Prise en Compte des Contraintes Transactionnelles dans les Services Web: Modélisation et Analyse d'Equivalence

KHEBIZI Ali[1], SERIDI Hassina [2]
[1] Département d'Informatique , Université 8 mai 1945 - BP 401 Guelma, Algérie
[2] Université Badji Mokhtar, BP 12 Annaba 23000, Algérie
LabGed laboratoire de gestion du document
[1]kheali@hotmail.com
[2]seridi@labged.net.

Résumé: L'infrastructure actuelle des services Web reste insuffisante pour une réelle prise en charge des activités métiers et pour la gestion des transactions trans-organisationnelles. En effet, la description actuelle du comportement externe des services Web et les techniques de composition et de coordination ne garantissent pas une gestion fiable et efficace des transactions métiers complexes, de longue durée et manipulant des données critiques. La compensation des opérations inachevées, pour diverses raisons, s'avère, alors, inévitable. Cependant, la prise en compte des contraintes transactionnelles et la gestion de la compensation n'ont pas bénéficié de tout l'intérêt qu'elles revêtent.
Dans cet article, nous nous intéressons à cet aspect important des environnements services Web. Nous proposons une modélisation formelle des effets transactionnels. Les protocoles de services enrichis par l'injection du modèle d'effets proposé seront exposés et l'analyse d'équivalence des services sera revue, en conséquence. Nous identifierons les nouvelles classes d'équivalence des services enrichis et nous présenterons les algorithmes y afférents.

Mots clés: Protocoles de services, Effets transactionnels, Compensation, Equivalence des services, Protocoles à effets, Equivalence à base d'effets.

Introduction

Dotés d'une assise théorique solide (architecture SOA) et une infrastructure consistante (SOAP, WSDL et UDDI), les services Web offrent de grandes potentialités pour l'intégration des applications intra et interentreprises et pour la

gestion des interactions entre divers partenaires issus d'environnements hétérogènes et distribués sur le Web.

Cependant, la description actuelle du comportement externe des services Web, par leur interface et par leur protocole, reste limitée et n'exprime pas toute la sémantique des interactions engagées lors des invocations de services. Par ailleurs, la composition des services, en vue de réaliser des activités à valeur ajoutée, nécessite une description plus riche des fonctions offertes par les fournisseurs. Cette exigence est primordiale dans un contexte de substitution d'un service défaillant par un autre qui lui est fonctionnellement équivalent.

Effectivement, les interactions engagées entre les divers partenaires lors de l'exécution des activités métiers sont complexes, de longue durée et manipulent des données critiques, et diverses raisons peuvent entraver l'accomplissement du service invoqué (panne réseau, panne du serveur, service défaillant…). Cette situation exige une annulation des opérations inachevées. Mais, contrairement aux bases de données et aux Workflows traditionnels, les transactions dans les services Web sont complètement différentes par leur nature et par leurs effets. La relaxation des propriétés ACID, associée aux techniques de compensation, offre un cadre adéquat pour la gestion des transactions dans les environnements services Web.

Toutefois, si l'infrastructure actuelle des services Web ait été enrichie par les spécifications; telles que les Framework: WS-Coordination [1] et WS-Transaction [2], nécessaires à la gestion des transactions au niveau du Middleware, et si diverses caractéristiques décrivant le comportement externe des services Web; telles que les contraintes d'ordre [3] et les contraintes temporelles [4] ont été prises en compte dans la modélisation des protocoles de services, la prise en compte et la gestion des contraintes transactionnelles, au niveau des protocoles de services, qui revêtent un aspect critique et un impact immédiat sur tout le cycle de vie d'un service Web (modélisation, développement, déploiement), 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 impulser le déploiement de la technologie des services Web, qui ne pourra atteindre ses objectifs sans la gestion des vrais processus métiers trans-organisationnels, nous proposons dans cet article un enrichissement de la description des protocoles de service par la modélisation des contraintes transactionnelles et nous étudierons l'impact, d'un tel enrichissement, sur l'analyse d'équivalence des services.

La suite de l'article sera structurée comme suit : Dans la section 2, nous exposerons les motivations qui ont incité ce travail, en se basant sur un scénario réel. Un état de l'art de la gestion des effets des transactions dans les protocoles de service est présenté en section 3. Dans la section 4, nous proposerons une modélisation des effets transactionnels et nous exposerons le modèle de protocole associé. La section 5, sera consacrée à l'analyse d'équivalence des protocoles de services à effets transactionnels et à l'identification des classes d'équivalence qui en découlent. En fin, la section 6 contiendra nos conclusions et nos travaux futurs.

