Patrons de Coopération à base de Services pour l'Adaptabilité des Modèles de Workflow

Saida Boukhedouma, Zaia Alimazighi, Mourad Chabane Oussalah, Dalila Tamzalit

Patrons de Coopération à base de Services pour l'Adaptabilité des Modèles de Workflow

Saida BOUKHEDOUMA(*, **) , Zaia ALIMAZIGHI (*), Mourad OUSSALAH (**), Dalila TAMZALIT(**)

(*) USTHB- FEI- Département Informatique- Laboratoire LSI – Equipe ISI
✉: El Alia BP n°32, Bab Ezzouar, Alger, Algérie.
Tél. / Fax : 213.21.24.79.17 {Sboukhedouma, zalimazighi} @usthb.dz

(**) Université de Nantes- Laboratoire LINA- Equipe MODAL
✉: 2, Rue de la Houssinière, BP 92208, 44322 – Nantes, cedex 3- France
Tél. / Fax : 33. 2. 51.12.58.00/ 33.2.51.12.58.12
{Mourad.oussalah, Dalila.tamzalit} @univ-nantes.fr

RESUME. *Les processus métiers sont continuellement soumis à des changements qui doivent être supportés par les modèles de processus et les systèmes qui les implémentent. Cet article s'intéresse à l'adaptabilité des modèles de processus obéissant à des architectures spécifiques de WF inter-organisationnels (WFIO) dont la « sous-traitance » et l' « exécution chainée ». Dans le but d'obtenir des modèles de processus facilement adaptables, nous proposons d'abord des patrons de coopération basés sur les* ***services*** *qui sont des composants faiblement couplés. Pour cela, nous introduisons le concept de* ***fonction d'orchestration*** *pour décrire les modèles de processus à base de services. Sur ces modèles, nous décrivons différentes opérations d'adaptation. Une adaptation de modèle revient à une transformation de la (ou des) fonction(s)d'orchestration. Nous distinguons particulièrement l'adaptation évolutive (ou évolution) des autres types d'adaptation.*

ABSTRACT. *Business processes are continually subject to changes which must be supported by process models and systems that implement them. This paper deals with the adaptability of process models obeying to specific architectures of Inter-Organizational Workflow (IOWF) such as the "subcontracting" and the "chained execution". In order to obtain process models easily adaptable, we first propose cooperation patterns based on* ***services*** *which are loosely coupled components. For that, we introduce the concept of* ***orchestration function*** *to describe our process models on which we describe different operations of adaptation. An adaptation consists of transformation of the orchestration function(s). We distinguish particularly the evolutive adaptation (or evolution) from the other types of adaptations.*

MOTS CLES : *WFIO, Service, Fonction d'orchestration, Adaptation, Evolution, Patron de coopération.*

KEY WORDS *: IOWF, Service, Orchestration function, Adaptation, Evolution, Cooperation pattern.*

1 Introduction

Depuis les années 2000, plusieurs travaux ont été orientés vers l'utilisation combinée de la technologie Workflow (Aalst, 99), du paradigme SOA (Papazoglou et al, 2007) et des services web pour la construction d'applications métiers collaboratives mettant en oeuvre soit une coopération *ad-hoc* (Wombacher et al, 2003) où le modèle de processus n'est pas entièrement défini au niveau « builtime », soit une coopération *structurée* (Spetch et al, 2005) où le modèle de processus est entièrement défini et toutes les instances suivent ce modèle.

Dans nos travaux de recherche, nous nous intéressons à la *coopération structurée* supportée par le concept de workflow inter-organisationnel (WFIO). Dans (Aalst, 99), un ensemble d'*architectures génériques de WFIO* ont été définies; il s'agit du *partage de charge*, *l'exécution chainée*, la *sous-traitance*, le *transfert de cas*, le *transfert de cas étendu* et le WF *faiblement couplé*. Ces architectures constituent la base des modèles de WFIO que nous considérons, elles définissent différentes formes de coopération pouvant lier les partenaires métiers entre eux et supportent ainsi un large éventail de processus métiers inter-organisationnels existants. Cependant, dans leur forme initiale, ces architectures ont été critiquées à cause de leur rigidité et leur manque de flexibilité (Chebbi, 2007) rendant l'adaptation des processus difficile et coûteuse.

Le présent article traite le problème d'*adaptabilité* des modèles de WFIO selon les perspectives contrôle de flux et interactions. L'adaptation d'un modèle de processus peut être justifiée par plusieurs raisons dont l'amélioration du processus, la prise en charge de contraintes nouvelles ou encore la correction des erreurs dans le modèle. Dans (Bastide, 2007), ces adaptations sont respectivement qualifiées de *perfective*, *adaptative* ou *corrective* ; on parlera plus généralement d'*adaptation* de modèles de processus. Une autre raison de l'adaptation est *l'évolution du processus* dite adaptation *évolutive* que nous percevons selon deux axes : l'évolution des *fonctionnalités* et l'évolution de la *coopération*, on parlera plus généralement, d'*évolution* des modèles de WFIO.

Partant du principe que l'adaptabilité d'un modèle dépend fortement des entités qui le composent et des liens entre ces entités, nous proposons des *patrons de coopération* à base de *services* correspondant aux architectures de base de WFIO ; dans cet article nous nous focalisons sur les architectures de « sous-traitance » et « exécution chainée ». Les patrons proposés définissent des modèles de WFIO plus flexibles et facilement adaptables grâce au couplage faible et l'interopérabilité des services. Les architectures de WFIO considérées sont implémentées comme un ensemble de services orchestrés à l'aide d'une ou de plusieurs *fonctions d'orchestration*. Ainsi, dans le cas d'un contrôle centralisé ou hiérarchisé, la structure du processus est décrite à l'aide d'une *fonction d'orchestration globale* implémentée au niveau du site d'un partenaire. Par contre dans le cas d'un contrôle décentralisé, la structure du processus est décrite à l'aide de plusieurs *fonctions d'orchestration locales* implémentées chacune au niveau d'un site de partenaire. Par conséquent, les adaptations de modèles reviennent à des modifications de services et

des fonctions d'orchestration. Dans la suite, la section 2 expose quelques travaux similaires et met en évidence la motivation de ce travail. La section 3 situe le contexte du travail et les principaux concepts qui y sont rattachés. La section 4 présente les patrons de coopération correspondant aux deux architectures considérées. Les sections 5 et 6 décrivent respectivement, les opérations d'adaptation et d'évolution de modèles de WFIO. La section 6 conclut le travail en mettant l'accent sur les travaux futurs.

