

# Outils pour le passage à l'échelle dans la composition des protocoles métier

Kaouthar FAKHFAKH<sup>1</sup>,  
Hamamache KHEDDOUCI<sup>2</sup> et Hamida SEBA<sup>3</sup>

<sup>1</sup> Unité de recherche ReDCAD, Département d'Informatique et de Mathématiques Appliquées. Ecole Nationale d'Ingénieurs Sfax, Tunisie.

Kaouthar.fakhfakh@gmail.com

<sup>2</sup> Dept. Informatique, IUT A, 71 rue Peter Fink, 01000 Bourg

[Kheddou@bat710.univ-lyon1.fr](mailto:Kheddou@bat710.univ-lyon1.fr)

<http://www.710.univ-lyon1.fr/~q2ap/>

<sup>3</sup> Dept. Informatique, IUT A, 71 rue Peter Fink, 01000 Bourg en Bresse, FranceVerlag

[hseba@bat710.univ-lyon1.fr](mailto:hseba@bat710.univ-lyon1.fr)

<http://www.710.univ-lyon1.fr/~hseba/>

**Abstract.** Dans cet article, nous présentons, une nouvelle solution permettant d'assurer une composition concise des services d'entreprises. Nous utilisons les protocoles métier (business protocol) pour modéliser les échanges entre les différentes entités (fournisseurs, clients, intermédiaires). Nous proposons un algorithme optimisé pour la composition client/serveur ainsi qu'un algorithme permettant la composition de plusieurs services à base de deux compositions : composition série et composition parallèle. Nous présentons aussi un prototype permettant de valider nos algorithmes.

## 1 Introduction

Les services Web sont un paradigme d'actualité qui vise la transposition des architectures par composant dans le cadre du Web. Un service Web est un composant logiciel qui offre des fonctionnalités à travers une interface standardisée. La particularité des services Web est l'utilisation de la technologie Internet comme infrastructure pour la communication. Les services Web se basent sur un modèle à trois couches : un protocole de communication permettant de structurer les messages échangés entre les composants logiciels, une spécification de description des interfaces des services et enfin une spécification de publication et de localisation des services. Actuellement, ce modèle supporte principalement des composants logiciels présentant des services sous forme d'une collection d'unités de traitements (opérations). Le paradigme des services Web est suffisant pour mettre en place des composants interopérables et facilement intégrables. Toutefois, certains composants logiciels nécessitent l'exécution d'une interaction plus complexe et qui obéit à un protocole spécifique en vue de parvenir à la fonctionnalité attendue. La composition des services permet donc de créer des applications et des processus à l'aide de services issus d'environnements hétérogènes, quelques soient les détails et les différences de ces environnements. Grâce à un éven-

tail de services à granularité forte conçus et composés de manière efficace, un expert en processus métier peut composer de manière productive de nouveaux processus métier et de nouvelles applications.

Les approches existantes de composition des services Web proposent des langages tels que (BPEL4WS<sup>1</sup>, WSFL<sup>2</sup>) qui permettent de générer un client pour interagir avec plusieurs services. Cependant, une architecture complète et concise pour la composition des services Web manque dans ce domaine. Notre but est de fournir un ensemble d'outils pour la composition de services. Dans ce travail, nous développons un ensemble d'outils qui offrent à l'utilisateur de nouveaux services complexes et évolués à partir de plusieurs services élémentaires. Contrairement aux approches existantes qui utilisent généralement les graphes ou les réseaux de Pétri, nous nous sommes basés sur les protocoles métiers « Business protocols » qui nous semblent plus adéquats pour la composition.

Nous avons d'abord proposé un algorithme optimisé pour la composition client/serveur. Ensuite, nous avons développé deux algorithmes plus généraux de composition de services.

Le reste de l'article est structuré comme suit : la section 2 présente un état de l'art sur les différentes approches de modélisation pour la composition de services Web. Dans la section 3, nous présentons notre contribution dans le domaine de la composition des services client/serveur. La section 4 présente deux nouveaux algorithmes de composition de services web qui répondent au passage à l'échelle. La section 5 décrit le prototype développé pour valider nos algorithmes. Finalement, nous terminons par quelques remarques concluantes et quelques perspectives.

