Un modèle d'interaction de services pour la coopération des procédés

Karim Baïna, Samir Tata, Khalid Benali

Un Modèle d'Interaction de Services pour la Coopération de Procédés

Karim Baïna, Samir Tata, Khalid Benali
e-mail: {baina,tata,benali}@loria.fr
LORIA - INRIA - CNRS (UMR 7503)
BP 239, F-54506 Vandœuvre-lès-Nancy Cedex, France

Résumé.

La conception ou la réalisation de tout projet un tant soit peu conséquent sous-entend l'implication d'un certain nombre de personnes, voire d'un certain nombre d'équipes ou d'entreprises. Ces entreprises interagissant et échangeant des données, de plus en plus via Internet et le "Web", on parle alors d'entreprise virtuelle. Cependant, il ne suffit pas simplement d'échanger des données pour travailler ensemble, il faut aussi contrôler et gérer ces échanges dans le cadre d'une démarche. Chaque entreprise possédant sa propre démarche et donc son propre procédé d'entreprise, la collaboration entre ces entreprises signifie l'interconnexion de ces procédés d'entreprise. Si plusieurs outils de coordination de travail existent, ils ont été développés pour les besoins internes d'une entreprise et sont mal adaptés à la collaboration inter-entreprises. L'approche qui nous a semblés la plus prometteuse pour l'interconnexion de procédés d'entreprises différentes et hétérogènes est l'approche orientée services. L'objectif de cet article est la description d'un modèle d'interaction de services pour l'interconnexion de procédés d'entreprise. Ce modèle se base sur le partage d'information entre services procédés et la coordination de services procédés. Dans cet article, après la description de l'approche orientée service, nous présenterons notre modèle d'interaction de services de manière formelle tout en nous servant d'un exemple support pour illustrer notre propos et appliquer cette formalisation à un cas concret. Nous présenterons enfin et de manière succincte comment ce modèle d'interaction de services procédés est mis en œuvre au sein de notre prototype DISCOBOLE (DIStributed CO-operation and Business prOcess on LinE) réalisé en Java au-dessus d'un bus CORBA.

Mots clefs : *systèmes d'information coopératifs, interconnexion de procédés, intégration de services, institutions virtuelles, technologie Web, interopérabilité entre systèmes.*

1 Introduction

Sachant l'importance que connaît l'automatisation des procédés d'entreprises, l'interconnexion de ces procédés devient de plus en plus difficile à contourner.

Si un large panel d'outils de coordination de travail existe (workflows, agendas partagés, outils de gestion de projets, outils d'édition coopérative, gestionnaires de versions et de configurations, etc.), ils ont été essentiellement développés pour les besoins internes des entreprises. Ils sont, en général, mal adaptés à la coopération inter-entreprises. Concernant les systèmes de gestion de workflows (SGWF), les solutions d'interconnexion existantes restent, en majorité, propriétaires (*i.e.* basées sur : des formats propriétaires de données, des langages spécifiques de définition de procédés-métiers, des plates-formes particulières de gestion de procédés,...). Afin d'améliorer le support générique d'interconnexion de procédés au sein des SGWF existants, de nouveaux modèles d'interconnexion sont en cours de développement. Ces modèles traitent de la conscience de groupe, de la formalisation du flux de données et de contrôle entre les procédés coopérants (*e.g.* modèles de gestion du partage de données [29], modèles de réplication de données [1], mécanismes de communication par envoi de messages [2, 5], paradigme d'abonnement/notification d'événements[9, 19, 34], invocation d'objets distants [26, 35], extension de protocoles de transfert [7],...). D'autres modèles traitent du contrôle du cycle de vie des procédés coopérants (*e.g.* protocoles d'échanges transactionnels [8, 16, 30], ...).

L'approche qui nous a semblés la plus prometteuse et la plus générique pour interconnecter des procédés et que nous avons, donc, choisie est une approche orientée service. Pour cela, nous avons besoin d'une architecture souple et flexible qui puisse supporter les ressources des procédés à interconnecter ainsi que les fonctionnalités nécessaires à leur interconnexion. Pour résumer notre approche, nous visons le développement d'un modèle d'interconnexion des procédés d'entreprises à travers l'interaction de services.

Le concept de service a été défini dans différents domaines de recherche : recherche sur l'orienté objet [25], recherche sur la modélisation de procédés [10, 17, 18, 20, 27, 33], recherche sur les systèmes

distribués [6, 21], etc. Dans le domaine de recherche sur le workflow, un service procédé peut être vu comme étant une entité logicielle capable de présenter les particularités et les objectifs d'un procédé sans en révéler la structure (*i.e.* son implantation dans un SGWF ou dans un gestionnaire de projets). En effet, un service offre une abstraction fonctionnelle d'un procédé (ou d'une partie d'un procédé) fourni par une organisation. Un service procédé spécifie la quantité de travail qu'une organisation promet de réaliser sous un certain contrat de coopération. De plus, il spécifie les parties du procédé qu'il couvre et les moyens d'y accéder. Le concept de service procédé a été étudié de divers points de vues : sémantique abstraite de l'exécution d'un service [17], sélection d'une sous-partie d'un service procédé [20], configuration dynamique des activités d'un service procédé [10], niveaux abstraits de contrôle de flux d'un service procédé [18], adaptation des méthodes et événements de deux services [6], etc. Un service, lié à un procédé, se comporte comme un patron de conception qui supporte, d'une manière pertinente, la coopération et l'interconnexion dynamique entre procédés d'entreprises.

