Supervision Décentralisée des Chorégraphies de Services

Aymen Baouab, Olivier Perrin et Claude Godart

LORIA - INRIA Nancy - Université de Lorraine
F-54500 Vandœuvre-lès-Nancy, France
{aymen.baouab, olivier.perrin, claude.godart}@loria.fr

RÉSUMÉ. *Les chorégraphies de services sont de plus en plus adoptées comme un moyen de collaboration entre des organisations partenaires. Cependant, la quasi-totalité des approches proposées pour le suivi des compositions de services sont limitées à une même zone de confiance et ne peuvent donc pas être adaptées pour superviser les échanges de messages des processus métiers qui s'étendent à travers de nombreuses organisations administrativement indépendantes. Dans cet article, nous proposons une approche décentralisée et événementielle pour la supervision des interactions d'une chorégraphie inter-organisationelle. Des évènements sont définis et traités par des unités de supervision implantées le long des frontières de chaque organisation et invoqués à chaque détection d'un nouveau message échangé avec une organisation partenaire. Après la vérification de chaque message par rapport au schéma de la chorégraphie, une notification est automatiquement générée, envoyée et propagée à un sous ensemble pré-calculé de participants.*

ABSTRACT. *Cross-organizational choreographies are increasingly adopted by different companies. In order to guarantee that all involved partners are informed about errors that may happen in the collaboration, it is necessary to monitor the execution process by continuously observing and checking message exchanges during runtime. This allows a global process tracking and evaluation of process metrics. In this paper, we present an approach for decentralized monitoring of cross-organizational service-based processes. We define an hierarchical propagation model for exchanging external notifications between the collaborating parties in order to limit data exposure and avoid bandwidth overhead. The generation of notifications relies on state change of choreographies allowing to transparently observe the external behavior of each participant without providing information about the internal processes. The collected monitoring data can be further used for the evaluation of global process metrics by expressing and evaluating statistical queries over execution traces.*

MOTS-CLÉS : *décentralisation, service, chorégraphie, supervision*

KEYWORDS: *decentralization, service, choreography, monitoring*

1. Introduction

De nos jours, la croissance continuelle des entreprises incite celles-ci à externaliser et à sous traiter certaines de ses activités. Cela a conduit à une forte augmentation du nombre d'interactions entre les différents partenaires en collaboration. Pour être capable d'externaliser leurs opérations ou vendre une partie de leurs processus métiers, certaines entreprises éprouvent le besoin de partitionner leurs processus et de séparer les éléments qui les constituent (Wetzstein *et al.*, 2010). Une chorégraphie inter-organisationnelle peut être mise en oeuvre afin de permettre la coordination et la synchronisation de ces différentes partitions. Ainsi, chaque organisation est en charge d'exécuter une partie du processus global. Parmi les inconvénients majeurs de la sous-traitance, nous pouvons citer la perte de maîtrise, le manque de flexibilité de la part du prestataire, le risque de délais trop longs et le manque d'information et de transparence. Afin de pallier ces problèmes, il est nécessaire pour chaque organisation de superviser ses fragments externalisés, et ce, de façon abstraite sans pour autant dévoiler la logique métier de chaque partenaire. En effet, la propagation des données de supervision entre les partenaires permet une gestion décentralisée des informations recueillies, et ce, dans le but d'offrir plus de maîtrise au niveau de chaque organisation, et de réduire les délais et les coûts en cas d'occurrence d'exceptions.

Durant la dernière décennie, la supervision des compositions de services a été l'objet de plusieurs travaux de recherche (Ardissono *et al.*, 2008 ; Francalanza *et al.*, 2010 ; Moser *et al.*, 2011 ; Chafle *et al.*, 2004 ; Halle et Villemaire, 2009). Néanmoins, les approches proposées sont limitées aux compositions regroupant des services d'une même zone de confiance et ne peuvent pas être adaptées aux chaînes de production et d'approvisionnement qui s'étendent à travers de nombreuses organisations administrativement indépendantes. Ainsi, il est nécessaire de trouver un moyen efficace pour permettre une propagation instantanée des exceptions internes à travers les frontières organisationnelles. Dans ce sens, nous avons proposé un modèle architectural et un mécanisme de supervision et d'échange de notifications (Baouab *et al.*, 2011 ; Baouab *et al.*, 2012a ; Baouab *et al.*, 2012b). Dans cet article, nous généralisons notre approche de supervision décentralisée afin de traiter un plus large spectre de modèles de chorégraphies. Dans notre approche, les échanges inter-organisationels de messages sont perçus comme des évènements. Ces évènements sont traités par des unités de supervision implantées sous forme de services déployés le long des frontières de chaque organisation et invoqués à chaque détection d'un nouveau message échangé avec une organisation partenaire. Après la vérification de chaque message par rapport au schéma de la chorégraphie, une notification est automatiquement générée, envoyée et propagée à un sous ensemble pré-calculé de participants.

La section 2 montre l'intérêt et l'applicabilité de notre approche. La section 3 présente l'approche de classification hiérarchique des participants. Ensuite, la section 4 décrit les algorithmes de configuration et d'exécution. La section 5 montre comment appliquer notre approche sur un cas de chaîne d'approvisionnement. La section 6 décrit comment généraliser notre approche. Finalement, la section 7 présente quelques travaux apparentés, et la section 8 conclut cet article.

2. Chaînes d'approvisionnement et suivi des processus inter-entreprises

Pour illustrer les concepts et la démarche de l'approche présentée dans cet article et pour bien comprendre les problèmes qui peuvent exister, nous adoptons le scénario d'une chorégraphie inter-organisationnelle qui englobe trois types de participants (si on ne compte pas le client final) jouant les trois rôles suivants : Revendeur, Fournisseur et Constructeur dans une chaîne d'approvisionnement. Dans ce type de chaîne inter-organisationnelle, un revendeur peut interagir avec plusieurs fournisseurs qui, à leur tour, interagissent avec plusieurs constructeurs dans le but d'obtenir un devis pour des biens ou des services. La *figure* 1 montre un exemple de chorégraphie impliquant sept organisations principales : un client (C), un revendeur (R), deux fournisseurs (SA et SB), et trois constructeurs ($A1$, $A2$, et $A3$).