## 2 Etat de l'art

La composition des services Web [2] est considérée comme une révolution qui permet la distribution sur le Web, non seulement des données et des documents mais aussi des applications. Un service Web est une interface qui décrit une collection d'opérations [8] qui sont accessibles à travers des messages décrits en XML. La composition de services Web ne consiste pas seulement à composer les services appartenants à un seul partenaire mais aussi à chercher les services des autres partenaires. Ce nouveau paradigme se focalise sur la composition des services complexes à partir des services élémentaires se trouvant sur Internet. En effet, La composition peut être statique ; cela veut dire que les services interagissent entre eux d'une manière prédéfinie. Elle peut être aussi dynamique en définissant à la volée l'interaction des services.

Il existe plusieurs formalismes pour la modélisation de la composition de services Web telque les réseaux de Pétri, les contrats et les automates [3], [4], [5]. Chacune de ces approches a ses avantages et ses inconvénients. Dans la modélisation par automate on distingue une approche particulière [1] dite « protocoles métier » (Business protocols). Les « business protocols » constituent une partie importante de la description de

---

<sup>1</sup> Business Process Execution Language for Web Services

<sup>2</sup> Web Services Flow Language

services Web. D'abord, ils assurent la spécification de la conversation [7], [15] supportée par les services. En outre, ils permettent une interaction [9] efficace entre les clients et les services. Enfin, ils garantissent la simplification, le développement, le déploiement, la gestion et l'exécution du service [10]. Les « business protocols » définissent donc le comportement des services. Si deux services ont besoin d'interopérer, leurs protocoles doivent être compatibles, équivalents ou remplaçables [11], [12], [14]. Les protocoles servent à simplifier, développer et à gérer le cycle de vie des services. Pour cela, ils ont besoin de langages, de modèles formels et d'opérateurs de gestion de protocoles.

On modélise un service par un « business protocols »  $P$  comme une machine à état fini tel que  $P=(S, S_0, F, M, R)$  dont:

- $S$ : est l'ensemble des états finis : ils définissent les différentes phases qu'un service peut mettre durant son interaction avec le demandeur et vice versa,
- $S_0$ : est l'état initial,
- $F$ : est l'ensemble des états finaux,
- $M$ : l'ensemble des messages. Chaque message est défini par une polarité : soit positive si  $m$  est un message d'entrée soit négative si  $m$  est un message de sortie suivant la notation  $m (+)$ ,  $m (-)$ ,
- $R$ : est l'ensemble de transitions qui sont déclenchées par les messages envoyés par le demandeur au fournisseur ou vice versa. Ces transitions sont étiquetées par des messages d'entrées ou de sorties. On identifie l'état source, l'état destinataire et le message produit durant cette transition ( $s, s', m$ ).

Le modèle des automates de « business protocols » [1], [6] est construit en utilisant le formalisme de graphe pour représenter les messages chorégraphiques [13]. Un message correspond à l'invocation d'une opération de service ou à sa réponse. Cependant, chaque état identifie un ensemble de transitions sortant et un ensemble de messages qui sont envoyés ou reçus. Chaque transition est étiquetée par un message que ce soit entrant ou sortant suivi par une polarité. Les protocoles sont déterministes, c'est-à-dire qu'ils ont un seul état initial et que chaque état ne fournit pas plus d'une transition sortante étiquetée avec le même nom de message. Le modèle supporte aussi la notion des états finaux qui correspondent à la fin d'une conversation prospère.