Nous avons commencé le travail sur les services par le développement d'un modèle de négociation pour le travail coopératif [22], puis nous nous sommes intéressés à la sélection dynamique des contrats de services [4, 3]. Dans cet article, nous nous intéressons à la coopération et aux interactions des services pour l'interconnexion des procédés d'entreprises.

Cet article présente notre modèle d'interaction des services pour l'interconnexion des procédés d'entreprises. Nous commencerons par définir de manière précise, dans la section 2, la notion de service procédé sur laquelle se base notre modèle d'interaction de services. Nous attaquerons ensuite la présentation de notre modèle d'interactions de services procédés. Ce modèle se base sur le partage d'informations entre services procédés que nous traiterons dans la section 3 et la coordination de services procédés, définie et décrite dans la section 4. Le modèle d'interaction sera ensuite synthétisé dans la section 5. La section 6 donnera en outre une vision globale de la mise en oeuvre de notre modèle sur notre plate-forme d'expérimentation et conclura cet article en présentant de nouvelles perspectives de notre approche.

2 Approche orientée service

2.1 Services procédés

L'échange de service est un paradigme de coopération de haut niveau. Il se base sur les paradigmes de coopération existants : de **production** (partage d'espaces communs de données), de **communication** (partage de format commun de messages, invocations de méthodes distantes, échange de messages de notification, conscience de groupe -*group awarness*-) et de **coordination** (synchronisation, vérification de critères de cohérence). De ce fait, il offre une plus grande richesse en termes de l'expressivité de son utilisation dans les situations de coopération. Notre objectif est de développer la structure de service comme un patron de conception (*design pattern*) supportant la coopération des procédés inter-organisationnels.

Un service procédé est une description d'un travail qu'un demandeur souhaite externaliser (*e.g.* faute de temps, de maîtrise de coûts ou de compétences). Ce travail peut être géré par un procédé automatique, semi-automatique ou manuel. Il peut être à la portée d'un seul fournisseur (service procédé atomique) comme il peut nécessiter l'intervention de plusieurs fournisseurs (service procédé composé). Nous parlerons alors de composition de services procédés. On distingue alors deux types de services procédés : le service procédé atomique et le service procédé composé. Un service procédé atomique représente l'interface d'un procédé dont on cache la structure. Quant au service procédé composé, il regroupe un ensemble de services qui peuvent être atomiques ou composés à leur tour. Un demandeur peut donc composer son service requis à partir de services fournis par plusieurs fournisseurs.

Un service procédé atomique, qui n'intervient pas dans un service composé, est une entité indépendante isolée des autres services procédés. La figure 1-a montre trois services procédés requis atomiques, le procédé de l'entreprise R coordonnera ces services commes des activités internes sans permettre leur interaction. De ce fait, les fournisseurs de services ne se connaissent pas et donc ne peuvent pas s'échanger leurs résultats intermédiaires. En effet, les fournisseurs n'interagissent qu'avec le demandeur de services et n'interagissent jamais entre eux.

Néanmoins, au sein d'un service composé, les services procédés composants peuvent interagir entre eux. Une telle interaction assure une meilleure coopération non seulement entre demandeur et fournisseur mais aussi entre fournisseurs eux mêmes (cf. Figure 1-b). Elle assure, en plus, une meilleure conscience des différents services demandés au sein d'une communauté et permet de supporter des services qui font appel à plusieurs compétences et qui ne sont pas nécessairement à la portée d'un seul fournisseur. Ce type de services constitue une composition de différents services rendus par les fournisseurs interactifs. Pour l'accomplir, les fournisseurs vont devoir partager les données du service, coordonner leur interaction et être chacun conscient de ce qui se fait au sein du service composé.

