Impact de la prise en compte des Contraintes Transactionnelles sur l'Orchestration des Services Web

Khebizi Ali[1], Hassina Seridi[2]
[1]Département d'Informatique, Université 8 Mai 45, Guelma 24000, Algérie
kheali@hotmail.com
[2]Université Badji Mokhtar, BP 12 Annaba 23000, Algérie
LabGed Laboratoire de Gestion du Document
seridi@labged.net

Résumé: Les progrès réalisés par la technologie des services Web demeurent insuffisants pour la prise en charge des vrais processus métiers trans-organisationnels. En effet, dans les scénarios de composition des services, l'élaboration des schémas d'orchestration et la gestion de la compensation demeurent des processus complexes et exigent une description plus riche des interactions engagées entre les divers services participants.
Dans cet article, basé sur l'analyse d'un scénario réel d'une application e-gouvernement dénommée «*Retraite*», nous aborderons l'analyse de compatibilité et d'équivalence des protocoles de services enrichis par les contraintes transactionnelles, et ce lors de l'orchestration des servies Web et nous proposerons une approche pour la modélisation des protocoles de compensation associés aux transactions inachevées.

Mots Clés: Protocoles de services, Contraintes Transactionnelles, Orchestration de Services, Compatibilité de services, Equivalence de services, Compensation.

1. Introduction

Les services Web offrent de fortes potentialités pour l'intégration des applications intra et interentreprises issues d'environnement hétérogènes et distribués sur le Web. Cependant, la première génération de services Web basée sur l'infrastructure: WSDL [1], SOAP [2] et UDDI [3], n'a permet que de surmonter le problème d'interopérabilité entre plateformes hétérogènes et reste, alors, insuffisante pour l'automatisation complète des vrais processus métiers trans-organisationnels. En effet, cette infrastructure de base est inadéquate pour la prise en charge de la composition des services afin de créer de nouvelles applications à forte valeur ajoutée sur la base de celles déjà existantes. La seconde génération de standards basée sur WS-Coordination [4], WS-Transaction [5] et BPEL [6] renforce l'infrastructure existante par les outils et mécanismes appropriés permettant la description des processus métiers, la composition des services et la gestion des interactions. Néanmoins, la nature intrinsèque des transactions métiers qui sont de longue durée et consommatrices de ressources exige des techniques de compensation dépassant les classiques propriétés ACID (Atomicité, Cohérence, Isolation, Durabilité). En effet, dans les environnements services Web, la compensation d'un protocole inachevé est assurée par le middleware d'une manière transparente en exécutant un autre protocole de compensation prédéfini par le développeur du service [7].

Or, la description actuelle du comportement externe des services, par leur interface et par leur protocole de conversation, n'exprime pas d'une manière consistante les caractéristiques inhérentes à la sémantique réelle des interactions engagées lors de la composition des services, bien que les contraintes d'ordre sur les opérations [8] et les contraintes temporelles [9] ont été prises en compte et injectées dans le modèle des protocoles de services modélisés par les automates d'états finis déterministes. En effet, les contraintes transactionnelles et les effets qu'elles induisent n'ont pas bénéficié de tout l'intérêt qu'elles revêtent.

Dans un tel contexte, le rehaussement de la description des protocoles de services par la prise en compte des contraintes transactionnelles impactera, promptement, l'analyse de compatibilité et d'équivalence des protocoles de services telles qu'elles ont été formalisées dans [8].

Nous proposons dans cet article un réexamen de l'analyse de compatibilité et d'équivalence des protocoles de services enrichis par les contraintes transactionnelles [10], et ce lors de l'orchestration des services Web. Nous exposerons, par la suite, une formalisation du processus de compensation des opérations et/ou services avortés et nous proposerons une modélisation des protocoles de compensation.

La suite de l'article sera structurée comme suit : dans la section 2, nous motivons notre travail, par l'exposé d'un scénario réel illustrant la substitution, la compatibilité et la compensation des services engagés lors de l'orchestration des services. L'analyse de compatibilité des protocoles de services, enrichis par les contraintes transactionnelles est présentée en section 3, et leur analyse d'équivalence est abordée au niveau de la section 4. La section 5, sera consacrée à la modélisation du processus de compensation. En fin, la section 6 contiendra nos conclusions et nos travaux futurs.