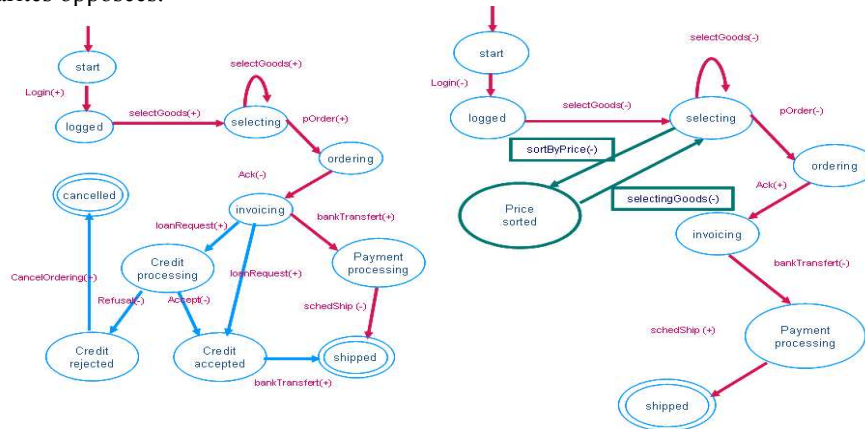
Les modèles basés sur les « business protocols » sont généralement plus riches et reflètent une expressivité qui facilite le processus de la composition.

### **3 Nouvel algorithme optimisé pour la composition client/serveur**

#### **3.1 Motivation**

Benatallah et al. présentent dans [1] un algorithme de composition de « business protocols ». Cet algorithme permet de composer deux protocoles : un protocole fournis-

seur et un protocole client. Le protocole généré, qui est le protocole d'intersection des deux protocoles d'entrées, est construit en combinant les états initiaux, les états intermédiaires et les états finaux communs entre les deux protocoles. Les transitions du protocole composé sont construites à partir des transitions de mêmes labels mais de polarités opposées.



**Fig. 1. Le protocole P1 d'un fournisseur de commerce électronique (à gauche) et le protocole P2 de son client (à droite).**

En étudiant cet algorithme, nous avons mis en évidence les manques suivants :

- Messages internes du client: La Fig. 1 décrit un exemple de conversations de protocoles de services Web dans le cadre d'une application de commerce électronique. L'application de l'algorithme de composition de Benatallah et al. sur ce modèle génère quelques anomalies. En effet, en arrivant à l'état « selecting » du protocole P2 et une fois que le client choisit le message « sortByPrice », il va sélectionner par la suite quelques articles à commander. Ces deux messages internes du client sont très importants. Nous devons bien les conserver dans le protocole de composition car ils ne s'exécutent pas d'une manière automatique, mais sous l'action du client. Cette faiblesse est justifiée par l'absence de messages internes du client dans l'automate de composition. Ces messages de polarités négatives manquent à cause de l'absence des messages de polarités positives dans le protocole du fournisseur correspondant. Ceci cause une incohérence dans le protocole de composition. Nous devons tenir compte donc des messages internes du client qui correspond à des opérations effectuées localement chez le client. Nous allons attribuer à ces messages une autre polarité qui ne peut être ni positive ni négative. Nous leur affectons la polarité « 0 » pour faciliter la procédure de composition.
- Les paramètres dans les automates : Les automates définis dans les travaux de Benatallah et al. peuvent être améliorés en terme de précision. En effet, les paramètres des messages correspondants aux paramètres des entrées et des sorties des opérations ne sont pas modélisés. Ces paramètres peuvent définir de nouveaux critères importants dans la composition de « business protocols ».

- Les messages dans les automates : Deux protocoles peuvent être composés si et seulement s'ils supportent des séquences communes d'échanges de messages. Or selon cette approche deux services supportant des séquences de messages communs mais labellisant de manière différente ne peuvent pas être composables.

### 3.2 Algorithme

Pour pallier les défauts de l'algorithme de Benatallah et al., nous avons développé un nouvel algorithme optimisé de composition d'un protocole client avec un protocole fournisseur illustré sur la Fig. 2. Cet algorithme permet d'avoir une composition plus efficace et plus concise. Il prend en entrée deux protocoles pour générer un protocole composé. Le protocole résultant est défini tel que : les états représentent la fusion des états des deux protocoles d'entrées et les transitions représentent la plus supérieure en terme d'inclusion. Notre algorithme se base sur des nouvelles notions que nous avons défini pour améliorer la composition :