Afin de pouvoir interagir avec les autres services, un service doit offrir un certain nombre de point

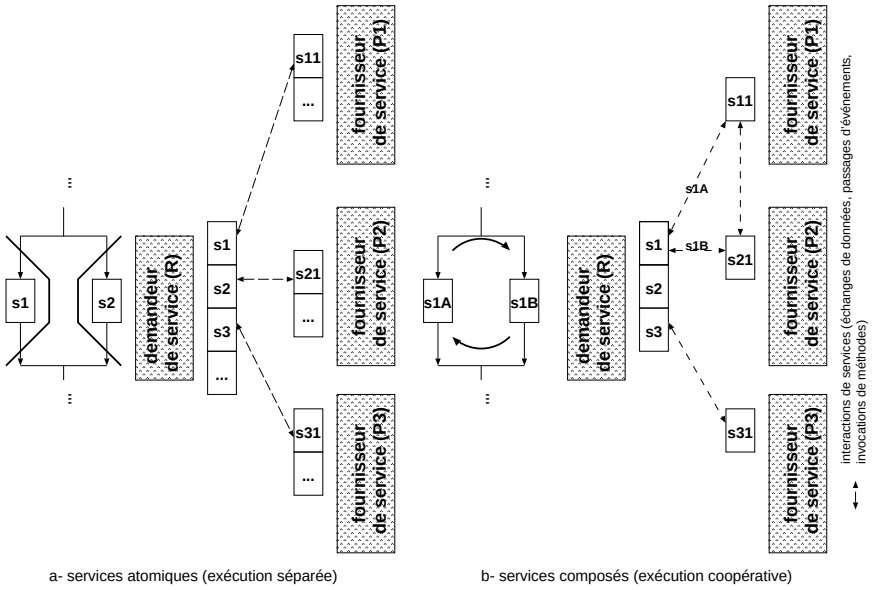

Figure 1: un exemple d'interconnexion de services

d'entrées aux ressources manipulées par le procédé qu'il encapsule (*e.g.* données, contexte d'évolution, structure du procédé, API du procédé -méthodes, événements), etc-. Cependant, un service doit assurer le respect d'un nombre de droits d'accès et d'utilisation de ses ressources. Nous parlerons alors de visibilité d'un service procédé.

Dans les systèmes de gestion de procédés existants, une instance de procédé est souvent considérée comme une boîte noire (ni les appels de méthodes de procédé ni les événements de son exécution, ni les données intermédiaires ne sont visibles pour un observateur externe). Cette approche n'est pas viable du point de vue de la coopération des procédés.

Un des objectifs de notre modèle d'interaction de services procédés est de permettre aux procédés de coopérer en rendant une partie de leurs ressources visibles de l'extérieur (*e.g.* partage de données, conscience de groupe) par le biais de l'ouverture d'accès à un ensemble de méthodes et d'événements du procédés [4, 3]. Cependant, ceci ne doit pas se faire sans contrôle de la cohérence de cette coopération.

2.2 Présentation du modèle

Notre modèle d'interaction intègre le support de partage d'information et la coordination entre services coopérants. Il peut, ainsi, décrire les besoins de gestion de rôles et la coordination de services dans la coopération. Nous présenterons notre modèle d'interaction de services procédés coopérants suivant cinq ensembles complémentaires:

(1) un ensemble de droits d'accès sur les données communes, (2) un ensemble de droits d'exécution des méthodes rendues accessibles par les services, (3) un ensemble de droits de réception des événements émis par les services, (4) un ensemble de règles de coordination basées sur les états des données communes, (5) et un ensemble de règles de coordination basées sur les états des services.

2.3 Présentation d'un exemple support

Pour illustrer notre approche, la figure 2 présente un exemple d'interconnexion des services procédés d'une entreprise de FOAD. Soit R (*Requester*), une entreprise de FOAD (Formation Ouverte et à Distance), un demandeur de service. Soient P_1 (*Provider*), une agence Web, P_2, un hébergeur de sites Web et P_3, une entreprise de collection de contenu de FOAD, trois (entre autres) fournisseurs de services. Ces entreprises forment un partenariat dont les acteurs sont des membres actifs qui opérent en proposant (respectivement demandant) des services à réaliser (respectivement externaliser) au sein du partenariat. L'entreprise R est l'acteur principal de ce partenariat, son objectif est de produire des sessions de formations ouvertes et à distance. Pour ce faire, elle fait appel aux services offerts par les fournisseurs du partenariat. La figure 2 montre comment notre modèle peut être appliqué dans l'interconnexion des procédés d'entreprises en

supportant les acteurs du partenariat dans la gestion de l'externalisation et de la composition des services procédés.