1 Motivations

A cause de leur coût excessif et de leur durée relativement longue, les transactions dans les services Web diffèrent, par leurs effets et par leurs processus d'annulation, de celles liées aux bases de données. Dès lors, les fournisseurs des services procèdent, plutôt à une *compensation* des transactions et affectent, souvent, une partie des coûts associés aux clients [3]. La compensation est assurée par le middleware d'une manière transparente en exécutant le protocole de compensation prédéfini par le développeur du service [5]. Dans ce contexte, il s'avère opportun d'aborder une réflexion approfondie sur la question de la modélisation des effets des transactions, afin de rehausser la description du comportement externe des services à sa sémantique réelle caractérisée par les effets des opérations et dans la perspective d'une éventuelle compensation.

A notre connaissance, peu de travaux existent au sujet de cette problématique qui exige des efforts de recherche approfondis. Nous détaillons, ci-après, les raisons directes qui motivent ce travail.

1.1 Vérification de la compatibilité des services

La définition de la compatibilité de deux protocoles (totale ou partielle); telle que décrite dans [3], restreint le critère de test à l'ordre des opérations et à la polarité des messages. Bien qu'elle ait été étendue pour prendre en compte les contraintes temporelles liées aux messages (délai, date, heure) dans [4], elle reste insuffisante et doit prendre en considération les effets transactionnels liés aux messages.

Le scénario présenté dans la Figure 1, dans lequel deux protocoles de services sont modélisés par des automates d'états finis déterministes, et sont relatifs au service de retrait d'un montant d'un compte par le client, illustre clairement cette exigence.

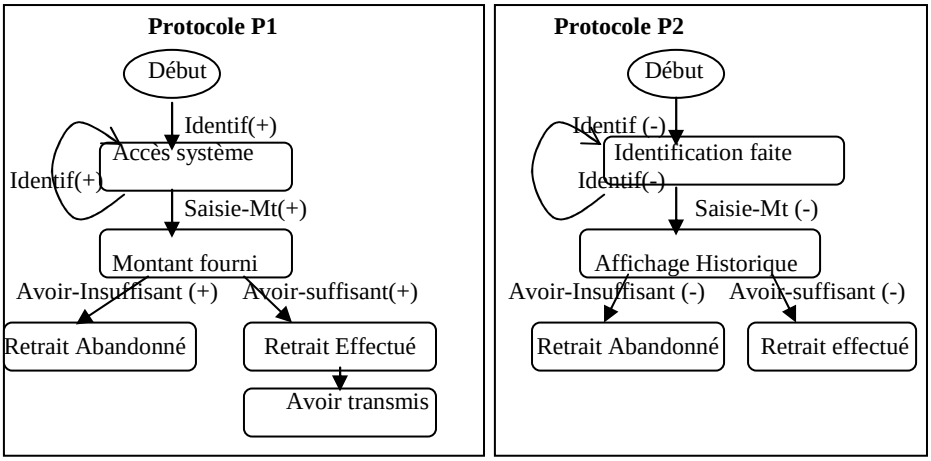

Figure 1. Deux protocoles partiellement compatibles sans considération des effets

Le chemin d'exécution [3]: (*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 conforme avec le protocole P_2. Donc, les deux protocoles sont partiellement compatibles [3].

Cependant, cette compatibilité, même partielle, n'est pas solide du fait que P_1 et P_2, lors de leurs transitions d'un état à l'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, déduction des frais bancaires du compte client. Alors que l'opération correspondante dans P_2 peut avoir d'autres effets: envoie historique des trois derniers mouvements et solde, compte fournisseur débité. Par conséquent, les protocoles P_1 et P_2 auront des protocoles de compensation différents en cas d'échec des opérations de retraits (abandon volontaire par les clients, annulation, transactions inachevées), car, seul P_1 applique des frais bancaires sur les opérations de retraits, en cas d'avoir insuffisant.

Nous constatons que la compatibilité des deux protocoles de services ne peut être abordée sans la prise en compte des effets engendrés par les messages.

1.2 Substitution fidèle des services lors de la composition

La composition des services vise à prendre en charge les processus métiers réels, en articulant plusieurs services élémentaires. Dans un tel contexte, la défaillance d'un service participant *P* exige une recherche d'un autre service P_s offrant, au moins, les

mêmes fonctionnalités. L'identification des services équivalents au service *P* pour le substituer doit être, impérativement, perçue en termes d'équivalence des effets des transactions et aux effets de la compensation. Le cas contraire, elle demeure superficielle (limitée à l'ordre des opérations).