- Paramètres d'opérations dans les « protocoles métier » : Nous avons ajouté au modèle existant les paramètres d'entrée et de sortie des opérations au niveau des transitions pour enrichir le processus de composition. Une transition est composée d'une opération. Une opération est composée d'un message. Un message comporte un ensemble de paramètres. Les paramètres peuvent être des paramètres d'entrées ou de sorties.
- Notion de messages internes : Pour assurer une composition plus précise, nous avons besoin de conserver les messages internes dans un protocole métier. Nous attribuons à ce type de messages la polarité « 0 ». Ces messages sont automatiquement émis puisqu'ils sont généralement des messages internes qui n'exigent pas une interaction avec un protocole externe.
- Inclusion de messages: Nous avons défini la notion d'inclusion de messages pour faciliter la composition de protocoles. Soit un message M1 appartenant au protocole du fournisseur et un message M2 appartenant au protocole du client. Nous disons que  $M1 \subseteq M2$  ssi tous les paramètres de M1 sont inclus dans M2 et la tâche effectuée par l'opération correspondante à M1 est effectuée dans l'opération correspondante à M2.
- Notion de fusion d'états «opérateur  $\nabla$  » : Afin d'avoir une composition plus riche, nous avons défini un nouvel opérateur de fusion d'états noté 'opérateur  $\nabla$ '. Soient deux états S1 et S2 appartenant respectivement aux protocoles P1 et P2. Soient M1 un message qui précède immédiatement S1 et M2 un message qui précède immédiatement S2 selon deux chemins d'exécutions. Si M1 est inclus dans M2 alors le message composé sera M2 et l'état composé sera  $S1 \nabla S2$ .

Pour illustrer la robustesse et l'efficacité de notre algorithme, nous l'avons appliqué sur un exemple de composition de protocole client et du protocole fournisseur.

```

Require : Two protocols  $P_1=(S^1, s_0^1, F^1, M^1, R^1)$  and  $P_2=(S^2, s_0^2, F^2, M^2, R^2)$  .
Ensure : A protocol  $P=P_1 \parallel^{c2} P_2$  .
Let  $R := \emptyset$ ,  $S := \{s_0^1, s_0^2\}$ ,  $s_0 = s_0^1 \nabla s_0^2$ ,  $F := \emptyset$ ,  $M := \emptyset$ ,  $cur := s_0$ ,  $cur1 := s_0^1$ ,
 $cur2 := s_0^2$ 
While  $\exists (s^1 \in S^1 \text{ and } s^2 \in S^2 \text{ and } m^1 \in M^1 \text{ and } m^2 \in M^2) \mid (R^1(cur1, s^1, m^1) \text{ and } R^2(cur2, s^2, m^2))$  do
  if  $(m^1 \sqsubseteq m^2 \text{ and } ((\text{polarity}(P_1, m^1) = '+' \text{ and } \text{polarity}(P_2, m^2) = '-') \parallel (\text{polarity}(P_1, m^1) = '-' \text{ and } \text{polarity}(P_2, m^2) = '+')))$  Then
     $S := S \cup \{s^1 \nabla s^2\}$ 
     $R := R \cup \{cur, s^1 \nabla s^2, m^2\}$ 
     $M := M \cup \{m^2\}$ 
     $cur := s^1 \nabla s^2$ 
  else if  $(\text{polarity}(P_2, m^2) = '0')$  Then
     $S := S \cup \{s^2\}$ 
     $R := R \cup \{(cur, s^2, m^2)\}$ 
     $M := M \cup \{m^2\}$ 
     $cur := s^2$ 
  else return NULL // composition impossible
   $cur1 := s^1$ ,  $cur2 := s^2$ 
end While

  while  $(s^1 \in F^1 \text{ and } s^2 \in F^2)$  do
     $F := F \cup \{s^1 \nabla s^2\}$ 
  end while
Let  $P=(S, s_0^1 \nabla s_0^2, F, M, R)$  with  $\text{polarity}(P, m) = \text{NULL}$ ,  $\forall m \in M$ 
Return P.

```

Fig. 2. Nouvel algorithme optimisé de composition client/serveur.

## 4 Passage à l'échelle dans la composition de services Web

Un des problèmes de base de la composition de services web est le passage à l'échelle. Comment procéder à la composition de plusieurs services ? Pour répondre à ce besoin, nous proposons un algorithme de chaînage de composition à base de deux types de compositions élémentaires : une composition série et une composition parallèle.