2. Intérêt et impact de la prise en compte des contraintes transactionnelles lors de l'orchestration des services Web

Avec la prolifération des services Web, leur composition est devenue un enjeu majeur pour les développeurs. Cette technique d'implémentation de nouveaux services, par articulation de ceux déjà existants, répond parfaitement aux objectifs de réutilisabilité des composants logiciels. Cependant, les problèmes de compatibilité des services découverts et susceptibles d'être intégrés dans la composition, les difficultés liées à la recherche de services équivalents afin de substituer ceux qui sont défaillants ou devenus incompatibles suite à leur évolution, ainsi que la complexité de la gestion des transactions et de leur compensation demeurent résiduels. Ces constats sont dus, principalement, à des déficits conceptuels au niveau de la description des protocoles des services eux-mêmes, et plus particulièrement, à l'absence de la prise en compte des contraintes transactionnelles dans la modélisation des comportements externes des services.

Afin de mettre en évidence les insuffisances précitées, nous exposons dans ce qui suit un scénario réel d'un service « *Retraite* » relatif à une application e-gouvernement du domaine de la sécurité sociale qui permet de prendre en charge les demandes de retraite des citoyens demandeurs.

Pour l'élaboration du service Web *Retraite*, permettant de liquider les droits en matière de retraite d'un citoyen, le développeur compose les services *Pension* (à son

tour composé des services : *Carrière* et *Etat- civil*), le service *Déclaration-Salaires* et le service *Assurance sociale*, comme illustré dans la **Fig.1.**
L'objectif du service *Retraite* est de permettre au client qui l'invoque, de prendre connaissance de ses droits de retraite, d'être notifié en cas de validation de sa demande et de bénéficier des prestations sociales.

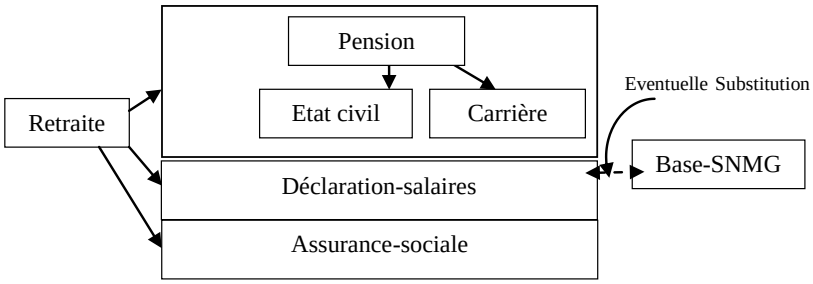

Fig. 1 : Composition du service *Retraite* par articulation d'autres services

Description des fonctionnalités des différents services à composer:

- Service ***Pension***: Permet de vérifier et de calculer les droits en matière de retraite sur la base des données état-civil et des données carrière.
- Service ***Carrière***: Fournit les données carrière (période d'activité, date entrée,…)
- Service ***Etat civil***: Fournit les données sur la situation matrimoniale et la liste des ayant-droits. Ces éléments entrent en compte dans le calcul des droits de retraite.
- Service ***Déclaration-salaires***: Fournit les salaires de référence servant au calcul de la retraite (moyenne des salaires des cinq meilleures années d'activité).
- Service ***Assurance-sociale***: le futur retraité fera l'objet d'une déclaration au niveau de l'organisme de sécurité sociale (service **Assurance-sociale)** pour pouvoir bénéficier des prestations sociales.
- Service **Base-SNMG** : En cas de défaillance du service ***Déclaration-salaires***, ce service fournira le salaire de référence (Salaire National Minimum Garanti).

Les différents services sont exposés par divers organismes (fournisseurs): la caisse des retraites pour les services *Pension, carrière* et *base-SNMG*, la caisse d'assurance sociale pour *Assurance-sociale*, la mairie pour le service *Etat-civil* et le service *Déclaration-salaires* est exposé par les divers employeurs.
D'un point de vue implémentation, le service *Retraite* est composé car il invoque d'autres services et l'orchestration traite, justement, de la façon dont ces différents services sont composés en un tout cohérent. Elle spécifie l'ordre dans lequel les services sont invoqués et les conditions sous lesquelles un service peut ou ne peut pas être invoqué. [7]. C'est un processus coopératif impliquant différents partenaires qui vise à intégrer de façon cohérente la collaboration des diverses activités des services participants. La notion de *protocole de service* [8] permet de représenter le processus métier composite «*Retraite*».

Nous détaillons, dans la **Fig.2**, le protocole du service composé *Retraite*. Le formalisme des automates d'états finis déterministes est utilisé comme outils de représentation. Les messages expriment les opérations permettant la transition entres les différents états de l'automate.

