Annulation explicite par le client;
- Date ou heure du vol dépassée ;

Quant aux effets de l'annulation ils peuvent être:
- Retenue de 3% du montant du billet pour une annulation explicite avant la date du vol ;
- Débiter le compte du client par le montant de remboursement.
- Envoi d'une lettre d'information au client.

Dans ce contexte, une modélisation des effets des transactions permettra d'identifier les événements à prendre en compte et les actions à déclencher; donc de construire le protocole de compensation et de s'assurer qu'il permettra, effectivement, d'annuler les effets du protocole déclencheur.

<u>Scénario 2:</u> Substitution d'un protocole par un protocole équivalent et produisant les mêmes effets

Soit $\boldsymbol{P_C}$ (Figure 4.3), le protocole de compensation de ***P,*** il décrit la logique métier du fournisseur pour la compensation d'une annulation de la réservation d'un client.

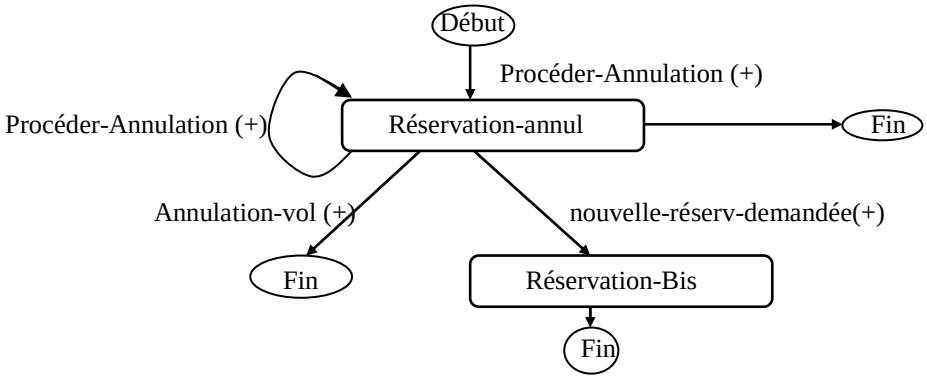


Figure 4.3: Protocole (*Pc*) pour la compensation d'une réservation.

Le message *annulation-vol* permet d'annuler complètement le vol par le fournisseur dans le cas où le nombre de places réservées est limité (Programmation du vol suite à des réservations, puis son annulation suite à l'annulation considérable des réservations). Alors que, la transition à l'état *Réservation-Bis* est déclenchée par le message: *nouvelle-réserv-demandée(+)* qui est spécifique à la logique du fournisseur. Par exemple, se message aura pour effets de ne pas permettre une nouvelle réservation d'un client ayant déjà fait un nombre d'annulation considérable. Il est évident qu'elle reflète une logique métier de chaque fournisseur et diffère d'un fournisseur à un autre.

Dans une situation de composition de service où P_1 contribue dans la composition et par la suite se trouve dans une situation de défaillance, il sera intéressant de pouvoir chercher d'autres protocoles équivalents pouvant remplacer P_1. Autrement, quels seront les services pouvant substituer $\boldsymbol{P_1}$? Le concept d'équivalence est étendu dans ce cas, à l'équivalence des effets des transactions et aux effets de la compensation.

La modélisation des propriétés transactionnelles et de leurs effets permettra de répondre à cette question, en offrant des modèles sémantiquement riches ainsi que des opérateurs de manipulation des protocoles (équivalence, compatibilité,..) qui prendront en charge les effets des messages.

4.3 Etude comparative de la gestion des transactions dans les protocoles des services Web

Dans cette section, nous nous attachons à dresser une étude comparative de la prise en compte des contraintes transactionnelles dans les protocoles des services Web.
En dépit des avancés réalisées dans le cadre de la gestion des transactions au niveau middleware et ayant abouti aux consensus autour des spécifications, largement adoptées, par la communauté des chercheurs et les industriels activant dans la technologie des services Web (WS-Transaction, WS-Atomic Transaction, WS-Business Activity), le domaine de la gestion des transactions au niveau protocole reste faiblement exploré par ces acteurs.

Dans ce qui suit, un panorama des travaux réalisés au sujet de la gestion des transactions au niveau protocole est exposé. Nous terminerons la section par une discussion d'évaluation des travaux réalisés dans ce contexte et en dressant une analyse comparative des approches proposées.

L'analyse des spécifications existantes pour la gestion des transactions au niveau protocole, a dégagé quatre approches qui traitent d'une manière plus ou moins rigoureuse cet aspect. Nous les explorons dans ce qui suit.

4.3.1 Les langages de modélisation des protocoles

Différentes extensions au standard WSDL ont été proposées pour modéliser les protocoles des services. Les langages WSFL et XLANG apportent des améliorations considérables en offrant des structures de composition et de coordination des services basés sur des règles. Ceci est réalisé par la spécification d'un ordre d'exécution des opérations, la définition de leur interdépendance et la gestion des exceptions. Cependant, aucun modèle n'est fourni pour la gestion des transactions distribuées. Par ailleurs, la compensation des transactions n'est abordée que d'un point de vue traitement de flux de données et n'est considérée que par la manipulation des données en sortie.

Ces deux langages ne traitent les transactions et la compensation que d'un point de vue manipulation de données. Cette façon de faire reste faiblement efficace et exige un effort de programmation supplémentaire, particulièrement dans le cas d'une annulation d'une transaction par le client. Par ailleurs, la manipulation des données seules, ne garantie aucune cohérence du processus de compensation des effets avec ceux d'un protocole déclenché. Cette situation est de plus en plus compliquée si les protocoles invoqués sont relatifs à des services composés.

4.3.2 Les protocoles de transaction sur le Web (Web Transactions Protocols)

L'infrastructure actuelle des services Web ne fournit pas de modèle pour l'exécution fiable d'un ensemble de services. Les spécifications proposées fournissent uniquement un modèle pour la coordination automatique de services basé sur la définition d'un ensemble de messages (opérations) transactionnels. Pour contourner le problème de la gestion des transactions longues, ces spécifications introduisent un nouveau concept des transactions remettant en cause les propriétés ACID (Atomicité, Cohérence, Isolation et Durabilité) par leur relaxation. L'ensemble des spécifications proposées n'aborde ni la notion d'effet transactionnel ni celui de la gestion de la compensation par rapport à ces effets. Les deux seuls protocoles ayant abordé cette problématique d'une manière plus ou moins précise sont: Business Transaction Protocol (BTP) et Tentative Hold Protocol (THP).

- **Business Transaction Protocol (BTP) d' OASIS**

BTP est une spécification d'OASIS [26], basée sur l'échange de messages XML et permettant la coordination des tâches de plusieurs participants autonomes (services). Il définie un protocole d'échange et de négociation à deux phases (interaction à deux phases) entre le client et le fournisseur pour s'assurer que les deux parties sont d'accord sur les résultats de la transaction. Afin de garantir la cohérence des résultats d'exécution de l'application composée, BTP relaxe les propriétés ACID en définissant deux types d'activité: *BTP-atom* et

BTP-cohésion. Cette cohérence est assurée par la confirmation ou l'annulation d'un processus complet exprimant la coordination d'activités des participants. Un deuxième mécanisme offert pour assurer la cohérence est la notion de *cohésion* qui consiste à sélectionner la tâche à confirmer.

BTP est basé sur la notion de *contrat entre participants*. Il coordonne les changements d'états causés par les échanges de massages des applications participantes et vérifie la cohérence de l'état résultant avec les contrats. Les effets des transactions sont abordés suivant trois axes: effets prévisionnels, effets finaux et contre-effets, qui doivent être spécifiés et visibles dans les contrats des différents participants.

En résumé BTP offre des mécanismes de manipulation des transactions dans le contexte de la coordination des services entre divers participants, mais il reste limité du fait que la spécification des effets et des contre-effets reste manuelle est spécifique à chaque coordination engagée, en plus, les contrats sont renégociés à chaque nouvelle composition. Par ailleurs, aucun modèle formel garantissant la cohérence des contre-effets avec les effets n'est présenté. Ceci restreint la manipulation des effets et engendre des efforts d'analyse, de conception et de programmation considérables.

- **Tentative Hold Protocol (THP) de W3C**

Tentative Hold Protocol (THP) [27], est un framework ouvert, à couplage faible et basé sur les échanges de messages entre activités en donnant la priorité à la transaction courante. Son objectif est de faciliter la coordination des transactions métier multiples.
Pour gérer les transactions dans un environnement ouvert, THP propose trois concepts clés: *tentative, non-blocking holds* et *reservation* pour l'affectation des ressources. Il vise à minimiser l'exécution des protocoles d'annulation par l'affectation préalable des ressources en cas de demandes par les clients. Ceci à pour bénéfice de minimiser les annulations et de donner une vision réelle sur les données aux clients. En effet, les ressources (Produits, matières, places…) ne sont réellement affectées que suite à une validation définitive par les clients qui vérifient les conditions relatives à l'accomplissement de la transaction. A titre d'exemple, un produit A n'est acheté auprès d'un fournisseur X que si un autre produit B est disponible chez un autre fournisseur Y. Une autre conséquence, est la minimisation de l'exécution des protocoles de compensation, du fait que les annulations sont peu fréquentes. Les effets des transactions ne sont observables dans le monde du client que suite à une validation engendrée par la vérification de certaines conditions.
THP offre un cadre de gestion des transactions dans un environnement d'échanges multiples basé sur la réservation et l'affectation des ressources de la transaction courante, suite à la vérification des conditions conjoncturelles de disponibilité d'autres ressources y afférentes pour l'accomplissement d'une activité métier quelconque. Cette approche pour aborder les transactions reflète une situation réelle et gère des données fiables pour les différents intervenants. Les processus d'annulation et de compensation sont, alors, réduits d'une façon significative.

En définitif, le protocole THP se base sur un ensemble de garde-fous garantissant une éventuelle terminaison de l'exécution du protocole. Or, dans la réalité les clients n'ont pas une vision précise de leurs besoins qui peuvent évolués en cours de l'exécution du processus. Ceci

est d'autant plus vrai dans le contexte de la découverte et de la composition dynamique des services. Par ailleurs, la manipulation des protocoles par l'affectation, le blocage ou la réservation des ressources s'avère fastidieuse pour le développeur qui doit garantir la manipulation de ces opérations ainsi que celle des opérations de compensation les concernant, en cas d'annulation d'activités.

Malgré les primitives qu'il offre pour éviter l'annulation et la compensation des protocoles, Le protocole THP reste limité dans la gestion des effets des transactions et de leur manipulation. Ce constat est la conséquence de l'absence d'un modèle formel pour la représentation des effets transactionnels compliquant, ainsi, leur manipulation et le processus de compensation.

4.3.3 Les Environnements: Entreprise Java et les transactions XML

Les environnements de développement de services Web pour les entreprises souffrent des faiblesses et insuffisances conceptuelles dues au manque de modèles permettant la représentation des effets transactionnels, de leur gestion et par conséquents des outils de comparaison des effets et de leur compensation. En effet, l'état de l'art pour le traitement des transactions dans les environnements logiciels destinés aux entreprises, fait ressortir que la gestion et la manipulation des transactions dans les protocoles de services restent lourdes et imprécises.

Les produits phares dans ce domaine sont J2EE Activity Service Specification [28] et les API (Application programming Interface) Transactionning API for Java [29], qui offrent un framework pour la coordination des transactions. Ces deux environnements définissent un ensemble d'interfaces et de messages pour réaliser la coordination parmi les applications participantes et les composants gestionnaires des transactions au niveau middleware, et ce indépendamment du protocole de transaction particulier qui est supporté.

Les parties participantes échangent les messages XML accompagnés des informations relatives aux transactions (Contexte des transactions) pour accomplir des activités de coordinations et des transactions multi-tiers.

Par leurs interfaces et primitives, ces environnements de développements offrent effectivement un cadre favorable au déploiement des services Web avec prises en comptes de aspects transactionnels des protocoles. Cependant, aucun mécanisme ne permet de vérifier que les effets observés dans le monde réel sont ceux désirés, à part les procédures de test classiques. En plus, aucune modélisation des effets n'est offerte, d'où l'impossibilité de leur manipulation. Par ailleurs, la compensation des effets n'est abordée qu'en termes d'un nouveau processus à exécuter sans corrélation avec le processus déclencheur.

Vu l'absence d'abstraction de haut niveau et de modèles formels de représentation d'effets transactionnels, aucune manipulation des protocoles et des transactions n'est offerte dans ces environnements. De ce fait, la charge de développement et de programmation est sous l'entière responsabilité du développeur.

4.3.4 Web Service Transaction (WSTx)

Dans le modèle Web Service Transaction (WSTx) [30], une extension du langage WSDL est proposée afin de traiter les comportements transactionnels du client et du fournisseur d'un service Web.
Le modèle WSTx introduit le concept de: *Comportements transactionnels* (*Transactional Attitudes*) pour décrire explicitement les attitudes transactionnelles, au lieu de les considérer d'un point de vu sémantique implicite. Ce modèle offre un cadre conceptuel consistant pour fiabiliser les transactions et décrire d'une manière détaillée, comment les attitudes explicites du fournisseur et du client sont décrites et exploitées par le middleware afin d'automatiser la composition fiable des services dans un environnement transactionnel, tout en gardant l'autonomie des applications.

La modélisation des comportements du fournisseur PTA (Provider transational attitudes) repose sur les trois primitives transactionnelles: *Pending-commit, Group pending-commit* et *Commit/Compensate*.

Par contre celle du client CTA (Customer transactional attitude) est basée sur la notion d'atome d'exécution: *Flexible Atom*.

Le middleware de gestion des transactions SAM (Smart Attitude Monitor) est une infrastructure logicielle pour le modèle WSTx. Ce Middleware est une infrastructure intermédiaire entre le client et le fournisseur pour harmoniser PTA et CTA. Il assure la fiabilité d'exécution et la gestion des transactions.

Les fonctions du Middleware: SAM (*Smart Attitude Monitor*)

- Comprend et supporte des ensembles de PTA (Fournisseur)
- Comprend et supporte des ensembles de CTA (Client)
- Intercepte toute les interactions transactionnelles entre les clients et les services web fournisseur qui participent.
- Etabli et gère le contexte des transactions du côté client.
- Associe des interactions transactionnelles (messages) au contexte global.

Le système SAM est défini lui-même comme un service. Il sert d'intermédiaire entre le client qui déclenche la transaction et le (s) service (s) fournisseur.

Bien que les concepts de comportement transactionnel et de compensation soient clairement abordés dans ce modèle, il reste ambigu quant à la spécification du comportement lui-même. En effet, un comportement transactionnel dans ce modèle n'est autre qu'une opération WSDL. D'où une insuffisance dans la prise en charge des effets des opérations.

En résumé, WSTx est un cadre de réflexion intéressant pour la gestion des transactions dans les services Web, car il introduit les concepts de comportement client et fournisseur. Cependant, il souffre d'un déficit dans la modélisation des effets et ne propose qu'une classification des types d'opérations WSDL suivant des critères transactionnels, chose qui reste conceptuellement difficile. Il n'offre aucun outils pour la modélisation de ces

comportements, d'où l'impossibilité de la gestion des protocoles transactionnels, même si la compensation des opérations est introduite comme type de comportement.

4.3.5 Evaluation

Le tableau 4.1 illustre une comparaison des approches exposées pour la gestion des transactions dans les services Web. Cette analyse est basée sur des critères de modélisations des effets transactionnels et de leur gestion, sur la prise en charge de la gestion de la compensation et sur la possibilité de comparaison de protocoles.

L'étude de la gestion des transactions dans les protocoles de services fait ressortir les constats suivants:

- Malgré l'existence de diverses spécifications visant la prise en charge des contraintes transactionnelles dans les protocoles de services, celles-ci restent limitées aux aspects opérationnels et ne traitent les transactions que d'un point de vue manipulation de données.
- L'approche de compensation est perçue indépendamment des effets. Cette façon de faire oblige certains modèles à un contournement artificiel par des garde-fous afin d'éviter sa prise en charge d'une manière automatique (Modèle THP).
- A part le modèle BTP, il n'existe pas d'autres modèles ni langages qui abordent les transactions par les effets qu'ils engendrent dans le monde réel. Mais, BTP ne les traite pas d'une manière formelle; c'est une simple classification des types d'effets.
- Absence totale d'un modèle formel pour la modélisation des effets transactionnels engendrant, de ce fait, une carence dans la vérification de la consistance d'un protocole de compensation avec son protocole déclencheur ainsi que la gestion automatique de la compensation.

4.3.6 Discussion

Les constats dégagés après analyse et comparaison des diverses approches visant la gestion des transactions dans les protocoles de services mettent en relief *un déficit caractérisé* dans la modélisation des effets des transactions. En effet, les modèles existants ne traitent les protocoles de services que par leurs aspects opérationnels exprimant une simple manipulation des données, et dans le meilleur des cas, une simple classification de ces effets est proposée (modèle BTP).

Cette situation a pour conséquence directe, de traiter les protocoles de compensation indépendamment de leurs protocoles déclencheurs, augmentant, par là même, la charge de développement du service et induisant une situation d'indécidabilité pour la vérification de la consistance des deux protocoles. Alors que dans la réalité, les effets des transactions ainsi que leurs effets compensatoires constituent, une caractéristique intrinsèque d'un protocole de service. C'est un élément fondamental de la description de son comportement externe.

Approches / Critère	**Langages de Modélisation des Protocoles (WSFL, XLANG)**	**Protocoles de Transactions sur le Web: BTP**	**Protocoles de Transactions sur le Web: THP**	**Entreprise JAVA et Transactions XML**	**WS Transaction WSTx**
Concepts Clés	-Coordination basée sur des règles -Interdépendance des opérations	-Relâchement propriétés ACID. -Négociation à 02 phases. -Contrat, cohésion, confirmation	-Relâchement propriétés ACID. -Transaction courante. -Tentative, Réservation	-Contexte de la transaction et contexte global.	-Comportement transactionnel - PTA, CTA, SAM
Principe de gestion des transactions	Manipulation des données	-Négociation et coordination	- Eviter les annulations par la réservation	-Programmation et gestion des données.	-Le middleware SAM coordonne PTA et CTA
Modélisation des effets des transactions	-Pas de Modèles -Basée sur les données	- Effets Prévisionnel, effets finaux et contre-effets.	- Aucune	- Aucune	-Aucune
Gestion des effets transactionnels	-Manipulations des Données -Gestion des exceptions	Changement d'état	- Affectation des ressources	- Gestion des messages avec le contexte.	A l'aide d'un ensemble de Primitives. - Classification des opérations WSDL.
Gestion de la compensation	- Manipulation de flux de données	-Contre-effets	- Eviter au maximum la compensation	-Manuelle	-Primitive: - Commit / compensate
Comparaison des protocoles	- Par le traitement des données	- Par la comparaison des effets.	- Aucune possibilité.	- Aucune possibilité.	Aucune possibilité.
Appréciations	- Effort de programmation accru. -Pas de garantie de cohérence de la compensation.	- Négociation à chaque composition. -Efforts d'analyse, de programmation accrus. -Pas de garantie de cohérence de la compensation.	- Les clients n'ont pas une vision précise des ressources nécessaires à leurs activités.	-Pas de garantie de cohérence de la compensation.	- Framework intéressant pour la gestion des transactions.

Tableau 4.1: Evaluation des différentes approches de gestion des effets des transactions dans les services Web.

Pour rehausser la description des protocoles de services à sa sémantique réelle et l'enrichir par les caractéristiques transactionnelles qui sont les effets engendrés dans le monde réel, il faut passer par une modélisation formelle des effets transactionnels qui s'avèrerait inévitable. En plus de la représentation des protocoles de services d'une manière fidèle reflétant leurs propriétés transactionnelles, cette modélisation offre un cadre conceptuel adéquat pour la prise en charge et la manipulation des protocoles de services et de leurs protocoles de compensation.

Nous sommes convaincus que l'enrichissement des modèles de protocoles de services par leurs effets transactionnels permettra, un dépassement qualitatif de l'analyse de leur compatibilité et de leur similarité, au-delà du simple aspect syntaxique et structurel. L'enrichissement nous permettra d'affronter une analyse plus réaliste basée sur la sémantique des transactions vue sous l'angle des effets qu'ils produisent dans le monde réel.

Dans la section suivante, nous allons aborder la modélisation des effets transactionnels dans les services Web. Notre démarche est la suivante:

- Définir en premier lieu la notion d'effet transactionnel pour éviter toute ambiguïté pouvant découler d'une mauvaise interprétation de ce concept.
- Explorer les modèles existants permettant une représentation des effets des services Web, même s'ils ne traitent pas des aspects transactionnels.
- Evaluer les forces et les faiblesses des modèles de représentation existants.
- Dégager le meilleur modèle, parmi ceux étudiés, qui offrira une représentation adéquate avec le raisonnement que nous voulons faire sur les effets et favorisant une gestion de la compensation.
- Adapter le modèle dégagé, par d'éventuelles extensions et/ou enrichissements, afin qu'il puisse représenter et prendre en compte les effets transactionnels.

4.4 Modélisation des effets des transactions

Dans un environnement service Web, l'invocation des opérations se traduit par des transactions qui auront des effets sur le monde réel. La prise en compte de ces contraintes passe inévitablement par une modélisation des effets transactionnels en premier lieu.

En effet, les messages échangés lors des interactions dans les services peuvent engendrer des effets qui diffèrent d'un service à un autre, pour le même message. Il est alors nécessaire, de pouvoir identifier les effets de chaque opération, de les représenter par des modèles et d'en faire les raisonnements utiles dans le contexte des services Web, à savoir:

- Pouvoir décider de la compatibilité réelle (à base d'effets) des services.
- Pouvoir décider de l'équivalence des services à base d'effets transactionnels.
- S'assurer qu'un protocole de compensation d'un service ou d'une opération et conséquent avec le protocole déclencheur et permet une annulation effective de ses effets.
- Pouvoir manipuler les protocoles et leurs effets pour d'éventuels traitements d'équivalence, et de substitution.

Dans cette section, nous introduisons la notion d'effet transactionnel et nous explorons les modèles de représentation d'effets existants. Nous discuterons de leur richesse sémantique et de leur utilité. Un modèle de représentation des effets transactionnels est présenté et illustré avec un exemple réel. L'impact du modèle proposé sur le modèle des protocoles de services est y exposé.

4.4.1 Notion d'effet transactionnel

Un *effet* représente le changement qu'apporte un service à l'état de l'environnement. Il correspond, alors aux modifications qu'engendre l'invocation de l'opération sur l'état du client et du fournisseur. Cependant, dans notre contexte, on s'intéressera à l'impact du service du côté client seulement, car du côté fournisseur, les *effets* sont traités par le fournisseur lui-

même suivant une logique métier interne, non dévoilée, et n'entrent pas dans le cadre de l'interaction des services, contrairement aux effets du côté client.
Il est important de différencier la notion d'*effet* de celle de *sortie (output)* de l'opération. Effectivement, si les *effets* d'une opération représentent les changements dans l'état du client, les *sorties* correspondent aux informations résultant des transformations opérées par l'opération. Les *effets* et les *sorties* constituent ensembles les *résultats* de l'opération L'exemple suivant, tiré de [26], illustre cette différence.
Un client invoque un service Web pour l'achat d'article et effectue le paiement avec carte de crédit. Après validation des achats et paiement, les résultats sont:

- Les Sorties: Envoi de facture détaillée ; c'est juste des informations ;
- Les Effets: Diminution du solde du compte du client, les articles achetés sont à la disposition du client, augmentation du compte du fournisseur et diminution des quantités en stock.