### 4.1 Composition série de deux protocoles métier

La composition série consiste à chaîner des services élémentaires entre eux pour construire un service composite qui fournit des fonctions avancées.

Pour modéliser la composition série, nous nous appuyons sur l'exemple de la Fig. 3 qui consiste à rechercher les pages Web contenant le numéro de téléphone d'une personne. Pour cela, nous avons eu recours à l'utilisation du service « annuaire » qui consiste à chercher le numéro de téléphone d'une personne donnée et le service de recherche de page web « Google ».

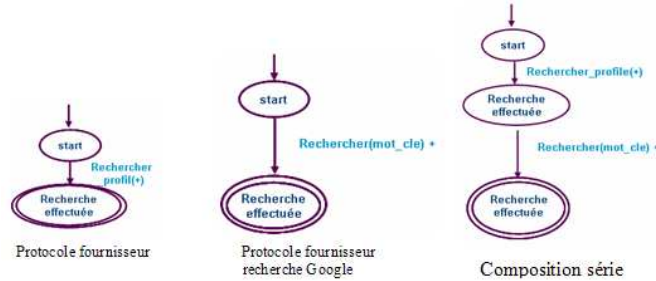


Fig. 3. Composition série des protocoles annuaire et recherche Google

Nous définissons le protocole fournisseur comme une machine à états finis déterministe dont les états décrivent les différentes phases de transformations de services. Pour chaîner un protocole avec un autre, nous avons eu recours à la fusion de l'état final du premier avec l'état initial du deuxième. L'Algorithme détaillé est montré dans la Fig. 4.

```

Require : Two protocols  $P_1 = (S^1, s_0^1, F^1, M^1, R^1)$  and  $P_2 = (S^2, s_0^2, F^2, M^2, R^2)$  .
Ensure : A protocol  $P = P_1 ||^s P_2$ 
Let  $R := \emptyset, S := \{s_0^1\}, s_0 = s_0^1, F := \emptyset, M := \emptyset, cur := s_0^1$ 
While  $\exists (s^1 \in S^1 \text{ and } m \in M^1 | R^1(cur, s^1, m))$  do
     $S := S \cup \{s^1\}$ 
     $R := R \cup \{(cur, s^1, m)\}$ 
     $M := M \cup \{m\}$ 
     $cur := s^1$ 
end While
if  $\exists (s^2 \in S^2 \text{ and } m \in M^2 | R^2(s_0^2, s^2, m))$  Then
     $S := S \cup \{s^2\}$ 
     $R := R \cup \{(cur, s^2, m)\}$ 
     $M := M \cup \{m\}$ 
     $cur := s^2$ 
end if
While  $\exists (s^2 \in S^2 \text{ and } m \in M^2 | R^2(cur, s^2, m))$  do
     $S := S \cup \{s^2\}$ 
     $R := R \cup \{(cur, s^2, m)\}$ 
     $M := M \cup \{m\}$ 
     $cur := s^2$ 
end While
Let  $F := F^2$ 
For all  $m \in M$ 
    if  $m \in M^1$  then
         $Polarity(P, m) = Polarity(P_1, m^1)$ 
    Else if  $m \in M^2$  then
         $Polarity(P, m) = Polarity(P_2, m^2)$ 
    End if
End for
Return  $P = (S, s_0, F, M, R)$ 

```

Fig. 4. Algorithme de composition série de deux "business protocols".