Par la prise en compte et la modélisation des effets transactionnels et de leurs effets de compensation, la notion d'équivalence est étendue à la sémantique des messages. Elles offriront aux développeurs de services les outils adéquats et des modèles plus riches pour la gestion des protocoles de services avec leurs effets (comparaison, inclusion, annulation, décomposition).

1.3 Vérifier la consistance d'un protocole (Outils de Conception)

L'objectif du protocole de compensation est d'annuler sémantiquement les effets induits par le protocole déclencheur. La question est de s'assurer que le protocole de compensation est, effectivement, consistant avec le protocole déclencheur ? Autrement dit, comment vérifier que la compensation garantie, d'une manière correcte, une « *annulation sémantique* » des effets observés dans le monde du client ?

La modélisation des effets des transactions offrira un cadre conceptuel solide pour aborder cette problématique. Les environnements logiciels adéquats offriront aux développeurs des services Web les outils de conception et de développement nécessaires pour les décharger de certaines tâches et tests inutiles.

1.4 Inférer les propriétés transactionnelles des scénarios existants

Vu la disponibilité de programmes BPEL [6], il est opportun d'extraire leurs propriétés transactionnelles en vue de leur analyse et de leur manipulation. En effet, 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, dans leurs protocoles de services, des propriétés transactionnelles.

Dans cette perspective, la partie *Abstract*, décrivant le comportement d'un programmes BPEL, est analysée afin d'inférer ses propriétés transactionnelles. Les éléments de gestion des erreurs et des transactions; tels que:

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

Constituent des blocs d'activités liées aux propriétés transactionnelles qui doivent être identifiés, récupérés et modélisés pour leur éventuel traitement. Cette action offrira une assise solide pour la gestion de la compatibilité et de l'équivalence des protocoles de services existants, avec prise en charge des contraintes transactionnelles.

2. Etat de l'art de la gestion des transactions et de la représentation des effets dans les protocoles de services

L'étude de l'état de l'art de la gestion des transactions, au niveau protocole, fait ressortir quatre approches qui traitent, d'une manière plus ou moins rigoureuse, cet aspect, à savoir:

- Les **langages de modélisation des protocoles: (**WSFL et XLANG) qui apportent des extensions au standard WSDL, en offrant des structures de composition et de coordination basés sur des règles et des mécanismes pour la gestion des exceptions.

- **Les protocoles de transaction sur le Web:** Leurs spécifications actuelles contournent le problème de la gestion des transactions longues, en remettant en cause les propriétés ACID et en renforçant les mécanismes de compensation. Les deux seuls protocoles ayant traité cette problématique, de façon consistante sont: Business Transaction Protocol (**BTP**) [7] et Tentative Hold Protocol (**THP**) [8].

Pour 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*. La cohérence est assurée par la confirmation ou l'annulation d'un processus complet. En parallèle, les effets sont abordés suivant trois dimensions: *effets prévisionnels, effets finaux et contre-effets*, qui doivent être spécifiés et visibles dans les contrats des différents participants. Quant au protocol THP, il est basé sur le principe de la réservation et de l'affectation des ressources de la transaction courante en manipulant les concepts: *tentative, non-blocking holds* et *reservation*.

- **Les environnements de développement de services Web pour les entreprises** (Entreprise Java et les transactions XML). Les participants échangent des messages XML accompagnés des informations liées aux transactions (Contexte des transactions) pour accomplir des activités de coordination et des transactions multi-tiers.

- **Le modèle des transactions sur le Web** (**Web Service Transaction Model) (WSTx)** [9] qui propose une extension du langage WSDL pour décrire les

comportements transactionnels du client et du fournisseur. WSTx constitue un cadre de réflexion intéressant pour la maîtrise des mécanismes des transactions dans les services Web. Cependant, il souffre d'un déficit dans la modélisation des effets et ne propose qu'une classification des opérations WSDL à base de critères transactionnels.

En résumé, l'étude des approches existantes pour la gestion des transactions dans les protocoles de services fait ressortir les constats suivants:

1. Les spécifications existantes restent limitées aux aspects opérationnels et ne traitent les transactions que du point de vue manipulation de données.

2. La compensation est perçue indépendamment des effets. Cette façon de faire contraint certains modèles à son contournement artificiel par des garde-fous (THP).

3. Mise à part le modèle BTP, il n'existe pas d'autres modèles ni langages qui abordent les transactions par les effets engendrés dans la réalité. Mais, BTP ne les traite pas d'une manière formelle; c'est une simple classification des types d'effets.