Seulement, les deux premiers effets correspondent à l'impact sur l'état du client et aux changements observés dans son monde.

La notion d'effet transactionnel est d'une importance capitale dans le contexte des transactions liées aux protocoles de service. Il est alors, primordial d'aborder leur modélisation. Dans ce qui suit, nous présentons un panorama des modèles de représentation d'effets existants et une analyse comparative est exposée à la fin de cette section afin de d'évaluer la richesse des modèles existants et d'opter, par conséquent, pour le modèle qui répondra, au mieux, à notre problématique.

4.4.2 Exploration des modèles de représentation des effets transactionnels

Dans cette partie, nous présentons trois modèles de représentations des effets des transactions dans les services Web, à savoir: OWL-S, BPEL et le modèle *Colombo* et nous tenterons de tirer profit de l'expressivité du meilleur parmi eux, afin de répondre aux objectifs du raisonnement que nous avons fixé préalablement. Nous terminerons la section par une évaluation des modèles en termes d'expressivité et de richesse sémantique.

a. Représentation d'effets dans OWL-S (Ontology Web Language For Web Services)

OWL-S (OWL for Web-Services) est une ontologie basée sur OWL, à laquelle elle ajoute la notion de service. Cette ontologie fournie une description des services avec des informations sémantiques. La Figure 4.4 illustre les différentes ontologies sur lesquelles est bâti OWL-S.

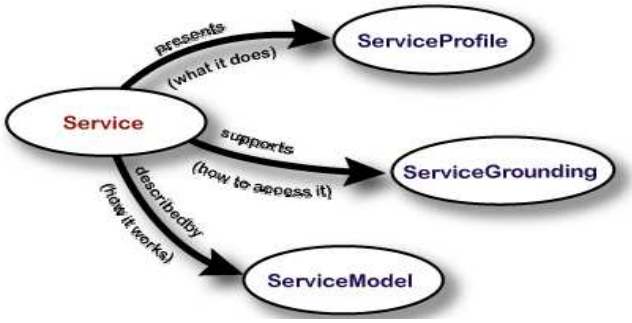


Figure 4.4: Représentation des différents éléments de l'ontologie d'OWL-S, [31]

Le ***ServiceProfile*** fourni une description générale du service Web. Il peut inclure, non seulement, les propriétés fonctionnelles (entrées, sorties, conditions préalables, et résultats), mais aussi les propriétés non fonctionnelles (nom de service, description des textes, information de contact et catégorie de service). Quant au ***serviceModel***, il décrit l'activité du service en termes de processus internes, en incluant les entrées, les sorties, pré et post-conditions, et les effets. Nous allons nous intéressés particulièrement au modèle de représentation des effets dans cette ontologie de service.

OWL-S utilise une ontologie ***ServiceProfile*** pour décrire les entrés, les sorties, les pré-conditions et les effets des services. Cette ontologie définie les propriétés des classes relatives aux IOPE (Input, Output, Pre-conditions, Effets). Le schéma de description des instances des IOPE existe au niveau de l'ontologie du Processus (**ServiceModel**). Les deux ontologies sont composées de classes décrites par un ensemble de propriétés.

En effet, dans ***ServiceProfile*** « La propriété ***hasResult*** spécifie un des résultats du service, comme défini dans la classe ***Result*** de l'ontologie S***erviceModel*** qui spécifie sous quelles conditions les résultats sont générés. En plus des résultats, elle spécifie aussi quels sont les changements produits après l'exécution du service » [31]. La description de la classe ***Result*** de l'ontologie S***erviceModel***,qui correspond à la propriété ***hasResult***, est schématisée dans la **Figure 4.5**.

En résumé, les effets des transactions des services dans OWL-S sont représentés par une classe de l'ontologie ***ServiceProfile,*** dont les propriétés sont fournies dans le **ServiceModel.** Ils sont exprimés sous forme d'expressions logiques.

```
<owl:Class rdf:ID="Result">
  <rdfs:label>Result</rdfs:label>
</owl:Class>
<owl:ObjectProperty rdf:ID="hasResult">
  <rdfs:label>hasResult</rdfs:label>
  <rdfs:domain rdf:resource="#Process"/>
  <rdfs:range rdf:resource="#Result"/>
</owl:ObjectProperty>
```

Dont les propriétés sont:

```
<owl:ObjectProperty rdf:ID="inCondition">
  <rdfs:label>inCondition</rdfs:label>
  <rdfs:domain rdf:resource="#Result"/>
  <rdfs:range rdf:resource="&expr;#Condition"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasResultVar">
  <rdfs:label>hasResultVar</rdfs:label>
  <rdfs:domain rdf:resource="#Result"/>
  <rdfs:range rdf:resource="#ResultVar"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="withOutput">
  <rdfs:label>withOutput</rdfs:label>
  <rdfs:domain rdf:resource="#Result"/>
  <rdfs:range rdf:resource="#OutputBinding"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasEffect">
  <rdfs:label>hasEffect</rdfs:label>
  <rdfs:domain rdf:resource="#Result"/>
  <rdfs:range rdf:resource="&expr;#Expression"/>
</owl:ObjectProperty>
```

Les effets du service sont des propriétés des objets de la classe ***hasResult***

Figure 4.5: Représentation des effets des services dans OWL-S, [31]

b. Représentation des effets dans BPEL

En BPEL, l'exécution d'un service est définie par la notion d'activité qui peu être basique ou structurée. Ces activités correspondent à l'invocation d'opération WSDL et sont décrites en termes d'entrée et de sortie. Toutes les données impliquées dans l'exécution du service (dans les opérations) sont explicitement nommées et listées.

Nous constatons que, BPEL ne prend pas en compte la notion d'état et ne considère pas les opérations par rapport à leurs d'effets. La conséquence est que ce langage doit prévoir alors, des mécanismes de compensation des opérations. En effet, les structures *<compensate>*, *<compensate scopes> et <compensation Handler>* permettent de répondre à cette insuffisance de représentation des effets. Ces derniers ne sont traités que par rapport aux résultats de sorties. Dès lors, l'approche de traitement et de transfert de données en BPEL est de type *BlackBoard*, analogue aux langages de programmation classiques, où toutes les données impliquées dans l'exécution des services sont explicitement nommées et listées.
En résumé, BPEL fourni les constructions et les mécanismes nécessaires pour manipuler les flots de données inhérentes à l'exécution des activités. Cependant, les effets des opérations sont considérés comme des résultats et la notion d'état est absente.

c. Modèle *Colombo* pour la représentation des effets: [32]

Dans le modèle *Colombo* de composition automatique des services Web [32], les effets reflètent les changements observés dans le monde du client. Ce modèle est basé sur le concept d'effet et des paramètres qui invoquent les processus. L'état du monde (*World state*), désigne le monde réel et il est représenté par une base de données relationnelle ayant un schéma universel (*Schéma du monde*).
Dans ce modèle, les effets d'une transaction sont exprimés par l'impact des arguments de sortie sur le monde du client. Ils correspondent à des opérations de ***mise à jour de la base de données*** ayant au schéma universel.
L'exemple suivant, tiré de [32], illustre la notion d'effet dans le modèle *colombo*.

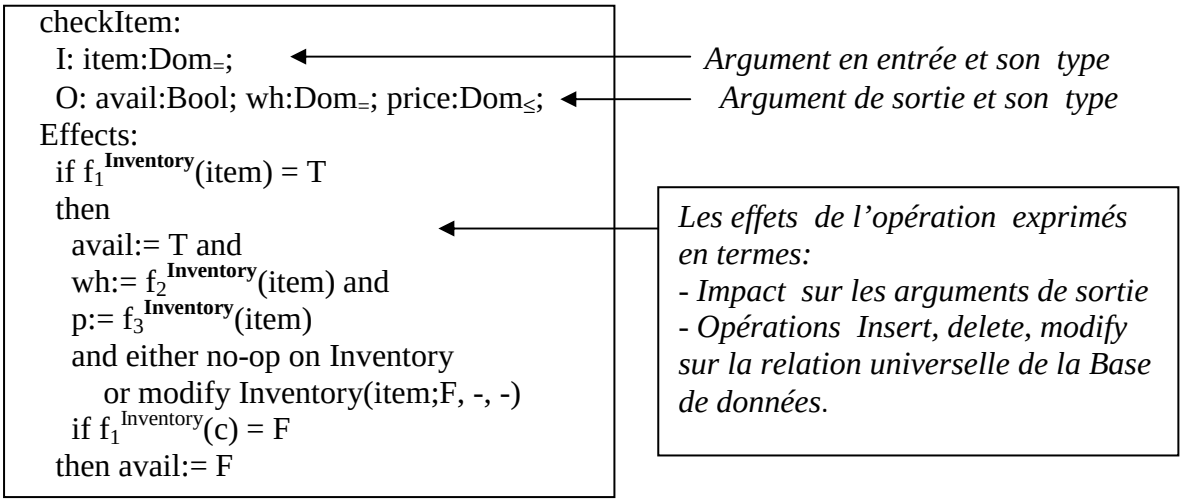


Figure 4.6: Exemple de Représentation des effets dans le Modèle Colombo, [32]

La définition formelle d'un effet dans le modèle *colombo* est donnée en fonction de celle d'un processus atomique, [32].
Etant donné une relation universelle *R*, un processus atomique est un objet *p* qui a une signature de la forme (*I, O, CE*) avec les propriétés suivantes: La *signature d'entrée I* et la *signature de sortie O* sont des ensembles de variables typées. La condition de l'*effet CE*, est un ensemble de couple de la forme *(c, E)*, où *c* est la *condition* exprimée sous forme d'une expression booléenne.
Un effet $e \in E$, est un couple (*es, ev*), où:

- *es*: désigne les effets sur le monde; c'est un ensemble d'expression de la forme:

(i) *Insert* $R(t_1, \ldots,t_k; s_1,\ldots,s_l)$
(ii) *Delete* $R(t_1, \ldots,t_k)$
(iii) *Modify* $R(t_1, \ldots,t_k; r_1,\ldots,r_l)$

Où $t_1, \ldots,t_k$ est une clé de la relation R et les t_i et s_j sont des termes formés par ensemble de constantes et les constantes $u_1,\ldots,u_n$. Les r_j sont des termes avec le symbole spécifique (-) exprimant que les enregistrements identifiés dans *R* doivent rester inchangés.

- *ev:* désigne les effets sur les sortie; c'est un ensemble d'expression de la forme:

(i) $v_j := t$, où $j \in [1\ldots m]$ et *t* est un terme accessible par des ensemble de constante $u_1,., u_n$. ou bien;
(ii) $v_j := \omega$, où $j \in [1\ldots m]$ (il doit y avoir une seule expression pour chaque v_j.)

De la définition précédente, nous pouvons dire que la représentation des effets dans le modèle *Colombo*, ne diffère pas globalement de celle décrite en OWL-S. En effets, les arguments d'une transaction dans OWL-S (entrées, les pré- et post-conditions, effets et sorties) ont été repris dans le modèle *Colombo* avec un raffinement très intéressant des effets. Ces derniers sont perçus dans une perspective plus opérationnelle et considérés comme des requêtes de mise à jour d'une base de données relationnelle.

4.4.3 Evaluation des divers modèles de représentation des effets transactionnels

Modèles / Critères de Comparaison	OWL-S	BPEL	Colombo
Concepts clés	ServiceProfil, ServiceModel, Ontologie, Classe, Effets	Variables, Activity, Scope Compensation	Requête de Mise à jour base de données, Relation universelle
Notion d'état	Oui	Non	Oui
Notion d'Effet	Oui	Non	Oui
Existence Modèle formel	Oui (logique de description)	Non (langage)	Oui (Modèle Relationnel, requêtes)
Gestion de la Compensation	Non	Oui	**Non**
Gestion du flux de données	Oui (prédicats)	Oui (blackboard)	Non
Modèle formel pour effets	**Non**	**Non**	**Non**

Tableau 4.2: Etude comparative des modèles de représentation des effets transactionnels

Le tableau comparatif 4.2 des différents modèles de représentation des effets transactionnels fait apparaître clairement la richesse du modèle *Colombo* par rapport aux deux autres.

Néanmoins, ce dernier reste défaillant dans la gestion de la compensation des transactions et ne permet pas la manipulation comparative des effets. Cette insuffisance s'explique par le fait que le modèle a été conçu dans le contexte de la composition des services, plutôt que de celui de la modélisation et de la gestion des transactions. Cependant, cette carence peut être surmontée par d'éventuels enrichissements, en vue d'une modélisation des effets transactionnels, sous entendant leur compensation et dans une perspective d'un traitement comparatif des protocoles à base d'effets. Dans la section suivante, nous allons nous basé sur le principe du modèle *Colombo* pour modéliser les effets des transactions, en considérant un effet comme une action de mise à jour de la base de données et nous allons étendre ce modèle afin qu'il puisse représenter les effets des transactions.

4.4.4 Proposition d'un modèle de représentation des effets transactionnels

a. Introduction

Pour parer au déficit constaté dans la modélisation des effets transactionnels et leur gestion ainsi que celle de la gestion de la compensation dans un environnement service Web constatés, comme il apparaît clairement dans le tableau 4.2, nous proposons un modèle pour la représentation des effets transactionnels.

Notre proposition est guidée par les objectifs du raisonnement et de la manipulation que nous voulons faire sur les effets transactionnels et par conséquent, sur l'analyse et la gestion des protocoles et leur protocole de compensation [33]. En effet, nous visons à modéliser les effets de façon à pouvoir:

- Faire la comparaison des effets (***équivalence et différence***);
- Identifier les effets élémentaires d'un effet complexe, afin d'induire les effets de compensation élémentaires correspondants, ou s'il existe un effet compensatoire global pour cet ensemble d'effets ; (***Analyse des effets: Région Atomique***);
- Vérifier qu'un effet est effectivement compensatoire d'un autre effet; (***effet inverse***)
- Composer les effets d'un ensemble d'opérations et pouvoir chercher quel sera l'effet global correspondant; (***Composition d'effet***)
- Composer les effets compensatoires d'un ensemble de protocoles et chercher s'il existe un protocole de service correspondent. (***effet inverse d'une composition d'effets***).

Cette modélisation répondra, inévitablement, aux besoins de manipulation des services et de leur protocole en prenant en compte, en plus de l'ordre des opérations, les effets des messages. Le nouveau *modèle de protocole à effets transactionnels* permettra de réaliser les opérations utiles à la gestion des protocoles de services permettant de:

- Comparer des protocoles et déceler leurs points communs et leurs différences, cette comparaison est basée sur les effets et l'ordre des opérations, (**Compatibilité à base d'effets**);
- Remplacer des protocoles par d'autres protocoles qui produisent les mêmes effets (**équivalents et substitution à base d'effets**).
- Composer des services en considérant leurs effets, (**Composition à base d'effets**);

- Annulation de transactions en invoquant des protocoles de compensation, (**différence d'effets, effets inverses**).

b. Démarche

Dans le modèle formel de représentation des protocoles de services introduit dans [15], un raffinement et un enrichissement des abstractions de haut niveau relatives aux conversations dans les services Web, sont apportés au modèle de base des protocoles. Cependant, ces abstractions de haut niveau, n'ont été qu'identifiées et aucun modèle pour la spécification des abstractions introduites n'a été présenté. En effet, les contraintes transactionnelles n'ont été qu'introduites est catégorisées dans le modèle de base présenté dans [15]. (Voir Chapitre 3 section 3.3.2.b).

Pour enrichir le modèle de base, nous visons à prendre en compte les contraintes transactionnelles et les raffiner par des modèles de représentation des effets, permettant de ce fait d'enrichir les protocoles de services pour qu'ils puissent exprimer d'une manière consistante le comportement externe du service et toute la sémantique véhiculée lors des interactions. Pour atteindre cet objectif, nous avons adopté la démarche suivante:

1. Partir du modèle de base des protocoles de services modélisé avec des automates d'états finis déterministes, comme modèle de base de protocoles de services.
2. Proposer un modèle de représentation des effets transactionnels, pour pallier aux insuffisances constatées dans ce contexte.
3. Illustrer la proposition de modélisation par un scénario réel.
4. Injecter le modèle d'effet proposé dans le modèle de base des protocoles et présenter le nouveau modèle.
5. Analyser les conséquences d'un tel enrichissement sur l'étude de la compatibilité et d'équivalence des protocoles.
6. Identifier les nouvelles classes d'équivalence et de compatibilité.
7. Proposer les nouveaux algorithmes nécessaires pour les tests d'équivalence et de compatibilité des protocoles à effets transactionnels.

Notre préoccupation principale est de représenter les effets transactionnels par des modèles et de les intégrer dans la représentation des protocoles métiers, pour aboutir à un modèle de protocole riche sémantiquement et doté d'un aspect formel pour la modélisation et la manipulation des ***protocoles de services enrichi par les d'effets transactionnels***, exprimant aussi bien, l'ordre des messages et leur polarité, ainsi que les effets observés dans le monde du client.

Cette démarche sera adoptée dans la suite du mémoire et vu l'importance et la consistance des étapes 5, 6 et 7, celles-ci seront traitées dans un chapitre à part.

c. Un nouveau modèle de représentation des effets

En partant de l'étude du comportement externe des services, de leurs interactions et plus particulièrement de l'analyse des transactions et le besoin de leur modélisation dans la

perspective d'une éventuelle compensation, nous proposons un modèle formel pour les représenter. Dans notre modèle, la représentation des effets et de leurs effets de compensation est basée sur *la perception d'un effet comme une requête de mise à jour de la base de données*. Ainsi, un message d'un protocole aura comme effets sur le monde du client *une requête de type*: *Insert(R), Delete (R)* ou *Modify (R)* où R est un enregistrement de la base de données [33].

Il ne nous échappe pas de signaler que notre modélisation est guidée par un souci d'enrichissement et de compréhension des mécanismes inhérents aux effets transactionnels. Ce qui justifie notre proposition de représenter un effet par une seule requête, au lieu de compliquer plus le problème en considérant l'effet comme plusieurs requêtes de mise à jour. Les modèles et/ou résultats obtenus peuvent être exploités pour aborder le modèle à base de requêtes multiples pour chaque effet.

La problématique de la manipulation des effets transactionnels est ramenée, en conséquence, à celle de la manipulation des requêtes de mise à jour de la base de données. Dès lors, les préoccupations précédentes liées à la manipulation des effets transactionnels sont traduites en termes de manipulation des requêtes de mise à jour au niveau de la base, comme suit:

- Comparaison des requêtes de mise à jour. (***équivalence et différence des effets***);
- Rechercher une requête inverse d'une requête de mise à jour. (***effet inverse d'un effet pour gérer la compensation***)
- Evaluer l'équivalence de deux séquences de requêtes de mise à jour qui auront des résultats identiques en produisant des états identiques de la base de données. (**Pour décider de l'équivalence de deux protocoles à base d'effets transactionnels).**
- Décomposition d'une requête complexe en un ensemble de requêtes plus élémentaires produisant la même mise à jour de la base, afin de rechercher les effets élémentaires et faciliter la recherche de requêtes élémentaires de compensation.
- Décider de l'inclusion des requêtes (Une Requête R_1 traite un sous ensemble de mise à jour d'une requête R_2);
- Rechercher une requête dont les résultats d'exécution correspondent à l'annulation d'un ensemble de requêtes (***effet inverse d'une requête complexe***).

La problématique de la prise en compte des contraintes transactionnelles dans les protocoles de services lors de la découverte et de la composition est ramenée à celle de la manipulation des requêtes de mise à jour de base de données relationnelles.

Définition formelle de la compensation: *Une requête R_j appartenant à un ensemble d'effets E_{cj} d'un monde client, compense les effets d'une autre requête R_i du même monde client, lors de l'exécution du service Web S, et on notera: $[R_j\, comp(R_i)]_S$, si l'exécution séquentielle R_i o R_j, de la requête R_i suivi de R_j n'engendre aucun effet sur le monde client, sauf ceux désirés volontairement par le fournisseur de S et exprimant les charges des transactions engagées par les fournisseurs envers les client. On parlera de* ***couple effet-compensation***.

Plus formellement, soit un message ***m (p, e, e')*** **tel que:**

e: Ensembles des effets d'un message d'un protocole de service représenté par la requête R_i.
e': Ensembles d'effets d'un message du protocole compensatoire représenté par la requête R_j.

Cet ensemble d'effets de compensation peut être décomposé en deux types d'effets obtenus après décomposition:

e'_a: Ensembles des effets d'annulation des effets du message *e*, représenté par la requête R_a.

e'_f:Ensemble des effets volontairement désiré par le fournisseur du service, représenté par la requête R_f.

Comme e'_f désigne le sous ensemble de e' relatif aux effets volontairement désirés par le fournisseur (charges affectées au client suite à l'annulation des transactions), la décomposition des effets donnera:

$e' \equiv e'_a \, U \, e'_f$, où e'_a correspond aux effets d'annulation simple de la transaction; d'où

$e'_f \equiv e' \setminus e_a$, et la décomposition des requêtes correspondantes est:

$R_j = R_a \, o \, R_f$; d'où: $[(R_a \, o \, R_f) \; comp(R_i)]_S$; exprimant que la séquence de requêtes $R_a \, o \, R_f$ compense la requête R_i dans le protocole de service *S*.

Il est claire que si: $R_f \equiv \emptyset$ (pas de charges) alors, l'annulation de la transaction n'affecte pas l'état de la base et on aura: $[R_a \; comp(R_i)]_S$; c'est à dire que si l'ensemble des requêtes exprimant les effets de compensation, volontairement désirés par le fournisseur est nul, alors, les ensembles de requêtes d'exécution et d'annulation sont équivalents.

d. Scénario de modélisation: Vers un modèle de protocoles de services enrichis par les effets transactionnels

Pour illustrer notre proposition de la modélisation des effets, nous allons reprendre le protocole de service de la réservation au niveau d'une compagnie aérienne de la figure 4.2. Nous nous situons dans le cas de l'aboutissement de la réservation et sa confirmation par le fournisseur (envoi message **confirmation(-)**), pour analyser les effets des transactions, analyser les effets de compensation les concernant et déduire les incidences conceptuelles sur l'analyse et la comparaison des protocoles à base d'effets. L'ensemble des effets engendrés par le service réservation (figure 4.2), permettant de réserver une place dans un vol d'une compagnie aérienne et de procéder au paiement par carte de crédit est décrit par les requêtes de mise à jour de la base de données et les résultats suivants:

- R_{11}: *Insert* (**IdenVo**l, Type-Classe, NumPass-port) // affectation du passager au Vol.
- R_{12}: *Insert* (**NumPass-port**, Nom, Prénom, Mode-Payement) // Saisie données client.
- R_{13}: *Modify* (**NumPass**, solde-client) // solde-client:= Solde-client- Montant-Billet.
- R_{14}: *Modify* (**compte**, solde-comp-f) // solde-cpt-f:= Solde-cpt-f + Montant-Billet.
- R_{15}: Envoi du billet de réservation au client. // Résultat
- R_{16}: *Modify* (**IdenVo**l, Type-Classe) // Nbre-place-disp:= Nbre-place-disp – 1. // Maj

Soit E_1 l'ensemble des effets engendrés par le service, conformément à notre modèle c'est une séquence de requêtes de mise à jour de la base. Les requêtes: R_{11}, R_{12}, R_{13}, R_{14}, R_{15} et R_{16} sont soumises à une relation d'ordre dans leur exécution. (la requête R_{15} est un effet matériel, donc non considéré parmi les requêtes de mise à jour de la Base de données). Notons *o* un opérateur exprimant la relation d'ordre d'exécution des requêtes. A l'ensemble correspond la séquence de requêtes $s(E_1)$ suivante: $s(E_1) = \{R_{11} \, o \, R_{12} \, o \, R_{13} \, o \, R_{14} \, o \, R_{16}\}$.