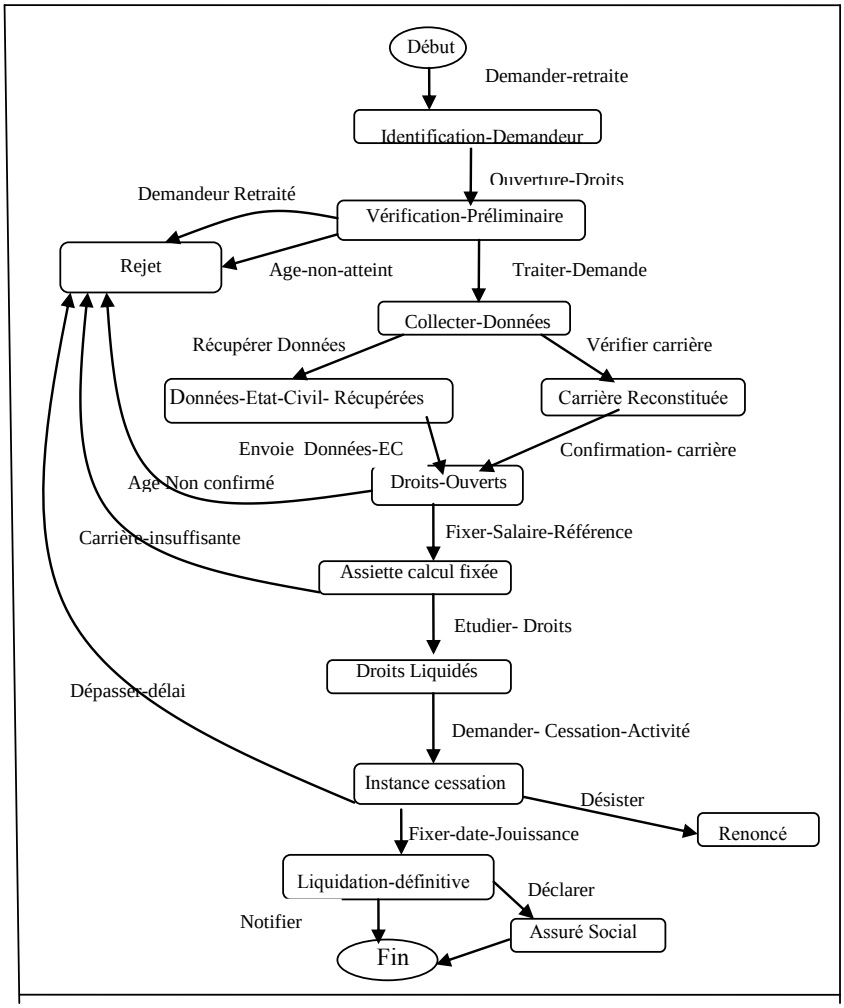

Fig. 2 : Protocole du service composé *Retraite*

Du point de vue interaction, la composition exige les conditions suivantes:

(i) Le protocole du service composite « *Retraite* » doit être compatible avec ceux des fournisseurs (*Déclarations-salaires*, *Assurance-sociale*) pour garantir que des conversations correctes puissent être engagées. Cette compatibilité doit être étendue aux effets engendrés par les différents messages.

(ii) En cas de défaillance d'un service (Déclaration-salaires par exemple), le service composite doit localiser un autre service fonctionnellement équivalent pour le substituer. La notion d'équivalence doit maintenir la cohérence de l'orchestration tout en tenant compte des contraintes transactionnelles. Le service Base-SNMG ne pourra substituer le service Déclaration-salaires sans que leurs effets ne soient équivalents.

(iii) Si le demandeur ne fournit pas une cessation d'activité dans un délai déterminé (06 mois), alors tout le processus est annulé, entrainant le déclenchement d'un nouveau protocole de compensation pour annuler sémantiquement les effets engendrés. L'élaboration du protocole de compensation doit, impérativement, tenir compte des effets transactionnels afin de pouvoir les défaire.

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 (WS-Transaction [5], WS-Transaction Atomic [12] et WS-Business Activity [13]), le domaine de la gestion des transactions au niveau protocole reste faiblement exploré engendrant des difficultés d'analyse et de programmation lors de la composition des services Web. Par la prise en compte et la modélisation des effets transactionnels les notions de compatibilité, d'équivalence et de compensation prennent une dimension plus riche, car basées sur la sémantique réelle des interactions vue sous l'angle des effets engendrés par les transactions déclenchées.