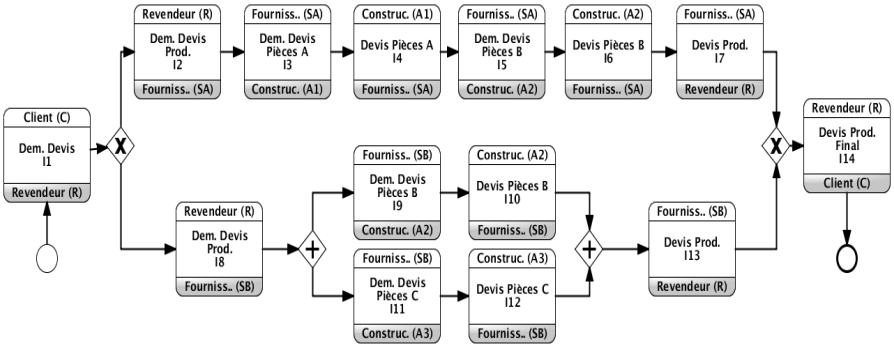

Figure 1. *Diagramme d'interaction d'une chaîne d'approvisionnement (BPMN 2.0).*

Dans cette chorégraphie, 14 interactions bi-partenaires sont définies ($I_{1,\ldots,14}$). Dans un premier temps, le revendeur reçoit une demande de devis de la part du client (I_1). Ensuite, il sélectionne un des deux fournisseurs (suivant le contenu de la demande) et élabore une nouvelle demande. En cas de sélection du fournisseur SA (à travers I_2), les constructeurs $A1$ et $A2$ seront invoqués en séquence ($I_{3,\ldots,6}$). Dans l'autre cas (i.e. SB sélectionné), les constructeurs $A2$ et $A3$ seront invoqués en parallèle ($I_{9,\ldots,12}$). Après avoir retourné les résultats finaux (I_7 ou bien I_{13}), le revendeur pourra enfin répondre au client par un devis complet (I_{14}). La chorégraphie ainsi présentée permet d'offrir au client final un service de demande de devis préalable. Dans le cas d'une chaîne de fabrication sur mesure de pièces originales, l'exécution d'une telle chorégraphie peut prendre des jours (voire des semaines pour des produits complexes nécessitant des pièces particulières).

Dans un cadre inter-organisationnel, chaque organisation n'est en relation qu'avec les partenaires qui lui sont directement connectés et ne peut donc pas voir au-delà. Techniquement, un service (ou un sous-processus exposé en tant que service) interagit en invoquant ses fournisseurs et en étant invoqué par ses consommateurs avec lesquels il a déjà conclu des accords de niveau de service. Lors de l'exécution, il n'a

donc aucune information sur le reste des participants de la chorégraphie. Toutefois, un service dépend de tous ses services clients, que ce soit par un lien direct ou transitif. Toute organisation peut donc s'intéresser à la surveillance de tous les fragments de processus externalisés et/ou exécutés par ses sous traitants. Dans notre exemple, le client C interagit uniquement avec le revendeur R et ne peut donc pas voir ce qui se passe au-delà. Après avoir envoyé sa demande à R, C restera en attente de la réponse sans avoir aucune information sur l'état actuel de sa demande. Ainsi, il serait intéressant de garder C informé de l'évolution de l'exécution de la chorégraphie. Un tel exemple montre la nécessité d'échanger des notifications afin de permettre le suivi d'une chorégraphie. En effet, dans une configuration centralisée (e.g. orchestration), ce problème ne se pose pas puisque tout est coordonné par une unité centrale qui peut connaître à tout moment l'état actuel de l'exécution. Par contre, lorsqu'il s'agit d'une chorégraphie qui franchit les frontières de plusieurs organisations administrativement indépendantes, la tâche s'avère délicate. En outre, l'échange des notifications permet d'avoir une vue plus large que la vue locale de chaque participant et de recueillir des données supplémentaires qui pourraient être utilisées a posteriori pour mesurer la performance globale du processus et le suivi de la réalisation des objectifs de l'entreprise.

3. Mécanisme décentralisé pour l'échange de notifications entre partenaires

Un superviseur traditionnel est censé superviser la conformité de l'exécution des processus locaux par rapport aux modèles définis et aux règles métiers de sa propre organisation administrativement indépendante (i.e. en se référant uniquement à ce qu'il peut voir localement). Ce type de supervision ne permet donc pas le suivi global d'une chorégraphie inter-organisationnelle puisqu'il n'est pas possible de voir ce qui se passe au-delà des frontières de l'organisation (e.g. les communications entre d'autres partenaires de collaboration). Pour superviser l'ensemble de la chorégraphie, des solutions ont été proposées (Ardissono *et al.*, 2008 ; Chafle *et al.*, 2004 ; Wetzstein *et al.*, 2010). Ces approches sont centralisées et se basent donc sur un superviseur central implanté dans une organisation à laquelle tous les autres participants doivent faire confiance. Cette entité centrale est notifiée par chaque participant chaque fois qu'un événement se produit, et ce, en utilisant, par exemple, le mécanisme de « $publish/subscribe$ » ou publier/ s'abonner (Wetzstein *et al.*, 2010) . Néanmoins, elle peut présenter un point de défaillance unique et peut ne pas être adaptée lorsqu'il n'y a pas d'entité de confiance commune à tous les participants (e.g. cas d'une chaîne d'approvisionnement avec plusieurs pays). En plus, on pourra aussi citer tous les autres inconvénients d'une configuration centralisée (e.g. goulot d'étranglement).

Pour faire face à ce genre de problème, nous proposons un mécanisme automatisé et décentralisé pour l'échange de données de supervision entre les participants. Nous définissons de nouveaux canaux pour l'échange de notifications entre les différents participants. Notre approche n'est pas intrusive, c'est-à-dire que ces canaux nouvellement définis sont indépendants des canaux dans lesquels les messages de la chorégraphie sont échangés. Une unité de supervision locale, appelée EFM, peut voir « passivement » ce qui est échangé sans rien modifier dans le modèle de chorégraphie.

La *Figure* 2 montre un aperçu sur notre approche.