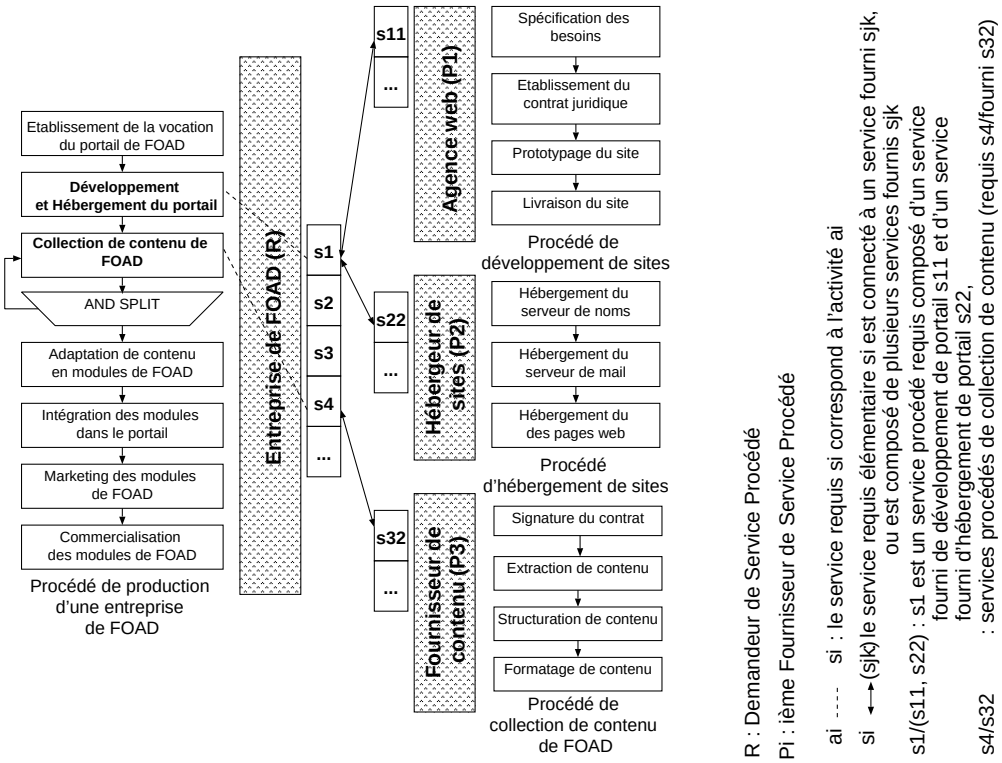

Figure 2: Un exemple d'interconnexion des services procédés d'une entreprise de FOAD

Dans cet exemple, nous considérons l'ensemble des services S, des données D, des droits d'accès aux données DR, des méthodes M, des événements E suivants :

$S = S_{frn} \cup S_{req}$: union des services fournis et des services requis suivants : $S_{frn} = \{s_{11}$: service de développement de portail, s_{22} : service d'hébergement de portail, s_{32} : service de collection de contenu$\}$ et $S_{req} = \{s_1 := (s_{11},\ s_{22})$ (service composé), $s_4 := s_{32}$ (service atomique)$\}$

$D = \{\ d_1$: contrat juridique du développement du portail, d_2 : contrat juridique de l'hébergement du portail, d_3 : charte graphique de la réalisation du portail, d_4 : prototype du portail, d_5 : cahier des charges du module de FOAD,...$\}$

$DR = \{$lecture, écriture$\}$

$M = M_{s_{11}} \cup M_{s_{22}} \cup M_{s_{32}}$: union des méthodes rendues visibles par les services s_{11}, s_{22} et s_{32} où $M_{s_{11}} = \{$getPortalDevelopmentAllocatedHumanResources(), getPortalDevelopmentStatus(), checkinDocument(), checkoutDocument(),...$\}$, $M_{s_{22}} = \{$getPortalHostingStatus(), setPortalHostingStatus(..), checkinDocument(), checkoutDocument() ...$\}$ et $M_{s_{32}} = \{$getProcessInstanceAchievedModuleDuration(), setProcessInstanceEffectiveAchievedDurationInDays(..), checkinDocument(), checkoutDocument() ...$\}$

$E = E_{s_{11}} \cup E_{s_{22}} \cup E_{s_{32}}$: union des événements rendus visibles par les services s_{11}, s_{22} et s_{32} où
$E_{s_{11}} = \{$ASK_FOR_MORE_SPECIFICATION, END_PORTAL_DEVELOPMENT, ...$\}$,
$E_{s_{22}} = \{$BEGIN_PORTAL_HOSTING, ASK_FOR_MORE_SPECIFICATION, ...$\}$ et
$E_{s_{32}} = \{$END_MODULE_COLLECTION, PRODUCED_NEW_MODULE_VERSION, ...$\}$

Dans ce qui suit, nous montrons comment notre modèle supporte les notions de **partage de données**, de **coordination**, de **cohérence** et de **conscience** entre services procédés.

3 Partage d'informations entre services procédés

La forme des interactions entre plusieurs fournisseurs dépend, entre autres, de la nature des opérations que ces fournisseurs exécutent sur les données qu'ils partagent. Le partage peut se faire par accès aux données communes aux différents services procédés et/ou par accès aux données privées à un service procédé en utilisant les méthodes qu'il rend visible. L'accès aux données communes et/ou privées doit être contrôlé pour que chacun des fournisseurs puisse jouer son rôle et assumer sa responsabilité dans la coopération.