2 Travaux similaires et motivation

Plusieurs travaux de recherche tels que (Leymann et al, 2002), (Gorton et al, 2009) ont été orientés vers l'utilisation du WF et du paradigme SOA dans les applications métiers collaboratives. Ceci a eu un impact important dans la promotion de la relation B2B entre partenaires métiers. Ainsi plusieurs approches et plateformes ont été proposées pour supporter la coopération utilisant le WF et le paradigme SOA. Dans la coopération *structurée*, nous pouvons citer les approches telles que CoopFlow (Chebbi, 2007), CrossFlow (Grefen et al, 2001), CrossWork (Mehandjiev et al, 2005), Pyros (Belhajjame et al, 2005) et e-Flow (Casati et al, 2001).

Par ailleurs, la flexibilité est une propriété importante devant être satisfaite par les processus métiers et leurs systèmes afin de supporter les changements souvent imposés par les contraintes du marché et les exigences des clients potentiels. Bien que certaines approches telles que CoopFlow, Pyros et e-Flow offrent la possibilité d'adaptation interne des WF sans compromettre la cohérence du processus global, un grand nombre de solutions de WFIO ne sont pas flexibles car elles sont étroitement liées aux plateformes qui les supportent.

Notons que la flexibilité du WF est perçue à deux niveaux complémentaires : (i) au niveau des systèmes, la flexibilité définit la capacité d'un système de gestion de WF (SGWF) à faire face aux situations erronées et aux exceptions (Sadiq et al, 2001), (Meng et al, 2006). (ii) Au niveau des modèles de processus définissant la capacité d'un modèle à être adaptable, évolutif et réutilisable. Pour supporter la flexibilité des modèles, plusieurs travaux de recherche ont été proposés utilisant différentes techniques d'adaptation comme les patrons d'adaptation (He et al, 2008), (Döhring et al, 2011), les patrons d'adaptation à base de règles (Döhring et al, 2010), (Geebelen, 2010) et la modélisation à base de contraintes (Pesic et al, 2007).

Le présent article s'intéresse à l'adaptabilité des modèles de WFIO inspirés d'architectures de base définies dans (Aalst, 99). Afin de pallier à la contrainte de rigidité des modèles et les rendre aisément adaptables, nous définissons un *patron de coopération à base de services* pour chacune des architectures de WFIO considérées dont la « sous-traitance » et l' « exécution chainée ». Pour cela, nous utilisons deux concepts clés : le concept de *service* et le concept de *fonction d'orchestration* qui est une abstraction de la structure du processus en termes de contrôle de flux et d'interactions entre services *internes* et *externes*. Nous décrivons conceptuellement, les opérations d'adaptation des modèles ainsi définis et nous terminons par mettre en

évidence la possibilité de combiner les deux patrons afin de définir des WFIO étendus modélisant des processus plus complexes. L'utilisation des services permet de gagner en flexibilité car le paradigme SOA supporte la composition, l'intégration et la réutilisation de services.

3 Définitions et Principaux Concepts

3.1 Définition et Dimensions du WFIO

Le WFIO peut être défini comme un gestionnaire d'activités faisant intervenir deux ou plusieurs WF autonomes, interopérables et éventuellement hétérogènes dans le but d'atteindre un objectif métier commun (Aalst, 99).

Dans (Aalst, 99), un ensemble d'architectures de base de WFIO ont été définies selon deux dimensions principales : le *partitionnement du processus* et le *contrôle d'exécution*. Le partitionnement du processus définit la manière dont les fragments de processus WFIO sont répartis sur les sites des partenaires et la localisation des instances de processus sur les différents sites au moment de l'exécution. La dimension *contrôle d'exécution* définit la manière dont les instances sont contrôlées par les systèmes des partenaires au niveau « runtime » ; le contrôle est centralisé si l'exécution des instances est gérée par un seul système, le contrôle est décentralisé si l'exécution est contrôlée au niveau des sites de partenaires, chacun pour le fragment de WF qu'il implémente. Le contrôle est hiérarchisé si un partenaire principal gère l'exécution du processus global et contrôle le déclenchement des fragments implémentés au niveau des partenaires secondaires.

3.2 Méta-modèle de WFIO, Adaptabilité et Evolutivité

Sur la base des définitions énoncées plus haut, nous établissons le méta-modèle de processus WFIO décrit dans la figure 1. Un WFIO est défini par un ensemble de fragments (WF) de processus et un *patron de coopération*. Chaque WF est rattaché à un partenaire métier, manipule des *données* et est soumis à une *condition d'invocation*. Un WF interagit avec un autre WF à travers des points d'interaction

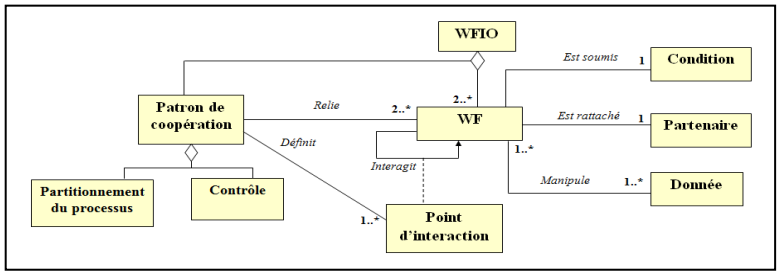

Figure 1. *Méta-modèle générique de WFIO*

définis conformément au patron de coopération. Ce dernier décrit une architecture spécifique de WFIO et est lui-même défini selon les deux dimensions principales du WFIO : le *partitionnement du processus* et le *contrôle d'exécution*.

A travers les concepts illustrés sur le méta-modèle, on peut dire qu'un modèle de processus WFIO couvre quatre axes principaux: ***processus*** (à travers les concepts de WFIO, WF, patron de coopération et condition), ***organisation*** (le concept de partenaire*),* ***données*** (le concept de donnée) et ***interaction*** (les concepts de patron de coopération et point d'interaction). Par conséquent, les contraintes d'adaptation ne s'appliquent pas à un seul axe mais couvrent les quatre axes de définition de WFIO.

L'*adaptabilité* d'un modèle de WFIO est sa capacité à supporter facilement les changements sur une ou plusieurs des entités qui le composent (WF, condition, donnée et point d'interaction) sans impact sur la cohérence du processus, la fonctionnalité globale et la coopération (les partenaires impliqués).

L'*évolutivité* (ou adaptabilité évolutive) d'un modèle de processus définit sa capacité à étendre sa fonctionnalité globale et/ou sa coopération, c'est-à-dire le nombre de partenaires et les fragments de WF, sans perturber le fonctionnement du processus initialement implémenté.