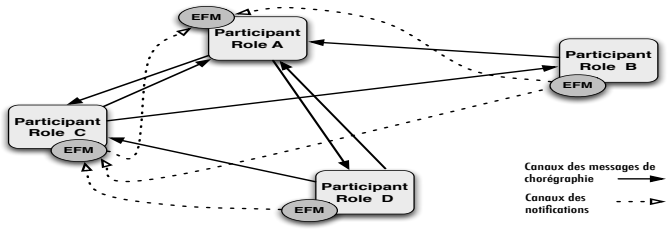

Figure 2. *Mécanisme décentralisé pour l'échange de notifications entre partenaires.*

3.1. *Définition des notifications externes*

Lors de l'exécution, le modèle chorégraphie est utilisé comme une base pour observer le comportement des participants. Notre approche repose uniquement sur les changements d'états de la chorégraphie (i.e. quand un message est envoyé ou reçu) permettant ainsi le suivi des échanges de messages d'une manière discrète (i.e. les messages échangés ne sont pas modifiés par les EFMs et les services composant la chorégraphie de base ne sont pas conscients des EFMs et des notifications créées et échangées). Chaque notification est corrélée à un message défini dans le modèle de la chorégraphie. Nous définissons une notification comme suit.

Definition 1 (Notification) *Une notification $n \in \mathcal{N}$ est un tuple (c_i,t,s,d,m_t) avec c_i l'identifiant de l'instance de chorégraphie (utilisé pour la corrélation), t le timestamp de la génération de l'évènement, s,$d \in \mathcal{P}$ (respectivement, la source et la destination du message associé), et $m_t \in \mathcal{M}_\mathcal{T}$ (le type/structure de message).*

3.2. *Classification hiérarchique des partenaires*

Échanger des notifications entre les partenaires risque d'induire un trafic supplémentaire très important. Afin de réduire cette charge supplémentaire du réseau, nous choisissons un échange sélectif. En d'autres termes, nous essayons de réduire le nombre de partenaires qui doivent être notifiés. Pour ce faire, nous introduisons un nouveau type de relation entre les participants de la chorégraphie. Avec une telle relation, nous définissons une direction particulière pour chaque notification.

Afin de permettre l'envoi sélectif des notifications, il faut commencer par déterminer le sous-ensemble des partenaires potentiellement intéressés par l'événement correspondant. Dans le cas des chaînes d'approvisionnement, les partenaires intéressés sont ceux qui sont considérés comme des consommateurs ou clients de ce participant (e.g. une entreprise est considérée comme cliente chez ses sous-traitants et est donc intéressée par l'accomplissement de toutes ses parties métiers externalisées). Nous procédons à une classification hiérarchique des participants suivant la relation producteur/consommateur et l'évolution de la valeur ajoutée dans une chaîne d'approvisionnement de services donnée. Nous commençons par mettre le client (i.e. l'initiateur

de la chorégraphie) au plus haut dans la hiérarchie (sur la couche supérieure). Les couches les plus basses sont constituées d'un ensemble de fournisseurs de services. Ainsi, chaque participant d'une couche donnée fournit un service à un service de la couche qui est juste au dessus (voir *figure* 3). Comme chaque couche dépend (directement ou transitivement) de toutes les couches inférieures, tout participant qui attend un service de la couche inférieure sera intéressé par le suivi de son exécution. Nous autorisons ainsi la propagation des notifications uniquement du bas vers le haut de la hiérarchie. Pour permettre une telle classification, nous introduisons la notion de *super / sous-partenaire* comme suit.

Definition 2 (Super / sous partenaire) *Un participant $P_i \in \mathcal{P}$ est appelé le $super-partenaire$ d'un participant P_j ssi P_i est l'émetteur dans la première interaction définie dans la vue locale de P_j, ou encore, l'instance du sous-processus de P_j est créée suite à un message venant de P_i. En d'autres termes, P_i est responsable de la participation de P_j dans la chorégraphie. Ainsi, P_j est appelé $sous-partenaire$ de P_i. Formellement, on note :*

$$Super(P_i) = P_j \Leftrightarrow P_i \in Sub(P_j)$$

Dans notre exemple de chaîne d'approvisionnement (présenté précédemment dans *Figure* 1) : $Super(SA) = R$; $Super(SB) = R = Super(Super(A2))$; $SA, SB \in Sub(R)$; $A1, A2 \in Sub(SA)$; $A2, A3 \in Sub(SB)$. Il faut remarquer que notre approche assume que la chorégraphie soit réalisable (Lohmann et Wolf, 2010) et avec un seul initiateur. Généralement, dans une chaîne d'approvisionnement, un fournisseur de services peut avoir, lui même, un ou plusieurs fournisseurs, qui eux aussi, peuvent avoir, à leur tour, d'autres fournisseurs et/ou sous-traitants, etc. Ceci fait qu'une telle structure est hiérarchique (ul Haq *et al.*, 2009). De cette façon, chaque participant aura exactement un *super-partenaire* (sauf celui qui a initié la chorégraphie) et peut avoir un ou plusieurs *sous-partenaires*. En tant que tel, les participants peuvent être classés dans une arborescence avec l'initiateur comme racine.

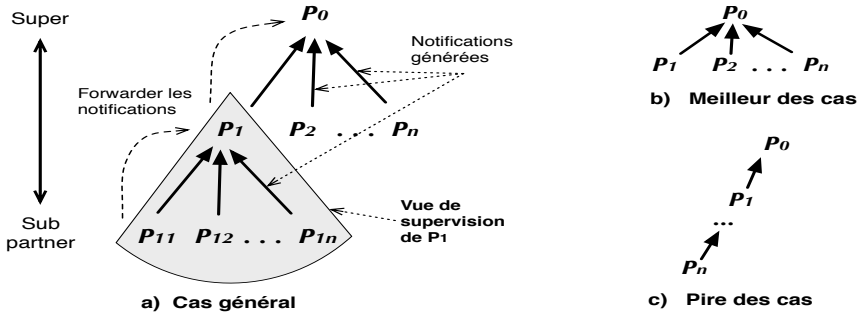

Figure 3. *Classification hiérarchique des partenaires.*

Chaque participant notifie son *super-partenaire* en générant une nouvelle notification chaque fois qu'il échange un message (envoi ou réception). En plus des notifications qu'il génère, il lui transmet aussi toutes celles reçues de la part de ses

sous-partenaires, et ce, directement dès qu'il les reçoit. Ainsi, les notifications se propagent toujours du bas vers le haut de l'arborescence des partenaires (i.e. de chaque *sous-partenaire* vers son *super-partenaire*). L'utilisation d'une telle architecture hiérarchique permet de réduire le nombre de notifications échangées et donc la surcharge du réseau. Ceci permet aussi de recevoir chaque notification au maximum une fois et élimine donc toute redondance suite à une réception multiple des notifications sur le même évènement de la part de deux sources différentes.