Parmi cette séquence de requêtes, seule la requête R_{13}, a un impact sur le monde du client, elle fait une modification de son solde.

On notera E_{1c} l'ensemble des effets qui ont un impact sur le monde du client.

$E_{1c} = \{R_{13}\}$, avec $E_{1c} \subset E_1$.

Soit E_2 l'ensemble des effets engendrés suite à l'annulation de la réservation par le client. Les résultats et requêtes correspondants sont:

- R_{21}: *Delete* (**IdenVo**l, NumPass-port) // Suppression du client du Vol.
- R_{22}: Modify (**NumPass**, solde-client) // solde-client:= Solde-client+ Montant-Billet-Retenues // rembourser montant et retenir des charges correspondantes à l'annulation.
- R_{23}: Modify (**compte**, solde-cpt) //solde-cpt:= Solde-cpt - Montant-Billet+Retenue.
- R_{24}: Modify (I**dentVol**, Nbre-place-dispon) // Nbre-place-disp:= Nbre-place-disp + 1 // Incrémenter le nombre de places disponibles suite à l'annulation de la réservation.
- R_{25}: Envoi avis au client. // Résultat

De même que pour les effets, les requêtes correspondantes à la compensation: R_{21}, R_{22}, R_{23}, R_{24} et R_{25} sont soumises à une relation d'ordre dans leur exécution. (la requête R_{25} est un effet matériel, donc non considéré parmi les requêtes de mise à jour de la Base de données). Si l'opérateur o exprime l'ordre d'exécution des requêtes, alors la séquence de requêtes est la suivante: $s(E_2) = \{R_{21} \circ R_{22} \circ R_{23} \circ R_{24}\}$. ($R_{25}$ est un effet matériel, donc non considéré)
Les effets qui ont un impact sur le monde du client: $E_{2c} = \{R_{22}\}$, avec $E_{2c} \subset E_2$.
Seule la requête R_{22}, a un impact sur le monde du client, elle correspond à la compensation de l'effet de la requête R_{23} de E_1 suivant la logique métier du fournisseur. (Retenue des charges).
On dira que la requête $R_{22} \in E_{c2}$ *compense* les effets de la requête $R_{13} \in E_{c1}$ lors de l'exécution du service Web réservation de places.

Ce scénario de modélisation, met en relief la nécessité de prendre en compte l'impact qu'aura l'invocation d'un message lors de l'exécution d'un service Web. Ainsi, les effets observés dans le monde du client, suite à l'invocation d'un message et leurs effets de compensation, exprimés sous forme de requêtes, sont une propriété intrinsèque au message. Dans ce contexte les messages du protocole de service *S* sont exprimés par rapport aux effets et leurs effets compensatoires.

4.5 Impact du modèle d'effet proposé sur le modèle de base des protocoles de service

Dans le modèle de base des protocoles de services prenant en compte des contraintes d'ordre sur les opérations [13], un message est caractérisé, simplement, par sa polarité. Par contre, la prise en compte des effets transactionnels permet une représentation plus riche de la réalité des interactions dans les services Web. En effet, un message sera caractérisé, en plus de sa polarité, par les effets qu'il engendre dans le monde du client, aussi bien que par les effets de compensation associés. Les effets de compensation sont représentés conjointement avec les effets observés, afin d'exprimer le fait que les fournisseurs de services appliquent des charges qui diffèrent, même si les effets sont les mêmes. Cette représentation reflète bien la réalité relative à la diversité des logiques de compensation spécifique à chaque fournisseur.

4.5.1 Nouvelle structure du message pour les protocoles de services enrichis par les effets transactionnels

Si on considère qu'à chaque effet est associé une requête de mise à jour de la base et sa requête correspondante liée aux effets de compensation, alors le message est modélisé comme suit [33]:

***m (p, e, e'),* où:**

m: Désigne le message et *p* sa polarité (+,-) selon que le message est en entrée ou en sortie.

e = E_{cj}: C'est l'ensemble des effets observés dans le monde du client; conformémement à notre modèle de représentation des effets, c'est une requête de mise à jour de la base de données (*Insert(R), Delete (R)* ou *Modify (R)) ; o*ù R est un enregistrement de la base de données

e'= E'_{cj}: C'est l'ensemble des effets de compensation pour annuler *e*; c'est une requête de mise à jour de la base de données pour défaire les effets *e*, tout en appliquant des charges liés à l'annulation de la transaction par le client. (*Insert(R), Delete (R)* ou *Modify (R)); o*ù R est un enregistrement de la base de données

Cette modélisation des effets transactionnels apporte un enrichissement important des protocoles de services. Elle exprime d'une manière formelle (ensemble de requêtes relationnelles) les effets des transactions et celle des opérations de compensation pour chaque message du protocole du service Web. Les conséquences d'une telle modélisation et d'un tel enrichissement induiront une révision plus fine et plus complète de l'analyse de compatibilité et d'équivalence des protocoles de services.

4.5.2 Modèle formel du protocole de service enrichi par les contraintes transactionnelles

Pour définir les protocoles de services à effets transactionnels, nous utilisons le formalisme des automates finis déterministes auquel nous apporterons des raffinements afin de prendre en compte les effets des transactions [33].

On considère un protocole service comme un tuple $P = (S, s_0, F, M, R, S_B)$ constitué des éléments suivants:

S: Ensemble fini d'états;

$s_0 \in S$ est l'état initial du protocole;

F: Ensemble des états finaux de l'automate*; Avec* $F \subset S$.

S_B: état de la base de données associé à un état quelconque du protocole;

M: Ensemble fini de messages, pour chaque message du protocole $m \in M$, on défini une fonction de polarité. Elle prend la valeur (+) si le message est en entrée, (-) si le message est en sortie. [13]

Par ailleurs, on associe au message m deux types d'effets E_c et E'_c, pour désigner les effets observés dans la base et leurs effets de compensation associés.

On notera *e* et *e'*, les effets engendrés dans le monde du client et leurs effets de compensation. A ces effets correspondent deux requêtes, respectivement: *Ri* et *Rj* de mise à jour de la base.

$R \subset (S \times S_B)^2 \times M$: Ensemble de transitions. Chaque transition concerne un état source auquel est associé un état de la base de données vers un état cible avec son état de la base suite à la réception du message d'entrée ou de sortie suivant qu'il est consommé ou produit durant la transition. On notera:

$R\ ((s, s_b), (s', s'_b), m)$, au lieu de $((s, s_b), (s', s'_b), m) \in R$.

Par ailleurs on définit une *fonction d'effets* qui à chaque message permettant la transition d'un état à un autre (avec leur état de la base) associe les effets (en termes de requêtes) et les effets de compensation correspondant (requêtes correspondantes).

La *fonction d'effet* $f_{eff}((s, s_b), e, e')$ associe à chaque état du protocole s et son état de la base, s_b, les effets et les effets de compensation correspondants lors de sa transition vers un autre état s' ayant un état de la base s'_b.

Le message est alors, de la forme: ***m (p, e, e')***; où *e* et *e'* sont les effets affectant le monde du client, exprimés par les requêtes de mise à jour de la base de données et définis préalablement par le fournisseur conformément à sa logique métier.

Le nouveau modèle de protocole de service enrichi par les contraintes transactionnelles est présenté dans la figure 4.7.

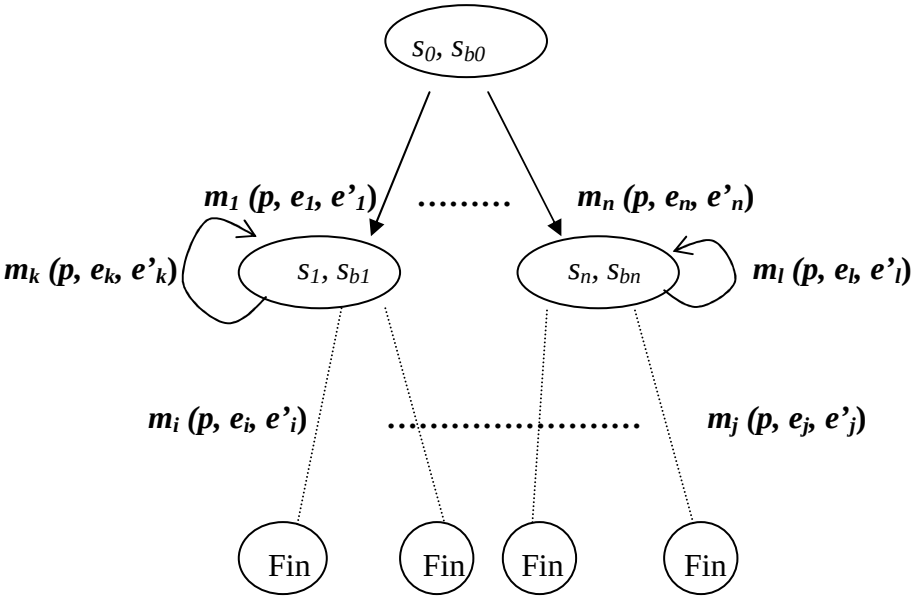


Figure 4.7: Automate représentant le protocole de service à effets transactionnels.

Légende:
p: Polarité des messages (+: Message entrant, -: Message Sortant)
m: Message d'une opération WSDL.
e: effets transactionnels induits par le message et modélisés par une requête de mise à jour.
e': effets compensatoires d'un effet transactionnel et modélisés par une requête de mise à

4.6 Conclusion

Dans ce chapitre, nous avons mis en évidence l'intérêt de la modélisation des contraintes transactionnelles au niveau protocole, en tant qu'éléments intrinsèques de la description du comportement externe des services Web. Ces contraintes ont été perçues comme effets affectant le monde du client et ont été abordées dans la perspective de leur compensation. Nous avons proposé un modèle basé sur les requêtes de mise jour de la base de données pour la représentation des effets transactionnels. A notre avis, ce modèle est adéquat avec le raisonnement que nous voulons faire sur les protocoles et il répond d'une manière consistante aux objectifs exprimés dans le début du chapitre à travers les différentes motivations. Le modèle de base des protocoles de service a été enrichi par la modélisation des effets transactionnels. Le nouveau modèle de protocole de service, enrichi par les effets des transactions, a été présenté et formalisé d'un point de vue structure des messages et séquencement des opérations. Cependant, cet enrichissement a affecté l'analyse des similarités et des différences pour le modèle de base, du fait qu'il l'a rehaussé à un degré dépassant la simple expression de l'ordre des messages.

Dès lors, une révision de l'analyse de compatibilité et d'équivalence des protocoles de services est inévitable dans le nouveau contexte des protocoles caractérisés par un nouveau format des messages et dont les effets sont exprimés par des requêtes de mise à jour de la base de données. Dans le prochain chapitre, nous allons aborder la nouvelle analyse de compatibilité et d'équivalence à la lumière des enrichissements apportés.

Chapitre 5

Analyse de Compatibilité et d'Equivalence des Protocoles de Services à Effets Transactionnels

Introduction

Dans cette partie, nous aborderons l'analyse des protocoles de services du point de vue compatibilité et équivalence, tout en prenant en compte les effets associés aux contraintes transactionnelles, représentés par des requêtes de mise à jour de la base de données.

Notre analyse s'articulera sur le nouveau modèle formel des protocoles de services enrichi par les contraintes transactionnelles présenté dans le chapitre précédent. Nous commençons notre étude par la présentation d'un ensemble de scénarios mettant en évidence les problèmes inhérents à l'analyse de compatibilité et d'équivalence des protocoles de services à effets transactionnels, à partir desquels de nouvelles définitions de la compatibilité et de l'équivalence seront élaborées et les algorithmes adéquats seront spécifiés. Par la suite, nous identifierons de nouvelles classes d'équivalence des protocoles, auxquelles nous présenterons les algorithmes correspondants. Nous terminerons le chapitre par la proposition d'un ensemble d'opérateurs de manipulation des protocoles de services, en vue de répondre aux besoins spécifiés dans le chapitre précédemment. Ces opérateurs permettront une comparaison des protocoles, la vérification de leur compensation, l'analyse de compatibilité et d'équivalence des protocoles.

5.1 Analyse de compatibilité et d'équivalence des protocoles à effets transactionnels

Comme il a été explicité dans le chapitre 3 (section 5), l'intérêt de l'analyse de compatibilité, d'équivalence et de substitution de deux protocoles consiste à vérifier si les deux protocoles peuvent interagir correctement en se basant sur leur définition de protocole et si un service peut remplacer un autre [13].

Cependant, l'enrichissement des protocoles de services par les effets transactionnels et leur représentation en tant que requêtes de mise à jour de la base de données aura des conséquences sur les définitions de la compatibilité et de l'équivalence des protocoles de services telles qu'elles ont été introduites dans [13]. En effet, les définitions introduites dans [13] n'ont été exprimées que par rapport au modèle de base, et ne prennent en compte que l'ordre des messages et leur polarité. Il en découle que la prise en compte des contraintes transactionnelles dans les protocoles de services comme effets produits dans le monde réel et dans un objectif d'une représentation plus riche des processus métiers, ainsi que pour gérer le processus de compensation et de s'assurer de sa consistance avec le protocole déclencheur, exige de nouvelles définitions de la compatibilité et de l'équivalence des protocoles de services.

Dans cette section, nous commencerons par présenter un ensemble de scénarios mettant en relief différentes situations dans lesquelles la compatibilité et l'équivalence des protocoles, telles que spécifiées dans le modèle de base [13], sont insuffisantes et/ou inadéquates dans le nouveau contexte du modèle enrichi, suite à l'injection des contraintes transactionnelles dans le modèle de protocoles. Ces scénarios nous permettront de dégager, dans un premier temps, les nouvelles exigences pour pouvoir élaborer une définition plus riche de la compatibilité et de l'équivalence qui prendra en compte les effets des transactions. Ils permettront, par la suite,

de dégager la substance nécessaire pour aborder l'étude détaillée des classes d'équivalence et la conception des algorithmes y afférents.

5.1.1 Scénarios pour l'analyse de compatibilité

L'analyse de compatibilité de deux protocoles de services permet de vérifier s'ils peuvent interagir, totalement ou partiellement, d'une manière correcte. C'est un aspect très important pour le développeur de service qui doit s'assurer que la conversation, au moins partielle, avec le service du fournisseur est possible. Elle permet, au client, de cerner les chemins d'exécution supportés par le protocole candidat à la conversation.

Contrairement aux protocoles de base, l'injection des effets des transactions dans les protocoles de service exige, non seulement une compatibilité en terme de conversations possibles, mais la rehausse à une compatibilité plus fine qui prendra en considération, en plus de l'aspect structurel des messages, leurs effets et leurs effets de compensation, et ce pour chaque message impliqué dans l'interaction. Nous exposons dans ce qui suit un ensemble de scénarios pour illustrer notre analyse.

<u>Scénario 1:</u> Compatibilité avec prise en compte des effets des transactions

Soient P_1 et P_2 deux protocoles de services, modélisés avec des automates à effets transactionnels, conformément au modèle proposé dans la section (4.5.2).

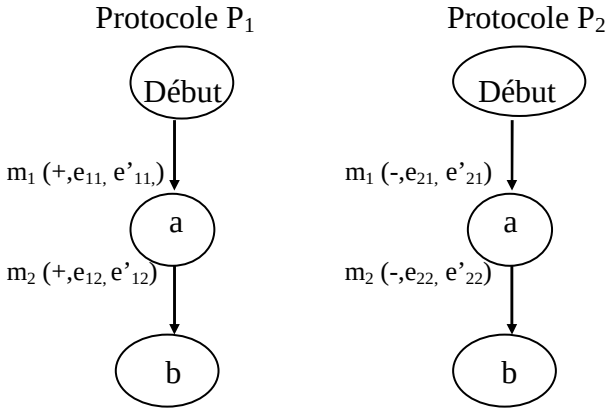


Figure 5.1: Deux Protocoles compatibles sans les effets transactionnels

Sans la prise en compte des contraintes transactionnelles, c'est à dire des effets, les protocoles des services P_1 et P_2 sont **compatibles.**

En effet, la trace d'exécution de P_1: m_1 $(+,e_{11},\ e'_{11,}).m_2$ $(+,e_{12},\ e'_{12})$ est supportée par le protocole P_2 du fait que si P_1 envoie le message m_1 (ayant pour effets e_{11}) suivi par m_2 (ayant pour effets e_{12}), il peut interagir correctement avec P_2 , car il s'agit des mêmes noms de messages avec le même ordre, les mêmes états et les polarités sont inversées (en entrée dans P_1 et en sortie dans P_2), donc les polarités sont compensées.

Cependant, avec la prise en compte des effets des messages, P_1 et P_2 ne pourront être compatibles qui si les effets des messages m_1 et m_2 soient compatibles. En partant de la représentation des effets des transactions par des requêtes de mise à jour de la base de

données, telle que décrite dans la section 4.4.4 du chapitre précédent; à chaque effet des messages e_{11}, e_{12} de P_1 et e_{21}, e_{22} de P_2 correspond une requête de mise à jour de la base de données de type (*Delete*, *Insert* ou *Modify*). L'interaction entre les deux protocoles n'est possible que si, chaque requête d'un message de P_1 **est de même type que celle du message correspondant dans P_2**.

Cette extension de la notion de compatibilité des effets, en se basant sur les types de requêtes de mise à jour, offre des garanties supplémentaires pour une interaction plus correcte entre les deux protocoles. En effet, au lieu de restreindre le critère de compatibilité, simplement, aux messages, il est nécessaire de tenir compte des effets qui seront observés dans le monde du client. Dans cette optique, P_1 ne pourra interagir correctement avec P_2 que si, les requêtes de chaque message de P_1 seront de même type que celles des messages correspondant dans P_2. Plus explicitement, les deux protocoles ne peuvent interagir que si à chaque requête d'un message de P_1 correspond une requête de même type dans P_2 pour le message correspondant. Ainsi, un protocole client ne pourra engager des conversations correctes avec celui d'un fournisseur, que si les types de requêtes sont identiques. A titre d'exemple, si la requête du message m_1 de P_1 est de type *Insert*, alors que celle de P_2, pour le même message, est de type *Delete*, les deux protocoles ne peuvent interagir correctement, à cause des effets non conformes qui seront produits au niveau des bases de données du client et du fournisseur.

Par conséquent, le chemin d'interaction [13] entre P_1 et P_2 est étendu, à son tour, à la notion de type de requête de mise à jour. Il est défini en fonction des types de requêtes, comme suit:

Etat. Message. Type de requête, et la compatibilité sera abordée en fonction de ce nouveau modèle de chemin d'interaction.

Ce scénario met en évidence, la nécessité de prendre en compte des effets des transactions lors de l'analyse de compatibilité des protocoles de services à bases d'effets transactionnels.

<u>Scénario 2:</u> Compatibilité avec prise en compte des effets de compensation

Soient P_1 et P_2 deux protocoles de services modélisés avec des automates à effets transactionnels et intégrant une opération de compensation.

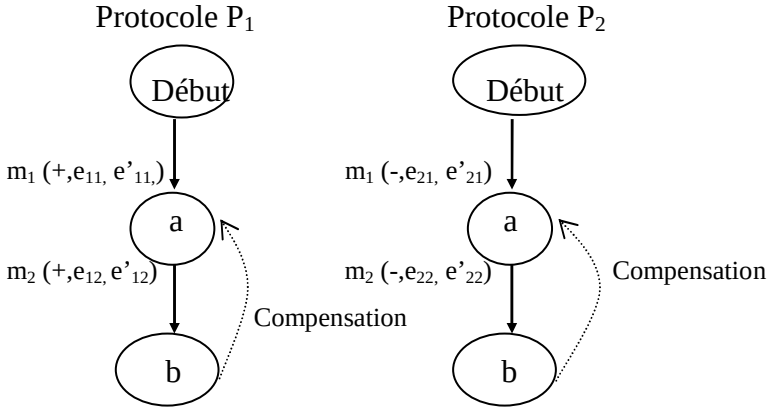


Figure 5.2: Deux Protocoles compatibles sans les effets de la compensation

Si on considère que les protocoles de services P_1 et P_2 sont compatibles en prenant en compte les effets des messages m_1 et m_2, c'est à dire que les requêtes de mise à jour sont de même

type, nous allons réexaminer l'analyse des deux protocoles en nous intéressant aux protocoles de compensation de P_1 et P_2.

A Chacun des messages m_1 et m_2 correspond un ensemble d'effets décrivant la compensation des messages dans le cas de l'annulation de l'opération par les clients ou par le fournisseur. Ainsi, à m_1 correspondent les effets de compensation $e'_{11,}$ permettant de défaire les effets e_{11} pour le protocole P_1 et e'_{21} pour le protocole P_2. De même pour le message m_2 correspond les effets de compensation e'_{12} permettant de défaire les effets e_{12} dans le protocole P_1 et e'_{22} pour le protocole $P_{2.}$ Supposons, pour simplifier, que la compensation concerne uniquement le message m_2, alors la question est de savoir si la requête associée à l'effet e'_{12} est de même type que associée à e'_{22} ?

Du fait que les types des deux requêtes à exécuter pour matérialiser les effets au niveau de la base, lors de l'opération de compensation, peuvent être différents, alors, les deux opérations ne seront pas compatibles, et par conséquent, la compatibilité des deux protocoles est remise en cause. Si les effets de la compensation sont différents, autrement si les types des requêtes de mise à jour exécutées lors de la compensation ne sont pas les mêmes, alors les deux protocoles ne peuvent pas interagir correctement.

L'analyse de compatibilité exige la prise en compte des effets des opérations de compensation en considérant les types de requêtes de mise jour.

<u>Scénario</u> 3: Compatibilité indirecte des effets des protocoles

Soient P_1 et P_2 deux protocoles de services décrits dans la figure 5.3

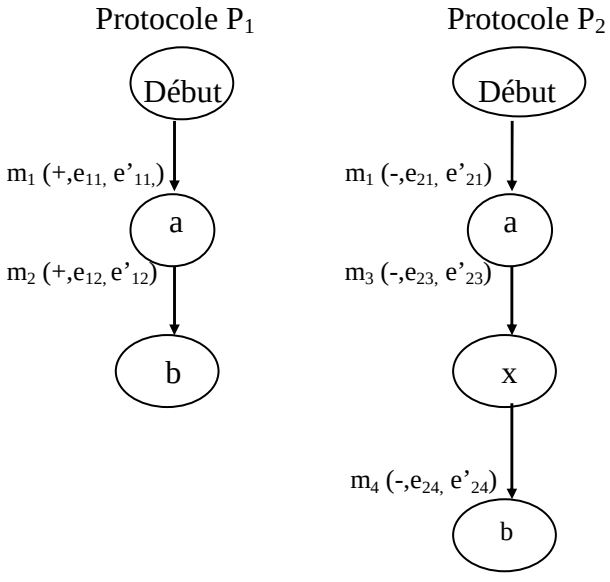


Figure 5.3: Deux Protocoles incompatibles, le sont-ils transitivement ?

Les protocoles de services P_1 et P_2 sont incompatibles en considérants, simplement, la séquence des messages échangés. En effet, en partant de l'état initial *a*, P_1 transite à l'état *b*, suite à la réception du massage m_1, alors que P_2 transite l'état *x* suite à la réception du message m_3. C'est un cas clair d'incompatibilité conformément à sa définition dans les

protocoles de base [13]. Par ailleurs, les effets e_{12} observés lors de la réception du message m_2 pour le protocole P_1 sont représentés par une seule requête, alors que pour P_2, la transition de l'état *a* à l'état *b* impliquent l'exécution séquentielle de deux requêtes relatifs aux effets de m_3 (e_{23}) et à ceux de m_4 (e_{24}).