3.1 Données partagées

Le modèle d'interaction que nous proposons ici permet aux différents fournisseurs coopérants de jouer des rôles dans le partage des données communes. Pour ce faire, ce modèle offre la possibilité de définir des contrats de partage de données. Ces contrats sont négociés entre les fournisseurs qui doivent les respecter pendant la phase d'interaction. Ils sont inspirés des modèles et des systèmes de contrôle d'accès [13, 14, 12, 31].

Un contrat de partage de données est une relation entre plusieurs services fixant leurs droits d'accès sur des données partagées. Nous exprimons cette relation sous forme d'un ensemble de contrats d'accès que chacun définit par rapport à un service sur un objet partagé. L'ensemble de contrats d'accès pour un service définissent sont rôle dans le partage de données.

$Acces(s, d, dr) \stackrel{\text{def}}{=}$ le service procédé s a le droit dr sur la donnée d. Le droit d'accès peut être positif, auquel cas il exprime des opérations permises sur la donnée partagée. Il peut aussi être négatif auquel cas il exprime des opérations refusées sur la donnée partagée. La notion de droit négatif est inspiré des travaux sur le contrôle d'accès dans les systèmes de gestion de bases de données [28].

Nous exprimons les contrats de partage de données entre les services de l'exemple présenté dans la section 2.3 comme suit : $Acces(s_1,\ d_1,\ lecture)$: signifie que s_1 (service requis composé de développement et d'hébergement du portail) peut lire la donnée d_1 (contrat juridique du développement du portail), $Acces(s_1, d_1,\ écriture)$: signifie que s_1 peut écrire la donnée d_1 et $Acces(s_{22},\ d_3,\ lecture)$: signifie que s_{22} (service fourni d'hébergement de portail) peut lire la donnée d_3 (charte graphique de la réalisation du portail).

3.2 Informations privées

La visibilité fonctionnelle, qu'un service procédé offre à un autre service procédé, concerne un ensemble de point d'accès à ses ressources (*e.g.* données, etc.). Cet ensemble est constitué, entre autres, de méthodes qu'un autre service procédé peut exécuter pour manipuler les données privées.

Au sein d'un service composé, chaque service procédé rend visible un ensemble de méthodes pour les autres services procédés coopérants. Du moment où chaque service procédé joue un rôle bien spécifique dans la coopération, les méthodes visibles sont par conséquent protégées par des droits d'exécution.

$Exec(s_i, M') \stackrel{\text{def}}{=} \forall m_{s_j} \in M', s_i$ peut exécuter la méthode visible m_{s_j} du service procédé s_j $(i \neq j)$.

Le support de la conscience de groupe permet aux différents services procédés de se coordonner de manière informelle, de situer leurs actions l'une par rapport à l'autre et de travailler de manière autonome tout en étant conscients de ce qui se passe dans le service composé. Ce support notifie aux services procédés les changements et les actions qui se produisent au sein du service composé (création d'une nouvelle version de telle ressource par un tel service procédé, etc.). Les notifications doivent être filtrées et structurées dans le sens où il faut savoir extraire les informations pertinentes dont il faut notifier et il faut déterminer les services procédés concernés par ces informations. Puisque les contrats de coopération décrivent les règles de travail au sein d'un service composé, ces règles peuvent à notre sens être utilisées comme un cadre pour l'extraction et la structuration des informations utiles pour la conscience de groupe.

Au sein d'un service composé, chaque service procédé rend visible un ensemble d'événements pour les autres services procédés coopérants. Ces événements donnent des informations sur les états des données privées du service procédé ainsi que l'état de son exécution. Du moment où chaque service procédé est intéressé et/ou concerné par des événements spécifiques, les événements visibles sont par conséquent protégés par des droits d'exécution.

$Recept(s_i, E) \stackrel{\text{def}}{=} \forall e_{s_j} \in E$, le service procédé s_i peut recevoir l'événement visible e_{s_j} du service procédé s_j $(i \neq j)$.

Dans ce qui suit, nous montrons un exemple de contrats de visibilité de méthodes que peuvent invoquer respectivement les services requis s_1 (service requis composé de développement et d'hébergement du portail) et s_4 (service requis de collection de contenu) sur les services fournis s_{11} (service fourni de

développement de portail), s_{22} (service fourni d'hébergement du portail) et s_{32} (service fourni de collection de contenu).
$Exec(s_1, M'_{s_{11}} \cup M'_{s_{22}})$ avec $M'_{s_{11}} = \{$getPortalDevelopmentAllocatedHumanResources(), getPortalDevelopmentStatus(), checkinDocument(), checkoutDocument()$\} \subset M_{s_{11}}$ et $M'_{s_{22}} = \{$getPortalHostingStatus(), checkinDocument(), checkoutDocument()$\} \subset M_{s_{22}}$ et $Exec(s_4, M'_{s_{32}})$ avec $M'_{s_{32}} = \{$getProcessInstanceAchievedModuleDuration(), checkinDocument(), checkoutDocument()$\} \subset M_{s_{32}}$