Nous mesurons la complexité de notre approche par rapport au nombre total de notifications échangées. En effet, ce nombre dépend du nombre de messages de la chorégraphie et du nombre de participants. Soit p le nombre de participants, la complexité, dans le meilleur des cas (lorsque la profondeur est égale à 1), est évidemment linéaire (voir *figure* 3.b). Dans la moyenne des cas (la quasi-totalité des cas), notre approche nécessite $O(p.Log(p))$ notifications. Dans le pire des cas et lorsque l'arbre est totalement déséquilibré (voir *figure* 3.c.), nous avons besoin de $O(p^2)$ notifications. Ici, l'arbre ressemble à une liste chaînée (aussi appelé arbre dégénéré).

Preuve 1 *Soit :*

– c *le nombre moyen de fils pour chaque nœud (i.e. le nombre moyen de* sous-partenaires *pour chaque participant),*

– m *le nombre maximum de messages envoyés par chaque participant,*

– *et* I_i *(avec* $i \in [0..d]$*) l'étage numéro* i *de l'arbre (à partir du bas de l'arbre).*

Pour chaque noeud (participant) dans le premier niveau I_1*, il y aura au maximum* m *notifications à envoyer. Pour* I_2*, il y aura* $2 * m$*, et pour* I_{d-1}*, il y aura* $(d-1) * m$*, et ce pour chaque participant. Donc, pour chaque participant nous aurons au maximum* $(d-1) * m$ *notifications. Dans un arbre quelconque, le nombre moyen de noeuds du niveau i est égal à* c^i *qui est majoré par* c^d*. Ainsi, le nombre total de notifications* n *est majoré par* $d * m * c^d$*, c'est à dire,* $n = O(d * c^d)$*. Nous savons que la profondeur d'un c-arbre (arbre avec* c *fils en moyenne par noeud) est* $d = O(log_c(p))$*. Nous trouvons finalement :* $n = O(log_c(p) * c^{log_c(p)}) = O(p * log_c(p))$*, si* $c > 1$*. En effet, le cas particulier* $c = 1$ *décrit le pire des cas représenté dans la figure 3.c.*

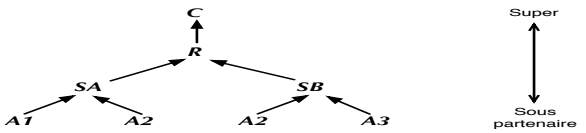

Figure 4. *Arborescence des partenaires pour l'exemple de motivation.*

La *figure* 4 montre l'arbre généré pour notre exemple de chaîne d'approvisionnement. Nous remarquons que le participant $A2$ figure deux fois dans l'arbre. En effet, suivant le chemin emprunté dans le schéma de la chorégraphie (i.e. le choix du fournisseur entre SA et SB fait par le revendeur R), le constructeur $A2$ pourra avoir un des deux fournisseurs comme *super-partenaire*. Nous rappelons que l'affectation des

super-partenaires se fait au cours de l'exécution de la chorégraphie au moment de la réception du premier message pour chaque participant et ce de façon indépendante pour chaque instance. Donc, ici $A2$ aura, à chaque instance, SA ou bien SB comme *super-partenaire* et non pas les deux. En d'autre termes, le *super-partenaire* de chaque participant est toujours unique. Ceci permet, entre autres, de garantir à notre classification de partenaires de garder sa structure hiérarchique.

3.3. *Vue de supervision externe (EFM-View)*

Une vue locale dans une chorégraphie représente la visibilité du point de vue d'un des participants. Elle fait référence au modèle restreint à toutes les interactions d'un seul partenaire. Par défaut, chaque participant est limité seulement à sa propre vue. En utilisant notre approche hiérarchique d'échange de notifications, chaque EFM reçoit des notifications sur tout changement d'état relatif à l'ensemble de ses *sous-partenaires*. Ainsi, l'EFM de chaque participant aura une vue plus large que la vue locale de ce dernier. Nous appelons cette vue nouvellement créée « vue de supervision externe » (*EFM-View*). L'*EFM-View* d'un participant P_i inclut toutes les interactions ayant comme émetteur ou receveur un des *sous-partenaires* de P_i. Cette vue comprend également l'ensemble des contraintes sur le séquencement de ces interactions. Formellement, nous définissons la vue de supervision (*EFM-view*) comme suit.

Definition 3 (Vue de supervision (EFM-View)) *Soit $\mathcal{VS}_i$ la vue de supervision du participant P_i et $\mathcal{V}_i$ sa vue locale. $\mathcal{VS}_i = \cup_{j \in Sub(P_i)} \mathcal{VS}_j \cup \mathcal{V}_i$*

Note. Si un participant P_i n'a pas de *sous-partenaires* ($Sub(P_i) = \emptyset$), alors nous avons $\mathcal{VS}_i = \mathcal{V}_i$, ce qui veut dire que sa vue de supervision *EFM-view* est restreinte à sa vue locale.

4. Algorithmes de configuration et d'échange de notifications

Pour permettre la mise en oeuvre de notre approche d'échange de notification entre partenaires, nous commençons par distinguer deux phases : la phase de configuration et la phase d'exécution. La phase de configuration consiste à calculer le *super-partenaire* et les *sous-partenaires* de chaque participant et la définition des notifications nécessaires pour chaque modèle de chorégraphie (i.e. quelles sont les interactions qui doivent être notifiées et à quel partenaire). L'*algorithme* 1 montre un pseudo-code de l'algorithme de configuration.

Soit C est une chorégraphie et $\mathcal{P}$ l'ensemble des participants. Pour chacun des participants P_i, nous cherchons la première interaction dans sa vue locale et nous regardons le champ «émetteur» dans le message associé. Ce dernier est considéré comme responsable de l'introduction de P_i dans la collaboration (i.e. son *super-partenaire*). Ensuite, nous définissons, à partir du modèle défini de la chorégraphie, l'ensemble des notifications nécessaires (i.e. celles qui doivent être générées pendant la phase d'exécution).