Pour rehausser la description des protocoles de services à sa sémantique interactionnelle en prenant en considération les effets des transactions, nous avons proposé dans nos travaux précédents [10] un modèle formel de représentation des effets basé sur leur perception en tant que requêtes de mise à jour de la base de données de type: *Insertion, Modification et Suppression.* Les requêtes de consultation n'affecteront, en aucun cas, l'état du client. Conformément à notre modèle, la structure des messages échangés lors des interactions est de type: ***m (p, e, e')***, où :

m: le message et *p* sa polarité (+,-). Une polarité (+) indique que le message est en entrée (consommée) et une polarité (-) signifie qu'il est en sortie (produit) [8].

e: Ensemble des effets observés du coté du client, représenté par une requête Q de mise à jour de la base de données.

e': Ensemble des effets de compensation pour défaire les effets *e* représentés par une requête *Q'* de mise à jour pour compenser les effets *e*. On notera: *Q' compense Q*.

A noter que si les requêtes sont de type consultation qui n'affectent pas le client, le message sera noté : ***m (p, *,*)***

Par ailleurs, le nouveau modèle de protocole de service enrichi par les contraintes transactionnelles (***Protocole à effets transactionnels***) est représenté par l'automate d'état fini déterministe, décrit par le tuple $P = (S, s_0, F, M, R, S_B)$ [10].

Néanmoins, cet enrichissement des protocoles de services par la prise en compte des propriétés transactionnelles, impactera directement le processus d'orchestration des services et exige, par conséquent, un réexamen de leur gestion lors de l'orchestration. Nous aborderons, dans la suite de l'article, l'analyse de compatibilité et d'équivalence des protocoles à effets transactionnels et nous formaliserons le processus de compensation.

3. Analyse de Compatibilité des Protocoles à Effets Transactionnels lors de l'Orchestration des Services

L'analyse de compatibilité de deux protocoles permet de tester s'ils peuvent interagir correctement en engageant des conversations correctes entre eux [8]. Pour les protocoles de services à effets transactionnels, cette analyse consiste à vérifier si les deux protocoles en interaction engendrent les mêmes types d'effets.

Conformément au modèle d'effets proposé [10], les effets induits par les messages d'un service Web sont des actions affectant le client qui l'invoque et qui sont matérialisées par la modification de son état. La conséquence directe de cet apport est la révision de la notion de compatibilité des services.

Dans cette optique, les protocoles du service client et celui du fournisseur doivent induire des effets qui seront compatibles. Par *compatibilité des effets*, nous sous-entendons que les requêtes de mise à jour au niveau des bases de données sont de même type. Cette condition conduit à une redéfinition du concept de chemin d'interaction, tel qu'il a été spécifié dans [8] pour l'étendre au type de requête.

- **Chemin d'interaction étendu:** La définition du chemin d'interaction dans [8] est limitée aux messages et aux états. Elle est décrite conformément à l'expression générique: ***((Etat1.Etat2).Message)****

Pour prendre en compte les effets transactionnels, nous proposons une extension de cette spécification au type de requêtes de mise à jour, comme suit:

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

Cette extension permettra de garantir, lors de l'analyse, la vérification de la compatibilité des requêtes de mise à jour. Elle favorise une spécification plus riche de l'interaction entre les protocoles. Ainsi, deux protocoles de services ne pourront être compatibles sans que les requêtes associées aux messages ne le soient. Le besoin d'une nouvelle spécification de la compatibilité nous conduit à une redéfinition des concepts: *Effets compatibles* et *messages compatibles*.

- **Effets 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 celui de l'analogie des types de requêtes des deux messages. Cependant, dans certaines situations, 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 Modifier dans un premier protocole, alors qu'au niveau du 2ième protocole, c'est une séquence d'une requête Insertion suivie d'une autre requête de Suppression. Dans ce cas, la compatibilité des effets est préservée, malgré l'incompatibilité des messages. Un deuxième cas concerne la décomposition d'une requête Modifier sur plusieurs attributs et/ou des sous ensembles d'attributs, où chaque modification est portée par un message à part. Pour illustrer ce deuxième type de compatibilité, nous reprenons le scénario de la Fig.2. La logique métier du service client peut traiter la cessation d'activité différemment de celle du fournisseur. Par exemple, au niveau du protocole du client de la Fig. 3, où les messages sont décrits par leurs effets et effets de compensation représentés par des requêtes (pour simplifier nous partons de l'état *Droits-Liquidés*). Les deux attributs : *Date-jouissance* et *Situation* de l'enregistrement concerné sont

modifiés en une seule passe en invoquant le message: *Notifier- Décision*, permettant la transition de l'état *Instance-cessation* vers l'état *Fin*. Dans ce cas, est-ce que des conversations correctes peuvent être engagées avec le service du fournisseur de la **Fig. 2** ? La réponse à cette question est positive, si les requêtes de mise jour sont de même type, même si la mise à jour est faite de manières différentes.