#### 4.2 Composition parallèle de deux protocoles métiers

Pour définir notre algorithme de composition parallèle de protocoles, nous avons utilisé une notion de logique temporelle pour pouvoir exprimer le parallélisme dans les automates. Ce parallélisme consiste à exécuter les transitions des deux services à composer en même temps. L'opérateur temporel utilisé dans nos automates est noté «opérateur  $\diamond$ » ou «opérateur now». Pour illustrer notre proposition, nous présentons

un exemple qui consiste à composer les deux protocoles de la Fig. 6: un service de réservation de vols et un service de location de voitures. Le principe de notre algorithme est d'exécuter en parallèle les services demandés. Ce mécanisme a pour but d'avoir un service composé plus performant et plus riche en fonctionnalité que les deux services initiaux. Notre algorithme utilise un ensemble de nouvelles notions (fusion de messages, synchronisation parallèle...). L'Algorithme détaillé est montré dans la Fig. 5.

```

Require : Two protocols  $P_1=(S^1, s_0^1, F^1, M^1, R^1)$  and  $P_2=(S^2, s_0^2, F^2, M^2, R^2)$  .
Ensure : A protocol  $P=P_1 \parallel P_2$  .
Let  $R := \emptyset, S := \{s_0^1 \nabla s_0^2\}, s_0 = s_0^1 \nabla s_0^2, F := \emptyset, M := \emptyset, cur := s_0, cur1 = s_0^1, cur2 = s_0^2$ 
While  $\exists (s^1 \in S^1 \text{ and } s^2 \in S^2 \text{ and } m^1 \in M^1 \text{ and } m^2 \in M^2) \mid (R^1(cur1, s^1, m) \text{ and } R^2(cur2, s^2, m^2))$  do
  if  $(m^1 = m^2 \text{ and } polarity(P_2, m^1) = '+' \text{ and } polarity(P_1, m^1) = '+')$  Then
     $S := S \cup \{decorticatingMessage, s^1, s^2, synchronization\}$ 
     $R := R \cup \{(cur, decorticatingMessage, m^1 \cup m^2), (decorticatingMessage, s^1, m^1),$ 
     $(decorticatingMessage, s^2, m^2), (s^1, synchronization, synchronize1), (s^2, synchronization, synchronize2)\}$ 
     $M := M \cup \{m^1 \cup m^2, m^1 \cup m^2, synchronize1, synchronize2\}$ 
     $cur := synchronization$ 
  else if  $(m^1 = m^2 \text{ and } polarity(P_2, m^1) = '-' \text{ and } polarity(P_1, m^1) = '-')$  Then
     $S := S \cup \{s^1 \nabla s^2\}$ 
     $R := R \cup \{(cur, s^1 \nabla s^2, m^1 \cup m^2)\}$ 
     $M := M \cup \{m^1 \cup m^2\}$ 
     $cur := s^1 \nabla s^2$ 
  else return NULL // composition impossible
   $cur1 = s^1, cur2 = s^2$ 
end While
Let  $F := \{cur\}$ 
For all  $m \in M$ 
  If  $\exists (m^1 \in M^1 \text{ and } m^2 \in M^2) \mid m = m^1 \cup m^2 \text{ and } polarity(P_1, m^1) = polarity(P_2, m^2)$  Then
     $polarity(P, m) = polarity(P_1, m^1)$ 
  else
     $polarity(P, m) = '0'$ 
  end if
end For all
Return  $P=(S, s_0, F, M, R)$ .

```

**Fig. 5. Algorithme de composition parallèle de deux "business protocols".**

## 5 Implémentation

Nous avons mis en œuvre un prototype pour valider nos algorithmes. Chaque service est modélisé par un protocole métier d'une part et chaque protocole est décrit sous forme d'un fichier XML d'autre part. Notre prototype, qui s'appuie sur les langages java et XML. Il prend en entrée deux fichiers XML décrivant deux protocoles métier de services WEB et leurs descriptions WSDL pour générer en sortie un protocole composite décrit sous forme d'un fichier XML avec son WSDL. Pour valider nos algorithmes, nous avons déployé des services Web dans un container OSGI<sup>3</sup>. Nous avons développé et appliqué nos algorithmes sur ces services. Pour des raisons de limitation en nombre de pages nous ne donnons ici qu'un test de validation de l'algorithme de composition série.