Dans cet article, nous nous intéressons à l'adaptabilité des modèles aux niveaux *processus* (notamment le contrôle de flux) et *interaction*.

4 Patrons de coopération à base de services

Dans cette section, nous décrivons les patrons de coopération à base de *services* correspondant aux deux architectures de WFIO considérées. Globalement, la question à laquelle il faut répondre pour définir ces patrons est : quelles sont les parties de WF à encapsuler dans des services afin de pouvoir les invoquer de l'extérieur? Lors du découpage en services, on essayera de préserver les points d'interaction entre les fragments de processus des différents partenaires. Selon les architectures, il s'agit *d'encapsuler un processus WF ou un sous-processus dans un service* (Boukhedouma et al, 2011). Un sous-processus est une partie d'un processus ayant des entrées et des sorties et remplissant une fonctionnalité partielle.

4.1 Le patron « Sous-traitance »

L'architecture de « sous-traitance » suppose l'existence d'un partenaire principal implémentant localement, son propre modèle de WF et son propre SGWF devant sous-traiter certaines activités auprès d'autres partenaires (secondaires). Ces derniers hébergent localement leurs WF et leurs SGWF respectifs.

Ainsi pour le patron de « sous-traitance », nous proposons *d'encapsuler entièrement* chaque WF secondaire impliqué dans la coopération dans un service. Sur la figure 2, le partenaire 1 implémente le WF principal et le partenaire 2 fournit son WF secondaire comme un service *S2* global, *S2* peut être un service composite

mais du point de vue du partenaire principal, il est abstrait dans une entité élémentaire. Donc, le partenaire 1 invoque le service *S2* fourni par le partenaire 2.

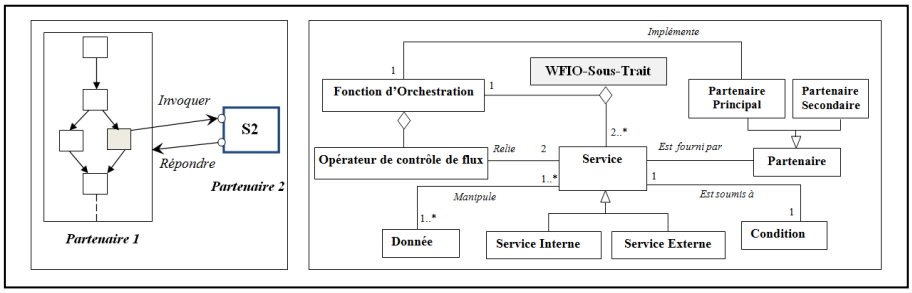

Figure 2. *Schéma et méta-modèle du patron « Sous-traitance »*

Afin d'obtenir un WFIO basé complètement sur les services, le WF du partenaire principal peut-être lui-même implémenté sous forme d'un ensemble de *services internes* et l'invocation de *services externes*. Pour les activités nécessitant l'intervention humaine, il existe des mécanismes pour les abstraire en services comme le service « TaskManager » fourni par Oracle-BPEL. L'ensemble des services est *orchestré* au niveau du partenaire principal à l'aide d'une fonction d'orchestration *globale*. Dans cette architecture, le contrôle d'exécution est *hiérarchisé* car le WF principal gère l'exécution du WFIO global et contrôle l'invocation du (ou des) service(s) externe(s). Sur le méta-modèle de la figure 2, un modèle de processus obéissant au patron de « sous-traitance » peut être défini par la un ensemble de services internes et externes rattachés à différents partenaires et une fonction d'orchestration qui décrit le contrôle de flux global.

4.2 Le patron « Exécution chainée »

Dans l'architecture *« exécution chainée »*, le WFIO global est partitionné en fragments disjoints s'exécutant en *séquence*, chaque fragment est implémenté localement au niveau du partenaire concerné et est géré par un SGWF local.

Pour le patron « exécution chainée » à base de services, nous proposons d'*encapsuler entièrement* le WF de chaque partenaire dans un service ; sur la figure 3, *S1* pour partenaire 1 et *S2* pour partenaire 2, l'instance de processus est exécutée selon la succession de services (*S1*, *S2*). D'une manière générale, le premier service (*S1*) de la séquence est déclenché par l'arrivée d'une nouvelle instance, les autres services de la séquence sont invoqués chacun, par le service précédent. En d'autres termes, un service *Si+1* est invoqué par le service *Si* qui le précède dès que ce dernier termine son exécution. Naturellement, cette architecture est implémentée comme une chorégraphie de services avec un contrôle *décentralisé*. Le patron *« exécution chainée »* est décrit par le méta-modèle de la figure 3.

Notons qu'au niveau interne, les services *S1* et *S2* sont implémentés comme des services composites. En effet, un service interne ***Sij*** peut encapsuler un sous-processus (ou activité) du partenaire ***i.*** Ainsi, un service ***Si*** fourni par le partenaire ***i*** est implémenté comme un ensemble de services orchestrés localement à l'aide d'une

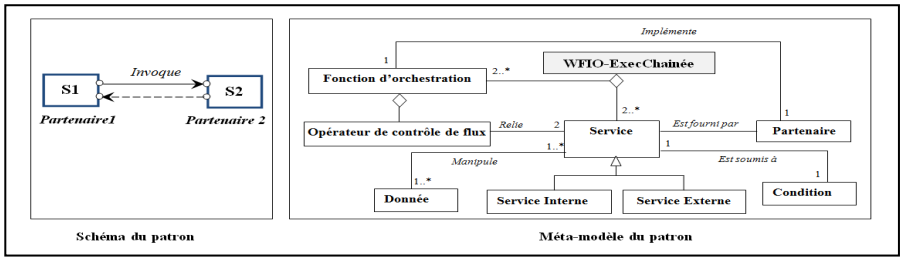

Figure 3. *Schéma et méta-modèle du patron « Exécution chainée »*

fonction d'orchestration locale. Afin de découpler les services métiers des interactions entre services, deux services particuliers ***Sini*** *et* ***Souti*** assurent les interactions avec les services des autres partenaires *: Sini* pour la réception des données et *Souti* correspondant à l'invocation du service suivant (*Si+1*) de la séquence ou la sortie de résultats. Chaque orchestrateur local ***i*** reçoit les données d'entrée à travers le service *Sini*, invoque son service métier local *Si* (composée des services *Sij*) et invoque ensuite le service *Si+1* avec les résultats d'exécution du service *Si*. Particulièrement, le service *Sin* du premier service de la séquence reçoit les données d'une nouvelle instance et le service *Sout* du dernier service de la séquence renvoie les résultats d'exécution complète de l'instance.