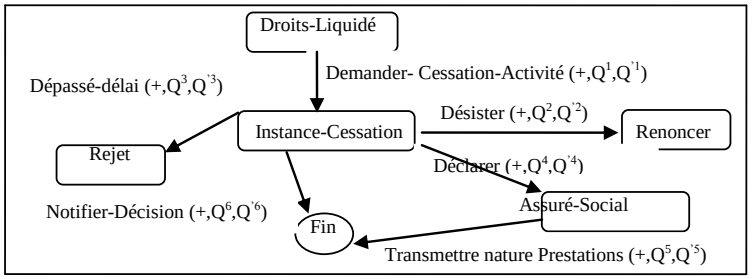

Fig. 3: Un protocole client qui fonctionne différemment. Pourra-t-il interagir correctement ?

- **Messages compatibles:** Deux messages appartenant à deux protocoles différents sont compatibles si les requêtes de mise à jour associées sont de même type.

A titre d'exemple : le message *Traiter-Demande* du protocole fournisseur du service *Retraite* **de la Fig. 2**, dont la requête de mise à jour est de type *Modification* ne pourra être invoqué correctement par le message : *Demander-retraite* d'un autre protocole client sauf si la requête associée à ce dernier est aussi de type *Modification.* Le cas contraire, il est évident que les effets observés seront non conformes et par conséquent, la conversation ne sera pas possible.

- **Définition formelle de la compatibilité des protocoles à effets transactionnels:**

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* $P_2=(S^2\ s^2_0, F^2, M^2, R^2, S^2_B)$ [10]

P_1 *et* P_2 sont compatibles s'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 de P_1 sont compatibles avec ceux de P_2.

Soit m_i le message à invoquer pour la transition courante, avec: $m_i(p^1,Q_1,Q'_1) \in P_1$ et $m'_i(p^2, Q_2, Q'_2) \in P_2$. La compatibilité des messages m_i et m'_i est vérifiée si :

- -Les polarités p^1 *et* p^2 se compensent (+, -) ;
- -En partant des mêmes états sources, les états cibles sont identiques, ou transitivement identiques; (transition par des états intermédiaires).
- -Les requêtes de mise à jour Q_1 et Q_2 sont de même type ou transitivement de même type (l'exécution d'une séquence de requêtes est de même type).
- -Les requêtes de compensation Q'_1 et Q'_2 sont de même type ou transitivement de même type (l'exécution d'une séquence de requêtes de compensation est de même type).

Plus concrètement et en se basant sur le concept de *la compatibilité des effets*, deux situations peuvent se présenter :

(i) Même type des requêtes de mise à jour: C'est le cas le plus simple. Il correspond à une compatibilité évidente des messages. Par exemple les messages *Etudier-droits* du service *Retraite* de la **Fig. 2**, permettant la transition de l'état *Assiette-calcul-fixée* vers l'état *Droits-liquidés* à pour effets de modifier l'enregistrement du demandeur en actualisant les attributs (tels que : salaire-mensuel, taux-retraite). Si le message du protocole du client est aussi de type *modification*, alors la compatibilité est garantie. Le cas contraire, l'interaction ne sera pas permise, même si elle est vérifiée du point du vue ordre et polarité des messages.

(ii) Cas de compatibilité des types de séquences de requêtes: Induisant les mêmes types d'actions au niveau des bases de données. C'est le cas le plus général et correspondant à des situations de transitivité. Dans ce cas les logiques métiers du service client et celui du fournisseur peuvent être différentes, en termes d'ordre des opérations ou des états parcourus, mais elles demeurent canalisées par des garde-fous qui sont les types de requêtes, garantissant une conversation correcte.

A titre d'illustration, considérant un client dont la logique métier pour le traitement des demandes de retraite est schématisée par le protocole de service à effets transactionnels de la **Fig. 3.** Si les requêtes des messages: *Demander-Cessation-Activité*, *Désister* et *Délai-dépassé* sont du même type que ceux correspondantes dans le protocole *Retraite* de la **Fig. 2**, la compatibilité partielle des chemins aboutissant aux états finaux *Renoncer* et *Rejet* est vérifiée. Par contre, pour les chemins d'exécution complets: *Droits-Liquidés.Instance-Cessation.Assuré-Social.Fin* et *Droits-Liquidés.Instance-Cessation.Fin, elle* n'est pas évidente. En prenant en compte les effets des opérations, l'interaction ne sera correcte que si les effets soient compatibles. Comme, la transition de l'état : *Instance-cessation* vers l'état *Fin* est déclenchée par le message unique *Notifier-Décision*$(+,Q_6,Q'_6)$ dans le protocole du service client, alors qu'elle est réalisée par deux messages distincts : *Fixer-date-jouissance*$(-,Q_7,Q'_7)$ et *Notifier-Retraite* $(-,Q_8,Q'_8)$ dans le protocole du service fournisseur de la **Fig.2**, la compatibilité est préservée seulement si: Q_6 est de même type que la séquence $Q_7 o\, Q_8$, où *o* désigne d'ordonnancement des requêtes.