Comme les deux traces d'exécution complètes: $m_1(+,e_{11},\ e'_{11}).m_2(+,e_{12},\ e'_{12})$ de P_1 et $m_1(-,e_{21},\ e'_{21}).\ m_3(-,e_{23},\ e'_{23}).\ m_4(-,e_{24},\ e'_{24})$ de P_2 aboutissent au même état *b* en partant du même état initial *a*, la question est de savoir si la séquence des requêtes successives associées à e_{23} et e_{24} ne produit pas le même type d'effets que celle associée à e_{12} ? Autrement est-ce qu'une conversation n'est pas possible, si nous considérons les effets cumulés de P_2 ?

Ce scénario illustre parfaitement la situation d'une compatibilité réelle, prenant en compte les effets produits dans la réalité, au-delà de la simple compatibilité syntaxique.

L'analyse de la compatibilité des deux protocoles nécessite la prise en compte de la composition des effets transactionnels.

(L'exemple peut-être étendu à plusieurs états transitoires: on parlera de compatibilité transitive multiple)

Un exemple typique, dans ce contexte, est le traitement des requêtes de mise à jour de type: *Modify*, que certains protocoles considèrent comme une séquence de deux requêtes: *Insert* suivie de *Delete.* Dans ce cas la compatibilité est vérifiée.

<u>Scénario</u> 4: Compatibilité indirecte des effets de compensation (annulation d'une région atomique)

Soient P_1 et P_2 deux protocoles de services décrits dans la figure 5.4 et dont la transition de l'état *a* vers l'état *b* est annulée, soit par le client soit par le fournisseur et exigeant une compensation de l'opération.

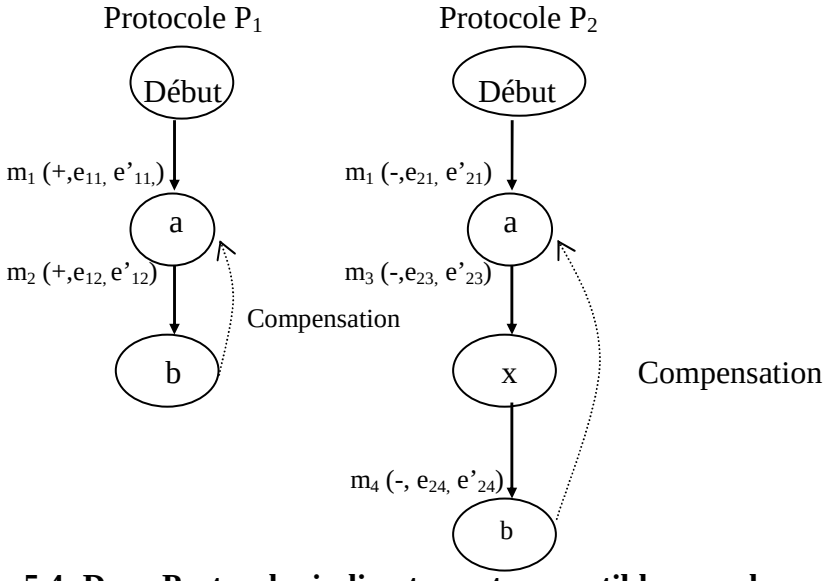


Figure 5.4: Deux Protocoles indirectement compatibles sans la compensation

Si nous considérons que les protocoles P_1 et P_2 sont compatibles: Ils aboutissent au même état des bases de données et la séquence des requêtes liées à m_3 et m_4 et de même type que celle de m_2. Alors, après exécution de toutes les opérations des deux protocoles, les deux états des bases de données seront équivalents. Cependant, est-ce que les effets des opérations de

compensation du message m_2 *(+, e_{12}, e'_{12})*, c'est à dire e'_{12} sont compatibles avec les effets de compensation cumulés des requêtes liées au message m_3 *(-, e_{23}, e'_{23})* suivi de m_4 *(-,e_{24}, e'_{24})* ? Autrement, est-ce que les effets de la compensation relative à e'_{12} ne sont pas de même type que ceux associés à la séquence des deux requêtes e'_{23} et e'_{24} ?

La question est de savoir si la composition des effets de compensation m_3 et m_4 est de même type que celle associée aux effets de compensation de m_2 ?

L'analyse de compatibilité est incomplète et nécessite la prise en compte de la composition des effets de compensation.

<u>Scénario</u> 5: Compensation unique d'un ensemble d'effets: Protocole cyclique

Soient *P* un protocole d'un client décrivant l'achat de produits auprès d'un fournisseur.

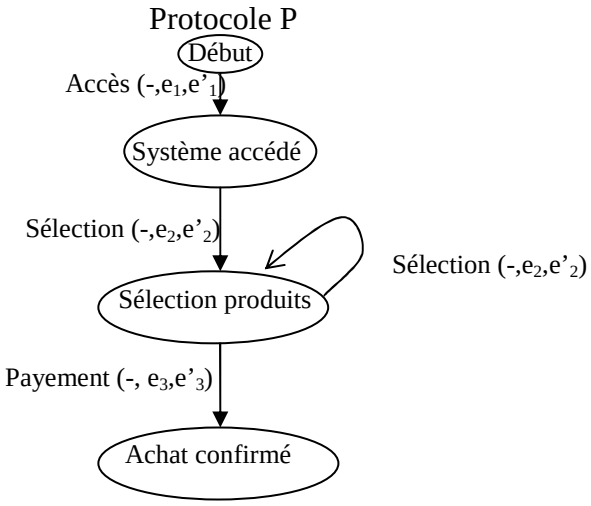


Figure 5.5: Protocole cyclique achat abandonné, comment le compenser ?

Ce protocole présente une opération itérative pour le message *sélection (-, e_2, e'_2)*, qui exprime qu'un client peut sélectionner plusieurs articles tout en restant dans le même état (*Sélection produits*). Cette structure itérative permet au client de choisir plusieurs articles, engendrant pour chacun des effets sur le monde client (e_2) décrit par une requête de mise à jour.

La question est de savoir si, en cas d'annulation du processus achat par le client <u>*en une seule action*</u> (le client ne va pas s'amuser à défaire chaque sélection !), comment compenser la totalité des effets en une seule action ? Autrement, quelle sera la composition des effets de compensation d'une annulation totale d'un protocole ? La compatibilité du protocole de compensation est-elle traitée par rapport à l'état initial de la base de données (en termes d'effets) ou en termes de compatibilité individuelle des messages ?

Ce cas diffère de celui du scénario 4, car il s'agit d'une annulation globale d'un protocole au lieu d'une composition simple d'effets de compensation.

Un exemple similaire est la réservation par un client d'une chambre, d'un spectacle et d'un vol aérien, puis après vérification de la météo du jour du spectacle, il annule le tout en une seule action (En fermant toutes les fenêtres actives, par exemple).

Ce scénario traite de l'annulation d'un protocole contenant une opération itérative. L'annulation, exige-t-elle la composition des effets de compensation? Et quels seront les effets de compensation cumulés d'une telle annulation ?

L'analyse des protocoles nécessite la prise en compte des effets de compensation lors de l'annulation globale d'un protocole.

- **Synthèse sur les scénarios d'analyse de compatibilité:**

Les différents scénarios précédents mettent en évidence le besoin réel d'une nouvelle spécification de la compatibilité des protocoles de services à effets transactionnels. En effet, la modélisation de ces effets par des requêtes de mise à jour de la base, a induit des bouleversements structurels et sémantiques dans les protocoles de base, exigeant une nouvelle réflexion profonde autour de l'analyse de la compatibilité des protocoles. Par ailleurs, la modélisation des effets des opérations de compensation conjointement aux effets déclenchés, en représentant un message par les attributs liés (e, e') exprime parfaitement les situations réelles dans lesquelles les fournisseurs associent des effets de compensation pour chaque effet observé. Les conséquences conceptuelles seront abordées dans la suite du chapitre, par l'introduction d'une définition conséquente de la compatibilité des protocoles de services à effets transactionnels.

5.1.2 Scénarios pour l'analyse d'équivalence

L'analyse d'équivalence a pour objectif de tester si deux protocoles peuvent être utilisés interchangeablement dans n'importe quel contexte et d'une manière transparente à l'utilisateur. La situation la plus opportune pour exploiter l'équivalence des protocoles est la substitution d'un protocole défaillant par un autre qui offrira, au moins, les mêmes fonctions (cas de panne de service, nouvelle version non compatible, composition de services…). L'équivalence des protocoles à effets transactionnels diffère, évidemment, de celle des protocoles de base. C'est ce que nous verrons dans les scénarios suivants.

<u>Scénario</u> 1: Equivalence étendue aux effets des protocoles

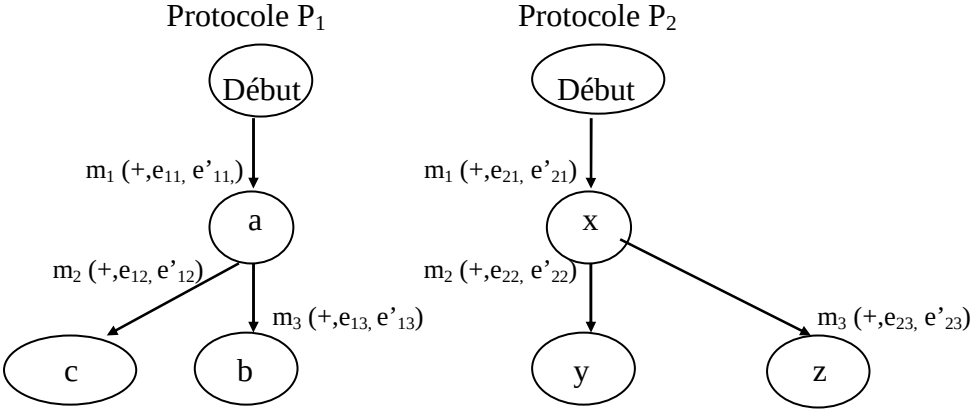


Figure 5.6: Les protocoles P_1 et P_2 sont-ils équivalents en considérant leurs effets transactionnels ?

L'analyse syntaxique des deux protocoles P_1 et P_2 fait ressortir qu'ils sont équivalents, du fait que l'ensemble des conversations permises {[m_1(+).m_2(+)], [m_1(+).m_3(+)]} est supporté par les deux protocoles. Ils peuvent, ainsi, être utilisés indifféremment.

Cependant, l'examen approfondi de l'équivalence qui prendra en compte les effets des messages, fait ressortir d'autres constats. En effet, en partant de deux états identiques des bases de données de chaque protocole, les états finaux de ces bases peuvent être différents, du fait que les effets cumulés engendrés par l'exécution des deux séquences de messages peuvent êtes différents. Ainsi, les séquences de messages $m_1(+).m_2(+)$ et $m_1(+).m_3(+)$ du protocole P_1 engendre l'ensemble des effets: $\{e_{11}, e_{12}\}$et $\{e_{11}, e_{13}\}$, auxquels correspondent les séquences de requêtes de mise à jour: $\{R_{11} \text{ o } R_{12}\}$ et $\{R_{11} \text{ o } R_{13}\}$, qui peuvent être non équivalentes à celles engendrées par les traces d'exécution de P_2, soient: $\{R_{21} \text{ o } R_{22}\}$ et $\{R_{21} \text{ o } R_{23}\}$.

Les deux protocoles ne seront équivalents, et par conséquent ne peuvent être interchangés d'une manière transparente aux utilisateurs, que si leurs effets soient identiques. Autrement, si les deux séquences de requêtes de mise à jour soient équivalentes. C'est-à-dire, l'exécution séquentielle des deux séquences de requêtes produit le même état de la base, en partant d'un même état initial, soient:

$$(1) \begin{cases} \{R_{11} \text{ o } R_{12}\} \equiv \{R_{21} \text{ o } R_{22}\} & \text{(a)} \\ \{R_{11} \text{ o } R_{13}\} \equiv \{R_{21} \text{ o } R_{23}\}. & \text{(b)} \end{cases}$$

L'examen poussé du test d'équivalence des séquences de requêtes (a) et (b), conduit à deux situations distinctes:

<u>Situation 1</u>: Equivalence stricte des requêtes

Dans cette situation, les deux séquences de requêtes ne seront équivalentes que si, à chaque requête de P_1 correspond, pour le même message de P_2, <u>exactement une seule requête équivalente</u>. Plus formellement:

$$(2) \begin{cases} R_{11} \equiv R_{21} \\ R_{12} \equiv R_{22} \\ R_{13} \equiv R_{23} \end{cases}$$

L'équivalence des requêtes de mise à jour est exprimée par la vérification des conditions suivantes:

(3)
a. Même schéma de la base de données;
b. Même type d'opération de mise jour (***Delete, Insert*** ou ***Modify***);
c. Le même enregistrement est concerné par la mise à jour.
d. Mêmes actions sur l'ensemble des attributs de l'enregistrement concerné par la mise jour.

En se référant à la description formelle des protocoles de services enrichis qui a été exposée dans le chapitre précédent (Section 4.5.2), Si S_{BF1} et S_{BF2} désignent les états finaux des deux bases de données des protocoles P_1 *et* P_2 ; on aura:

$$S_{BF1} \equiv S_{BF2}, \quad (4)$$

Et si S_{BFT1} *et* S_{BFT2} désignent les états transitoires des deux bases de données, on aura:

$$S_{BFT1} \equiv S_{BFT2}, \quad (4')$$

Quelque soit l'état dans lequel se trouve le protocole au cours de ses différentes transitions.

Si le système (2) est vérifié, les séquences de messages invoquées dans l'un ou l'autre des protocoles, aboutiront à des états équivalents des deux bases de données, en partant d'un même état initial. Dans ce cas, les deux bases seront identiques après l'exécution conjointe de n'importe quelle trace d'exécution des protocoles, et les deux protocoles sont **strictement équivalents**, suite à l'équivalence individuelle des requêtes.

Situation 2: Equivalence Convergente.

La situation d'équivalence convergente ne traite pas de l'équivalence individuelle des requêtes de chaque message, ni de l'équivalence des états transitoires des bases; elle concerne simplement l'état final des deux bases de données. Cette convergence s'explique par la différence dans la façon de réaliser les mises à jour à chaque phase du processus, mais les états des deux bases doivent converger vers un état final identique. Dans la réalité, les séquences de certaines opérations, mêmes si elles ne sont pas exécutées dans le même ordre, aboutissent aux mêmes effets finaux dans le monde réel. A titre d'exemple, un fournisseur peut permettre le paiement d'une commande juste après sa confirmation, comme il peut le laisser à la livraison des produits. Dans ce cas, deux protocoles où les produits sont pré ou post payés seront équivalents, du fait qu'ils aboutissent tous les deux à l'accomplissement de l'opération de paiement, même si elle est faite différemment, ce qui conduit à des bases de données équivalentes.

Formellement cette situation est décrite par l'équation: (4) $S_{BF1} \equiv S_{BF2,}$ et ce indépendamment de l'équivalence des requêtes intermédiaires (Indépendamment du système (2)).

On parlera d'**équivalence convergente** pour exprimer le fait que les deux bases de données convergent vers un même état final identique.

En résumé, ce scénario illustre la nécessité d'étendre le critère de test d'équivalence aux effets lors de l'analyse d'équivalence des protocoles de service à effets transactionnels, donc à l'équivalence des requêtes de mise à jour de la base de données et à l'équivalence des états finaux de bases de données.

Scénario 2: Equivalence étendue aux effets de compensation des protocoles

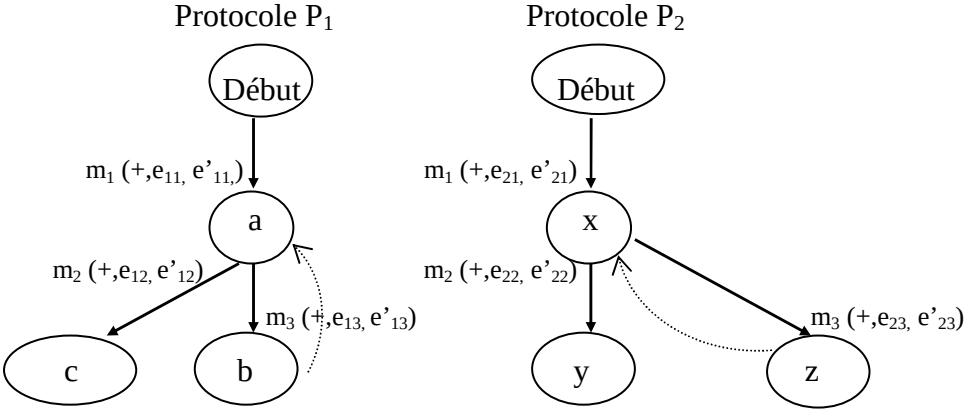


Figure 5.7: P_1 et P_2 sont-ils équivalents en considérant leurs effets de compensation?

La même analyse que pour le scénario précédent peut être faite sur les effets de compensation des deux protocoles. En remarque que les protocoles P_1 et P_2 sont équivalents, du fait que les deux traces d'exécution possibles; $[m_1(+).m_2(+)]$ et $[m_1(+).m_3(+)]$ sont supportées par les deux protocoles.

Mais si en considère les effets de compensation de chaque protocole, l'équivalence n'est plus évidente, du fait que les effets de compensation peuvent être différents. Ainsi, si on considère à titre d'exemple le message m_3, ayant pour effets de compensation e'_{13} pour le protocole P_1 et e'_{23} pour le protocole P_2, les deux requêtes de mise à jour des deux bases de données associées: R'_{13} et R'_{23} peuvent être non équivalents. Dans ce cas, si le protocole P_1 est défaillant (nouvelle version, serveur en panne…), P_2 ne pourra pas le substituer, car les effets du processus de compensation de l'opération d'annulation dans P_2 sont différents de ceux dans P_1. Ce constat s'explique par le fait que les logiques métiers exercées par les fournisseurs de services et relatives au processus de compensation des deux protocoles sont différentes et par conséquents, les deux protocoles ne seront pas équivalents, même s'ils vérifient une équivalence syntaxique exprimée uniquement par les mêmes noms de messages et un respect des contraintes d'ordre. En termes de requêtes de mise à jour, P_1 est équivalent à P_2 si les séquences de requêtes de compensation sont équivalentes. Plus formellement:

$$(5)\ \begin{cases} \{R'_{12} \circ R'_{11}\} \equiv \{R'_{22} \circ R'_{21}\} \\ \{R'_{13} \circ R'_{11}\} \equiv \{R'_{23} \circ R'_{21}\} \end{cases}$$

Notons que l'ordre dans les séquences de requêtes du système (5) est inversé par rapport au protocole déclencheur, pour exprimer le caractère ascendant du mécanisme de compensation. Dans ce contexte, les transactions à compenser affecteront soit la totalité du protocole, ou certaines opérations seulement. Dans le cas de la compensation partielle, nous désignerons par S_{IC}, l'état intermédiaire cible du protocole, auquel est associé un état intermédiaire de la base de données S_{BIC}. En cas d'annulation totale du protocole, cet état cible n'est autre que l'état initial du protocole.

Ce scénario illustre clairement, dans le cas de l'annulation d'une opération, soit par le client soit par le fournisseur, la nécessité de prendre en compte les charges qu'imposent chaque fournisseur de services pour compenser les opérations annulées ou abandonnées par les clients. Il est évident que ces charges diffèrent d'un fournisseur à un autre suivant la logique métier propre à chacun.

De même que pour le **scénario 1**, deux situations d'équivalence des effets de compensation sont envisageables;

<u>Situation 1</u>: Equivalence Stricte des requêtes de compensation

Dans cette situation, les deux séquences de requêtes ne seront équivalentes que si, à chaque requête de compensation de P_1 correspond, pour le même message de P_2, <u>exactement une seule requête équivalente</u>. Plus formellement:

$$(6)\ \begin{cases} R'_{11} \equiv R'_{21} \\ R'_{12} \equiv R'_{22} \\ R'_{13} \equiv R'_{23} \end{cases}$$

L'équivalence des requêtes est exprimée par la vérification des conditions énumérées dans le scénario précédent. (Système 3).

En se référant à la description formelle des protocoles de services enrichis, exposée dans le chapitre précédent (Section 4.5.2), si S_{BIC1} et S_{BIC2} désignent les états des deux bases de données relatifs aux phases intermédiaires cibles S_{IC1} et S_{IC2} du processus de compensation des protocoles P_1 *et* P_2, respectivement, on aura:

$$(7)\ S_{BIC1} \equiv S_{BIC2},$$

Quelque soit l'état dans lequel se trouve le protocole de compensation au cours de ses différentes transitions, avant d'atteindre l'état ciblé par la compensation.
Si les équivalences individuelles des requêtes du système (6) sont vérifiées, les séquences de messages de compensation invoquées, soit dans P_1 ou dans P_2, conduiront à des bases de données identiques, en partant d'un même état initial identique. Dans cette situation, les deux bases seront identiques après exécution de chaque message de compensation, conduisant évidemment, à une équivalence des états intermédiaires ciblés par le processus de compensation. On est dans un cas d'**équivalence stricte par rapport aux requêtes de compensation**.

Situation 2: Equivalence Convergente durant la compensation

Dans cette situation, nous nous intéresserons, particulièrement, à l'état intermédiaire cible visé par la compensation, sans se soucier de la comparaison des requêtes de chaque message de compensation.

Lors du déclenchement d'un processus de compensation, les clients, ou les fournisseurs dans le cas des transactions implicites, visent à revenir à un état antérieur du protocole, en annulant certaines opérations. Donc, la notion d'état intermédiaire est capitale dans ce contexte. Pour vérifier l'équivalence des protocoles à base d'effet lors de la compensation, nous allons tester l'équivalence des bases de données associées à cet état intermédiaire.

Les requêtes associées aux séquences de messages de compensation, peuvent traiter les mises à jour des deux bases de données différemment. Cependant elles peuvent conduire vers une convergence des deux états des bases de données. En effet, les fournisseurs appliquent les charges de compensation, chacun conformément à sa logique métier qui lui est spécifique. A titre d'exemple, un fournisseur procède, lors de l'annulation d'une commande déjà confirmée par le client, aux actions décrites de la manière suivante:

1) Mettre à jour les quantités en stock, suite au retour des livraisons;
2) Procéder à la retenue d'un montant sur le compte client relatif aux pertes de vente;
3) Envoi d'un avis au client;
4) Décrémenter le score de fidélité du client.

Notons qu'il s'agit d'une annulation partielle d'un protocole (le client maintient son abonnement, son contrat …etc.)

Chez un deuxième fournisseur, ces quatre opérations peuvent se faire dans un autre ordre, tout en conduisant au même état de la base de données. Dans ce cas, les deux protocoles de compensation seront équivalents, du fait qu'ils aboutissent à des états équivalents des bases de données intermédiaires.

Formellement cette situation est décrite par l'équation (7): $S_{BIC1} \equiv S_{BIC2}$ indépendamment de l'équivalence des requêtes intermédiaires (Indépendamment de l'équation (6)).

Il s'ensuit que l'analyse d'équivalence des protocoles de service à effets transactionnels doit étendre le critère de test d'équivalence aux effets de compensation des protocoles, donc au test d'équivalence des requêtes de mise à jour associées à l'opération de compensation et aussi au test d'équivalence des bases de données associées aux états intermédiaires cibles.