4.3 Fonction d'orchestration

Sur la figure 4, nous introduisons les opérateurs de base de contrôle de flux utilisés pour décrire la *fonction d'orchestration*. Ces opérateurs sont exprimés à l'aide d'une notation générale indépendamment de tout langage de spécification ou plateforme.

Remarque. Pour le choix-multiple (resp. le multi-parallèle), nous décomposons en plusieurs branches de choix simple (resp. parallélisme simple). Par exemple, *Alt* (S1, S2, S3) est exprimé par *Alt* (*Alt* (S1, S2), S3) ou *Alt* (S1, *Alt* (S2, S3)).

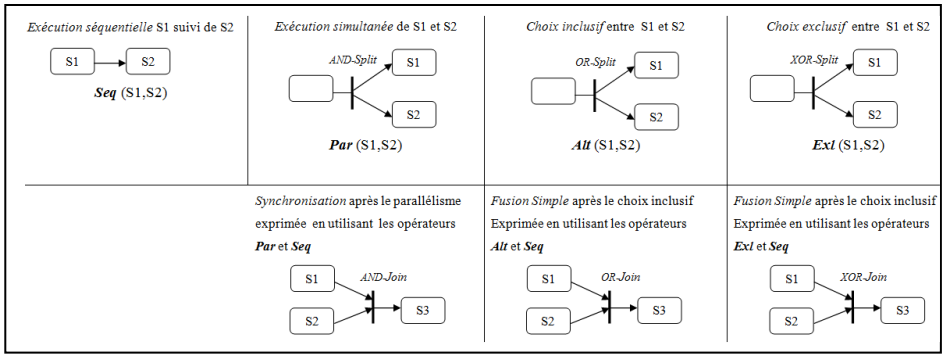

Figure 4. *Opérateurs de base de contrôle de flux*

La partie supérieure de la figure 5 illustre le concept de *fonction d'orchestration* sur un exemple de WFIO obéissant au patron « sous-traitance ». Le WFIO implique deux partenaires métiers, partenaire 1 implémente le WF principal composé de

services internes *S11, S12, S13, S14* et l'invocation d'un service *S2* externe fourni par le partenaire 2. La fonction d'orchestration peut être représentée par l'arbre binaire à droite avec deux types de nœuds : les *services* et les *opérateurs*. Le parcours *préfixé* de l'arbre permet de générer la structure du processus.

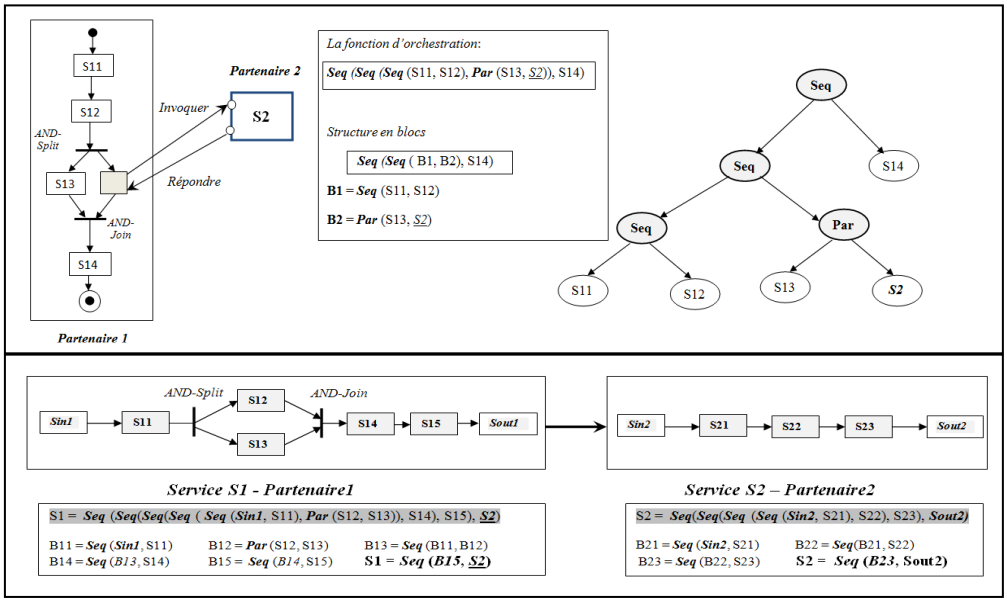

Figure 5. *Illustration du concept de fonction d'orchestration*

Dans la partie inférieure de la figure 5, nous montrons un autre exemple de WFIO obéissant au patron *« exécution chainée »*. Le WFIO implique aussi deux partenaires, le partenaire 1 implémente un WF composé de services internes *S11, S12, S13, S14* et partenaire 2 implémente un WF composé de services internes *S21, S22, S23*. Les services *Sin1, Sin2, Sout1* (sert à invoquer *S2* à partir de *S1*) et *Sout2* assurent les interactions entre les services des deux partenaires et l'environnement.

Pour une meilleure lisibilité, le schéma du processus peut être structuré en blocs pouvant être exprimés par composition d'autres blocs.

4.4 Définition formelle de WFIO

Formellement, un WFIO est défini par une paire <S, F> où S est un ensemble de services *Si* (ou *Sij* au niveau interne) et F est un ensemble de fonctions d'orchestration *f*, où ***f*** (Si1, Si2, … , Sin) = *Si1* ***op1*** *Si2*… ***opn-1*** *Sin* = ***Si***
et op1, … opn-1 sont des opérateurs de contrôle de flux.

Selon le patron de coopération implémenté, une fonction d'orchestration ***f*** associe à un ensemble de services ***Sij*** de S, un service composite ***Si*** correspondant au WFIO global ou un fragment du WFIO implémenté au niveau d'un partenaire. Les interactions internes (resp. externes) sont définies par les interactions entre les services d'un même partenaire (resp. entre les services de partenaires distincts).

5 Adaptabilité des modèles de WFIO

Conformément à la définition précédente, adapter un modèle de processus revient à apporter des modifications aux deux entités principales constituant le WFIO à savoir : les *services* et la (ou les) *fonction(s) d'orchestration*. Généralement, une modification de service peut être un ajout, une suppression, une substitution, une fusion de deux services ou la décomposition d'un service. Notons que l'adaptation de l'entité *service* induit souvent une modification de la *fonction d'orchestration* qui l'utilise ou des attributs tels que *condition* ou *donnée* rattachés au service. Par ailleurs, d'autres opérations d'adaptation affectent le contrôle de flux (donc la fonction d'orchestration) sans altérer les services impliqués dans le processus.