4. Absence totale d'un modèle formel pour la représentation des effets transactionnels.

La prise en compte des contraintes transactionnelles dans les environnements services Web passe, inévitablement, par la modélisation de leurs effets engendrés dans le monde réel et qui constituent des caractéristiques intrinsèques du protocole de service. C'est un élément fondamental de la description du comportement externe du service. Le tableau 1, synthétise les caractéristiques des modèles rapprochant la représentation des effets des transactions dans les services Web.

Modèles / Critères	OWL-S [10]	BPEL	*Colombo* [11]
Concepts de base	Ontologie, Classe, Effets ServiceProfil, ServiceModel,	Activités, Variables, Scope Compensation	Requête de mise à jour de la base de données, Relation universelle
Méta-modèle formel	Logique de description	Langage	Modèle Relationnel
Notion d'Effet	Oui	Non	Oui
Notion d'état	Oui	Non	Oui
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 1: Evaluation et comparaison des modèles de représentation des effets transactionnels

Le déficit apparent dans la modélisation des effets des transactions a pour conséquence directe, de traiter les protocoles de compensation indépendamment de leurs protocoles déclencheurs, augmentant de ce fait, la charge de développement du service et induisant une situation d'indécidabilité pour la vérification de la consistance des deux protocoles, en plus d'une ambiguïté manifeste dans l'analyse d'équivalence.

Pour parer à ce déficit nous proposons, dans la prochaine section, un modèle formel de représentation des effets transactionnels, que nous injecterons par la suite dans le modèle de base des protocoles de services, décrit dans [3].

3. Modélisation des effets transactionnels et enrichissement de la description du modèle de protocoles

Basé sur la perception des effets en tant que requêtes de mise à jour au niveau de la base de données, le modèle *Colombo* [11] offre des avantages dans la maîtrise des notions d'*effet* et d'*état*. Par contre, il reste défaillant dans la gestion de la compensation et ne permet pas une manipulation comparative des effets.

Compte tenu du raisonnement que nous voulons faire sur les effets des transactions, à savoir: la comparaison, l'annulation, la contenance et la composition des effets, nous adapterons le principe du modèle *Colombo* pour la représentation des effets qui s'avère adéquat avec notre contexte.

Nous proposons la représentation des effets transactionnels et de leurs effets de compensation en se basant sur la perception d'un effet en tant que requête de mise à jour de la base de données. Ainsi, un message d'un protocole de service aura comme effets sur le monde du client une requête de mise à jour de type: *Insert(R), Delete (R)* ou *Modify(R);* où *R* est un enregistrement de la base de données, exprimant l'impact du message sur le monde du client. Le message est caractérisé, en plus, par une deuxième requête de mise à jour exprimant la compensation des effets engendrés par la requête *R*. La conséquence directe de cette modélisation, est que la problématique de la gestion des effets transactionnels est ramenée à celle de la manipulation (équivalence, inclusion, décomposition) des séquences des requêtes de mise à jour.

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 et qui sont décrits par les deux requêtes de mise à jour. Il est important de préciser que les fournisseurs de services appliquent des charges qui diffèrent, même si les effets sont les mêmes. La représentation conjointe des effets et de leurs effets de compensation, au niveau des messages des protocoles, reflète bien la réalité de la divergence des logiques de compensation.

- **Nouvelle structure des messages des protocoles de services enrichis par les effets transactionnels**

Conformément à notre modèle, à chaque effet d'un message est associé une requête de mise à jour de la base et sa requête correspondante liée aux effets de

compensation. La nouvelle structure d'un message est décrite comme suit: ***m (p, e, e')***, où :

m: Désigne le message et *p* sa polarité (+,-) selon qu'il est en entrée ou en sortie [3]

e: Ensemble des effets observés dans le monde du client, représenté par une requête *R* de mise à jour de la base de données.

e': Ensemble des effets de compensation pour défaire, sémantiquement, les effets *e* représentés par une requête *R'* de mise à jour de la base de données pour compenser les effets *e*, tout en appliquant des charges imposées par le fournisseur et relatives à l'annulation ou l'abandon de la transaction. On notera: *R' compense R*.

Cette modélisation exprime d'une manière formelle (requêtes relationnelles de mise jour) les effets des transactions et les effets des opérations de compensation, pour chaque message du protocole du service.