Concernant la phase d'exécution, l'*algorithme* 2 décrit le processus d'échange de notifications qui est exécuté au niveau de chaque participant. Cet algorithme montre

Algorithme 1: Algorithme d'échange de notification (Phase de configuration)

Require : - C : un modèle de choreographie (une vue globale)
- $\mathcal{P}$: un ensemble de participants
for *(each P_i in $\mathcal{P}$)* **do**
    /* Identification of the Super of P_i */
    $I_{i0} \leftarrow GetFirstInteraction(P_i)$
    $Super_i \leftarrow GetSource(I_{i0})$
    $SubPartners(Super_i) \leftarrow SubPartners(Super_i) \cup \{P_i\}$
    /* Définition de l'ensemble des notifications nécessaires */
    $V_i \leftarrow ConstuctNotificationSet(P_i)$

comment et quand générer, envoyer et transférer les notifications pour chacun des EFMs. Nous soulignons le fait que chaque EFM traite trois types d'événements : (i) les événements liés aux messages échangés avec d'autres partenaires (messages de chorégraphie), (ii) les événements liés au transfert de notifications vers le sommet de l'hiérarchie (i.e. les événements liés aux messages échangés entre des sous-partenaires), et (iii) les événements indiquant des exceptions. Selon le type d'événement, l'EFM se comporte différemment :

Algorithme 2: Algorithme d'échange de notifications (Phase d'exécution)

Require : - S : Le super partenaire du participant en question
- $\mathcal{V}$: La vue de supervision EFM-view
for *(each event occurrence e_i)* **do**
    if *(e_i.type $\neq$ Exception and CheckConformance(e_i))* **then**
        $UpdateMonitorState(e_i)$
        if *($e_i.msrc \neq Super$* **and** *$e_i.mdst \neq Super$)* **then** /* Generate a new notification if e_i is not coming from/going to the Super */
            if *(e_i.type = message exchange)* **then**
                $n \leftarrow GenerateNotification(e_i)$
                $SendNotification(n, Super)$
            else
                /* e_i.type = notification exchange */
                $ForwardNotification(e_i, Super)$
    else
        $AlertInternalMonitor()$
        $ReportExceptionTo$(Super, Sub(P_i))

– Si le type d'événement est un échange de message, l'EFM vérifie si c'est conforme à sa vue de supervision ($EFM - view$). Ce contrôle consiste à vérifier si le message reçu ou envoyé est conforme à celui qui est attendu, et ce, en se basant sur les contraintes spécifiées dans le modèle de chorégraphie. Ceci permettra, entre

autres, d'éviter l'envoi de notifications inappropriées ainsi que la duplication des notifications. Si le message est conforme, l'EFM met à jour son statut (noeud actuel dans le graphe de vue de supervision), génère une nouvelle notification indiquant l'occurrence du message et l'envoie à son *super-partenaire*. En effet, nous n'avons pas besoin de notifier le *super-partenaire* si ce dernier figure dans le champ émetteur ou receveur du message car il est déjà au courant de son occurrence.

– Si le type d'événement est une réception de notification (i.e. un des *sous-partenaires* nous a envoyé une notification sur une occurrence de message), l'EFM vérifie si c'est conforme à sa vue locale et met à jour son statut. Ensuite, il transfère la notification à son *super-partenaire*. Dans ce cas aussi, nous optimisons le nombre de notifications envoyées en éliminant celles qui ne sont pas nécessaires (e.g. quand le *super-partenaire* est déjà au courant de l'évènement). L'EFM ne transfère donc pas les évènements ayant son *super-partenaire* comme émetteur ou receveur du message notifié.

– Si le type d'événement est une exception générée par un autre partenaire, l'EFM traite localement cette exception et la transmet tout de suite à son *super-partenaire* et à tous ses *sous-partenaires*. Les autres feront de même afin que l'exception soit propagée à tout le monde.

Dans les deux premiers cas, si l'événement n'est pas conforme à celui attendu, une exception est générée, le moniteur interne est alerté. Le *super-partenaire* et tous les *sous-partenaires* sont ensuite informés. Il est à noter que les *super-partenaires* peuvent déceler l'exception automatiquement sans avoir à être notifiés par leurs *sous-partenaires* car ils seront bloqués après un certain temps. Une telle propagation rapide des exceptions dans tout l'arbre permettra d'accélérer la détection des situations de blocage.

5. Application : Cas d'une chorégraphie d'une chaîne d'approvisionnement

Afin d'illustrer le fonctionnement de notre algorithme d'échange de notifications, nous avons appliqué notre approche sur la chorégraphie chaîne d'approvisionnement introduite dans *Section* 2. La *figure* 5 montre le diagramme de collaboration complet (modélisé en BPMN 2.0). Ce diagramme montre les vues locales de tous les participants ($C, R, SA, SB, A1, A2$ et $A3$) ainsi que les messages échangés. Nous ajoutons les canaux nouvellement créés pour les notifications ainsi que la définition des *super-partenaires*.

Après l'exécution de l'algorithme de configuration (*cf. Algorithme* 1), on définit l'ensemble des relations entre les participants (i.e. *super/sous-partenaires*). Par la suite, l'ensemble des notifications qui seront envoyées par chaque participant à son *super-partenaire* est automatiquement défini à partir du schéma de la chorégraphie. Dans cet exemple, nous discernons douze types de notifications $N_{2,\ldots,13}$ respectivement associées aux messages $M_{2,\ldots,13}$ définis dans la chorégraphie (e.g. Le fournisseur SA définit les quatre notifications $N_{3,\ldots,6}$). Cela permet de construire, pour chaque participant, une vue de supervision (*EFM-view*) qui va inclure la vue locale et