5.1 Ajout, Suppression, Substitution de services

Pour les opérations d'ajout ou de suppression de services, nous distinguons l'ajout ou la suppression de service dans un bloc séquentiel (i.e une seule branche) de l'ajout ou la suppression de service dans un bloc parallèle ou alternatif (composé de deux branches). La partie supérieure de la figure 6 décrit les opérations de base d'*ajout de services*, celles-ci sont illustrées à l'aide de schémas génériques de parties de processus et les fonctions d'orchestration correspondantes.

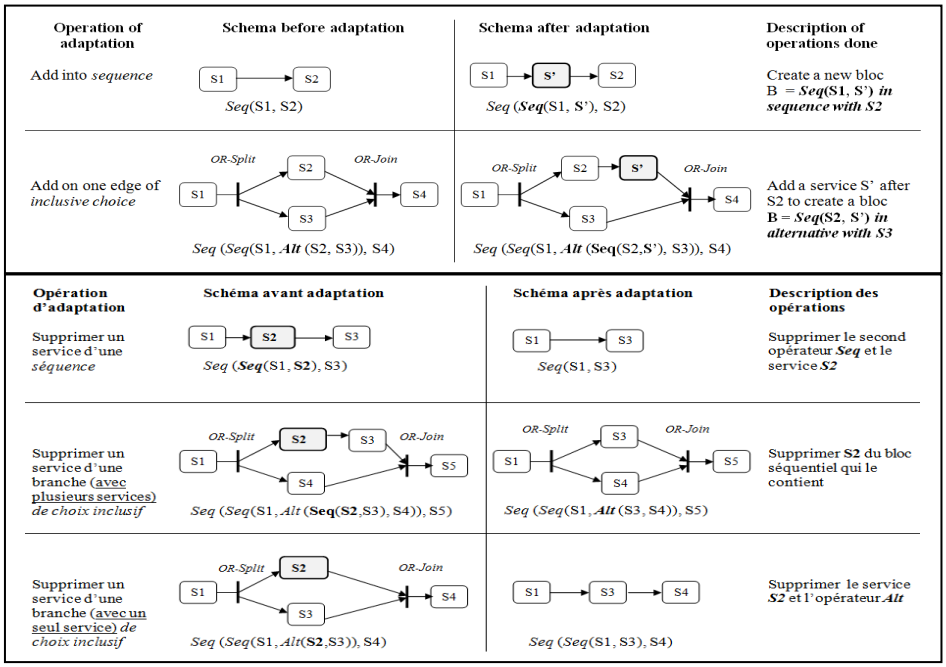

Figure 6. *Ajout et suppression de services*

La *suppression* est l'opération inverse de l'ajout, la partie inférieure de la figure 6 illustre les opérations de base de suppression de services (S2 par exemple). Pour

les blocs non séquentiels, nous décrivons la suppression dans un bloc alternatif (choix inclusif) ; le même scénario est appliqué dans le cas d'un bloc parallèle ou de choix exclusif. Notons que deux configurations sont envisageables dans le cas d'une suppression dans un bloc avec deux branches : (i) le service à supprimer fait partie d'une séquence de services, (ii) le service à supprimer est seule sur la branche.

Une autre opération de base de l'adaptation de modèles concerne la *substitution* de services. Il s'agit de la suppression d'un service suivie de l'ajout d'un nouveau service.

5.2 Fusion et décomposition de services

La *fusion de services* peut concerner deux services reliés par un opérateur de séquence, de choix inclusif, de choix exclusif ou de parallélisme. Une fusion de services peut être appliquée pour simplifier le modèle de processus par abstraction d'un ensemble de services dans un seul.

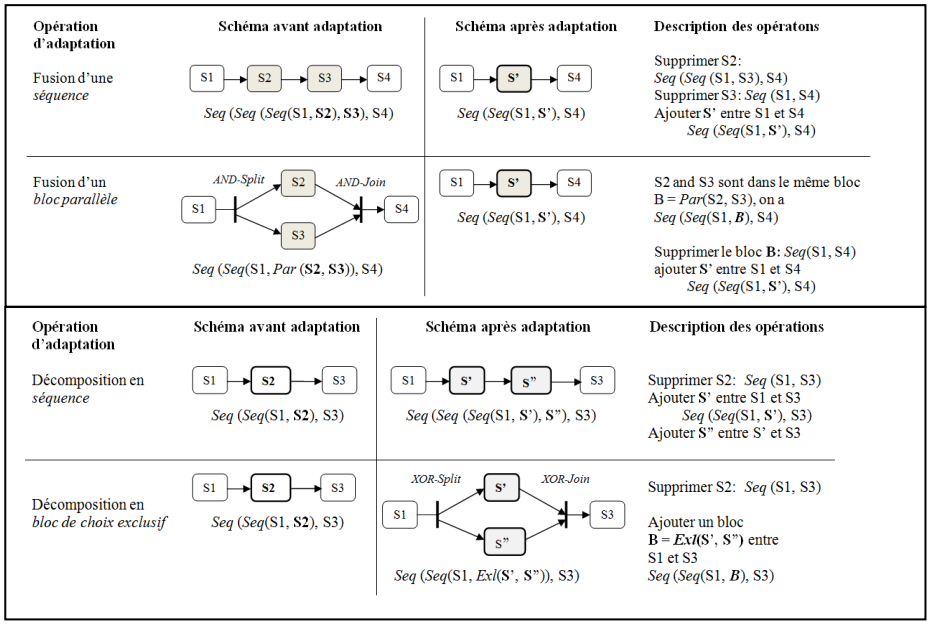

Figure 7. *Fusion et décomposition de services*

La partie supérieure de la figure 7 montre ces opérations de fusion de base, l'ensemble des transformations effectuées sur la fonction d'orchestration initiale pour obtenir la fonction d'orchestration décrivant le modèle après adaptation. Sur la figure 7, il s'agit de fusionner les services *S1* et *S2* en un service *S'*. On peut remarquer que lorsque les services à fusionner sont dans le même bloc, l'opération de fusion consiste à supprimer un bloc et le remplacer par un service, en effet le bloc *Alt (S2, S3), Par (S2,S3)* ou *Exl (S2, S3)* est considéré comme une *seule entité.*

Des opérations de fusion plus élaborées concernent des configurations où les services à fusionner ne sont pas dans le même bloc. Par exemple, dans un modèle décrit par la fonction d'orchestration *Seq(Seq(**S1**, Par(**S2**,S3)), S4),* la fusion des services *S1* et *S2* ne peut pas se faire directement car elle peut donner deux résultats différents selon que le parallélisme soit maintenu ou non; ceci peut être fourni comme un paramètre additionnel afin d'effectuer correctement l'opération de fusion. Dans les deux cas, la transformation consiste à supprimer puis ajouter un service.