- **Modèle formel des protocoles de services enrichis par les effets transactionnels**

L'injection du modèle d'effets transactionnels proposé, dans le modèle de base des protocoles de service [3], en considérant la nouvelle structure du message, permet d'enrichir le modèle des protocoles représentés par des automates d'état finis déterministes. Le nouveau modèle de protocole de service (***Protocole à effets transactionnels***) est décrit par le tuple $P = (S, s_0, F, M, R, S_B)$ constitué des éléments:

S: Ensemble fini d'états; $s_0 \in S$ est l'état initial du protocole;

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

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

M: Ensemble fini de messages. On associe à chaque message *m* deux types d'effets *e* et *e'*, auxquels correspondent, les requêtes *Ri* et *Rj* de mise à jour de la base.

$R \subset (S \times S_B)^2 \times M$: Ensemble de transitions. Chacune concerne un état source, auquel est associé un état de la base, vers un état cible avec son état de la base, suite

à la réception du message. 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 leurs états de la base), associe les effets et les effets de compensation correspondants (en termes de requêtes de mise à jour).

La *fonction d'effet* $f_{eff}((s, s_b), e, e')$ associe à chaque état s du protocole 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.

4. Analyse d'équivalence des protocoles de services à effets transactionnels

Vu l'intérêt que revêt l'analyse d'équivalence des services Web [12] et, particulièrement, dans le contexte de la substitution d'un service défaillant impliqué dans un scénario de composition, elle mérite d'être revue à la lumière des enrichissements proposés. Effectivement, la représentation des effets transactionnels permettra un dépassement qualitatif de l'analyse d'équivalence au-delà de son simple aspect structurel. Elle sera, plutôt, basée sur la sémantique des transactions vue sous l'angle des effets produits dans le monde réel. Pour illustrer cet impact, les deux protocoles P_1 et P_2 de la Figure 2, ne seront équivalents que si les effets produits dans le monde, le soient. Autrement, si les requêtes de mise à jour associées aux messages de P_1 sont équivalentes à celles correspondantes dans P_2. (Les flèches montantes correspondent aux messages de compensation pour défaire les opérations inachevées).

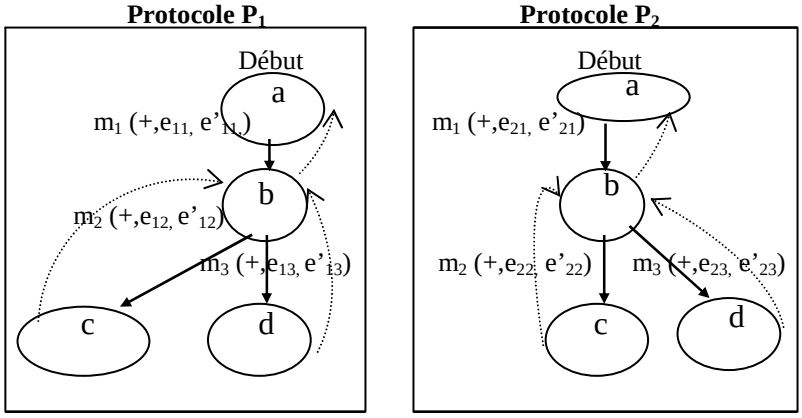

Figure 2 : Deux protocoles à effets transactionnels structurellement équivalents.

La condition d'équivalence est formalisée par les systèmes d'équivalence (1) et (2) des séquences de requêtes relatives aux chemins d'exécution complets, où les R_{ij} et R'_{ij} désignent les requêtes associées aux effets e_{ij} et aux effets de compensation e'_{ij}, respectivement. L'opérateur « *o* » exprime le séquencement des requêtes et l'opérateur « ≡ » exprime l'équivalence des séquences de requêtes.

$$\begin{cases} R_{11} \circ R_{12} \equiv R_{21} \circ R_{22} \quad \text{et} \quad R_{11} \circ R_{13} \equiv R_{21} \circ R_{23} & (1) \\ R'_{12} \circ R'_{11} \equiv R'_{22} \circ R'_{21} \quad \text{et} \quad R'_{13} \circ R'_{11} \equiv R'_{23} \circ R'_{21} & (2) \end{cases}$$

Nous schématisons dans la Figure 3, le processus réductionniste de transformation de la problématique de l'analyse d'équivalence des protocoles de services à effets transactionnels, en un problème de test d'équivalence des séquences de requêtes.