4. Analyse d'Equivalence des Protocoles à Effets Transactionnels lors de l'Orchestration des Services

L'analyse d'équivalence a pour objectifs de s'assurer que deux protocoles de services peuvent être utilisés interchangeablement dans n'importe quel contexte d'une manière transparente à l'utilisateur [8]. Elle est utile dans le contexte de la substitution d'un protocole défaillant (panne de service, évolution de version, …) par un autre qui offrira, au moins, les mêmes fonctions. Dès lors, son intérêt est grandissant lors de l'orchestration des services. Cependant, cette analyse exige une révision et un raffinement spécifique pour les protocoles à effets transactionnels, du fait qu'elle sera, plutôt, basée sur la sémantique des transactions vue sous l'angle des effets produits.

Pour illustrer cet impact, les deux protocoles des services *Déclaration-Salaires* et *Base-SNMG*, intégrés dans la composition du service *Retraite* et schématisés par la **Fig. 4**, ne seront équivalents que si les effets produits, le sont. Autrement, si les requêtes de mise à jour associées aux messages sont équivalentes.

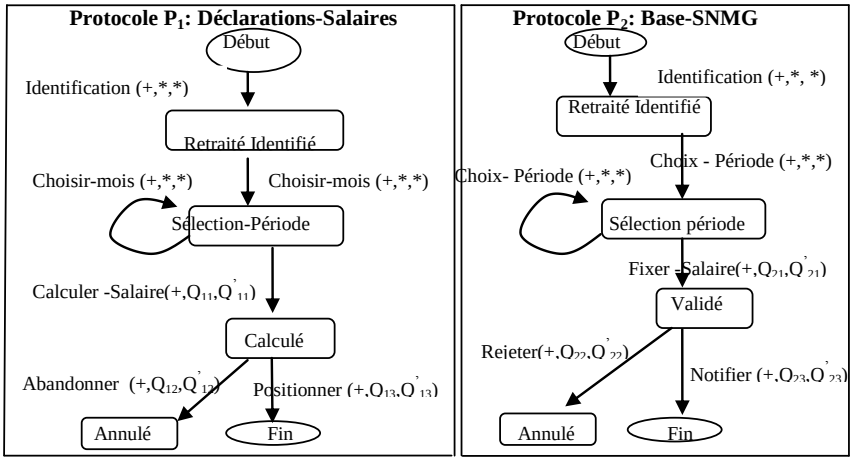

Fig. 4: Deux protocoles: Equivalence structurelle vs Equivalence à base d'effets

Les messages de la forme ***m(p,*,*)*** correspondent à de simple consultations de la base et n'induisent aucun effet. Pour que les protocoles P_1 et P_2 soient équivalents, il faut que les équations (1) et (2) soient vérifiées: (le symbole ≡ exprime l'équivalence des requêtes) de mise à jour au niveau de la base de données.

$$\begin{cases} Q_{11} \equiv Q_{21} \text{ et } Q_{12} \equiv Q_{22} \text{ et } Q_{13} \equiv Q_{23} & (1) \\ Q'_{11} \equiv Q'_{21} \text{ et } Q'_{12} \equiv Q'_{22} \text{ et } Q'_{13} \equiv Q'_{23} & (2) \end{cases}$$

La réalisation des équations (1) et (2), simultanément, explique une façon de faire, de la part des fournisseurs de services, parfaitement identique; chose qui n'est pas évidente dans la réalité, du fait que chaque fournisseur a sa propre logique métier qui lui est spécifique. Effectivement, certaines opérations peuvent être regroupées, d'autres sont éclatées ou ordonnées différemment. Par ailleurs, les logiques de compensation diffèrent, éventuellement, d'un fournisseur à l'autre, engendrant la non vérification de l'équation (2), même si (1) est vérifiée. Ces constats, nous conduisent à raisonner en termes d'effet globale induit par la séquence de requêtes pour chaque chemin d'exécution complet (qui commence d'un état initial et se termine par un état final [8]).