<u>Scénario</u> 3: Equivalence totale (étendue aux effets et aux effets de compensation)

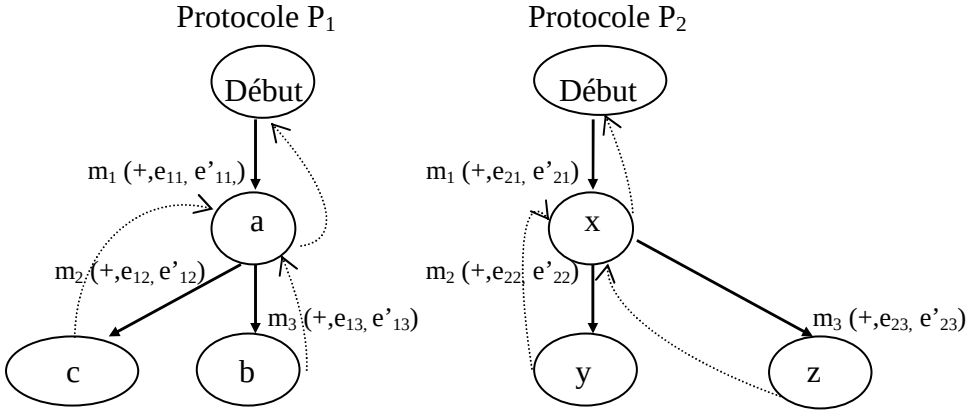


Figure 5.8: P_1 et P_2 sont-ils équivalents en considérant leurs effets et leurs effets de compensation?

Ce scénario combine les deux scénarios précédents. L'analyse d'équivalence des deux protocoles P_1 et P_2 est étendue, aussi bien aux effets des transactions, qu'aux effets de la compensation. Ainsi, au-delà du test d'équivalence syntaxique, les deux protocoles ne seront équivalents, et par conséquents ne peuvent être mutuellement inter-changés, que si leurs effets et leurs effets de compensation seront équivalents. De ce fait, même si les conversations possibles, soient: $[m_1(+).m_2(+)]$ et $[m_1(+).m_3(+)]$, soient supportées par les deux protocoles, ces derniers ne seront parfaitement équivalents sans l'équivalence des requêtes de mise à jour des deux bases de donnée ou sans l'équivalence des états des bases de données cibles, aussi bien pour les protocoles déclencheurs, que pour leur protocoles de compensation associés. En termes de requêtes, les séquences de requêtes du protocole déclencheur et celles du protocole de compensation des deux protocoles P_1 et P_2 sont équivalentes, plus formellement:

$$(8)\quad \begin{cases} \{R_{11} \circ R_{12}\} \equiv \{R_{21} \circ R_{22}\} \\ \{R_{11} \circ R_{13}\} \equiv \{R_{21} \circ R_{23}\}. \\ \{R'_{12} \circ R'_{11}\} \equiv \{R'_{22} \circ R'_{21}\} \\ \{R'_{13} \circ R'_{11}\} \equiv \{R'_{23} \circ R'_{21}\} \end{cases}$$

En définitive, l'équivalence des protocoles à effets transactionnels est déterminée par rapport à l'équivalence des requêtes de mise à jour de la base de données et/ou par rapport l'équivalence des états des bases de données cibles.

Néanmoins, nous pouvons dégager des classes d'équivalence des protocoles, au lieu de les traiter d'une manière binaire (équivalence/ non équivalence). Ces classes d'équivalence prennent en compte les équivalences partielles des protocoles, soit par rapport aux effets, soit par rapport aux effets de compensation.

- **Synthèse sur les scénarios d'analyse d'équivalence**

Vu l'importance de l'analyse d'équivalence des protocoles de services, elle mérite d'être réétudiée dans le nouveau contexte des protocoles enrichis par les contraintes transactionnelles. Les divers scénarios exposés dans cette section, ont montré que l'analyse d'équivalence dans les protocoles à effets transactionnels, modélisés par des requêtes de mise à jour de la base de données, nous a amené vers une exploration en profondeur de l'aspect contenu des bases de données, au-delà de la simple équivalence structurelle et syntaxique. Au moins deux conclusions majeures ont été dégagées à partir des scénarios présentés, à savoir:

- ✓ L'équivalence des protocoles est conditionnée par l'équivalence des états finaux des deux bases de données, dans le cas des protocoles déclencheur, et par l'équivalence des états bases associées aux états intermédiaires cibles dans le cas de la compensation.
- ✓ La vérification de l'équivalence des deux bases de données associées aux deux protocoles est examinée sur la base de l'équivalence des séquences des requêtes de mise à jour, en partant d'un même état initial.

Nous avons décelé deux types d'équivalence pour les séquences de requêtes; des équivalences strictes et des équivalences convergentes. Ces deux types contribueront fortement à la détermination des classes d'équivalence des protocoles.

5.2 Formalisation de l'analyse de compatibilité et d'équivalence des protocoles à effets transactionnels

Dans la section précédente nous avons dégagé la substance et les exigences nécessaires à une nouvelle perception de la compatibilité et de l'équivalence des protocoles de services à effets transactionnels. A présent, nous aborderons la modélisation formelle, en se basant sur le modèle formel des protocoles présenté dans le chapitre précédent (section 5.6.2) et en prenant en considération les différents contextes explorés, à travers les divers scénarios, et que nous considérons exhaustifs.

5.2.1 Compatibilité des protocoles de services à effets transactionnels

La compatibilité des protocoles de services à effets transactionnels diffère de celle des protocoles de base, à cause des effets induits par les messages. En ce sens, une interaction entre deux protocoles de services n'est permise que si les effets observés seront compatibles. Par compatibilité des effets, nous sous-entendons que les requêtes de mise à jour au niveau des bases sont de même type. Cette condition conduit à une redéfinition du concept de chemin d'interaction qui sera étendu au type de requête.

- **Chemin d'interaction étendu:** La définition du chemin d'interaction dans les protocoles de bases est limitée aux messages et aux états. Elle est décrite conformément à

l'expression générique: ***((Etat1.Etat2).Message)****. Pour la prise en compte des effets transactionnels, nous proposons une extension de cette spécification au type de requête de mise à jour, comme suit:

((Etat1.Etat2). Message. Type Requête)*

Cette extension permettra, inévitablement de garantir, lors de l'analyse, la vérification de la compatibilité des requêtes de mise à jour et favorise une spécification plus riche de l'interaction entre les protocoles. Elle offre un dépassement du simple aspect lié aux contraintes d'ordre sur les messages. Ainsi, deux protocoles de services ne pourront être compatibles sans que les requêtes associées aux messages ne le soient. Cette condition est nécessaire pour une interaction correcte entre les protocoles du client et du fournisseur, car elle garantira une conformité des effets produits au niveau des deux bases de données. Le besoin d'une nouvelle spécification de la compatibilité nous conduit à définir les concepts de *messages compatibles* et *Effets compatibles*.

- **Messages compatibles:** Deux messages, appartenant à deux protocoles différents, sont compatibles si les requêtes de mise à jour associées sont de même type. Par exemple: le message m_1 appartenant à un protocole client P_1 et dont la requête de mise à jour est de type *Delete* ne peut interagir avec un protocole fournisseur P_2, que si sa requête associée à m_1 est aussi de type *Delete*. Le cas contraire, il est évident que les effets observés seront non conformes et par conséquent, la conversation ne sera pas possible.
- **Effet compatibles:** Deux effets induits par un ou plusieurs messages sont compatibles si les actions produites au niveau des bases de données sont identiques. Le cas le plus évident de la compatibilité des effets est directement lié à la compatibilité des messages. Cependant, dans certaines situations (scénarios 3 et 4 de la section 5.1.1), les effets sont traités par rapport à la séquence des requêtes. Un cas typique est le traitement d'une requête de modification. Elle est considérée de type *Modify* dans un premier protocole, alors qu'au niveau d'un $2^{ième}$ protocole, c'est une séquence d'une requête *Insert* suivie d'une autre requête de type *Delete*. Dans ce cas la compatibilité des effets est vérifiée malgré l'incompatibilité des messages. Un deuxième exemple concerne la décomposition d'une requête *Modify* sur plusieurs attributs en plusieurs requêtes *Modify* sur des attributs et/ou des sous ensembles d'attributs.

Définition intuitive: Deux protocoles de services P_1 et P_2 sont compatibles, en considérant leurs contraintes transactionnelles, s'ils produisent des effets et des effets de compensation compatibles au niveau des deux bases de données. Autrement, s'ils existent des chemins d'interaction étendus aux effets des messages qui sont supportés par les deux protocoles. Dans ce cas les messages sont compatibles, même d'une façon transitive, et engendrent des conversations correctes.

Définition formelle: Soient P_1 et P_2 deux protocoles de services à effets transactionnels modélisés avec des automates d'états finis comme suit:
$P_1 = (S^1, s^1_0, F^1, M^1, R^1, S^1_B)$ *et* $P_2 = (S^2\ s^2_0, F^2, M^2, R^2, S^2_B)$
P_1 *et* P_2 sont compatibles si*:*

Il existe une conversation entre P_1 *et* P_2 (au moins un chemin d'exécution complet de P_1 est supporté par P_2) dans laquelle les effets et les effets de compensation de P_1 sont compatibles avec ceux de P_2.

Soient m_i le message à invoquer pour la transition courante:

$m_i(p^1, e^1, e^{1\prime}) \in P_1$ dont les requêtes associées aux effets et aux effets de compensation sont R_i et R'_i.

$m_j(p^2, e^2, e^{2\prime}) \in P_2$ dont les requêtes associées aux effets et aux effets de compensation sont R_j et R'_j.

La compatibilité des messages m_i et m_j est vérifiée si:

- Les noms des messages sont identiques ;
- Les états cibles sont identiques; ou transitivement identiques
- Les polarités p^1 *et* p^2 se compensent (+, -) ;
- Les requêtes R_i et R_j associées aux effets sont de même type.
- Les requêtes R'_i et R'_j associées aux effets de compensation sont de même type.

Après avoir défini la compatibilité des effets et des messages, nous pouvons à présent reprendre les définitions de la compatibilité partielle et de la compatibilité totale.

Compatibilité Partielle: Deux protocoles de services à effets transactionnels P_1 et P_2 sont partiellement compatibles s'il existe, au moins, un chemin d'exécution complet qui est supporté par les deux protocoles et dont les effets sont compatibles.

Compatibilité Totale: Un protocole de services à effets transactionnels P_1 est totalement compatible avec un protocole de service à effets transactionnel P_2, si tous les chemins d'exécution complets de P_1 sont supportés par P_2 et leurs effets sont compatibles.

L'analyse de compatibilité des protocoles de services à effets transactionnels est d'une importance capitale dans le processus d'interaction entre un protocole client et celui d'un fournisseur. Nous insistons, dans ce contexte, sur le nouveau concept de *la compatibilité des effets*. Deux situations peuvent se présenter:

- Même type des requêtes de mise à jour: C'est le cas le plus simple. Il correspond à une compatibilité des messages.(mêmes type de requêtes)
- Même type des actions au niveau de la base: C'est le cas le plus général. Il correspond à des situations de transitivités (cas des séquences de requêtes).

5.2.2 Equivalence des protocoles de services à effets transactionnels

L'analyse d'équivalence des protocoles de services à effets transactionnels diffère de celle des protocoles de bases, car elle permet de tester si les deux protocoles sont équivalents en termes d'effets produits dans le monde réel, au lieu de se limiter uniquement à celle des conversations supportées.

Comme les effets dans le monde réel sont modélisés par des requêtes de mise à jour de la base de données, alors nous sommes contraints de vérifier l'équivalence des états des deux bases de données associées aux protocoles en question, pour s'assurer de leur équivalence. Cette question est, à son tour, plus complexe. La vérification de l'équivalence des états finaux des

deux bases de données est réduite à celle du test d'équivalence des séquences de requêtes de mise à jour associées à chaque base, du fait qu'on suppose qu'on démarre d'un état initial identique.

Les scénarios d'analyse d'équivalence exposés dans la section précédente, nous ont permet de cerner le problème de l'analyse d'équivalence des requêtes de mise à jour, cependant une spécification plus formelle reste inévitable. Cette dernière permettra, d'une part, de consolider notre proposition quant à la modélisation des effets par des requêtes de mise à jour, et d'autre part d'offrir un cadre théorique cohérent pour ce type d'analyse d'équivalence des protocoles.

Nous observons clairement sur la figure 5.9, le processus réductionniste de transformation de la problématique de l'analyse d'équivalence des protocoles de services à effets transactionnels, modélisés par des requêtes de mise à jour de la base de données, en un problème de test d'équivalence de ces séquences de requêtes.

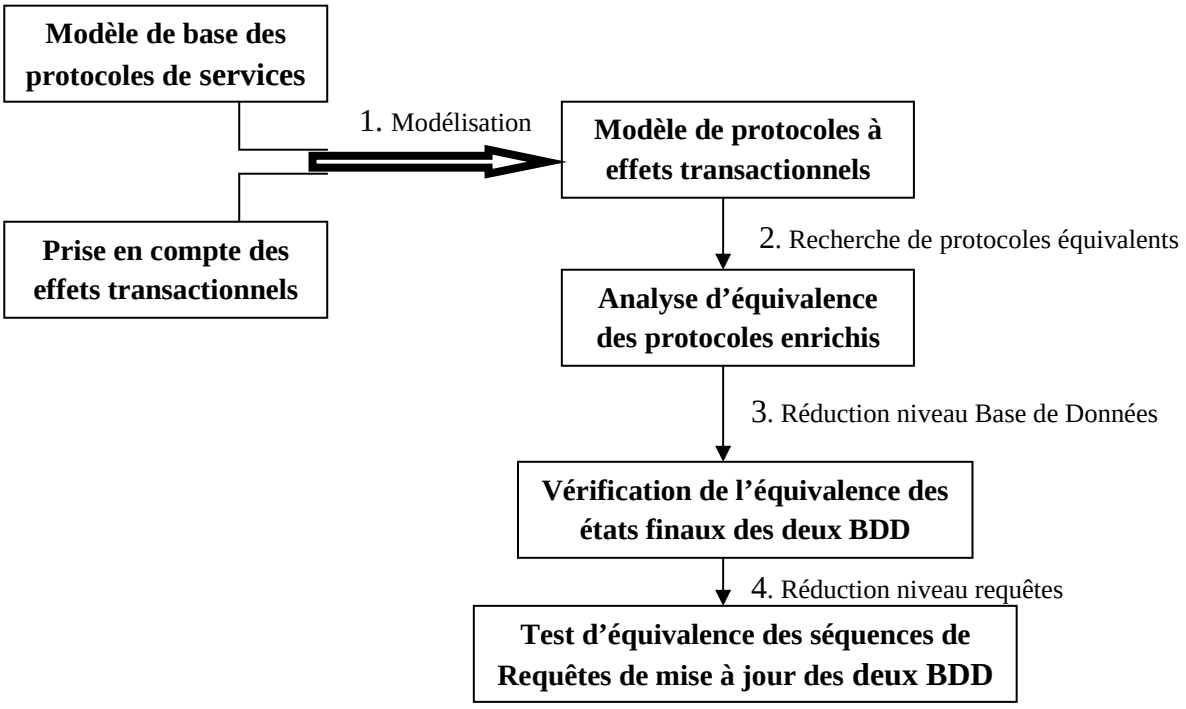


Figure 5.9: Processus d'analyse d'équivalence des protocoles de services à effets transactionnels.

a. Définition de l'équivalence des protocoles à effets transactionnels

Soient P_1 et P_2 deux protocoles de services modélisés avec des automates d'états finis étendus aux états initiaux des deux bases de données, exprimés comme suit:

$P_1 = (S^1, s^1_0, F^1, M^1, R^1, S^1_B)$ et $P_2 = (S^2 s^2_0, F^2, M^2, R^2, S^2_B)$ ayant un état initial identique des deux bases de données, exprimé par $S^1_{BI} \equiv S^2_{BI.}$

Les protocoles P_1 et P_2 sont dits équivalents en termes d'effets si: pour toutes les traces d'exécution complètes possibles de P_1 et P_2, $S^1_{BF} \equiv S^2_{BF}$ où *S^j_{BF} ; $j:1,2$ désigne l'état final de base de données des protocoles P_1 ou P_2.*

Il est important de signaler que cette nouvelle définition de l'équivalence des protocoles de services, peut être satisfaite conjointement avec la définition standard spécifiée dans les protocoles de bases exprimée en termes de conversations supportées [13], comme elle peut constituée, à elle seule, un type d'équivalence particulier basé uniquement sur les effets des transactions. Néanmoins, nous estimons que dans le contexte de la substitution des services, l'équivalence à base d'effet est plus rentable que celle basée sur les conversations, car elle s'adapte parfaitement à la finalité recherchée par le service, abstraction faite à la façon d'atteindre cette finalité. Plus explicitement, sans se soucier de l'ordre des séquences de requêtes de mise à jour. Donc, elle peut être exploitée dans le contexte de la substitution, même si l'équivalence syntaxique par rapport aux messages n'est pas satisfaite.

b. Définition de l'équivalence des protocoles de compensation à effets transactionnels

Soient P_1 et P_2 deux protocoles de services modélisés avec des automates d'états finis étendus aux états initiaux des deux bases de données, exprimés comme suit:

$P_1 = (S^1, s^1_0, F^1, M^1, R^1, S^1_B)$ et $P_2 = (S^2, s^2_0, F^2, M^2, R^2, S^2_B)$ ayant un état initial identique des deux bases de données associées aux états intermédiaires source; exprimé par $S^1_{BI} \equiv S^2_{BI}$.

P_1 et P_2 sont dits équivalents en termes d'effets de compensation si: Pour toutes les opérations de compensations, $S^1_{BIC} \equiv S^2_{BIC}$ où *S^j_{BIC} ; j:1,2 désigne l'état de la base de données associé à l'état intermédiaire ciblé par les protocoles de compensation de P_1 et P_2,, respectivement.*

Cette définition, même si elle présente de grandes ressemblances avec celle des protocoles à effets transactionnels (Définition 5.2.2.a), révèle des aspects très contradictoires, concernant:

1. **Les Etats:** les états des deux bases de données, dont il faut tester l'équivalence, sont les états finaux d'une trace d'exécution complète pour la première, mais concernent les états intermédiaires ciblés par la compensation pour la deuxième. En effet, durant le processus de compensation, on essaye toujours de revenir à un état antérieur intermédiaire ou à l'état initial du protocole (cas d'annulation total), suite à l'annulation d'une ou de plusieurs opérations, soit par le client, soit implicitement par le fournisseur.
2. **Les Effets:** Le modèle d'effets proposé représente le message par le triplet ***m (p, e, e')***; où *e* et *e'* sont les effets affectant le monde du client, exprimés par les requêtes de mise à jour de la base de données et définis préalablement par le fournisseur conformément à sa logique métier. Pour la première définition, seules les requêtes associées aux effets ***e*** seront exécutées. Par contre, seules les requêtes associées aux effets de ***e'*** seront exécutées durant la compensation. La formalisation des effets et de leurs effets de compensation par un modèle d'attributs liés (***e, e'***) est justifiée par un souci de préserver le lien intrinsèque qui associe les deux types d'effets et n'induit, en aucun cas, leur exécution simultanée.
3. **Les requêtes de mise à jour:** Dans la définition 5.2.2.a, les séquences de requêtes sont exécutées dans l'ordre descendant, exprimant ainsi, l'évolution du protocole vers l'état final, par contre dans la définition 5.2.2.b, elles sont exécutées dans le sens inverses pour rejoindre un état intermédiaire cible, en partant d'un état déjà atteint.

5.3 Classes d'équivalence des protocoles de services à effets transactionnels

Lors de notre analyse d'équivalence, nous avons conclu que le test d'équivalence de deux protocoles de services à effets transactionnels a été ramené à celui du ***test d'équivalence des états finaux*** des bases de données associés (Figure 5.9).

Une classe d'équivalence, dans le contexte des protocoles à effets transactionnels, définit un ensemble de protocoles de services qui sont équivalents en termes d'effet dans le monde réel, et dont la structure des séquences de requêtes de mise à jour obéit à un critère particulier.

Suite à l'analyse structurelle et sémantique des séquences de requêtes associées aux messages, nous avons pu distinguer deux types de classes d'équivalence des protocoles de services à effets transactionnels; à savoir: Classe d'équivalence stricte et classe d'équivalence à convergence d'état, que nous détaillerons dans la suite.

5.3.1 Classe des protocoles de service strictement équivalents

Cette classe d'équivalence est caractérisée par ***une équivalence stricte*** des requêtes de mise à jour associées à chaque message du protocole, conduisant par conséquent, à une équivalence des séquences de requêtes, et par la même, à deux états finaux identiques des deux bases de données des deux protocoles de services en question.

Si P_1 et P_2 sont les deux protocoles à effets transactionnels, nous pouvons formaliser cette classe d'équivalence par l'équation d'équivalence des requêtes suivante:

$R_{1i} \equiv R_{2i}$; $\forall$ *i l'ordre de la requête correspondante aux effets associés au message d'ordre i*. (10)

Cette équation conduit, inévitablement à une équivalence des états finaux des deux bases de données, du fait qu'on démarre d'un état initial identique.

Dans ce cas, les deux protocoles P_1 et P_2 peuvent être substitués dans n'importe quel contexte, et d'une manière transparente à l'utilisateur du service.

Exemple 1: Soient P_1, P_2 et P_3 trois protocoles de services à effets transactionnels de la figure 5.10.

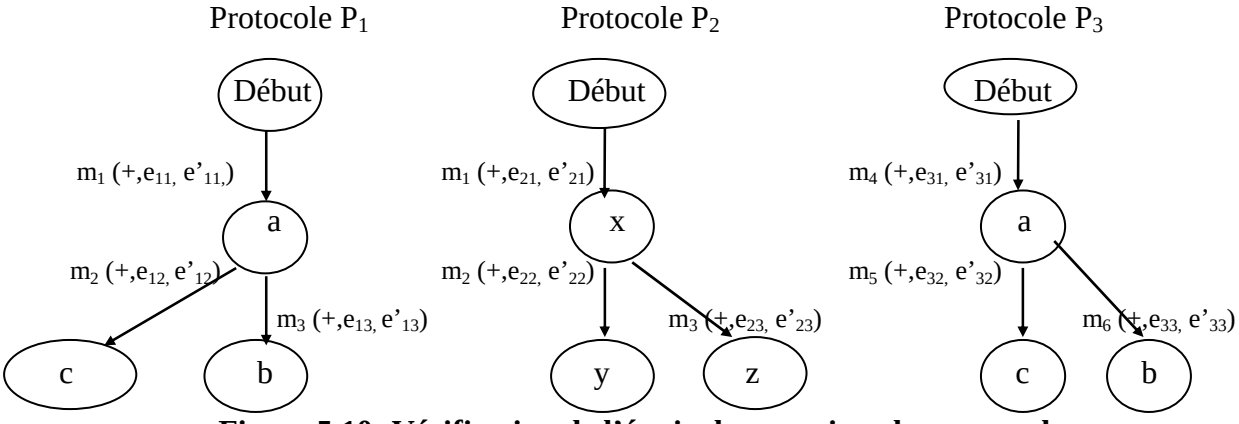


Figure 5.10: Vérification de l'équivalence stricte des protocoles

Pour éclaircir notre définition de l'équivalence stricte, examinons l'équivalence des trois protocoles à effets transactionnels P_1, P_2 et P_3.