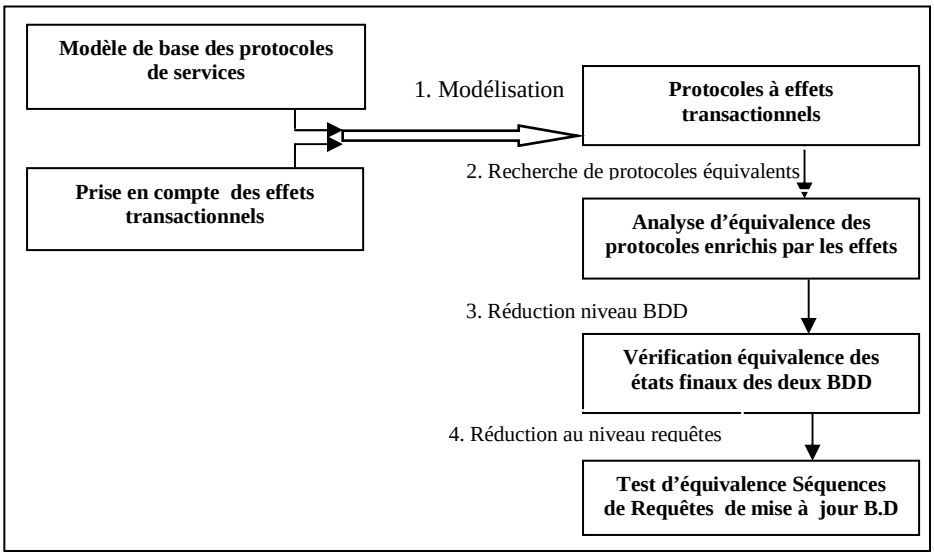

Figure 3 : Processus d'analyse d'équivalence des services à effets transactionnels

- **Classes d'équivalence des protocoles de services à effets transactionnels:**

L'identification des classes d'équivalence pour les protocoles de services a pour rôle de cerner les services fonctionnellement équivalents à un service donné, en vue d'une éventuelle substitution.

Conformément au processus d'analyse décrit dans la Figure3, et après étude de divers scénarios, nous avons déduit que l'équivalence des protocoles à effets transactionnels est conditionnée par l'équivalence des états finaux des deux bases de données. Mais, sans procéder à une vérification exhaustive, l'équivalence des états

finaux des deux bases de données, associées aux chemins d'exécution complets des protocoles [3] candidats, est examinée sur la base de l'équivalence des séquences des requêtes de mise à jour, en supposant qu'on démarre d'un même état initial.

Nous avons identifié deux types d'équivalence pour les séquences de requêtes: des équivalences strictes et des équivalences convergentes, qui vont induire deux classes d'équivalence pour les protocoles de services à effets transactionnels:

Equivalence Stricte et **Equivalence à convergence des états des bases,** que nous détaillerons dans ce qui suit :

- **Equivalence Stricte :** Dans cette classe, à chaque message *m* du protocole P_1 correspond, pour le message correspondant dans le protocole P_2, exactement, une seule requête qui lui est équivalente. *i.e* de même type (*Insert, Delete, Modify*), traite le même enregistrement et réalise les mêmes actions. Dans ce cas, les états des bases de données associées à P_1 et P_2 seront, évidemment, identiques, en partant d'un même état initial.

A titre d'exemple, si les systèmes d'équivalence des séquences de requêtes (1) et (2) sont vérifiés, alors, les protocoles P_1 et P_2 de la Figure 2 sont strictement équivalents.

Lemme 1: *Si deux protocoles de services à effets transactionnels P_1 et P_2 sont strictement équivalents, alors P_1 et P_2 sont structurellement équivalents (l'équivalence structurelle est basée sur l'ordre des messages uniquement).*

- **Equivalence à convergence des états des bases:** Dans ce cas, les séquences de requêtes relatives aux chemins d'exécution complets peuvent être équivalentes sans que ces chemins ne soient structurellement équivalents. A chaque séquence de requête d'un chemin d'exécution complet d'un protocole P_1 correspond, pour le même chemin dans P_2, une autre séquence de requêtes qui lui est équivalente. Ceci conduit à une convergence des mises à jour aboutissant à des états finaux équivalents des bases de données. Le protocole à effets transactionnels P_3 de la Figure 4 est équivalent au protocole P_1 de la Figure 2, même si leurs chemins d'exécution complets sont distincts.