La Fig. 7 montre le résultat de composition série des services « annuaire » et « recherche Google » (illustré dans le protocole composé de la Fig. 3). Pour rechercher le

<sup>3</sup> OSGi: OSGi Service Gateway Specification Release 1.0. Open Service Gateway Initiative. (2000)

numéro de téléphone d'une personne donnée, le service composé commence par demander de saisir le nom et le prénom. Ensuite, il invoque les opérations convenables pour récupérer le numéro du téléphone de la personne en question à partir du service "annuaire". Puis, le service composé utilise le service Web Google pour chercher les pages Web contenant ce numéro de téléphone. Enfin, le serveur finit par afficher les résultats de cette recherche sur l'écran.

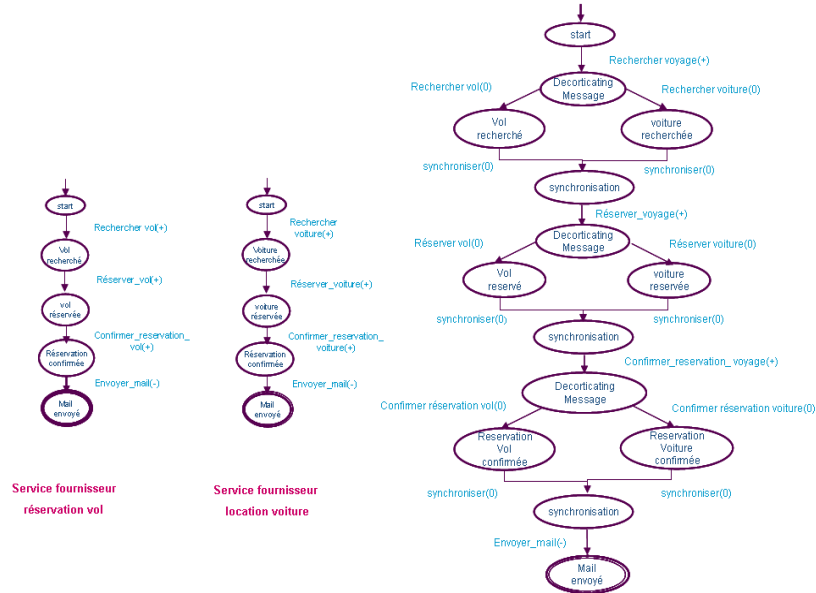


Fig. 6. Le protocole de la composition parallèle de deux services fournisseurs (réservation de vol et location de voiture).

```

----- start computing protocol -----
[DEBUG] looking for rechercherTelephone in wsdl ...
invoking rechercherTelephone ...
Prénom ?
Hamamache
Nom ?
Kheddouci
please Validate by ENTER to invoke the operation!

[DEBUG] connecting to: http://localhost/xmlrpc : annuaire.rechercherTelephone
03 80 39 37 83
[DEBUG] st12
[DEBUG] looking for rechercherPageWeb in wsdl ...
invoking rechercherPageWeb ...
mot ?
03 80 39 37 83
please Validate by ENTER to invoke the operation!

[DEBUG] connecting to: http://localhost/xmlrpc : rechercherGoogle.rechercherPageWeb
Kheddouci Hamamache Tel: 03 80 39 37 83. Fax : 03 80 39 60 04. Email : kheddouc@u-bourgogne.fr.
[DEBUG] sta22
----- end computing protocol -----

```

Fig. 7. Test et validation de l'algorithme de composition série

## 6 Conclusion et perspectives

Dans cet article, nous nous sommes basés sur les « protocoles métier » [1] pour réaliser la composition de services Web. Cette approche nous a semblée plus intéressante que les autres approches [3], [5] de composition de services Web. Elle reflète une expressivité qui facilite le processus de la composition. Notre contribution dans ce domaine a commencé par l'amélioration d'un algorithme de composition de protocoles client/fournisseur. Nous avons développé ensuite deux algorithmes de composition série et de composition parallèle de deux protocoles métiers représentant deux services Web. La combinaison de ces deux algorithmes permet d'obtenir des services évolués et riches en fonctionnalités. Nous avons aussi défini et formalisé un ensemble d'opérateurs nécessaires pour le processus de la composition. Nous avons utilisé des formalismes orientés objet pour enrichir le degré d'expressivité du modèle que nous avons utilisé pour décrire le fonctionnement des services Web. Pour valider ces algorithmes, nous avons représenté les protocoles métiers en XML et nous avons développé un prototype en java. Ce dernier nous a permis de valider les différents opérateurs et algorithmes que nous avons définis. Nous avons ouvert plusieurs perspectives à partir de ce travail. Nous envisageons d'enrichir notre modèle de description des services pour pouvoir partager des données entre les différentes opérations. Le modèle actuel exprime l'ordre d'exécution des opérations dans un service sans préciser les données qu'elles échangent. Nous voulons nous inspirer du travail de [16] pour ajouter les échanges de paramètres entre les opérations au niveau des états des services. En outre, nous comptons développer un algorithme de composition "en alternance" quand les protocoles ne sont pas totalement compatibles ou composables. Cette nouvelle composition englobe les deux types de compositions : la composition série et la composition parallèle. L'élaboration d'un algorithme de vérification de « composabilité » de protocoles métier nous intéresse également pour garantir une composition homogène du point de vue sémantique