L'exemple suivant montre des contrats de visibilité des événements que peuvent percevoir respectivement les services requis s_1 et s_4 depuis les services fournis s_{11}, s_{22} et s_{32} :
$Recept(s_1, E'_{s_{11}} \cup E'_{s_{22}})$ avec $E'_{s_{11}} = \{$ASK_FOR_MORE_SPECIFICATION, END_PORTAL_DEVELOPMENT$\} \subset E_{s_{11}}$ et $E'_{s_{22}} = \{$BEGIN_PORTAL_HOSTING, ASK_FOR_MORE_SPECIFICATION$\} \subset E_{s_{22}}$ et $Recept(s_4, E'_{s_{32}})$ avec $E'_{s_{32}} = \{$END_MODULE_COLLECTION, PRODUCED_NEW_MODULE_VERSION$\} \subset E_{s_{32}}$

4 Coordination des services procédés

La coordination des services procédés consiste à interagir (échanger des données, passer des événements, invoquer des méthodes) donc à se synchroniser durant l'exécution. Pour les services atomiques, la coordination se restreint aux interactions entre les deux vues du service procédé (service requis et service fournis) (cf. figure 1-a). Dans ce cas, les services fournis s'exécutent sans interaction. Pour les services formant un service composé, ils peuvent coopérer et interagir selon des règles de coordination basées sur deux dimensions complémentaires : les états de données et les états de services procédés.

4.1 Coordination basée sur les états des services procédés

Nous définissons une règle de coordination basée sur les états des services procédés comme l'association de deux services et d'un type de coordination sur les services. Chaque type de coordination définit une contraintes sur les états de deux services. Pour illustrer des exemples de types de coordination, nous présentons tout d'abord les notions d'état, de séquence d'états et de séquence de vues d'un service.

4.1.1 État et séquence d'états d'un service procédé

Tout service procédé est supposé avoir un état. Un état est la situation du service ou encore l'ensemble des circonstances le décrivant à un instant donné.

On appelle une séquence d'états d'un service s, que l'on note E^s, l'ensemble complet et ordonné des états par lesquels passe s. On peut définir E^s comme $E^s = \{e^s_0, e^s_1, \dots\}$, où e^s_0 est l'état initial du service et e^s_1, e^s_2, ... sont ses états successifs. Un état d'un service s_1 est vu par un service s_2 si ce dernier a exécuté une méthode de s_1 dont la valeur de retour est un attribut reflétant l'état de s_1 ou, s'il a reçu un évènement émis par s_1 reflétant son état.

Pour un service s_1, on appelle séquence de vues d'un service s_2, que l'on note $E^{s_1}_{s_2}$, le sous-ensemble ordonné de E^{s_1} (l'ensemble des états de s_1) vus par s_2.

Du moment où l'on considère qu'un service possède un début et une fin, l'ensemble de méthodes qu'un service peut exécuter et l'ensemble des évènements qu'il peut recevoir sont finis. De ce fait, l'ensemble $E^{s_1}_{s_2}$ est fini. Nous notons la séquence d'états du service s_1 vus par le service s_2 par $E^{s_1}_{s_2} = \{e^{s_1}_{s_2,0}, e^{s_1}_{s_2,1}, \dots, e^{s_1}_{s_2,n-1}\}$ où n dénote le nombre d'états de s_1 vus par s_2. $e^{s_1}_{s_2,0}$ est l'état initial de s_1 vu par s_2. Il s'agit du premier état de s_1 vu par s_2 après son début (*e.g.* $e^{s_1}_{s_2,n-1}$ est le dernier état de s_1 vu par s_2 avant sa fin). Notons que $e^{s_1}_{s_2,0}$ peut ne pas être l'état initial du service s_2. Notons en plus qu'il peut y avoir un ou plusieurs états de s_1 non vus par s_2 et qui ont été créés entre deux états vus par s_2.

On appelle séquence de vues d'un service s tous ses états ainsi que tous les états qu'il peut voir des autres services. Nous notons la séquence vues d'un service s par $V^s = E^s \cup_{s_1 \in S} E^{s_1}_s$. Cet ensemble représente les états des services dont le service s est conscient.

Nous définissons une règle de coordination basée sur les états des services procédés comme une contrainte reliant deux services procédés et un type de coordination. Formellement nous notons une telle règle par *SCoor*(*state based Service-Coordination* -coordination basée sur les états des services-), $SCoor(s_1, s_2, coortype)$ où s_1 et s_2 sont deux services procédés appartenant à l'ensemble de services procédés S et *coortype* est un type de coordination sur les états de services procédés appartenant à l'ensemble *SCoorT* (*state based Service-Coordination Type* -Type de coordination basée sur les états des services-).