L'opération de *décomposition* est l'inverse de la fusion, il s'agit de décomposer un service pour obtenir un bloc composé de deux services séquentiels, alternatifs ou parallèles (voir la partie inférieure de la figure 7). Une décomposition peut être faite pour avoir un meilleur contrôle sur l'exécution du processus (bloc séquentiel), ou pour améliorer le temps de réponse du processus (bloc parallèle) ou encore pour prendre en charge de nouvelles contraintes (bloc alternatif).

5.3 Adaptation du contrôle de flux

Une autre catégorie d'opérations d'adaptation concerne la modification de la fonction d'orchestration sans altérer les services impliqués dans le processus. Il s'agit particulièrement de remplacer un opérateur de contrôle de flux par un autre; on pourrait par exemple remplacer un opérateur de parallélisme (***Par***) par un opérateur de séquence (***Seq***) si les services deviennent dépendants l'un de l'autre, ou alors remplacer un opérateur de séquence par un opérateur de parallélisme dans le cas contraire pour augmenter le degré de parallélisme dans le processus.

Lorsque les services à restructurer sont dans le même bloc, l'adaptation peut facilement être effectuée par substitution d'opérateurs ; par exemple dans la fonction *Seq (**Seq (S1**,**S2**), S3),* si on veut mettre les services *S1* et *S2* en parallèle, il suffit de remplacer l'opérateur ***Seq*** par l'opérateur ***Par*** afin d'obtenir la fonction transformée *Seq (**Par(S1**,**S2**), S3).* Si au contraire, les services à restructurer ne sont pas dans le même bloc, l'opération d'adaptation est moins évidente ; par exemple, dans la fonction d'orchestration *Seq(Seq (Seq (S1,**S2**), **S3**),S4),* si l'on veut relier *S2* et *S3* par un opérateur de parallélisme, ceci ne peut pas se faire par une substitution directe de l'opérateur *Seq*. En effet, on doit supprimer *S2* pour obtenir *Seq (Seq (S1, **S3**), S4),* ensuite supprimer *S3* pour obtenir *seq (S1, S4),* finalement insérer le bloc ***Par (S2,S3)*** entre S1 et S4 pour obtenir la fonction *Seq (Seq (S1, **Par (S2**, **S3**)), S4).*

6 Evolutivité des modèles de WFIO

L'évolutivité d'un modèle de WFIO est reflétée sur deux perspectives : *fonctionnalité* et *coopération*. Ainsi, un WFIO évolue s'il peut être étendu à des fonctionnalités additionnelles ou s'il permet l'expansion du nombre de partenaires et des services externes fournis. Notons que les deux perspectives ne sont pas exclusives, l'expansion de la coopération peut induire une expansion de fonctionnalité et vice versa.

6.1 Expansion des fonctionnalités

L'évolution de modèles selon la perspective *fonctionnalité* peut se faire soit par l'ajout de services (resp. blocs) internes avec des fonctionnalités complémentaires, soit par l'ajout de services externes fournis par de nouveaux partenaires (cas d'expansion de la coopération), ou encore par substitution de services (resp. blocs) par d'autres remplissant des fonctionnalités additionnelles. Pour effectuer de telles opérations sur les modèles de processus, on peut se référer aux opérations explicitées dans la section 5.1. On dit que le processus évolue en fonctionnalité.

6.2 Expansion de la coopération

L'évolution de modèles selon la perspective de *coopération* se fait à travers l'ouverture du WFIO à de nouveaux partenaires. L'ajout de partenaires et donc de services externes peut préserver le patron de coopération du WFIO ou donner lieu à la combinaison de deux patrons de coopération ou plus. Nous allons illustrer ces deux configurations sur les deux patrons que nous avons considérés dans ce papier à savoir : le patron de *sous-traitance* ou le patron *exécution chainée*.

6.2.1 Expansion de la « sous-traitance »

L'expansion de la sous-traitance peut avoir lieu selon l'une des trois configurations suivantes décrites dans la figure 8 (*OP-Split* est soit OR, XOR ou AND-Split):

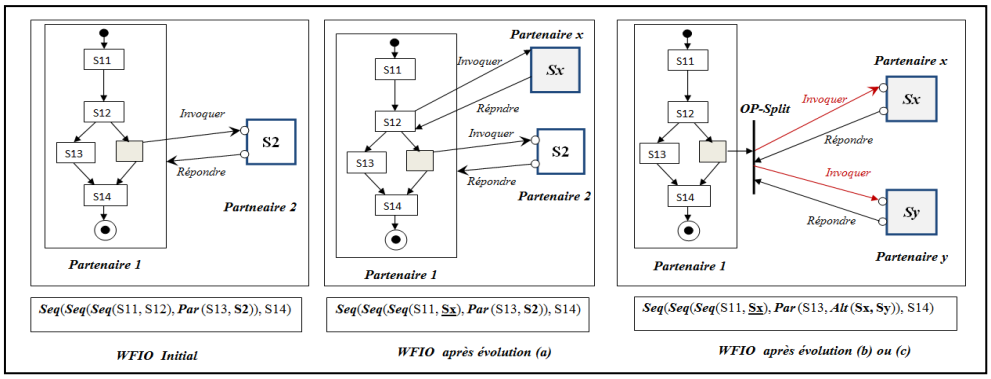

Figure 8. *Expansion de la coopération "sous-traitance"*

a) Externalisation de nouveaux services qui consiste à remplacer un service interne par un service externe si on a besoin d'étendre la sous-traitance.
b) Remplacer un service externe par un bloc alternatif composé de deux services *Sx* et *Sy* fournis par deux partenaires tels que le service *Sx* est invoqué dans certains cas (selon une condition) et dans d'autres cas c'est le service *Sy* qui est invoqué, afin de répondre à de nouvelles contraintes.

c) Remplacer un service externe par un bloc parallèle composé de deux services *Sx* et *Sy* fournis par deux partenaires et devant être exécutés simultanément dans le but d'améliorer le temps de réponse du processus.

Les scénarios d'évolution sus-décrits peuvent être exprimés à travers des opérations de substitution et de décomposition déjà présentées dans la section 5. La seule différence est que ces opérations concernent des services externes.