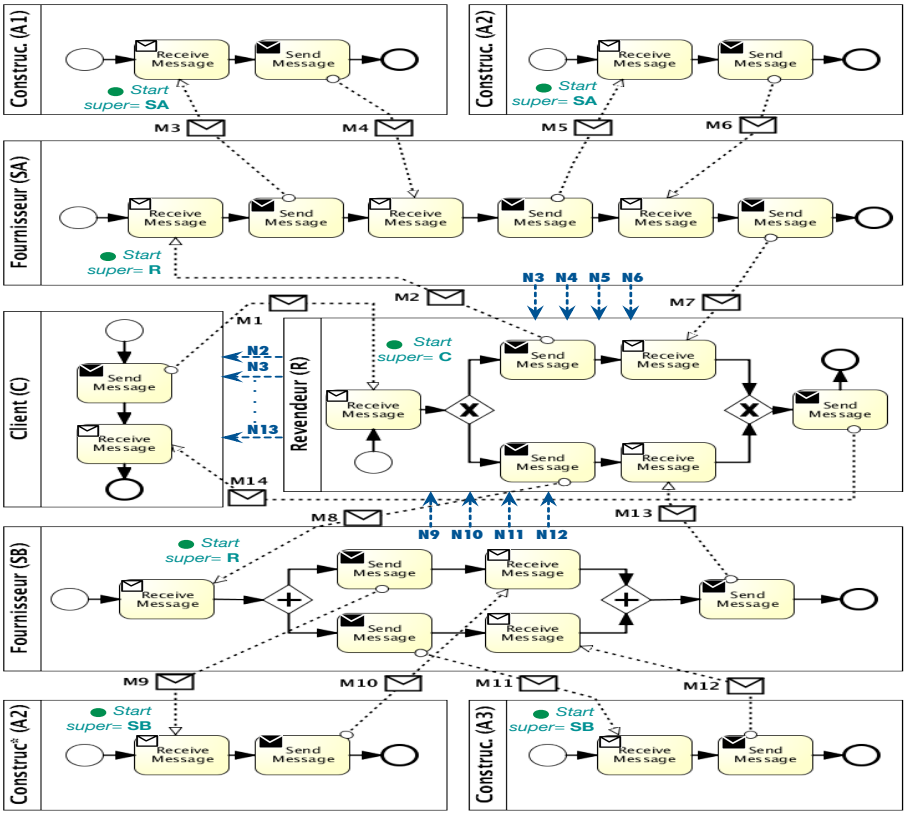

Figure 5. *Vues locales et échange de notifications : Diagramme de collaboration BPMN 2.0*

les différentes notifications à recevoir. La *figure* 6 montre l'*EFM-view* du Revendeur (R). Dans cette figure, nous pouvons remarquer l'ajout de huit notifications ($N_{3,\ldots,6}$ et $N_{9,\ldots,12}$) qui correspondent à tous les messages échangés par les *sous-partenaires* du Revendeur. Ces notifications lui sont envoyées de la part des *sous-partenaires* (i.e. les fournisseurs SA et SB). Nous remarquons que les constructeurs $A1$, $A2$, et $A3$ n'ont pas de *sous-partenaires* ($Sub(Ai) = \emptyset$). Ainsi, leurs vues de supervision sont égales à leurs vues locales ($\mathcal{V}_{Ai} = \mathcal{C}_{Ai}$).

En phase d'exécution et à la réception du premier message, chaque participant fixe son *super-partenaire* (e.g. $Super(R) = C$, $Super(SA) = R$ et $Super(SB) = R$) à l'exception du client C qui n'a pas de *super-partenaire* puisqu'il est l'initiateur de la chorégraphie (racine de l'arborescence). Ensuite, à chaque réception ou envoi de message par un participant, ce dernier envoie la notification associée à son *super-partenaire* (e.g. quand le fournisseur SA envoie le message M_3 il notifie son *super-partenaire* R avec la notification N_3). En plus, à chaque réception de notifi-

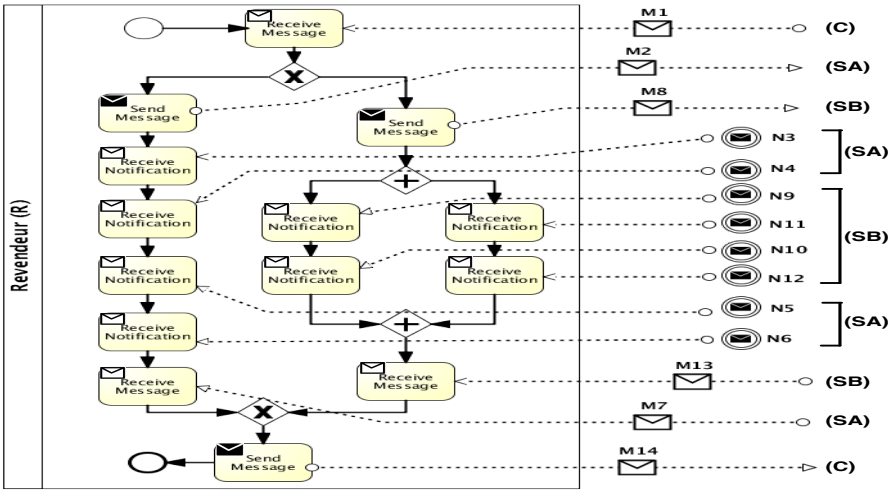

Figure 6. *Vue de supervision (*EFM-view*) du Revendeur (R).*

cation de l'un des *sous-partenaires*, le participant transfère cette notification à son *super-partenaire* (e.g. quand le revendeur R reçoit la notification N_3, il la transfère à son *super-partenaire* C). Afin de réduire le nombre de messages échangés et supprimer les transferts des notifications non nécessaires, les notifications associées aux messages ayant le *super-partenaire* comme émetteur ou receveur ne sont pas transférées au *super-partenaire* (il est déjà au courant). La *figure* 7 montre, à l'aide d'un diagramme de séquence, un scénario d'exécution réussie. Les messages de la chorégraphie sont modélisés par des flèches continues. Les notifications échangées, quant à elles, sont modélisées par des flèches en pointillé. $ci1$ est l'identificateur de l'instance. τ_i représente le $timestamp$ de l'occurrence du message M_i (temps de réception ou d'envoi). En fait, l'attribut $timestamp$ est nécessaire pour la vérification de l'ordre des messages. Les trois derniers attributs de la notification correspondent à l'émetteur, receveur et type du message associé.

Notre approche permet de renforcer les contraintes locales de chaque participant (relatives au séquencement des messages) au niveau de son *super-partenaire*. Par exemple, la contrainte de co-occurrence entre les messages M_9 et M_{11}, vérifiée localement par le fournisseur SB, peut être aussi contrôlée par l'EFM du revendeur R en vérifiant la co-occurrence des deux notifications N_9 et N_{11}. Cette dernière co-occurrence peut par ailleurs être contrôlée par le client C puisqu'elle figure aussi dans sa vue de supervision.