L'équivalence des effets globaux est formalisée par les équations (3) et (4).

$$\begin{cases} Q_{11} \circ Q_{12} \equiv Q_{21} \circ Q_{22} \text{ et } Q_{11} \circ Q_{13} \equiv Q_{21} \circ Q_{23} & (3) \\ Q'_{12} \circ Q'_{11} \equiv Q'_{22} \circ Q'_{21} \text{ et } Q'_{13} \circ Q'_{11} \equiv Q'_{23} \circ Q'_{21} & (4) \end{cases}$$

L'examen approfondi de la vérification des équations (1), (2), (3) et (4) conduit à plusieurs situations possibles. Chacune de ces situations détermine une ***classe d'équivalence*** des protocoles à effets transactionnels.

1. L'équation (1) seule est vérifiée : Equivalence Stricte des effets directs.
2. L'équation (2) seule est vérifiée: Equivalence Strict des effets de compensation.
3. Les équations (1) et (2) sont vérifiées: Equivalence Stricte Parfaite.
4. L'équation (3) seule est vérifiée: Equivalence convergente à effets directs des états des bases de données.
5. L'équation (4) seule est vérifiée: Equivalence convergente à effets de compensation des états des bases de données.

6. Les équations (3) et (4) sont vérifiées : Equivalence Parfaite à convergence des états des bases de données.

La dernière classe d'équivalence revêt un intérêt particulier. Elle reflète une manière 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 convergeant vers des bases de données équivalentes. Il s'ensuit que l'équivalence des protocoles à effets transactionnels peut être vérifiée sans que l'équivalence structurelle le soit. 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.

5. Processus de compensation des services lors de l'Orchestration

Le mécanisme de compensation offre un cadre conceptuel adéquat pour défaire sémantiquement les effets engendrés par les transactions inachevées durant l'orchestration d'un service Web. Mais, ce processus en lui-même, demeure fastidieux car basé sur la simple manipulation des données et aucune garantie n'est offerte pour vérifier sa **consistance** avec le protocole déclencheur. En se basant le modèle de représentation des effets intégrant, conjointement, pour chaque message les effets induits et leurs effets de compensation modélisés par des requêtes sous la forme ***m (p, Q, Q')***, il apparait que la problématique de la compensation des services lors de l'orchestration est réduite à celle de la manipulation des requêtes de mise à jour de la base de données. En effet, pour défaire les effets induits par une requête Q d'un message m, il suffira d'exécuter la requête inverse Q'. Cette simplicité, offerte par la richesse du modèle de représentation, permet d'identifier clairement les requêtes nécessaires à la modélisation du protocole de compensation et de les structurer dans l'ordre exprimant la logique de compensation afin de construire le protocole de compensation.

Définition formelle de la compensation: Une requête Q_j décrivant les effets observés du côté du client et induits par une opération d'un service Web S, compense les effets d'une autre requête Q_i, lors de l'exécution du service Web S, et on notera : $[Q_j\ comp(Q_i)]_{S,}$ si l'exécution de la séquence de requêtes : $Q_i\ o\ Q_{j,}$ n'engendre aucun effet du côté du client, sauf ceux, volontairement, désirés par le fournisseur de S et exprimant les charges des transactions imposées aux clients, suite à l'annulation de l'opération. On parlera de *couple effet-compensation*.

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

e : Les effets du message m induits par la requête Q_i.

e': Les effets du message du protocole compensatoire représenté par la requête Q_j.

Les effets de compensation peuvent être décomposés en deux types de sous-effets:

e'_a: Les effets d'annulation des effets du message e, représenté par la requête Q_a.

e'_f : Les effets volontairement désirés par le fournisseur du service et qui sont associés aux charges affectés au client suite à l'annulation de la transaction. Ils sont exprimés par la requête Q_f.

La décomposition des effets associés aux requêtes donnera: $e' \equiv e'_a + e'_f$;

$Q_j = Q_a\ o\ Q_f$; d'où : $[(Q_a\ o\ Q_f)\ comp(Q_i)]_{S;}$ exprimant le fait que la séquence de requêtes $Q_a\ o\ Q_f$ compense la requête Q_i dans le protocole de service S.