Conformément à notre modèle de représentation des effets, nous associons aux effets de chaque message des trois protocoles, une requête de mise à jour de la base de données, comme illustré dans le tableau 5.1 suivant:

Protocole	Message	Effets	Requête	Type de Requête
P_1	m_1	e_{11}	R_{11}	*Insert (I, a_1, a_2,…,a_n)*
P_1	m_2	e_{12}	R_{12}	*Modify (I, a_1, a_2,…,a_n)*
P_1	m_3	e_{13}	R_{13}	*Delete (I)*
P_2	m_1	e_{21}	R_{21}	*Insert (I, a_1, a_2,…,a_n)*
P_2	m_2	e_{22}	R_{22}	*Modify (I, a_1, a_2,…,a_n)*
P_2	m_3	e_{23}	R_{23}	*Delete (I)*
P_3	m_4	e_{31}	R_{31}	*Modify (I, a_1, a_2,…,a_n)*
P_3	m_5	e_{32}	R_{32}	*Modify (I, a_1, a_2,…,a_n)*
P_3	m_6	e_{33}	R_{33}	*Delete (I)*

Tableau 5.1: Exemple de requêtes associées aux effets des messages

Nous remarquons que les deux protocoles de service P_1 et P_2 sont ***strictement équivalents en*** termes d'effets transactionnels, car le système d'équivalence des requêtes (2) (Voir Scénario d'équivalence 1, situation 1) est vérifié. Dès lors, l'exécution conjointe dans P_1 et P_2 des séquences de requêtes relatives à n'importe quelle trace d'exécution complète conduira au même état de la base de données. Par ailleurs, P_1 et P_2 sont aussi, équivalents en termes de conversations supportées.

Contrairement à P_1 et P_3 qui ne sont pas équivalents en termes d'effets, abstraction faite à la différence des noms de messages, et ce à cause de la divergence des requêtes de mise à jour associées aux messages, même si la requête R_{33} est identique à R_{13} (*Delete (I)*). Dans ce cas les états finaux des deux bases de données seront différents. En plus, P_1 et P_3 ne sont pas équivalents en termes de conversations supportées, à cause de la différence des noms de messages. Alors, en aucun cas, P_1 et P_3 ne peuvent être inter-changés.

Lemme 1: *Si deux protocoles de services à effets transactionnels sont strictement équivalents et si les noms de messages sont identiques et dotés des mêmes polarités, alors les deux protocoles sont équivalents en termes de conversations supportées.*

Preuve par l'absurde. (Non équivalence des conversations implique une contradiction)

Supposons que deux protocoles P_1 et P_2 sont strictement équivalents en termes d'effets et qu'ils ne sont pas équivalents en termes de conversations supportées. Il existe, alors, au moins un message qui est supporté par P_1 mais pas par P_2. La requête associée à ce message aura des effets qui ne seront pas observés parallèlement dans le protocole P_2, même s'ils sont observés au niveau d'autres messages de P_2. Il en découle qu'il n'existe pas de requête équivalente pour P_2. Conformément à la définition de l'équivalence stricte, il s'ensuit, donc, que P_1 et P_2 ne sont pas strictement équivalents. Ce qui est contradictoire avec l'hypothèse de départ. On en déduit que: P_1 et P_2 ne peuvent être qu'équivalents en termes de conversations supportées.

5.3.2 Protocoles à équivalence convergente des états des bases de données

Contrairement à l'équivalence stricte, cette classe d'équivalence est caractérisée par une équivalence des états finaux des deux bases de données associées aux deux protocoles, sans que les requêtes correspondantes aux messages ne le soient individuellement.

L'équivalence des états finaux des bases de données ne peut être que le résultat d'une équivalence des séquences des requêtes de mise à jour associées aux messages.

Si P_1 et P_2 sont les deux protocoles à effets transactionnels, cette classe d'équivalence est formalisée par l'équation d'équivalence des séquences de requêtes suivante:

$R_{11} \circ R_{12} \circ \ldots R_{1i} \equiv R_{21} \circ R_{22} \circ \ldots R_{2j}$; $\forall i, \forall j$ *l'ordre des requêtes d'une même trace d'exécution complète*. (11)

L'opérateur «o » désigne le séquencement des requêtes, en déterminant une relation d'ordre d'exécution.

Cette équation conduit, inévitablement, à une équivalence des états finaux des deux bases de données, du fait qu'on démarre d'un état initial identique.

Dans ce cas, les deux protocoles P_1 et P_2 peuvent être substitués dans n'importe quel contexte, et d'une manière transparente à l'utilisateur du service.

Exemple 2: Soit P_4 le protocole de services à effets transactionnels de la figure 5.11:

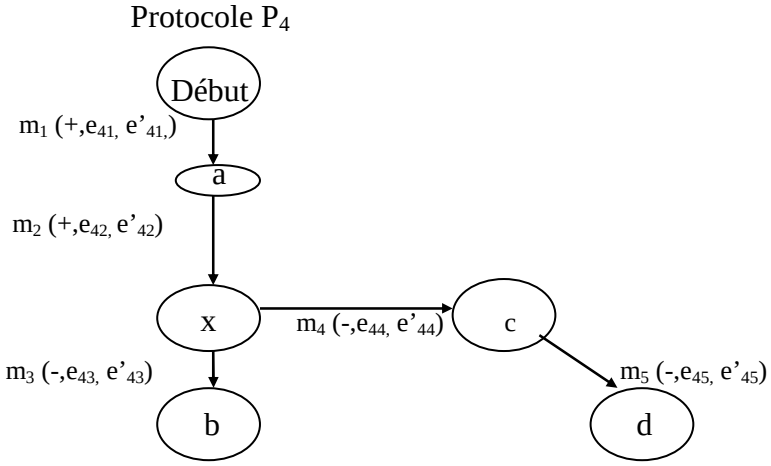


Figure 5.11: Un Protocole à équivalence convergente de l'état de la base avec P_1

Les requêtes de mise à jour associées aux effets des messages de P_4 sont:

$e_{41} \longrightarrow R_{41}$: *Insert (I, a_1, a_2,…,a_n)*

$e_{42} \longrightarrow R_{42}$: *Delete (I)*

$e_{43} \longrightarrow R_{43}$: *Modify (I, a_1, a_2,…,a_n)*

$e_{44} \longrightarrow R_{44}$: *Modify (I, a_1, a_2,…,a_n)*

$e_{45} \longrightarrow R_{45}$: *Delete (I)*

Les séquences de requêtes correspondantes aux deux traces d'exécution complète possibles sont:

Séquence 1: (état final b): $R_{41} \circ R_{42} \circ R_{43}$

Séquence 2: (état final d): $R_{41} \circ R_{42} \circ R_{44} \circ R_{45}$

Si ces deux séquences sont équivalentes aux deux séquences des mêmes traces d'exécution complètes du protocole P_1 de la figure 5.10, soit:

$$\begin{cases} R_{41} \circ R_{42} \circ R_{43} \equiv R_{11} \circ R_{12} \\ R_{41} \circ R_{42} \circ R_{44} \circ R_{45} \equiv R_{11} \circ R_{13} \end{cases}$$

Alors, leur exécution conduira aux mêmes états des bases de données, en partant d'un même état initial. Par conséquent, les deux protocoles P_1 et P_2 sont équivalents à convergence des états de leur base.

L'équivalence des séquences de requêtes s'explique par une façon de faire différente exercée par les fournisseurs de services. Elle est concrétisée au niveau de la base de données par des requêtes différentes, soit dans leur type, soit dans leur ordre, tout en aboutissant à des bases identiques.

A titre d'exemple, un fournisseur peut effectuer sa mise à jour en une seule opération *Modify* affectant plusieurs attributs, un autre peut la faire en plusieurs opérations *Modify* portant sur des sous ensembles d'attributs.

Un autre exemple est la façon de faire une modification d'un attribut d'un enregistrement par les fournisseurs. Elle peut être réalisée en une seule opération *Modify*, ou faite en deux requêtes consécutives: *Insert* puis *Delete*.

Remarque: *L'équivalence stricte est un cas particulier de l'équivalence à convergence des états des bases de données.*

5.4 Problème de l'équivalence des requêtes de mise à jour

Tout au long de notre étude nous étions contraints de vérifier l'***équivalence de requêtes de mise à jour*** des bases de données. Dans cette section spéciale, nous allons aborder cet aspect d'une manière détaillée en expliquant les cas d'équivalence des requêtes. Il est à préciser que nous parlons d'équivalence au lieu d'égalité des requêtes, à cause de la représentation ensembliste des relations dans le modèle de base de données relationnel, qui n'impose pas de relation d'ordre sur les attributs de la relation. A titre d'exemple les deux requêtes R_1 et R_2 suivantes sont équivalentes au sens base de données:

R_1 = *Modify* (**NUMSS**, Nom, Prénom, Date-Naissance)

R_2 = *Modify* (**NUMSS**, Date-Naissance, Prénom, Nom)

Dans notre modèle, la représentation des effets des transactions est fondée sur leur perception en tant que requêtes de mise à jour de la base de données de type: *Insert (I, a_1, a_2,…, a_n), Delete (I), Modify (I, a_1, a_2,-,…,-,…, a_n)* où *I* est l'identifiant et les a_i sont les attributs de l'enregistrement concerné par la mise à jour. Le symbole «- » exprime que l'attribut n'est pas concerné par la mise à jour. Dans ce contexte, deux requêtes de mise à jour, R_1 et R_2 sont équivalentes si elles vérifient les quatre conditions du système (9).

(9)
a. Elles agissent sur des schémas de bases de données identiques;
b. Elles réalisent le même type d'opération de mise jour (***Delete, Insert*** ou ***Modify***);
c. Elles traitent le même enregistrement concerné par la mise à jour ($I_1 = I_2$);
d. Les actions sur l'ensemble des attributs de l'enregistrement concerné par la mise jour sont identiques.

Pour éclaircir notre propos, soient R_1 et R_2 deux requêtes de mise à jour de la base de données. Différentes situations peuvent se présenter:

(i) Si $R_{1=}$ *Insert* $(\boldsymbol{I_1}, a_{11}, a_{12},\ldots, a_{1n})$, alors pour que $R_{2=}$ *Opération* $(\boldsymbol{I_2}, a_{21}, a_{22},\ldots, a_{2n})$ soit équivalente à R_1, il faut que le système (9) soit vérifié, c'est-à-dire:

a. Les schémas des relations impliquées dans R_1 et R_2 sont identiques;
b. *Opération* de *R2* est de type *Insert;*
c. Les identifiants des enregistrements sont égaux $(I_1 = I_2)$;
d. $\forall$ l'attribut a_{1i} de R_1, il existe un attribut a_{2j} de R_2 tel que: $a_{1i} = a_{2j}$

(ii) Si $R_{1=}$ *Delete* $(\boldsymbol{I_1})$, alors pour que $R_{2=}$ *Opération* $(\boldsymbol{I_2}, a_{21}, a_{22},\ldots, a_{2n})$ soit équivalente à R_1, il faut que le système (9) soit vérifié, c'est-à-dire:

a. Les schémas des relations impliquées dans R_1 et R_2 sont identiques;
b. *Opération* de R2 est de type *Delete*;
c. Les identifiants des enregistrements sont égaux $(I1 = I2)$;
d. Abstraction des noms d'attributs de R_2.(c'est une supression à base d'identifiant).

(iii) Si $R_{1=}$ *Modify* $(\boldsymbol{I_1}, a_{11}, a_{12},\ldots, a_{1n})$, alors pour que $R_{2=}$ *Opération* $(\boldsymbol{I_2}, a_{21}, a_{22},\ldots, a_{2n})$ soit équivalente à R_1, il faut que le système (9) soit vérifié, c'est-à-dire:

a. Les schémas des relations impliquées dans R_1 et R_2 sont identiques;
b. Opération de R_2 est de type *Modify*;
c. Les identifiants des enregistrements sont égaux $(I_1 = I_2)$;
d. Les mêmes mises à jours des attributs impliqués.

La vérification du point (d) n'est pas évidente dans le cas des requêtes de type *Modify*. Pour pouvoir traiter ce cas, nous allons le ramener à une séquence de deux cas simples *(i) et (ii)*. Nous considérons une opération de modification, comme une séquence de deux opérations. Une opération de suppression de l'ancien enregistrement à modifier, après sauvegarde des anciennes valeurs des attributs concernés par la mise à jour, suivie d'une opération d'insertion du nouvel enregistrement affecté des mises à jour voulues. La formalisation de cette transformation est:

Modify $(\boldsymbol{I_1}, a_{11}, a_{12},\ldots,\text{-}, a_{1i}, \text{-}, a_{1j}, a_{1n}) \equiv$ *Delete* $(\boldsymbol{I_1})$ *o Insert* $(\boldsymbol{I_1}, a'_{11}, a'_{12},\ldots,\text{-}, a'_{1i}, \text{-}, a'_{1j}, a'_{1n})$

Où: *o* désigne l'opérateur de séquencement des requêtes, ″-″: Un attribut non concerné par la mise à jour et les $a'_{11}, a'_{12}, a'_{1i}, a'_{1j}, a'_{1n}$ sont les nouvelles valeurs des attributs après traitement des expressions algébriques relatives à la mise à jour.

Une fois le problème de test d'équivalence des requêtes de mise à jour résolu, nous pouvons aborder l'étude algorithmique de l'équivalence des protocoles de services, avec comme préalable, l'existence de primitives qui permettront de décider de l'équivalence des requêtes en réalisant les transformations adéquates.

5.5 Algorithmes d’équivalences

Dans cette section, nous aborderons l’étude algorithmique de l’équivalence de deux protocoles de services à effets transactionnels. Cette étude est d’une importance capitale dans le contexte de notre travail. En effet, elle permettra d’une part, de vérifier si deux protocoles de services à effets transactionnels sont équivalents, sans passer par le processus laborieux de comparaison des états finaux des bases, qui n’est pas évident à cause du volume de données à tester. D’autre part, les algorithmes présentés doteront le middleware de l’infrastructure des services Web des outils logiciels permettant de chercher s’il existe un service équivalent en termes d’effets, afin de substituer un service défaillant. Nous aborderons, en fin de section, l’examen de la robustesse des algorithmes proposés en vérifiant leur test d’arrêt et leur solidité.

5.5.1 Algorithmes d’équivalence stricte

Cet algorithme répond à un problème de décision (Algorithme décidable). Il a en entrée deux protocoles de services à effets transactionnels, dont les états initiaux de leurs bases de données sont équivalents, et donnera en sortie une réponse positive ou négative sur leur équivalence stricte. Nous traiterons pour ce type d’algorithmes, deux situations différentes: Protocoles avec mêmes noms de messages, sans tenir compte de leurs états (Algorithme 1) et protocoles avec mêmes états, abstraction faite aux noms de messages (Algorithme 2).

Algorithme 1 Equivalence-Stricte-Mêmes-Noms-Messages

Entrée: Deux protocoles à effets transactionnels

$P_1 = (S^1, s^1_0, F^1, M^1, R^1, S^1_B)$ et P2= $(S^2, s^2_0, F^2, M^2, R^2, S^2_B)$

Avec $s^1_{0B} \equiv s^2_{0B}$ // Equivalence des états initiaux de deux bases de données.

Et les deux protocoles sont dotés des mêmes noms de messages

Sortie: Décision sur l'équivalence Stricte des protocoles P_1 et P_2

1. *Etat-testés: = Ø, Etats-Courant:= {(s^1_0, s^2_0)}, Continuer:= Vrai*
2. *Si $M^1 \neq M^2$ ALORS pas d'équivalence de messages ; Continuer:= Faux*
3. ***Tant que Existe (****$Etat^1, Etat^2) \in$ Etats-Courant* ***et (****Etat-testés $\neq S^1 \setminus F^1$)* ***et*** *Continuer=Vrai* ***Faire***
4. *s^1-test:= $Etat^1$; s^2-test:= $etat^2$*
5. *Etat-testés:= Etat-testés U {****(s^1****-test, s^2-test)}*
6. ***Pour tout*** *message ($m_{1i} \in M^1 \mid R^1$ ((s^1-test, $s_s{}^1{}_{\text{-test}}$), (s^1-cible, $s_s{}^1{}_{\text{-cible}}$), m_{1i})* ***Et***

   $m_{2j} \in M^2 \mid R^2$ ((s^2-test, $s_s{}^2{}_{\text{-test}}$), (s^2-cible, $s_s{}^2{}_{\text{-cible}}$), m_{2j}) ***Et*** $m_{1i=}\ m_{2j}$) ***Faire***
7. *Récupérer la requête R_{1i} associée aux effets e_{1i} de la transition de m_{1i} ($p, e_{1i,} e'_{1i,}$)*
8. *Récupérer la requête R_{2j} associée aux effets e_{2j} de m_{2j} ($p, e_{2j,} e'_{2j,}$)*
9. ***Si*** *$R_{1\,i} \not\equiv R_{2j}$* ***Alors*** // non équivalence des requêtes
10. *P_1 et P_2 ne sont pas strictement équivalents*
11. *continuer:= Faux* // arrêt de l'algorithme avec échec
12. ***Finsi***
13. *Etats-Courant:= Etats-Courant U {****(s^1****-cible, s^2-cible)}* // Passer à l'étape suivante
14. ***FinPour***
15. *Etats-Courant:= Etats-Courant \ Etat-testés* // Tester les états restants
16. ***Fin Tant que // tous les messages sont testés***
17. ***Si*** *Etat-testés = $S^1 \setminus F^1$* ***Alors*** *P_1 et P_2 sont strictement équivalents* // sortie avec tests positif, tous les états sauf finaux sont testés.
18. ***Fin Equivalence-Stricte-*** **Mêmes-Noms-Messages.**

Dans la réalité, mêmes les noms de messages peuvent être différents dans les deux protocoles, bien que leurs effets soient les mêmes. Alors, nous proposons une deuxième variante de l'algorithme qui est basée sur les mêmes noms d'états pour les deux protocoles.

Algorithme 2 Equivalence-Stricte-Mêmes-Noms-Etats

Entrée: Deux protocoles à effets transactionnels

$P_1 = (S^1, s^1_0, F^1, M^1, R^1, S^1_B)$ et P2= $(S^2, s^2_0, F^2, M^2, R^2, S^2_B)$

Avec $s^1_{0B} \equiv s^2_{0B}$ // Equivalence des états initiaux de deux bases de données.

Et les deux protocoles sont dotés des mêmes noms d'états

Sortie: Décision sur l'équivalence Stricte P_1 et P_2

1. *Etat-testés: = Ø, Etats-Courant:= {(s^1_0, s^2_0)}, Continuer:= Vrai*
2. *Si $S^1 \neq S^2$ ALORS pas d'équivalence d'états ; Continuer:= Faux*
3. ***Tant que Existe (****$Etat^1, Etat^2$)$\in$Etats-Courant **et (***Etat-testés$\neq S^1 \setminus F^1$***)et** Continuer=Vrai **Faire***
4. s^1*-test:= $Etat^1$; s^2-test:= $Etat^2$*
5. *Etat-testés:= Etat-testés U {**(**s^1-test, s^2-test)}*
6. ***Pour tout** message ($m_{1i} \in M^1$ | R^1 ((s^1-test, $s_s{}^1{}_{\text{-test}}$), (s^1-cible, $s_s{}^1{}_{\text{-cible}}$), m_{1i}) **Et***

   *$m_{2j} \in M^2$ |R^2 ((s^2-test, $s_s{}^2{}_{\text{-test}}$), (s^2-cible, $s_s{}^2{}_{\text{-cible}}$), m_{2j}) **Et** s^1-test = s^2-test) **Faire***
7. *Récupérer la requête R_{1i} associée aux effets e_{1i} de la transition de m_{1i} ($p, e_{1i,} e'_{1i,}$)*
8. *Récupérer la requête R_{2j} associée aux effets e_{2j} de m_{2j} ($p, e_{2j,} e'_{2j,}$)*
9. ***Si** $R_{1\,i} \not\equiv R_{2j}$ **Alors*** // non équivalence des requêtes
10. *P_1 et P_2 ne sont pas strictement équivalents*
11. *continuer:= Faux* // arrêt de l'algorithme avec échec
12. ***Finsi***
13. *Etats-Courant:= Etats-Courant U {**(**s^1-cible, s^2-cible)}* // Passer à l'étape suivante
14. ***FinPour***
15. *Etats-Courant:= Etats-Courant \ Etat-testés* // Tester les états restants
16. ***Fin Tant que // tous les messages sont testés***
17. ***Si** Etat-testés = $S^1 \setminus F^1$ **Alors** P_1 et P_2 sont strictement équivalents* // sortie avec tests +
18. ***Fin Equivalence-Stricte-*** **Mêmes-Noms-Messages.**

*NB: Il est possible de combiner les 2 algorithmes en un seul: mêmes noms de messages et mêmes noms d'états. Il suffit de changer la condition de test de la boucle **Pour** de la ligne 6, en ($m_{1i=}\, m_{2j}$ **et** s^1-test = s^2-test).*

- **Fonctionnement des Algorithmes d'équivalence stricte (Algorithme 1, Algorithme 2)**

Les deux algorithmes de test d'équivalence stricte sont basés sur l'exploration des deux automates représentant leur protocole de service, en partant des deux états initiaux ayant des bases de données équivalentes. Pour chaque couple d'états ($Etat^1$, $Etat^2$) appartenant à P_1 et P_2, respectivement, sauf pour les états finaux et tant que le test est encore positif (Ligne3), un test d'égalité des messages de transition vers l'ensemble des états cibles est effectué. Si c'est le cas, alors une extraction des requêtes associées aux deux messages est opérée, du fait que le modèle que nous avons proposé permet, justement, de représenter les effets par des requêtes de mise à jour de la base de données. Si les deux requêtes ne sont pas équivalentes, le traitement s'arrête (Ligne 11) en affectant la valeur *Faux* à la variable *Continuer* pour permettre la sortie de la boucle principale. Le cas contraire, l'exploration parallèle des automates continue en basculant vers les états cibles qui deviennent les nouveaux états courants à tester (Ligne 13 et 15), tout en excluant les états déjà testés. Dans le cas ou la sortie de la boucle principale est réalisée suite à l'échec de la condition: *Etat-testés* $\neq S^1 \backslash F^1$, exprime que touts les états sont testés (sauf finaux) et ont donné un résultat positif (existence de requête équivalente pour chaque message). Ce qui induit que les deux protocoles sont strictement équivalents.

- **Test d'arrêt des Algorithmes: Test de Convergence**

Pour vérifier la terminaison des deux algorithmes, nous analyserons leur structure. La boucle ***Tant Que*** (Ligne 3 à 16) permet de parcourir tous les états du protocole P_1 et les états qui sont associés aux mêmes noms de messages dans P_2. L'ensemble des états testés (Variable: *Etat-testés*) est initialisé à vide (1ère instruction de la Ligne 1), et la boucle s'arrêtera si cet ensemble est égal à l'ensemble des états du protocole diminué de ceux des états finaux, qu'il ne faut pas tester ou si la condition de test d'équivalence n'est pas vérifiée (*Continuer=Faux*). La ligne 13 permet d'ajouter un état cible, à partir d'un état en cours de test à l'ensemble des états à tester durant la prochaine itération, permettant ainsi, une exploration des états des deux automates. La ligne 15, garanti la terminaison de la boucle principale de test, car elle ôte l'ensemble des états déjà testés précédemment (variable *Etats-testés*), de l'ensemble des états à tester pour la prochaine itération (*Etats-courants*), ce qui assure une terminaison de la boucle une fois tout les états des automates testés (Nombre fini d'itération). L'ensemble des états testés est augmenté à chaque itération des états parcourus (ligne 5). Quant à la deuxième boucle (Ligne 6 à 14), elle permet de tester, à partir d'un état en cours de P_1, tous les messages permettant une transition vers des états cibles dans P_2. Comme ces états de destination, sont, évidemment limités, alors la boucle se terminera, une fois tous les messages vers les états cibles (s^1-*cible* et s^2-*cible*) seront examinés.