6. Généralisation de l'approche

Dans un cas plus général et en dehors du cadre des chaînes d'approvisionnement, la classification hiérarchique des participants peut paraître peu évidente. En effet, les

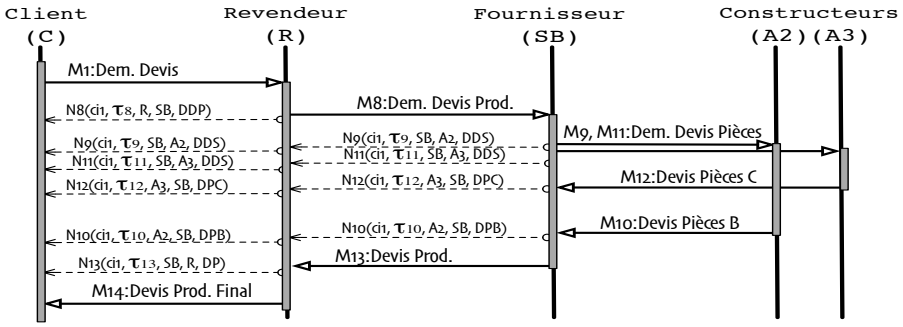

Figure 7. *Exemple d'une exécution correcte (Diagramme de séquence).*

chorégraphies inter-organisationnelles peuvent comporter des cycles dans les interactions (e.g. trois participants A, B, C avec A qui invoque B, B invoque C qui, lui même, invoque A). Dans de pareils cas, l'affectation des *super-partenaires* peut se faire de façon dynamique (i.e. affectation au premier message reçu). Reste que dans certains cas, nous pouvons avoir des notifications non nécessaires, voire redondantes parfois. Pour pallier ce problème, nous proposons les deux extensions suivantes :

6.1. *Élimination des redondances*

Les redondances peuvent se produire lorsqu'un participant reçoit plusieurs notifications sur le même message échangé, et ce, de la part d'au moins deux de ses *sous-partenaires*. Une telle situation peut se présenter par exemple si deux *sous-partenaires* B et C, d'un même participant A, échangent plusieurs messages entre-eux $M_3, ..., M_n$ (voir *figure* 8). A chaque message, chacun des deux participants va donc envoyer une notification à A. Ce dernier va donc recevoir deux notifications pour chaque message échangé et va à son tour les transférer à son *super-partenaire* S.

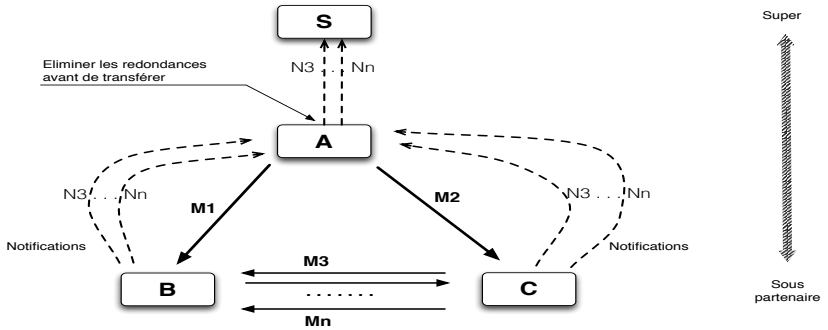

Figure 8. *Extension de l'approche (Elimination de la redondance des notifications).*

Ce type de problème peut être détecté au niveau du participant A (i.e. le point où converge les notifications). Une amélioration possible de l'algorithme de routage des

notifications consiste donc à filtrer les notifications sur le même message en éliminant les redondances et en ne transférant qu'un seul exemplaire.

6.2. *Élimination des cycles*

Dans le cas d'une chorégraphie qui comporte des cycles, notifier son *super-partenaire* peut parfois être non nécessaire. Il arrive que ce *super-partenaire* soit déjà au courant de l'occurrence de l'événement. La *figure* 9 montre, en utilisant un diagramme de séquence, un scénario d'exécution d'une chorégraphie de transfert d'argent par courrier électronique. Dans cet exemple, nous avons quatre participants : l'expéditeur S, sa banque SB, le bénéficiaire R et sa banque RB. Après avoir appliqué notre mécanisme de classification hiérarchique, nous pouvons obtenir facilement les relations suivantes : $Super(SB) = S$, $Super(R) = SB$ et $Super(RB) = R$.

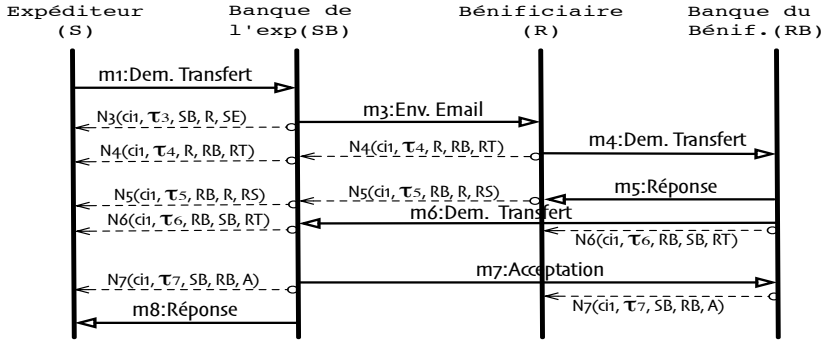

Figure 9. *Extension de l'approche (Elimination des notifications non nécessaires).*

Nous remarquons sur la figure que R n'a pas transféré les deux notifications N_6 et N_7 à son *super-partenaire* SB. En effet, notifier SB n'est pas nécessaire puisqu'il est déjà au courant de l'occurrence du message M_6 (c'est lui le destinataire) et du message M_7 (c'est lui l'émetteur).

7. Travaux connexes

La supervision des compositions de services a fait l'objet de plusieurs travaux de recherche. (Subramanian *et al.*, 2008) ont présenté une approche pour améliorer les moteurs BPEL en proposant un nouveau moteur appelé "SelfHealBPEL" qui met en œuvre des fonctionnalités supplémentaires pour la détection des violations et le traitement des exceptions. Cette approche peut avoir un impact négatif sur la performance et l'agilité du fait qu'il n'y a pas de séparation entre la logique de supervision et le moteur BPEL. (Barbon *et al.*, 2006) ont proposé une architecture qui sépare la logique métier d'un service de la partie de supervision. Cependant, leur approche centralisée cible les orchestrations BPEL et ne traitent pas le problème des chorégraphies dans un cadre inter-organisationnel. (Ardissono *et al.*, 2008) ont présenté un $framework$ permettant le suivi de l'avancement de l'exécution d'une chorégraphie afin d'assurer

la détection précoce des conflits et la notification des participants concernés. L'approche consiste en un moniteur central qui est notifié par chaque participant à chaque fois qu'il envoie ou reçoit un message. Néanmoins, cette approche est centralisée et reste non adaptée aux chorégraphies inter-organisationnelles quand il est difficile de trouver une partie commune auquelle toutes les organisations font confiance (e.g. cas de plusieurs pays).