Il est claire que si : $Q_f \equiv \emptyset$ (pas de charges : *Effectless transactions* dans [11]), alors, l'annulation de la transaction n'affecte pas l'état de la base et on aura: $[Q_a\ comp(Q_i)]_S$.
Cette approche pour aborder le processus de compensation, permet de doter les développeurs de services des outils adéquats pour la modélisation et la vérification des protocoles de compensation, tout en garantissant que la compensation, assure effectivement, l'annulation sémantique du protocole déclencheur. Cette façon de faire est très rentable en termes de productivité du fait qu'elle évite de procéder par des suites et scénarios de tests qui sont non exhaustifs.
Pour illustrer cet apport, nous exposons dans la **Fig. 5**, le protocole de compensation du protocole P_1 : *Déclaration-Salaires* de la **Fig. 4**.
Ce protocole est invoqué implicitement par le fournisseur de service dans le cas ou le client invoque l'opération *Abandonner* permettant la transition de l'état *Calculé* vers l'état *Annulé* du protocole P_1 de la **Fig. 4**.
La logique métier du fournisseur consiste à annuler les modifications réalisées par les requêtes : Q_{11} et Q_{12}, en exécutant séquentiellement les requêtes Q'_{11} et Q'_{12}. A noter que ces deux dernières requêtes ont des effets permanents et ne peuvent être compensées (*Definite transitions* dans [11]).

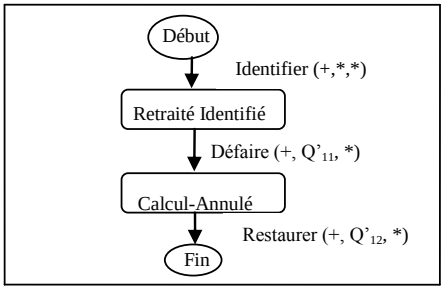

Fig. 5: Protocole de compensation du protocole P_1: *Déclaration-Salaires*

Dans la pratique, il est opportun de décomposer les requêtes complexes en un ensemble de requêtes basiques, dont la séquence d'exécution produira les mêmes effets au niveau de la base, permettant, ainsi d'identifier les effets de bases et de faciliter par conséquent la spécification des requêtes de compensation associées. En plus, la recherche d'une requête globale, dont l'effet résultant reflète l'annulation d'une séquence de requêtes, permettra de compenser une séquence d'opérations en une seule passe. (situation où tout le service est abandonné en une seule opération: fermeture de la fenêtre active, par exemple).

6. Conclusions

La composition des services Web est une technique permettant la réutilisation des services déjà existants pour créer de nouveaux services à forte valeur ajoutée. Pour sa mise en œuvre opérationnelle, la connaissance de la structure et du comportement des services constituants est inévitable, afin de s'assurer de la compatibilité et de l'équivalence des services à composer. Dans cet article, nous avons mis en évidence l'intérêt de la prise en compte des contraintes transactionnelles inhérentes aux interactions engagées lors des scénarios de composition de type orchestration des

services. Nous avons abordé l'analyse de compatibilité et d'équivalence des protocoles de services à base d'effets transactionnels, tout en identifiant les classes d'équivalence. Une approche, basée sur le modèle de requêtes de mise à jour de la base de données, et permettant de modéliser les protocoles de compensation des opérations et/ou services avortés a été proposée.
Comme travaux futurs, nous envisageons la spécification des algorithmes relatifs aux différentes classes d'équivalence déjà identifiées ainsi que la proposition des opérateurs algébriques de manipulation des effets transactionnels, afin de consolider l'assise théorique existante.

Références

1. R. Chinnici et al. Web Services description Language (WSDL) version 2.0 June 2007. http://www.w3.org/TR/wsdl20/
2. M. Gudgin et al. SOAP version 1.2, July 2001. http://www.w3.org/TR/2001/WD-soap12-20010709/
3. T. Bellwood et al. UDDI Version 3.0.2 UDDI Spec Technical Committee Draft, 2004. http://uddi.org/pubs/uddi_v3.htm/
4. 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
5. F. Cabrera et al. Web Service transaction (WS-Transaction), January 2004. http://dev2dev.bea.com/pub/a/2004/01/WS- Ttransaction.html/
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. Gustavo Alonso et al: Web services concepts Architectures and applications, Edition Springer Verlag Berlin 2004.
8. B. Benatallah et al : Representing, Analysing and Managing web Service Protocols. Data Knowledge Ingineering. 58 (3): 327-357, 2006.
9. J. Ponge et al: Fine-Grained Compatibility and Replaceability Analysis of Timed Web Service Protocols. ER 2007: 599-614
10. Ali Khebizi: External Behavior Modeling Enrichment of Web Services by Transactional Constraints, ICSOC PhD Symposium, December 2008. http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-421/paper12.pdf/
11. B. Benatallah et al : Web Service Conversation Modeling A cornerstone for E-Business automation, IEEE Internet computing 8 (1) (2004) 46-545 WSC
12. F. Cabrera et al. Web Services AtomicTransaction (WS-AtomicTransaction) August 2005 http://specs.xmlsoap.org/ws/2004/10/wsat/wsat.pdf/

[13] 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