## References

1. B. Benatallah, F. Casati, F. Toumani, Analysis and management of Web Service protocols., in: ER conference, China, Vol. 3288 of Lecture Notes in Computer Science, Springer, 2004, pp. 524-541.
2. S. Dustdar\* and Wolfgang Schreiner, A survey on Web services composition, Int. J. Web and Grid Services, Vol. 1, No. 1, 2005.
3. R. Hamadi and B. Benatallah, A Petri Net-based Model for Web Service Composition., in School of Computer Science and Eng., University of New South Wales, Australia ADC 2003: 191-200.
4. D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini and M. Mecella, "Automatic Composition of e-Services", Proc. of the First Int. Conf. on Service-Oriented Comp. (ICSOC), pp. 43-58, 2003.
5. R. Heckel, Towards contract based testing of Web service, in Electronic Notes in Theoretical Computer Science 116 (2005) 145-156.

6. S. Thone, R. Depke and G. Engels, "Process- Oriented, Flexible Composition of Web Services with UML", Workshop on Conceptual Modeling Approaches for e-Business (eCO-MO), 2002.
7. B. Benatallah, F.Casati, F. Toumani, Web Service Conversation Modeling: A Cornerstone for e-Business Automation, *IEEE Internet Computing* 8 (1) (2004) 46-54.
8. G. Alonso, F. Casati, H. Kuno, V. Machiraju, *Web Services Concepts, Architectures and Applications*, Springer Verlag, 2004.
9. B. Medjahed, B. Benatallah, A. Bouguettaya, A. Ngu, A. Elmagarmid, Business-to-Business interactions: issues and enabling technologies, *VLDB J* 12 (1) (2003) 59-85.
10. K.Baina, B.Benatallah, F.Casati, F. Toumani, Model-Driven Web Service Developpement, in; *CaiSE'04*, Vol. 3084 of LNCS, Springer, Riga, Latvia, 2004, pp. 290-306.
11. L. Bordeaux, G. Salaum, D. Berardi, M. Mecella, When are two Web Services Compatible?), in: *VLDB TES'04*. Toronto, Canada, 2004, pp. 15-28.
12. A. H. X. Dong, J. Madhavan, E. Nemes, J. Zhang, Similarity Search for Web services, in *VLDB'04*. Toronto, Canada, 2004, pp. 372-383.
13. A. Wombacher, B. Mahleko, P.Fankhauser, E. Neuhold, Matchmaking, for Business Processes based on Choreographies), in: *EEE'04*, Taipei, Taiwan, 2004, pp. 359-368.
14. B. Benatallah, F. Casati, J. Ponge, F. Toumani, Timed Web services protocols: compatibility and replaceability analysis, Tech. rep., LIMOS Cleremont-Ferrand, (2005).
15. B. Benatallah, F. Casati, F. Toumani, R. Hamadi, Conceptual Modeling of Web Service Conversations, in: *Proc of CaiSE'03*, Vol. 2681 of LNCS, Springer, Klagenfurt, Austria, 2003, pp. 449-467.
16. T. Chaari, F. Laforest, A. Celentano Adaptation in Context-Aware Pervasive Information Systems. *International Journal of Pervasive Computing and Communications*. vol3, 2006