Dans le cas des processus décentralisés au sein de la même organisation (i.e. même cercle de confiance), (Chafle *et al.*, 2004) ont modélisé un moniteur implémenté comme un service Web. Ce moniteur maintient l'état de toutes les activités du service composite. (Halle et Villemaire, 2009) ont introduit le concept de superviseur MBM qui intercepte et traite les messages échangés. Pour chaque participant un MBM local signe les messages sortants avec l'état actuel pour éviter les désynchronisations et offrir un protocole d'échange fiable. En ce qui concerne les solutions basées sur les événements, les approches sont principalement basées sur la technologie BAM (Business Activity Monitoring) (Dahanayake *et al.*, 2011). (Kikuchi *et al.*, 2007) ont utilisé un format d'audit commun qui permet le traitement et la corrélation d'événements entre plusieurs moteurs BPEL. (Wetzstein *et al.*, 2010) ont présenté une spécification d'événements complexes et utilisé un identificateur d'instance de chorégraphie (CIID) pour permettre la corrélation des événements inter-organisationnels (ce qui n'est pas pris en charge dans (Kikuchi *et al.*, 2007)). Contrairement à ces travaux, nous avons proposé un mécanisme automatisé et décentralisé pour l'échange de notifications entre les participants dans le but de superviser des chorégraphies franchissant les limites des zones de confiance.

8. Conclusion et perspectives

Nous avons présenté un mécanisme d'échange de notifications entre les différents participants d'une chorégraphie. Nous avons montré comment une telle approche pourra permettre une supervision décentralisée de l'exécution d'une chorégraphie, et ce, dans le but d'offrir plus de maîtrise au niveau de chaque organisation participante et de réduire les délais et les coûts en cas d'occurrence d'exceptions. La supervision se fait de façon abstraite (i.e. uniquement sur les échanges de messages inter-organisationnels) sans pour autant dévoiler la logique métier de chaque entreprise. La propagation hiérarchique des données de supervision permet de limiter l'exposition des données et d'éviter une surcharge du réseau (notification sélective). En outre, notre approche est non intrusive dans le sens où elle se base sur l'écoute passive des messages de la chorégraphie sans aucune modification de la structure ou du contenu des messages (i.e. les processus ne sont pas altérés et ne sont pas conscients de la supervision et l'échange de notifications). Notre approche permet aussi d'offrir une traçabilité de l'exécution des processus. En effet, les informations recueillies pourraient être utilisées a posteriori pour mesurer la performance globale du processus et le suivi de la réalisation des objectifs de chaque organisation. Des travaux futurs nous permettront l'extension de notre approche afin de gérer les actions à entreprendre si une exception (e.g. violation de contrainte, redondance, blocage, etc.) est détectée.

Nous prévoyons aussi aborder les questions de confidentialité, et ce, afin d'éviter la divulgation des informations critiques.

9. Bibliographie

Ardissono L., Furnari R., Goy A., Petrone G., Segnan M., « An Event-Based Model for the Management of Choreographed Services », vol. 5183, p. 51-60, Springer Berlin Heidelberg, 2008.

Baouab A., Perrin O., Godart C., « An Event-Driven Approach for Runtime Verification of Inter-organizational Choreographies », *2011 IEEE International Conference on Services Computing (SCC)*, july 2011, p. 640 -647.

Baouab A., Fdhila W., Perrin O., Godart C., « Towards Decentralized Monitoring of Supply Chains », *ICWS*, 2012, p. 600-607.

Baouab A., Perrin O., Godart C., « An Optimized Derivation of Event Queries to Monitor Choreography Violations », *ICSOC*, 2012, p. 222-236.

Barbon F., Traverso P., Pistore M., Trainotti M., « Run-Time Monitoring of Instances and Classes of Web Service Compositions », *Web Services, IEEE International Conference on*, vol. 0, 2006, p. 63-71, IEEE Computer Society.

Chafle G. B., Chandra S., Mann V., Nanda M. G., « Decentralized orchestration of composite web services », WWW Alt. '04, ACM, 2004, p. 134–143.

Dahanayake A., Welke R. J., Cavalheiro G., « Improving the understanding of BAM technology for real-time decision support », *Int. J. Bus. Inf. Syst.*, vol. 7, 2011, p. 1–26, Inderscience Publishers.

Francalanza A., Gauci A., Pace G., « Runtime Monitoring of Distributed Systems (Extended Abstract) », rapport, 2010, University of Malta, WICT.

Halle S., Villemaire R., « Flexible and reliable messaging using runtime monitoring », *Enterprise Distributed Object Computing Conference Workshops, 2009. EDOCW 2009. 13th*, sept. 2009, p. 116 -125.

Kikuchi S., Shimamura H., Kanna Y., « Monitoring Method of Cross-Sites' Processes Executed by Multiple WS-BPEL Processors », *CEC/EEE 2007*, 2007, p. 55 -64.

Lohmann N., Wolf K., *Realizability Is Controllability*, vol. 6194, Springer Berlin Heidelberg, 2010.

Moser O., Rosenberg F., Dustdar S., « Event Driven Monitoring for Service Composition Infrastructures. », *WISE*, Springer, 2011, p. 38-51.

Subramanian S., Thiran P., Narendra N., Mostefaoui G., Maamar Z., « On the Enhancement of BPEL Engines for Self-Healing Composite Web Services », *SAINT*, 2008, p. 33 -39.

ul Haq I., Huqqani A., Schikuta E., « Aggregating Hierarchical Service Level Agreements in Business Value Networks », *Business Process Management*, vol. 5701, Springer Berlin / Heidelberg, 2009, p. 176-192.

Wetzstein B., Karastoyanova D., Kopp O., Leymann F., Zwink D., « Cross-organizational process monitoring based on service choreographies », *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, NY, USA, 2010, ACM, p. 2485–2490.