Protocole P_3

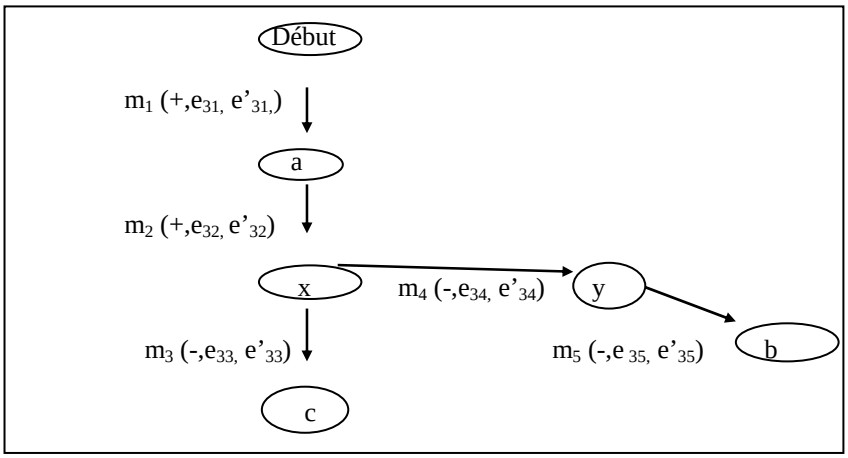

Figure 4. Un autre protocole qui fonctionne différemment de P_1. Est-il équivalent ?

Conformément à notre modèle d'effets, on associe des requêtes R_{ij} à chaque effet e_{ij} et des requêtes R'_{ij} aux effets de compensation e'_{ij} pour chaque message m_i. On abouti aux deux séquences de requêtes suivantes correspondantes aux deux chemins d'exécution complets de P_3.

Chemin d'exécution complet 1 : (vers l'état final c) : R_{31} o R_{32} o R_{33}

Chemin d'exécution complet 2 : (vers l'état final b) : R_{31} o R_{32} o R_{34} o R_{35}

Le test d'analyse d'équivalence des protocoles P_1 et P_3 est réduit à la vérification de l'équivalence des séquences de requêtes, exprimée par les systèmes (3) et (4).

$$\begin{cases} R_{11} \text{ o } R_{12} \equiv R_{31} \text{ o } R_{32} \text{ o } R_{33} \quad \text{et} \quad R_{11} \text{ o } R_{13} \equiv R_{31} \text{ o } R_{32} \text{ o } R_{34} \text{ o } R_{35} & (3) \\ R'_{12} \text{ o } R'_{11} \equiv R'_{33} \text{ o } R'_{32} \text{ o } R'_{31} \quad \text{et} \quad R'_{13} \text{ o } R'_{11} \equiv R'_{35} \text{ o } R'_{34} \text{ o } R'_{32} \text{ o } R'_{31} & (4) \end{cases}$$

La seconde classe d'équivalence revêt un intérêt particulier. Elle exprime une façon de faire différente de la part des fournisseurs de services qui est matérialisée 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 de données équivalente. Il s'ensuit que l'équivalence des protocoles à effets transactionnels peut être vérifiée sans l'équivalence structurelle. Par ailleurs, nous déduisons que l'équivalence stricte n'est qu'un cas particulier de l'équivalence à convergence des états des bases de données.

Vu cette importance, nous présentons, ci-après l'algorithme décisionnel permettant de vérifier si deux protocoles de services à effets transactionnels sont équivalents.

Algorithme 1 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* $P_2 = (S^2, s^2_0, F^2, M^2, R^2, S^2_B)$ *//les 2 automates*