- ***Sous-traitance multi-niveaux****:* Une autre configuration d'expansion de la *sous-traitance* est envisagée lorsqu'un partenaire *secondaire* opère à son tour une évolution de son modèle de processus pour sous-traiter une partie de son WF à un autre partenaire secondaire, ceci implique de nouveaux partenaires et donc de services externes et une expansion de la hiérarchie dans la sous-traitance, on parle de sous-traitance multi-niveaux. Dans ce cas, l'opération d'évolution concerne le WF secondaire et consiste à substituer un service interne par un service externe.

6.2.2 Expansion de l' « Exécution chainée »

L'exécution chainée peut être étendue dans l'un des cas suivants:

a) Ajouter à la séquence de services, un nouveau service externe (encapsulant un WF) fourni par un nouveau partenaire dans le but d'une expansion des fonctionnalités ou l'ajout d'une phase intermédiaire au processus.

b) Remplacer un service de la séquence par un bloc exclusif composé de deux nouveaux services fournis par deux partenaires. Dans tous les cas, les instances suivent exclusivement l'une des deux branches et on se trouve toujours dans un WFIO obéissant au patron « exécution chainée ».

Partant d'un WFIO composé initialement d'une séquence de trois services *Sx*, *Sy* et *Sz*, la figure 9 illustre les deux évolutions sus-décrites. On suppose qu'un service *Sp* du partenaire *p* est composé d'une séquence *Sinp*, *Spp* (le service métier composite) et l'invocation du service suivant ou *Soutp* pour le dernier de la séquence.

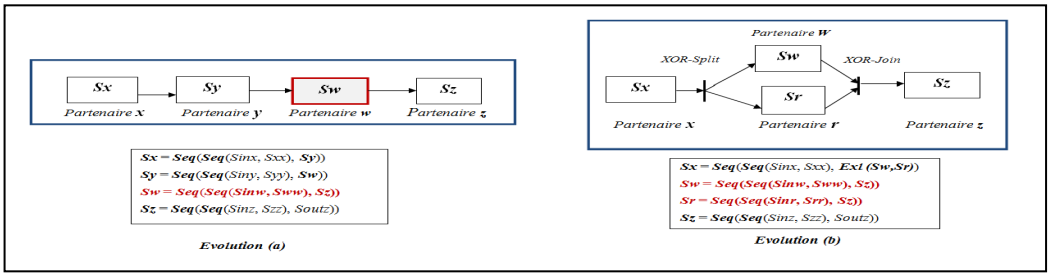

Figure 9. *Expansion de la coopération "Exécution chainée"*

6.2.3 Evolution par combinaison de patrons de coopération

Le fait de décrire le WFIO comme un service composite à l'aide du concept de fonction d'orchestration permet de combiner deux WFIO obéissant au même patron de coopération ou à des patrons de coopération différents. En effet, le WFIO est lui-même manipulé comme un service composite pouvant être ajouté, supprimé, ou substitué en respectant les contraintes de chaque patron de coopération. Dans ce qui suit, nous schématisons deux modèles de WFIO obtenus par combinaison des deux patrons « *sous-traitance* » et « *exécution chainée* ».

- **Une exécution chainée dans une sous-traitance**

Un partenaire principal peut sous-traiter une partie de son WF auprès d'un groupe de partenaires alliés dans un WFIO de type « *exécution chainée* » par exemple. Dans ce cas, le patron prédominant est la *sous-traitance*, le WFIO global est contrôlé par le partenaire principal qui invoque le WFIO du groupe de partenaires secondaires, ce dernier une fois invoqué est contrôlé de manière décentralisée par le groupe de partenaires, ceci nécessite un retour de résultats dans le sens inverse de la séquence afin de communiquer les résultats de la sous-traitance au partenaire principal. On tombe sur un WFIO avec un contrôle mixte *hiérarchisé/décentralisé*. Sur la figure 10, le partenaire 1 sous-traite une partie de son WF auprès du groupe de partenaires *x*, *y*, *z* fournissant un WFIO formé de la séquence de services *(Sx, Sy, Sz)*. Le partenaire 1 invoque le premier service (*Sx*) de la séquence.

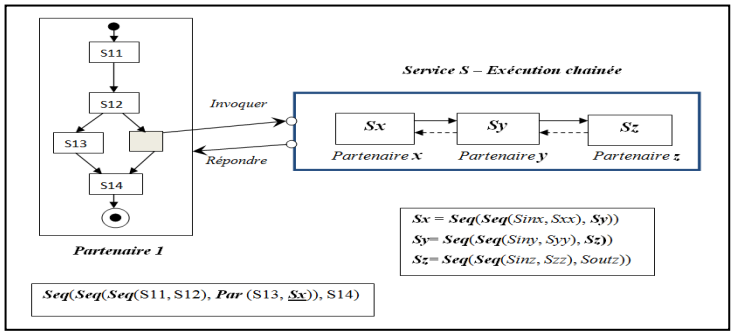

Figure 10. *Combinaison de patrons « Une exécution chainée dans une sous-traitance»*

- **Une sous-traitance dans une exécution chainée**

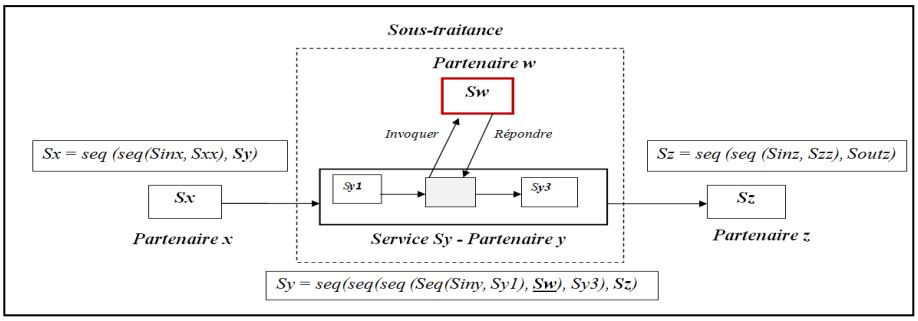

Figure 11. *Combinaison de patrons « Une sous-traitance dans une exécution chainée »*

Un partenaire impliqué dans un WFIO de type *exécution chainée* peut sous-traiter une partie de son WF implémenté localement à un nouveau partenaire externe. Dans ce cas, le patron prédominant est l'*exécution chainée* et sur un fragment de la séquence, on a une sous-traitance. Le contrôle est mixte *décentralisé/hiérarchisé*. Il est hiérarchisé sur le fragment de WFIO concerné par la sous-traitance. Sur la figure 11, partenaire *y* sous-traite une partie de son WF au partenaire *w*, une modification est apportée à la fonction d'orchestration du service *y* pour invoquer *Sw*.