4.1.2 Types de coordination sur les états des services

Les types de coordination que nous présontons ici sont inspirés de plusieurs travaux. Nous citons entre autres ceux réalisés dans les domaines de modèles transactionnels [11] ainsi que ceux présentés pour supporter les paradigmes d'interconnexion de workflows [35]. Pour présenter ces types, nous utilisons les notations suivantes. Soit s un service composé des deux services s_i et s_j. Soient E^{s_i} et E^{s_j} leur séquence d'états respectives. Soit E (la séquence d'états de s) l'union de E^{s_i} et E^{s_j}. E est muni d'une relation d'ordre partiel noté $\rightarrow$ et qui désigne la précédence des états dans E. Ci-dessous nous présentons les types de coordination basés sur les états de services :

• Coordination émanant des modèles transactionnels : Nous définissons une règle de coordination basée sur les états des services procédés comme une contrainte de dépendance sur les événements significatifs de ces services (Enacted, Prehempted, Committed, Aborted, etc.). Afin d'exprimer ces dépendances sur les états des services, nous avons choisi l'approche utilisée dans [11], sachant que l'on peut considérer qu'un service comme étant une transaction avec un début (enact), une fin (commit, abort) et un cycle de vie.

– **Dépendance de séquence :** Une dépendance de séquence DS entre deux services, peut s'exprimer de la manière suivante : $SCoor(s_2,\ s_1,\ DS)$ et signifie que le service s_2 ne peut pas s'exécuter avant que s_1 ne soit validé, plus formellement : $SCoor(s_2, s_1, DS) \stackrel{\text{def}}{=} \mathsf{Enacted}_{s_2} \in E \Rightarrow \mathsf{Committed}_{s_1} \rightarrow \mathsf{Enacted}_{s_2}$

– **Dépendance de validation :** Une dépendance de validation DV entre deux services, peut s'exprimer de la manière suivante : $SCoor(s_2,\ s_1,\ DV)$ et signifie que si s_1 et s_2 sont validés alors la validation de s_1 précède celle de s_2, plus formellement :

$SCoor(s_2,\ s_1,\ DV) \stackrel{\text{def}}{=} \mathsf{Committed}_{s_2} \in E \Rightarrow (\mathsf{Committed}_{s_1} \in E \Rightarrow (\mathsf{Committed}_{s_1} \rightarrow \mathsf{Committed}_{s_2}))$

• Coordination émanant des travaux de standardisation de la WfMC [35]. L'interface 4 de standardisation de la WfMC répond au problème de l'interconnexion des procédés par des paradigmes d'interopérabilité se basant essentiellement sur la communication par invocations de méthodes entre objets définissant les procédés :

– **Dépendance de chaînage** : Une dépendance de chaînage DC peut s'exprimer de la manière suivante : $SCoor(s_2,\ s_1,\ DC)$ et signifie qu'un service s_1 active un service s_2 à distance sans attendre sa complétion, plus formellement : $SCoor(s_2,\ s_1,\ DC) \stackrel{\text{def}}{=} \mathsf{Enacted}_{s_2} \in E \Rightarrow \mathsf{Enacted}_{s_1} \rightarrow \mathsf{Enacted}_{s_2}$

– **Dépendance d'imbrication** : Une dépendance d'imbrication DI peut s'exprimer de la manière suivante : $SCoor(s_2,\ s_1,\ DI)$ et signifie qu'un service s_1 active un service à distance et attend sa complétion, plus formellement : $SCoor(s_2, s_1, DI) \stackrel{\text{def}}{=}$

$(\mathsf{Enacted}_{s_2} \in E \Rightarrow \mathsf{Enacted}_{s_1} \rightarrow \mathsf{Enacted}_{s_2}) \wedge (\mathsf{Committed}_{s_1} \in E \Rightarrow \mathsf{Committed}_{s_2} \rightarrow \mathsf{Committed}_{s_1})$

– **Dépendance de synchronisation parallèle** : Une dépendance de synchronisation parallèle DSP peut s'exprimer de la manière suivante : $SCoor(s_2,\ s_1,\ DSP)$ et signifie que deux services s_1 et s_2 *"décident"* d'un point de rendez-vous où ils échangent des informations, puis chacun des services continue normalement son exécution. (voir la section 4.2.2 pour une expression formelle de cette dépendance).

Ces types de coordination basée sur les états des services ont été donnés à titre d'exemples, d'autres types de coordination définis de la même façon n'ont pas été détaillés dans cet article.

L'ensemble de contrats suivant montre un exemple de contrats de coordination basée sur les états des services procédés de la section 2.3 :
$SCoor(s_{11},\ s_1,\ DI)$: signifie que l'exécution de s_{11} (service fourni de développement de portail) est imbriquée dans l'exécution de s_1 (service requis composé de développement et d'hébergement du portail),
$SCoor(s_{22},\ s_1,\ DI)$: signifie que l'exécution de s_{22} (service fourni d'hébergement de portail) est imbriquée dans l'exécution de s_1,
et $SCoor(s_{22},\ s_{11},\ DSP)$: signifie que l'exécution de s_{11} et s_{22} est parallèle synchronisée.