Donc, nous voyons bien que l'algorithme est déterministe et se termine en un nombre fini d'itérations et les deux boucles ne pourront s'exécuter indéfiniment.

5.5.2 Algorithme d'équivalence à convergence des états de la base

Cet algorithme permet de vérifier si deux protocoles à effets transactionnels sont équivalents en considérant les effets produits au niveau des deux bases de données, abstraction faite aux noms de messages et aux noms des états.

Algorithme 3 Equivalence-Convergente-Etats-BDD

Entrée: Deux protocoles à effets transactionnels

$P_1 = (S^1, s^1_0, F^1, M^1, R^1, S^1_B)$ et P2= $(S^2, s^2_0, F^2, M^2, R^2, S^2_B)$

Avec $s^1_{0B} \equiv s^2_{0B}$ // Equivalence des états initiaux de deux bases de données.

Sortie: Décision sur l'équivalence à convergence des bases de P_1 et P_2

1. *Continuer:= Vrai, Trouver:= Faux*
2. *Etat-testés1: = Ø, Etats-Courant1:= $\{s^1_0\}$*
3. *s^1-test:= s^1_0 ; s^2-test:= s^2_0*
4. *Séquence1-Requêtes:= Ø, Séquence2-Requêtes:= Ø*
5. ***Tant que** Existe message $m_{1i} \in$ Etats-courant1 **Et** Continuer=Vrai **Et***

   *$m_{1i} \in M^1 \mid R^1$ ((s^1-test, $s_{s^1\text{-test}}$), (s^1-cible, $s_{s^1\text{-cible}}$), m_{1i}) **Faire***
6. *Récupérer la requête R_{1i} associée aux effets e_{1i} de la transition de m_{1i} (p, $e_{1i,}$ $e'_{1i,}$)*
7. *Séquence1-Requêtes:= Séquence1-Requêtes o R_{1i} // séquence*
8. *Etat-testés1:= Etat-testés1 U $\{s^1$-test$\}$*
9. ***Si** s^1-cible $\in F^1$ **Alors** // Chemin complet de P_1*
10. *Séquence2-Requêtes:= Ø ; s^2-test:= s^2_0*
11. ***Tant que** Existe $m_{2j} \in$ Etats-courant2 **Et** Trouver=Faux **Et***

    *$m_{2j} \in M^2 \mid R^2$ ((s^2-test, $s_{s^2\text{-test}}$),(s^2-cible, $s_{s^2\text{-cible}}$), m_{2j}) **Et** s^2-test $\in S^2 \setminus F^2$ **Faire***
12. *Récupérer la requête R_{2j} associée aux effets e_{2j} de m_{2j} (p, $e_{2j,}$ $e'_{2j,}$)*
13. *Séquence2-Requêtes:= Séquence2-Requêtes o R_{2j} // séquences équivalente*
14. *Etat-testés2:= Etat-testés2 U $\{s^2$-test$\}$*
15. ***Si** s^2-cible $\in F^2$ **Alors** // Chemin complet de P_2*
16. ***Si** Séquence1-Requêtes =Séquence2-Requêtes **Alors** Trouver:= Vrai*
17. *Séquence2-Requêtes:= Ø*
18. *s^2-test:= s^2_0*
19. *Etats-Courant2:= s^2_0*
20. *Etat-testés2:= Ø*
21. ***Finsi***
22. *Etats-Courant2:= Etats-Courant2 U $\{s^2$-cible$\}\setminus$ Etat-testés2*
23. ***Fin Tant que***
24. ***Si** Trouver = Faux **Alors** Continuer:= Faux // Aucune séquence équivalente*
25. ***Finsi***
26. *Etats-Courant1:= Etats-Courant1 U $\{s^1$-cible$\} \setminus$ Etat-testés1*
27. ***Fin Tant que***
28. ***Si** Continuer=Vrai **Alors** P_2 est équivalent à P_1 avec convergence des états des BDD*
29. ***Fin Equivalence*-Convergente-Etats-BDD.**

- **Fonctionnement de l'Algorithmes d'équivalence à convergence des états des bases (Algorithme 3)**

L'Algorithme à convergence des états des bases de données des deux protocoles en entrée permet de tester s'ils auront des bases de données équivalentes pour tous les chemins d'exécution possibles, en partant des états équivalents des bases ($s^1_{0B} \equiv s^2_{0B}$).
La boucle principale (Ligne 5 à 27) vérifie pour chaque chemin d'exécution complet du 1er protocole, si la séquence de requêtes correspondante aura une séquence de requêtes équivalente pour un chemin d'exécution complet du 2ième protocole. Si c'est le cas la sortie de la boucle maintient l'état de la variable *Continuer* à *Vrai* (Ligne 28). Quant à la boucle interne (Ligne 11 à 23), elle permet d'explorer tous les chemins possibles du 2ième protocole, tant qu'une séquence de requêtes équivalente, correspondante à un chemin d'exécution complet, n'est pas trouvée. Les instructions des lignes 10, 17, 18, 19 et 20 initialisent les variables liées à la recherche pour chaque itération. Dès qu'une séquence de requêtes équivalente est trouvée (ligne 16), la sortie de la 2ième boucle est observée et une nouvelle itération pour un nouveau chemin du 1er protocole est lancée. Les lignes 9 et 15 assurent les tests uniquement pour les chemins d'exécution complets. Quant aux lignes 8, 26 et 14, 22, ils permettent l'exploration des deux protocoles en changeant les états à tester pour une itération par les états cibles de l'itération précédente, tout en ôtant les états déjà testés. L'algorithme est déterministe est se termine en un nombre fini d'itérations, puisque les deux boucles s'arrêteront, une fois tous les chemins testés, ou dans le cas d'une recherche négative d'une séquence équivalente dans le 2ième protocole.

Avant de clore cette section, il est important de mentionner que les algorithmes proposés traitent particulièrement des protocoles déclencheurs et les tests sont basés sur l'équivalence des requêtes de mise à jour liées aux effets des messages. Cependant, le modèle de message proposé tient compte, aussi bien des effets que de leurs effets de compensation (***m (p, e, e')). Il s'ensuit que les algorithmes proposés sont aussi valables pour les protocoles de compensation, à condition de traiter les requêtes liées aux effets de compensation au lieu de celles liées aux effets.***

5.6 Opérateurs de manipulation des protocoles à effets transactionnels

Dans cette section, nous introduirons un ensemble d'opérateurs de manipulation des effets transactionnels. Ces opérateurs répondront aux objectifs visés dans le chapitre précédent (chapitre 4, section 4.4) et relatifs au fondement du raisonnement que nous voulons faire sur le modèle de protocole proposé.

La nécessité d'introduire les opérateurs répond directement à des objectifs opérationnels de manipulation des protocoles et de prise en charges des contraintes transactionnelles lors des interactions dans les services Web. Ces opérateurs, liés au modèle de représentation proposé, ont été élaborés sur la base d'objectifs précis et dont la substance a été extraite à partir des scénarios illustrés au début de ce chapitre.

5.6.1 Opérateurs de comparaison d'effets et d'effets de compensation: $\leftrightarrow$, $\leftrightarrow^c$

Cet opérateur Booléen permet de comparer les effets simples de deux messages. Il a pour opérandes deux messages en entrée et donnera une réponse binaire exprimant leur

équivalence ou leur différence. L'opérateur « ↔ » compare les effets directs des protocoles, alors que «↔^{c}» traite les effets de compensation.

Soient, $m_1\ (p, e_1, e'_1)$ et $m_2\ (p, e_2, e'_2)$ deux messages en entrée qui sont conformes au modèle de représentation. R_1 et R_2 sont les deux requêtes de mise à jour associées aux effets de m_1 et m_2, respectivement et R'_1 et R'_2 sont les deux requêtes de mise à jour associées aux effets de compensation de m_1 et de m_2.

L'expression $\mathbf{e_1 \leftrightarrow e_2}$: Désigne la comparaison des effets directs e_1 avec les effets e_2. Le résultat rendu est une valeur *Vrai* si les effets sont équivalents, *Faux* sinon. Du fait que ces effets sont modélisés par les requêtes de mise à jour au niveau des bases de données, alors:

$\mathbf{e_1 \leftrightarrow e_2}$ = *Vrai* si $R_1 \equiv R_2$, où ≡ est l'opérateur d'équivalence des requêtes (voir section 5.4)

$\mathbf{e_1 \leftrightarrow e_2}$ = *Faux* sinon, exprimant que les effets sont différents.

De même:

$\mathbf{e'_1 \leftrightarrow^{c} e'_2}$**:** Désigne la comparaison des effets de compensation e'_1 avec les effets e'_2. Il rend la valeur *Vrai* si les effets sont équivalents, *Faux* sinon.

$\mathbf{e'_1 \leftrightarrow^{c} e'_2}$ = *Vrai* si $R'_1 \equiv R'_2$, où ≡ est l'opérateur d'équivalence des requêtes, exprimant l'équivalence des effets de compensation.

$\mathbf{e'_1 \leftrightarrow^{c} e'_2}$ = *Faux* sinon, exprimant que les effets de compensation sont différents.

La manipulation par des opérateurs distincts des effets et des effets de compensation est justifiée par le fait que les effets de compensation, tels que nous les percevons, sont composés et exigent une décomposition en effets d'annulation et effets liés aux charges imposés par le fournisseur. En effet: $e'_1 \equiv e'_{1a}\ U\ e_1'_f$ et $e'_2 \equiv e'_{2a}\ U\ e'_{2f}$; d'où:

$$\mathbf{e'1 \leftrightarrow^{c} e'_2 =>\ (}\ e'_{1a}\ U\ e_1'_f\ \mathbf{)} \leftrightarrow^{c} \mathbf{(}e'_{2a}\ U\ e'_{2f}\mathbf{)}$$

L'opérateur de comparaison des effets, aussi simple qu'il apparaisse, offre une abstraction très forte pour la manipulation des effets des transactions. C'est un opérateur basique qui sera exploité dans la construction et la manipulation d'autres opérateurs plus complexes.

5.6.2 Opérateur de compensation: ↑

Dans la section 4.4.4.c du chapitre précédent, nous avons spécifié la définition formelle de la compensation des effets d'un message *m*. A présent, nous introduisons un opérateur de compensation pour gérer et manipuler les effets de compensation.

La richesse sémantique du modèle de message proposé, qui intègre les effets de compensation dans la structure du message même, facilite considérablement cette tâche.
Soient, $m_1\ (p, e_1, e'_1)$ et $m_2\ (p, e_2, e'_2)$ deux messages d'un protocole de service, tels que:
e: Ensembles des effets de m_1 représenté par la requête R_i.
e': Ensembles des effets de compensation de m_1 représenté par la requête R_j.
La décomposition des effets de compensation *e'* en deux types d'effets donne:
e'_a: Ensembles des effets d'annulation des effets du message *e*, représenté par la requête R_a.
e'_f: Ensemble des effets volontairement désiré par le fournisseur du service, représenté par la requête R_f.

Comme e'_f désigne le sous ensemble de e' relatif aux effets volontairement désirés par le fournisseur (charges affectées au client suite à l'annulation des transactions):

$e' \equiv e'_a U e'_f$, où e'_a correspond aux effets d'annulation simple de la transaction; d'où $e'_f \equiv e' \backslash e_a$.

Les requêtes élémentaires associées, après décomposition de la requête R_j sont: R_a et R_f.
Avec: $R_j = R_a \circ R_f$, et on notera:
$\mathbf{[e' \uparrow e]_S}$, pour exprimer que les effets *e' compensent* les effets *e* d'un message *m* dans le protocole de service *S*.
L'opérateur de compensation '↑' permet d'annuler les effets d'un message *m* tout en préservant les effets volontairement désirés par le fournisseur du service. Il a en entrées deux effets *e* et *e'*, et produira en sortie un nouvel état de la base de données affecté des mises à jour associées à la compensation.
Si S_B est l'état de la base à l'instant *t*, l'exécution consécutive des requêtes R_i et R_j aura pour effets sur l'état S_B de la base:
$S_B \circ e \circ e' \equiv S_B \circ R_i \circ R_j \equiv S_B \circ R_i \circ (R_a \circ R_f) \equiv S_B \circ (R_i \circ R_a) \circ R_f$;
Or, la séquence $(R_i \circ R_a)$ ne produit aucun effets sur la base, du fait que R_a annule R_i (par définition), alors l'état de la base à l'instant *t+1* est:
$S_B \circ R_f$ qui n'est autre que l'état initial après exécution des requêtes volontairement désirées par le fournisseur.
Il est possible d'étendre les fonctionnalités de cet opérateur pour traiter deux messages différents. Dans ce cas, l'expression algébrique de la compensation doit inclure les messages sources, de la façon suivante:

$\mathbf{[e'(m_2) \uparrow e(m_1)]_{S,,}}$ dans ce cas: le message m_2 compense les effets de m_1 si et seulement si les effets de compensation de m_2 sont équivalents à ceux de m_1.

$$\mathbf{e'_1}(m_1) \leftrightarrow^{\mathbf{c}} \mathbf{e'_2}(m_2) \Rightarrow [e'(m_2) \uparrow \mathbf{e}(m_1)]_{S,} \qquad \mathbf{(12)}$$

L'équation (12) est d'une importance capitale dans le cadre de la manipulation des protocoles de services et de leurs effets de compensation. En effet, les deux messages m_1 et m_2 peuvent appartenir au même protocole (mêmes effets de compensation pour divers messages du même protocole). Par exemple: l'annulation de n'importe quelle opération d'un protocole engendre l'abandon du processus et le retour à l'état initial. Mais le cas le plus intéressant, est celui de l'appartenance des messages à deux protocoles distincts. Dans ce cas, même si le protocole déclencheur est défaillant, une compensation peut être activée en se référant à des messages appartenant à d'autres protocoles, à condition que les effets des messages de compensation soient équivalents à ceux du protocole déclencheur (cas des protocoles à effets transactionnels équivalents strictement ou à convergence des états des bases).

5.6.3 Opérateur de composition des effets et des effets de compensation: ◊

Un protocole de service est généralement composé d'un ensemble de messages, et il est rare que le client invoque une seule opération. Il réalise, plutôt, des processus métiers. Par ailleurs la prise en charge des transactions dans les services Web diffère de celle des bases de données. Le critère d'atomicité est relaxé et le traitement des transactions est basé sur les *régions atomiques*. (Voir Chapitre 3, section 3.3.2.b).

Pour permettre une gestion des effets des protocoles de services, en tenant compte des spécificités de la région atomique, nous introduisons un opérateur de composition des effets des transactions. Son objectif consiste à identifier les effets élémentaires d'un effet complexe, ou d'une région atomique afin d'induire les effets de compensation élémentaires correspondants, ou s'il existe un effet compensatoire global pour cet ensemble d'effets élémentaire.

Soient, m_1 (p, e_1, e'_1), m_2 (p, e_2, e'_2),… m_i (p, e_i, e'_i)..., m_n (p, e_n, e'_n) un ensemble de messages appartenant à une région atomique, tels que:

e_i: Ensembles des effets de m_i représenté par la requête R_i.
e'_i: Ensembles des effets de compensation de m_i représenté par la requête R'_j.
Et soit: La composition des effets des messages m_1, m_2, …, m_n, formalisée par:
$\boldsymbol{e_1 \lozenge e_2 \lozenge \ldots \lozenge e_i \lozenge \ldots \lozenge e_n}$.
Cette composition exprime le changement de l'état de la base de données après l'exécution de la séquence des requêtes de mise à jour associées aux effets élémentaires, comme suit:
$S_B\, o\, (e_1 \lozenge e_2 \lozenge \ldots \lozenge e_i \lozenge \ldots \lozenge e_n) \equiv S_B\, o\, (R_1\, o\, R_2\, o \ldots R_i \ldots o \ldots o R_n)$.
D'une manière identique la composition des effets de compensation de la région atomique correspond à l'exécution de la séquences des requêtes y afférentes.
$S_B\, o\, (e'_1 \lozenge e'_2 \lozenge \ldots \lozenge e'_i \lozenge \ldots \lozenge e'_n) \equiv S_B\, o\, (R'_1\, o\, R'_2\, o \ldots o\, R'_i \ldots o R'_n)$.

Dans le contexte de la compensation, il sera intéressant de trouver un effet compensatoire global équivalent à la composition des effets de compensation élémentaires. Le problème est formalisé de la façon suivante:
Existe-t-il un effet global e'_g, telle que: $e'_g \leftrightarrow^c e'_1 \lozenge e'_2 \ldots \lozenge e'_i \lozenge \ldots \lozenge e'_n$? Autrement, existe-t-il une requête globale R'_g de mise à jour, dont les actions au niveau de la base produiront les mêmes effets que ceux de la séquence de requêtes: $R'_1\, o\, R'_2\, o \ldots R'_i \ldots o\, R'_n$?
Plus simplement: Existe-t-il R'_g, telle que: $R'_g \equiv R'_1\, o\, R'_2\, o \ldots R'_i \ldots o\, R'_n$?
La réponse à cette question ne peut être abordée sans l'utilisation de l'opérateur de compensation des effets, ci-dessus spécifié.
D'un point de vue pragmatique, cet opérateur est très utile dans le cadre de la manipulation des protocoles de services. Si nous voulons compenser une région atomique, il suffit de chercher un message $m_g\,(p, e_g, e'_g)$, tel que:

$$\boldsymbol{e'_g \uparrow e_g} \text{ et } \boldsymbol{e_g \leftrightarrow e_1 \lozenge e_2 \lozenge \ldots \lozenge e_i \lozenge \ldots \lozenge e_n \Rightarrow e'_g \leftrightarrow^c e'_1 \lozenge e'_2 \lozenge \ldots \lozenge e'_i \lozenge \ldots \lozenge e'_n} \quad \textbf{(13)}$$

En résumé, il suffit d'exécuter une seule requête de compensation, pour une région atomique, au lieu de procéder à l'exécution séquentielle d'une séquence de requêtes. Les effets de compensation de cette requête unique sont équivalents à la composition des effets élémentaires de compensation.

Il est clair que la problématique de la composition des effets et celle des effets de compensation est ramenée à un problème de composition des requêtes, comme il a été conclu dans le chapitre 4 (Section 4.4.4.c**).**
Par ailleurs, L'opérateur de composition des effets et des effets de compensation permet de prendre en charge le problème de la compatibilité transitive des effets et des effets de compensation de messages et, aussi, celui de l'équivalence à convergence des états des bases

de données. Les problèmes exposés dans les scénarios: 3, 4 et 5 de la section 5.1.1 et les situations de convergence de la section 5.1.2 peuvent être traités sans difficultés.

Une fois implémenté, les opérateurs proposés permettront une analyse plus efficace et plus concise des protocoles en les exploitant dans le processus d'analyse de compatibilité et d'équivalence. Les algorithmes proposés peuvent être remis en ordre en intégrant ces opérateurs, aux fins d'optimisation et de clarté.

5.7 Conclusion

Au niveau de ce chapitre, nous avons démarré d'un ensemble de scénarios mettant en évidence des situations inadéquates avec l'analyse standard de la compatibilité et d'équivalence des protocoles de base, et ce à cause des enrichissements apportés au modèle de base des protocoles de services. Nous étions contraints, de ce fait, de réexaminer cette analyse à la lumière des enrichissements apportées.

Une nouvelle formalisation du problème a été conçue et nous avons proposé une nouvelle démarche pour l'analyse des protocoles à effets transactionnels. Après l'étude de la compatibilité des protocoles, une attention particulière a été attachée à l'analyse d'équivalence, vue son importance. En ce sens, nous avons identifié des classes d'équivalences des protocoles à effets transactionnels, auxquelles les algorithmes adéquats ont été élaborés.

Le problème de l'équivalence des requêtes de mise des bases de données, face cachée des effets des transactions dans notre modèle de protocole, a été abordé d'une manière explicite. Notre contribution a été concrétisée par les algorithmes et opérateurs nécessaires à la gestion et à la manipulation des protocoles de services à effets transactionnels.

Conclusion et Perspectives

Conclusion et Perspectives

Tout au début, lorsque nous avons commencé à aborder la problématique de la prise en compte des contraintes lors de la découverte et de l'orchestration des services Web, nous étions partis avec le préalable de traiter les contraintes non fonctionnelles et, particulièrement, les contraintes de qualité de service (QoS). Après un parcourt considérable, nous avons conclu que le domaine des contraintes non fonctionnelles a fait l'objet de travaux de recherche importants qui ont abouti aux spécifications et standards adoptés par la communauté des acteurs du domaine des services Web.

Par conséquent, nous étions obligés de changer de cap et de nous orienter vers les contraintes fonctionnelles. Après exploration du domaine, nous avons décelé un déficit remarquable dans la prise en compte des contraintes transactionnelles. Nos efforts ont été focalisés sur cet aspect.

L'étude des protocoles de services s'est imposée dés le début de notre travail, comme élément fondamentale de la description du comportement externe d'un service Web. Nous avons étudié les modèles de représentation existants et nous avons évalué leurs forces et leurs faiblesses. Le modèle des automates d'états finis déterministes a été adopté dans la suite de notre travail, vue son formalisme rigoureux, son expressivité et la disponibilité des outils permettant leurs manipulation.

Par la suite, en analysant les contraintes fonctionnelles et les comportements externes des services Web, nous avons constaté qu'il y a une exigence réelle pour la modélisation des effets des transactions, afin de rehausser la description des services Web à sa sémantique interactionnelle. Ce n'est qu'à ce moment que nous avons pris conscience de l'importance et de la lourdeur de notre mission.

Dans ce mémoire nous avons mis en évidence l'intérêt de la modélisation des contraintes transactionnelles au niveau protocole, en tant qu'éléments intrinsèques de la description du comportement externe des services Web. Ces contraintes ont été perçues comme effets affectant le monde du client et ont été analysées dans la perspective de leur compensation. Nous avons proposé un modèle pour la représentation des effets transactionnels basé sur les requêtes de mise jour de la base de données. A notre sens, ce modèle est parfaitement adéquat avec les exigences des processus de compensation et il répond, d'une manière consistante, aux objectifs exprimés à travers les diverses motivations qui ont incité ce travail de recherche.

Le nouveau modèle de protocole de service, enrichi par les effets des transactions, a été présenté et formalisé. Cependant, cet enrichissement a affecté l'analyse de compatibilité et d'équivalence des protocoles de services, du fait qu'il l'a rehaussé à un degré dépassant l'aspect syntaxique et structurel exprimé par l'ordre des messages et leur polarité.

C'est à ce niveau qu'intervient notre seconde contribution. Elle est, justement, relative à la formalisation de l'analyse de compatibilité et à l'étude de l'équivalence des protocoles de services à effets transactionnels, où deux classes d'équivalences ont été identifiées et les algorithmes adéquats élaborés.

L’ensemble des opérateurs de manipulation des effets transactionnels que nous avons proposé contribuera d’une manière significative à l’optimisation des algorithmes inhérents à la gestion des protocoles à effets transactionnels. Ces opérateurs contribueront, par ailleurs, à consolider l’assise théorique existante pour la manipulation des protocoles de services.