7 Conclusion et Travaux Futurs

Dans cet article, nous nous sommes intéressés à l'*adaptabilité* des modèles de WFIO dans le cadre d'une coopération structurée. La première question sur laquelle nous nous sommes penchés est la définition de nouveaux modèles architecturaux appelés *patrons de coopération* correspondant aux architectures de WFIO définies dans (Aalst, 99), afin de pallier à la contrainte de rigidité et de difficulté d'adaptation des modèles. Les patrons de coopération que nous proposons reposent sur les concepts de *service* et de *fonction d'orchestration* et obéissent à une approche SOA. Dans cet article, nous nous sommes focalisés sur deux architectures de base que sont la « sous-traitance » et l' « exécution chainée » pour lesquelles nous avons défini les deux patrons de coopération. Sur ces patrons, nous avons défini les opérations d'adaptation de modèles pouvant être une adaptation évolutive (ou évolution). L'évolution de modèles peut être perçue sous deux perspectives principales : l'expansion des fonctionnalités du processus ou l'expansion de la coopération. L'expansion de la coopération peut préserver le patron de coopération initial ou induire une combinaison de deux patrons de coopération distincts, nous avons illustré ceci à travers des configurations combinant les deux patrons considérés.

Actuellement, nous travaillons sur l'implémentation des patrons de coopération que nous avons définis pour les différentes architectures et sur l'implémentation des opérations d'adaptation et d'évolution de modèles (sous forme de patrons d'adaptation) en considérant un langage de spécification approprié tels que BPEL ou jPDL. Nous projetons dans un second temps, de nous pencher sur la composition de WFIO afin de définir des processus de plus en plus complexes par réutilisation.

Bibliographie

(Aalst, 99): Aalst W.V.D., "Process oriented architectures for electronic commerce and interogranizational workflow"**,** *Journal of Information systems, volume 24 issue 9, 1999.*

(Bastide, 2007): Bastide G., « SCORPIO – Une approche pour l'adaptation structurelle des composants logiciels: Application aux environnements ubiquitaires ». *Thèse de doctorat, université de Nantes, 2007.*

(Belhajjame et al, 2005): Belhajjame K., Vargas-Solar G., and Collet C. "Pyros - an environment for building and orchestrating open services". *In Proceedings of the 2005 IEEE International Conference on Services Computing, pages 155–164, Washington, DC, USA, 2005. IEEE Computer Society.*

(Boukhedouma et al, 2011) : Boukhedouma S., Alimazighi Z., Oussalah M., Tamzalit D., « Une approche basée SOA pour l'interconnexion de WF : Application au transfert de cas ». Proceedings *INFOSID'2011, pp. 43-56*, Lille, France. 2011.

(Casati et al, 2001): Casati F. and Shan M., "Dynamic and adaptive composition of e-services". *Information Systems*, 26(3) :143–163, 2001.

(Chebbi, 2007) Chebbi I., «CoopFlow : une approche pour la cooperation ascendante des workflows dans les enterprises virtuelles ». *Thèse de doctorat, Institut national de Telecom*, France, 2007.

(Döhring et al, 2010): Döhring M., Zimmermann B., Godehardt E., "Extended workflow flexibility using rule-based adapatation patterns with eventing semantics". *In proc. of INFORMATIK'10*, pp. 216,226- 2010.

(Döhring et al, 2011): Döhring M., Zimmermann B., Karg L., "Flexible Workows at design- and Runtime using BPMN2 Adaptation Patterns". *In proceedings of BIS'2011- Springer*, 2011.

(Geebelen, 2010): Geebelen K., "A MVC Framework for policy-based adaptation of workflow processes: a case study on confidentiality". *In IEEE International Conference on Web Services*, 2010.

(Gorton et al, 2009): Gorton S., Montangero C., Reiff-Marganiec S., Semini L., "StPowla: SOA, Policies and Workflows**"**, *ICSOC 2007 workshops, LNCS 4907, pp. 351-362*, 2009.

(Grefen et al, 2001): Grefen P., Aberer K., Hoffer Y., and Ludwig H. "Crossflow : Cross-organizational workflow management for service outsourcing in dynamic virtual enterprises". *IEEE Data Engineering Bulletin*, 24(1) :52–57, 2001.

(He et al, 2008): He Q., Yan Y., Jin H., "Adaptation of web service composition based on WF patterns". *In proceedings of Service Oriented Computing*, ICSOC, 2008.

(Leymann et al, 2002): Leymann, F., D. Roller, and M.-T. Schmidt, "Web Services and Business Process Management", *IBM Systems, Journa*l, Vol. 41, No. 2, 2002.

(Mehandjiev et al, 2005): Mehandjiev N., Stalker I., Fessl K., and Weichhart G.,"Interoperability contributions of crosswork". *In* Proceedings *of INTEROP-ESA'05 Conference, Geneva, February 2005. Springer-Verlag.*

(Meng et al, 2006): Meng J., Su S.Y.W., Lam H., Helal A., Xian J., Liu X. and Yang S. "DynaFlow: a dynamic inter-organizational workflow management system", *Int. J. Business Process Integration and Management*, Vol. 1, No. 2, pp.101–115. 2006.

(Papazoglou et al, 2007): Papazoglou Mike P., Heuvel Willem-Jan van den, "Service Oriented Architectures : approaches, technologies and research issues", *the VLDB Journal*, vol.16, pp 389-415, 2007.

(Pesic et al, 2007): Pesic M., Schonenberg MH., Sidorova N., Aalst W. Van der. "Constraint-based workflow models: Change made easy". *In Proceedings of the OTM Conference CoopIS'2007*. In vol 4803 of Lecture Notes in Computer Science, pp 77–94.Springer-Verlag, Berlin.

(Sadiq et al, 2001) : Sadiq S.W., Orlowska M.E., "On capturing Exceptions in workflow process models". *In proceedings of ER'2001.*

(Spetch et al, 2005): Specht, T., Drawehn, J., Thranert, M., Kuhne, S., "Modeling cooperative business processes and transformation to a service oriented architecture". *In the proceedings of the 7th IEEE International Conference on E-Commerce Technology*, Pp 249 – 256, 2005.

(Wombacher et al, 2003) : Wombacher, A., Mahleko, B., "Ad-Hoc Business Processes in Web Services". *In proceedings of the Symposium on Applications and the Internet (SAINT) Workshops*, pp. 101-105, Florida, USA, 2003.