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*} // On démarre de l'état initial*
3. s^1*-test :=* s^1_0 *;* s^2*-test :=* s^2_0 *// Commencer àpartir des deux états initiaux*
4. *Sequence1-Req := Ø, Sequence2-Req := Ø // les deux séquences sont nulles*
5. ***Tant que*** *Existe message* $m_{1i} \in M^1 \mid R^1((s^1\text{-test}, s^1_{s\text{-test}}), (s^1\text{-cible}, s^1_{s\text{-cible}}), m_{1i})$ ***Et*** *Continuer=Vrai* ***Faire***
6. *Récupérer la requête* R_{1i} *des effets* e_{1i} *de la transition de* m_{1i} (p, e_{1i}, e'_{1i})
7. *Sequence1-Req := Sequence1-Req o* R_{1i} *//* 1^{er} *séquence de* P_1
8. *Etat-testés1 := Etat-testés1 U {*s^1*-test} //ajout l'état courant de* P_1
9. ***Si*** s^1*-cible* $\in F^1$ ***Alors*** *// Si un chemin complet de* P_1 *est atteint*
10. *Sequence2-Req := Ø ;* s^2*-test :=* $s^2_{0\,//}$ *Chercher une séquence ≡ de* P_2
11. ***Tant que*** *Existe* $m_{2j} \in$ *Etats-courant2* ***Et*** *Trouver=Faux* ***Et*** $m_{2j} \in M^2 \mid R^2((s^2\text{-test}, s^2_{s\text{-test}}), (s^2\text{-cible}, s^2_{s\text{-cible}}), m_{2j})$ ***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. *Sequence2-Req :=Sequence2-Requêtes o*R_{2j} *//* $2^{ième}$ *séquence de* P_2
14. *Etat-testés2 := Etat-testés2 U {*s^2*-test} // ajout l'état courant de* P_2
15. ***Si*** s^2*-cible* $\in F^2$ ***Alors*** *// Chemin complet de* P_2 *atteint*
16. ***Si*** *Sequence1-Req ≡Sequence2-Req* ***Alors*** *Trouver := Vrai*
17. ***Sinon*** *Sequence2-Req :=Ø //Réinitialiser toutes variables*
18. s^2*-test :=* s^2_0 *//de* P_2 *pour pouvoir continuer les tests*
19. *Etats-Courant2 :=* s^2_0 *// d'existence d'un chemin d'exécution*
20. *Etat-testés2 := Ø // complet ayant une séquence équivalente*
21. ***Finsi***
22. ***Finsi***
23. *Etats-Courant2:= Etats-Courant2 U {*s^2*-cible}\ Etat-testés2*
24. ***Fin Tant que***
25. ***Si*** *Trouver = Faux* ***Alors*** *Continuer := Faux // pour continuer*
26. ***Finsi***
27. *Etats-Courant1:= Etats-Courant1 U {*s^1*-cible} \ Etat-testés1 // Eliminer*
28. ***Fin Tant que***
29. ***Si*** *Continuer=Vrai* ***Alors*** P_2 *est équivalent à* P_1 *// Tous les tests sont positifs*

Fin Equivalence-Convergente-Etats-BDD.

Comme résultat, le déroulement artificiel de l'algorithme sur les protocoles P_1 et $P3$ permet de conclure qu'ils appartiennent à la deuxième classe d'équivalence.

Conclusion et perspectives

Dans cet article, nous avons mis en évidence l'intérêt de la modélisation des effets transactionnels dans les protocoles de service et nous avons proposé un modèle, basé sur les requêtes de mise jour de la base de données pour leur représentation. Ce modèle est adéquat avec le raisonnement désiré et répond d'une manière consistante aux objectifs exprimés à travers les différentes motivations. Le nouveau modèle de protocoles de service, enrichi par les effets transactionnels, a été présenté et formalisé. Cependant, cet enrichissement a affecté l'analyse 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.

Notre seconde contribution est relative à la formalisation de l'analyse d'équivalence des protocoles de services à effets transactionnels, où deux classes d'équivalences ont été identifiées et l'algorithme d'équivalence convergente a été présenté.

Comme travaux futurs, nous envisageons d'aborder l'analyse de compatibilité et d'identifier les classes de compatibilité et l'élaboration des algorithmes adéquats. Nous envisageons, par ailleurs, la proposition d'un ensemble d'opérateurs de manipulation des effets transactionnels qui permettront 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.

Références

[1] F. Cabrera et al. Web Service coordination (WS-coordination), August 2005

[2] F. Cabrera et al. Web Service transaction (WS-transaction), January 2004. http://dev2dev.bea.com/pub/a/2004/01/ws-transaction.html/

[3] B. Benatallah et al : Representing, Analysing and Managing web Service Protocols. Data Knowledge Ingineering. 58 (3): 327-357, 2006

[4] J. Ponge et al: Fine-Grained Compatibility and Replaceability Analysis of Timed Web Service Protocols. ER 2007: 599-614

[5] Gustavo Alonso, Fabio Casati, Hurumi Kuno,Vijay Machiraju : Web services concepts Architectures and applications, Edition Springer Verlag Berlin 2004

[6] Web Services Business Process Execution Language Version 2.0 OASIS Standard, 11 April 2007, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html/

[7] OASIS Business Transaction Protocol (BTP), http://oasis-open.org/committees/business-transactions/

[8] J. Roberts , K. Srinivasan. Tentative Hold Protocol Part 1: White Paper, W3C Note 28 November 2001, http://www.w3.org/TR/tenthold-1/

[9] T. Mikalsen et al, Transactional attitudes: Reliable composition of autonomous Web Services, WDMS 2002.

[10] 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/

[11] D. Berardi et al, Automatic composition of Transition-Based Semantic Web Services with Messagging, Proceeding of 31st VLDB conference, Trondheim, Norway, 2005

[12] D. Grigori, et al., Behavioral matchmaking for service retrieval: Application to conversation protocols, Informat. Systems (2008), doi:10.1016/j.is.2008.02.004