Comme travaux futurs, nous envisageons de raffiner l’analyse de compatibilité, d’identifier les types de compatibilité (à base d’effets, à base d’effets de compensation, parfaite…) et l’élaboration des algorithmes adéquats pour les différents types de compatibilité.

Nous envisageons, par ailleurs, une spécification plus riche de l’ensemble des opérateurs de manipulation des effets transactionnels afin de permettre une formalisation plus rigoureuse de l’analyse de compatibilité et d’équivalence. Ces objectifs renforceront, inévitablement, l’assise théorique existante et contribueront à la formalisation d’une algèbre des protocoles de services à effets transactionnels.

En conclusion, nous pouvons affirmer que ce travail nous a permis, en plus des acquis méthodologiques liés à la maîtrise de la conduite d’un travail de recherche, de se familiariser avec le domaine des services Web, de comprendre leur technologie et d’essayer de contribuer à leur évolution par l’originalité des propositions que nous avons apporté.

Références et bibliographie

Bibliographie

[1] Patrick Kellert et Farouk Toumani: Les web services sémantiques. Rapport de recherche LIMOS/RR 03-15 ; 11 Juillet 2003 http://www.isima.fr/limos/publi/RR-03-15.doc/ http://www.revue-i3.org/hors_serie/annee2004/revue_i3_hs2004_01_07.pdf/

[2] W3C, Web Services Architecture, W3C Working Group Note 11 February 2004 http://www.w3.org/TR/ws-arch/

[3] Hubert Kadima, Valérie Monfort: Les Web Services techniques, démarches et outils XML, WSDL, SOAP, UDDI, Rosetta, UML, Edition DUNOD, 2003

[4] Gustavo Alonso, Fabio Casati, Hurumi Kuno,Vijay Machiraju: Web services concepts Architectures and applications, Edition Springer Verlag Berlin 2004.

[5] F. Cabrera et al. Web Service Coordination (WS-coordination), August 2005. http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-tx/WS-Coordination.pdf

[6] F. Cabrera et al. Web Service transaction (WS-transaction), January 2004. http://dev2dev.bea.com/pub/a/2004/01/ws-transaction.html.

[7] F.Cabrera et al. Web Services Web Services Business Activity Framework (WS- BusinessActivity) August 2005 http://specs.xmlsoap.org/ws/2004/10/wsba/wsba.pdf

[8] F. Cabrera et al. Web Services AtomicTransaction (WS-AtomicTransaction) August 2005 http://specs.xmlsoap.org/ws/2004/10/wsat/wsat.pdf

[9] ebXML Technical Architecture Specification v1.0.4 February 2001 http://ebxml.org/specs/ebTA.pdf

[10] M. JURIC, BPEL and JAVA. April 2005

[11] C.PELTZ. Web Service Orchestration and Choreography – A look at WSCI and BPEL4WS. July 2003.

[12] S. GOUDEAU et al. Web services et Interopérabilité des S.I, Dunod, Paris 2004.

[13] B. Benatallah et al: Representing, Analysing and Managing web Service Protocols. Data Knowledge Ingineering. 58 (3): 327-357, 2006

[14] Xiaochuan Yi, Krys J. Kochut, Process composition of Web services with complex conversation Protcol: a Colored Petri Nets based Approch.

[15] B. Benatallah et al: Web Service conversation Modeling A cornerstone for E-Business automation, IEEE Internet computing 8 (1) (2004) 46-54

[16] J. Ponge et al: Fine-Grained Compatibility and Replaceability Analysis of Timed Web Service Protocols. ER 2007: 599-614

[17] P. bonnet, Cadre de reference Architecture SOA http://pie.bonnet.ifrance.com/ON-guideSOA-2005-02-23% 20(part1).pdf

[18] Typologie des contraintes OCL, L. Audibert
http://laurent-audibert.developpez.com/Cours-UML/html/Cours-UML024.html

[19] Blum, A. (2004). “UDDI as an Extended Web Services Registry: Versioning, quality of service, and more”. Web service journal, Juin 2004

[20] Blum, A. (2004). “Extending UDDI with Robust Web Services Information”.

[21] Ran, S. (2004). "A Model for Web Services Discovery with QoS". SIGEcom Exchanges, vol. 4, no. 1, 2004, pp. 1–10.

[22] Maximilien, E.M. & Singh, M.P. (2004). “A Framework and Ontology for Dynamic Web Services Selection”. *IEEE Internet Computing*, 8(5):84-93, 2004.

[23] Majithia, S. et Al, “Reputation-based Semantic Service Discovery“ *Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* (WET ICE’04), Italy, June 2004.

[24] Wishart, R et Al “SuperstringRep: Reputation-enhanced Service Discovery“, Proceeding of 21ième *Australasian conference on Computer Science*, Vol. 38: 49-57, 2005

[25] P. Bonnet, “ Cadre de référence Web services- Meilleure pratiques “
http://pie.bonnet.ifrance.com/ON-guideWS-2005-02-23%20(part2).pdf

[26] OASIS Business Transaction Protocol (BTP)
http://oasis-open.org/committees/business-transactions/

[27] J. Roberts , K. Srinivasan. Tentative hold Protocol Part 1: White Paper, W3C Note 28 November 2001, http://www.w3.org/TR/tenthold-1/

[28] JSR 95 J2EE Activity Service for Extended Transactions, http://jcp.org/jsr/detail/95.jsp

[29] JSR 156 XML Transactioning API for Java (JAXTX),
http://www.jcp.org/jsr/detail/156.jsp/

[30] T. Mikalsen et al, Transactional attitudes: Reliable composition of autonomous Web Services, WDMS 2002.

[31] D. Martin et al, OWL-S: Semantic Markup for Web Services, W3C Submission November 2004, http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/

[32] D. Berardi et al, Automatic composition of Transition-Based Semantic Web Services with Messagging, Proceeding of 31st VLDB conference, Trondheim, Norway, 2005

[33] Ali Khebizi: External Behavior Modeling Enrichment of Web Services by Transactional Constraints, ICSOC PhD Symposium, Sydney December 2008.
http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-421 /paper12.pdf/

ANNEXES

ANNEXES

ANNEXE 1: Structure, description et exemples de messages SOAP.

a/ Structure d'un message SOAP

Un message SOAP est un document XML ordinaire contenant les éléments suivants:

- Une **Enveloppe**: élément indispensable qui identifie le document XML comme un message SOAP.
- Un en-tête (**Header)**: élément qui contient des informations d'entête (optionnel).
- Un Corps (**Body):** élément qui contient les informations d'appel et de réponse, qui est aussi indispensable.
- Un **fault** élément qui fournit des informations sur une erreur qui surviendrait lors du traitement du message.

Il existe deux formats de messages SOAP correspondants aux messages sans pièces jointes et avec pièces jointes.

✓ Format d'un message SOAP sans pièce Jointe

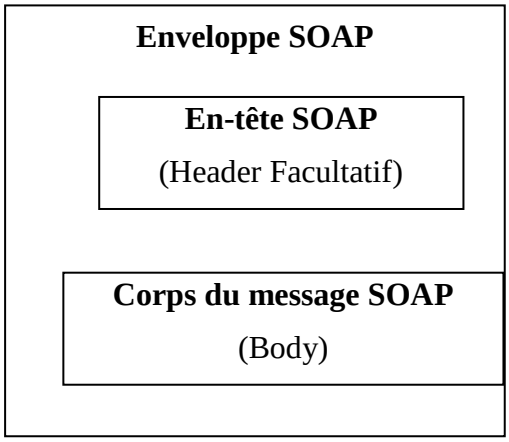


Figure A1: Structure d'un message SOAP sans pièce jointe

✓ Format d'un message SOAP avec pièce Jointe

Le message peut comporter une ou plusieurs parties d'attachement ajoutées au message SOAP de base. La partie SOAP ne peut contenir seulement que du XML, comme résultat, si un contenu de message n'est pas au format XML, il doit apparaitre dans la partie " attachement ". Si le message contient par exemple une partie image, on doit avoir une partie "attachement" pour les véhiculer dans les messages SOAP. Toute partie "Attachement" peut contenir tout type de contenu et donc aussi des données au format XML.

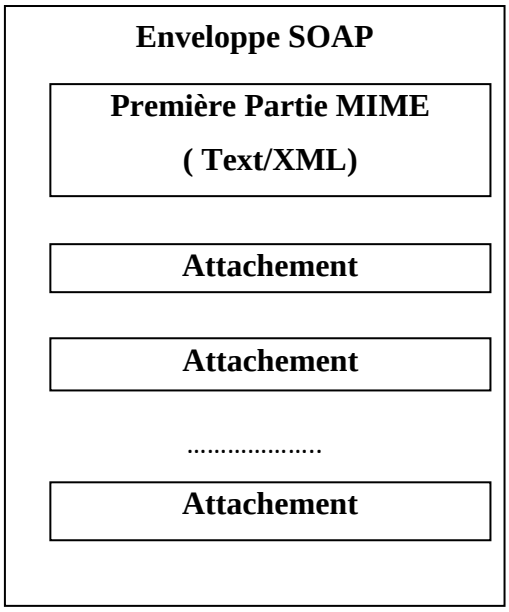


Figure A2: Structure d'un message SOAP avec pièces jointes

b/ Description d'un message SOAP

- L'enveloppe SOAP: <SOAP-ENV:Envelope>

L'enveloppe est la racine du document XML contenant le message SOAP, marquée par la balise <Envelope>. La spécification impose que tous les attributs de cette balise et de celles imbriquées soient explicitement associés à un *namespace*, de manière à supprimer toute ambiguïté. Par convention, la spécification SOAP définit deux *namespace* fréquemment utilisés:

• SOAP-ENV associé à l'URI:
- http://schemas.xmlsoap.org/soap/envelope pour définir le namespace de l'enveloppe dans la version 1.1.
- http://www.w3.org/2001/06/soap-envelope dans la version 1.2 reprise par le W3C.

• SOAP-ENC associé à l'URI:
- http://schemas.xmlsoap.org/soap/encoding pour la définition des formats de types de données dans la version 1.1.
- http://www.w3.org/2001/06/soap-encoding dans la version 1.2.

L'attribut «*EncodingStyle*», dont la valeur est un URI, spécifie quelles conventions sont employées dans cet élément (et dans tous ceux qu'il contient) pour le format des données. La spécification propose des formats identifiés par l'URI SOAP-ENC précédemment mentionné. Rien n'exclu de faire référence à des règles de codage et de représentation des données différentes de celles proposées par la norme, mais le cas reste rare.

- L'en-tête SOAP: <SOAP-ENV: Header>:

La balise <Header> permet de passer dans le message des informations complémentaires sur ce message. Cet élément est facultatif, mais s'il est présent, il doit être le premier dans l'enveloppe SOAP du message. L'en-tête peut, par exemple, contenir des informations authentifiant l'émetteur ou bien le contexte d'une transaction dont le message n'est qu'une

des étapes. Pour les couches de transport (FTP) ne fournissant pas à l'émission d'adresse de retour, on peut aussi utiliser l'en-tête pour identifier l'émetteur du message SOAP.
L'élément <Header> et les éléments qu'il contient, peuvent être qualifiés par l'attribut « *MustUnderstand* » qui prend la valeur 0 ou 1. La valeur 1 signale que le récepteur doit reconnaitre l'information présente dans l'en-tête et que son traitement est obligatoire ; la valeur 0 indique que l'en-tête peut être ignoré par le serveur. Ainsi, il est possible d'étendre les fonctions du serveur. L'exemple suivant montre un en-tête transportant une adresse de courrier électronique:

```
<SOAP-ENV: Header>
<exemple: emetteur xmlns: exemple= = «http://monURI»
 SAOP-ENV: mustUnderstand="0">
 Monnom@mondomaine.com
</exemple:emetteur>
</SOAP-ENV: Header>
```

Un message SOAP peut avoir à «traverser» plusieurs intermédiaires avant d'atteindre son destinataire. Dans ce cas, l'en-tête joue un rôle primordial: à chaque arrêt le long de l'itinéraire, le serveur intermédiaire extrait de l'en-tête ce qui le concerne en propre et ajoute e qui est nécessaire au serveur intermédiaire suivant. Les éléments concernant un serveur intermédiaire sont marqués par la présence de l'attribut: SOAP-ENV: ACTOR avec une valeur conventionnelle:

SOAP-ENV: ACTOR= "http://schemas.xmlsoap.org/soap/actor/next"

- Le corps du message SOAP: <SOAP-ENV: Body>
  Le corps du message est constitué d'un seul élément BODY, contenant un ou plusieurs sous-éléments. Ces sous-éléments sont les suivants:
  - Des données destinées au destinataire du message, à un format défini par les règles de codage SOAP.
  - FAULT, indiquant une erreur ou une défaillance en réponse à une requête.

Initialement, le corps de message SOAP était conçu pour passer des appels de procédures distantes, et des résultats ou des messages d'erreur. Mais, en fait, l'usage s'est élargi et le corps du message est souvent utilisé pour simplement passer des données structurées entre applications. Dans le cas de la défaillance (FAULT), de nouveaux sous-éléments sont employés pour signaler le type d'erreur et pour renvoyer à l'émetteur des informations supplémentaires indiquant la raison de l'échec de l'appel. Les codes d'erreurs sont également normalisés par la spécification SOAP.

c/ Exemple d'une requête SOAP

Pour illustrer la structure d'un message SOAP, voici un exemple basé sur un service fournissant le prix des pommes.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body>
    <m:GetPrice xmlns:m="http://www.w3schools.com/prices">
    <m:Item>Apples</m:Item>
    </m:GetPrice>
  </soap:Body>
</soap:Envelope>
```

Un exemple de réponse à ce message donnant le prix des pommes: 100 est le suivant:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
 <soap:Body>
  <m:GetPriceResponse
    xmlns:m="http://www.w3schools.com/prices">
  <m:Price>100 </m:Price>
  </m:GetPriceResponse>
 </soap:Body>
</soap:Envelope>
```

ANNEXE 2: Structure, description et exemples d'interface WSDL

a/ Structure d'une Interface WSDL

Une interface WSDL est structurée en deux parties. Une Abstraite qui est l'interface de service et l'autre concrète contenant l'implémentation du service.

La première étape dans la définition d'une interface WSDL est d'identifier et de définir toutes les structures de données qui seront échangées en tant qu'éléments entre les différentes applications.
La seconde étape est de définir les messages qui contiendront de telles structures. Dans WSDL, chaque type de message est divisé en parties. Chaque partie est caractérisée par un nom et par un type référant au type définit dans le schéma XML.
La troisième étape est de définir les opérations aussi appelées transactions primitives d'appels ou interactions.
Les opérations seront regroupées logiquement en type de port (*Port type*).

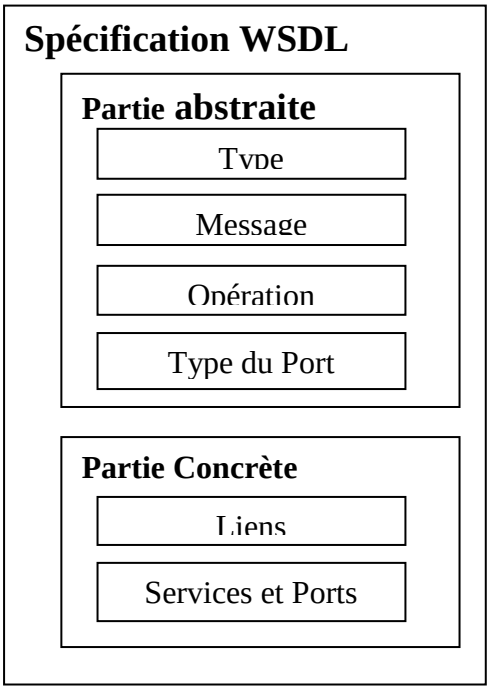


Figure A3: Eléments de description d'une interface WSDL

b/ Description des éléments d'une Interface WSDL

- Les type: balise <*type*>

  L'élément type contient les définitions des types de données appliquées aux messages échangés. Pour garantir une interopérabilité maximale et une grande indépendance au niveau des plates formes, WSDL utilise XSD (Schéma XML) en tant que système de type.

- Les messages: balise <*message*>

  C'est dans cette partie que la description des messages est faite. Selon leur complexité, les messages sont décomposés en une ou plusieurs parties (<*Part*>). Chaque partie est associée à un type à l'aide d'un attribut «*type*».

- Les opérations: balise <*opération*>

  Sont définies par la balaise <opération> et peuvent être de différentes natures:
  - Opération unidirectionnelle (*One-Way*): Le récepteur final du service reçoit un message mais ne renvoie jamais de réponse.
  - Requête/Réponse: On communique en mode question/réponse synchrone.
  - Sollicitation/Réponse: Idem que l'opération requête/Réponse ; hormis que les éléments en entrée et les éléments en sorite sont inversés.
  - Notification: Le récepteur final envoi un message mais sans attendre de réponse de la part du client.

  La première et la dernière opération impliquent un seul message et sont asynchrones, contrairement aux deux autres qui impliquent deux messages et sont synchrones.

- Les types de port: balise <*PortType*>

  Les types de port sont utilisés pour définir les traitements offerts par un service Web. Un type de port est un ensemble de messages regroupés en opération qui représentent une unité d'action pour le service décrit. Un type de port peut comporter plusieurs opérations.
  A noter que les quatre balises précédentes forment la partie abstraite du service. Elle est considérée abstraite pour les raisons suivantes: il n'est y a pas de lien (*Binding*) concret, ni un codage spécifique à ces constructions, non plus une définition d'un service qui implémente un ensemble de *PortType*.

- Les liens: balise <*Binding*>

  Une liaison permet de spécifier les propriétés du protocole utilisé et le format des messages. Un *PortType* peut supporter plusieurs protocoles.

- Les ports (ou *EndPoint*): balise <*Port*>

  Un port spécifie l'adresse d'une liaison. Un port combine les interfaces 'Binding' et les adresses réseau (spécifiées par un URI) à laquelle l'implémentation du PortType est accessible. Un port ne peut pas indiquer plus d'une adresse et ne peut définir d'autres informations de liaison.

- Les services: balise <*Service*>

  C'est un groupement logique de ports et constitue la liste des services mis à disposition par le service Web. C'est ce point qui va permettre de rechercher via l'UDDI les services offerts.

  Une fois tous ces constructeurs sont ajoutés à WSDL, la définition de l'interface du service devient plus concrète. Avec l'information de lien («*Binding*»), l'utilisateur

connaît le protocole à utiliser, la structure des messages XML pour interagir avec le système et le résultat attendu en contactant le service.
Avec l'information du port, l'utilisateur connaît l'adresse réseau et à quelle fonctionnalité de port elle est implémentée. Finalement avec la définition de service, l'utilisateur connaît tous les ports qui sont implémentés pour un groupe.

Le fichier WSDL est ensuite publié dans un annuaire UDDI par le fournisseur et récupéré par le client lorsque celui-ci décide d'invoquer le service Web.

c/ Exemple d'une interface WSDL

Voici un exemple détaillé d'un document WSDL concernant un service qui permet de commander des produits (**Figure A4**)

```
< ?xml version="1.0"? >
<definitions name= "Procurement"
TargetNamespace= "http://example.com/procurement/definitions"
xmlns: tns="http://example.com/procurement/definitions"
xmlns: xs="http://www.w3.org/2001:XMLSchema"
xmlns: soap="http://schema.xmlsoap.org/wsdl/soap/"
xmlns ="http://schema.xmlsoap.org/wsdl/" >

<message name= "OrderMsg">
        <Part name="ProductName " type= "xs:string"/>
        <Part name="quantity" type="xs: integer"/>
</message>

<portType name= "procurementPortType">
        <operation name="OrderGoods">
            <input message = "orderMsg"/>
        </operation>
</portType>

<binding name ="ProcurementSoapBinding" type="tns: procurementPortType">
        <soap: Binding style ="document"
                Transport="http://schema.xmlsoap.org/soap/htpp/">
        <operation name ="orderGoods">
                <soap: operation soapAction="http://example.com/orderGoods"/>
                <input>
                        <soap: body use ="literal1" />
                </input>
                <output>
                        <soap: body use ="literal1" />
                </output>
        </operation>
</binding>

<service name ="ProcurementService">
        <Port name ="ProcurementPort"  binding = "tns: ProcurementSoapBinding">
            < soap: address  location = "http:// example.com/procurement "/>
        </port>
</service>

</definitions>
```

Partie abstraite
Messages
Opération et PortType
Partie Concrète
Lien (Binding)
Nom de service
Port et services
Localisation du service

Figure A4: Description WSDL d'un service pour la commande de produits

ANNEXE 3: Structure, description et API du registre UDDI

Les spécifications UDDI définissent le modèle d'information, c'est-à-dire les structures de données UDDI et les API que l'opérateur de l'annuaire est censé fournir aux utilisateurs de l'annuaire.

a/ Structure des données UDDI

Dans l'annuaire UDDI, les descriptions des services contiennent quatre types d'informations: informations métier, information de service, information de lien, et information sur les spécifications du service décrit par plusieurs entités.

- **Business Entity**: Décrit l'organisation qui fournit le service Web. Contient la liste des noms d'entreprises, adresse et autres informations sur le fournisseur.

- **Business Service**: Décrit un groupe de services Web offerts par une Business Entity. Un Business Service correspondra à un type de service mais proposé à différentes adresses, dans de multiples versions et à travers différentes technologies. Une entrée de Business Service n'est rattachée qu'à une seule business entity.

- **Business Template**: Décrit l'information technique pour utiliser un service Web particulier. Il définit essentiellement l'adresse à laquelle le service est valable ainsi qu'un certain nombre d'informations telles que la référence du document (appelé *tModel*) décrivant l'interface du service Web ou autres propriétés du service. Il définit également comment les paramètres d'opération doivent être configurés et les valeurs par défaut pour ces paramètres. Une entrée du Business Service peut inclure de multiples éléments Business Template, mais une Business Template n'est rattachée qu'à une seule Business Service.

- **tModel "Technical Model ":** C'est un conteneur générique pour n'importe quel type d'informations. Il peut par exemple représenter une interface de service WSDL, une classification ou un protocole d'interaction; il peut aussi décrire la sémantique d'une opération.

b/ Les API de l'enregistrement UDDI

Les annuaires UDDI ont trois types principaux d'utilisateurs exploitants les API:
- Les fournisseurs de services qui publient les services Web.
- Les clients ou demandeurs qui recherchent les services Web.
- D'autres utilisateurs qui ont besoin d'échanger des informations.

Ces utilisateurs d'annuaires (clients de l'UDDI) peuvent utiliser six types d'API:

- **API d'Interrogation: Cette API inclut:**

- Les opérations permettant de trouver des entrées dans l'annuaire qui correspondent aux critères de recherches et donnent une vue d'ensemble des informations au sujet de cette entrée.
- Les opérations qui fournissent des détails sur une entrée spécifique.

Cette API, utilisée par le navigateur UDDI, permet au développeur de trouver les informations nécessaires.

- **API de Publication:** Cette API permet au fournisseur d'ajouter, modifier ou supprimer des entrées dans l'annuaire.

- **API de sécurité:** Elle permet aux utilisateurs d'obtenir des authentifications qui seront utilisées dans les communications futures avec l'annuaire.

- **API de propriété de transfert:** Autorise les annuaires à transférer les droits des structures d'un fournisseur à un autre.

- **API d'Abonnement à l'UDDI:** Permet de gérer les abonnements à un enregistrement (ajout, modification, suppression).

- **API de Réplication:** Permet la réplication entre deux annuaires.

Les annuaires UDDI maintiennent différents points d'accès (URIs) pour les demandeurs, les fournisseurs et les autres annuaires.