4.2 Coordination basée sur les états des données

Nous définissons une règle de coordination, basée sur les états des données des services procédés, comme l'association de deux services, d'une donnée et d'un type de coordination. Afin d'exprimer ces règles de coordination sur les données partagées par les services, nous avons choisi l'approche utilisée dans [32], sachant que l'on peut considérer qu'un service n'est qu'une activité particulière au sens de [32] qui accède à des données partagées avec d'autres activités.

4.2.1 État et séquence d'états d'une donnée

Toute donnée est supposée avoir un état. Un état est la situation de la donnée ou encore l'ensemble de circonstances décrivant la donnée à un instant t. Par exemple, l'état de la donnée "*compte en banque*" inclut le solde du compte, l'état de la donnée "*horloge*" inclut le temps actuel.

On appelle une séquence d'états d'une donnée d, et l'on note E^d, l'ensemble complet et ordonné des états de d. On peut définir E^d comme $E^d = \{e^d_0, e^d_1, \ldots\}$, où e^d_0 est l'état initial de la donnée et e^d_1, e^d_2, ...sont ses états successifs. Il faut noter cependant qu'aucune hypothèse n'est faite sur la granularité du changement d'état. Celui-ci peut varier d'un simple caractère dans un document textuel à une version complète d'un fichier source d'un programme. Un état de d est vu par un service s si celui-ci a exécuté une opération sur d dont la valeur de retour est un attribut reflétant l'état de d ou s'il a reçu un évènement reflétant l'état de la donnée. "lire" et "écrire" un fichier sont des opérations qui permettent de voir l'état du fichier. Exécuter la commande UNIX "ls" est une opération qui permet de voir l'état du répertoire. Par contre, "écrire" le répertoire (*i.e.* créer, supprimer ou déplacer un fichier de ce répertoire) est une opération qui ne permet pas de voir l'état du répertoire.

Pour une donnée d, on appelle séquence de vues d'un service s, que l'on note E^d_s, le sous-ensemble ordonné de E^d (l'ensemble des états de d) composé de tous les états de d vus par s.

Du moment que l'on considère qu'un service possède un début et une fin, l'ensemble des opérations qu'un service peut exécuter est fini. De ce fait, l'ensemble E^d_s est fini (à la différence de E^d qui peut être infini). Nous notons $E^d_s = \{e^d_{s,0}, e^d_{s,1}, \ldots, e^d_{s,n-1}\}$ la séquence d'états de d vus par le service s où n dénote le nombre d'états de d vus par s et $e^d_{s,0}$ est l'état initial de d vu par s. Cet état initial est le premier état de d vu par s après son début. $e^d_{s,n-1}$ est le dernier état de d vu par s avant sa fin. Notons que $e^d_{s,0}$ peut ne pas être l'état initial de la donnée d. Notons en plus qu'il peut y avoir un ou plusieurs états de d non vus par s et qui ont été créés entre deux états vus par s.

Pour une donnée, la coordination entre deux services procédés exprime le fait qu'il existe des contraintes sur les états des données communes vus par l'un et/ou l'autre. Formellement nous notons une règle de coordination basée sur les états de données par $DCoor(s_1, s_2, d, coortype)$ (*DCoor*: *Data state based service-Coordination* -Coordination basée sur les états des données de services-), où s_1 et s_2 sont deux services procédés appartenant à l'ensemble de services procédés S, d est une donnée appartenant à l'ensemble de données D et *coortype* est un type de coordination sur les états de services procédés appartenant à l'ensemble $DCoorT$ (*DCoorT*: Data state based service-Coordination Type (Type de Coordination basée sur les états des données de services)). Nous distinguons plusieurs types de coordination que nous présentons ci-dessous.

4.2.2 Type de coordination sur les données

Dans ce qui suit s_1 et s_2 dénoteront deux services, d une donnée partagée par ces deux services, $E^d_{s_1} = \{e^d_{s_1,0}, e^d_{s_1,1}, \ldots, e^d_{s_1,n-1}\}$ (respectivement $E^d_{s_2} = \{e^d_{s_2,0}, e^d_{s_2,1}, \ldots, e^d_{s_2,m-1}\}$) la séquence d'états de d vus par le service s_1 (respectivement le service s_2) et n (respectivement m) le nombre d'états de d vus par le service s_1 (respectivement le service s_2).

• **Coordination de service :** Pour une donnée d, le service s_1 est coordonné avec le service s_2 sur la donnée d quand la séquence d'états de d vus par s_1 est inclue dans la séquence d'états de d vus par s_2. Nous notons *CoorSv* (Coordination de Service) ce type de coordination. Le prédicat utilisé pour exprimer cette coordination est défini comme suit : $DCoor(s_1, s_2, d, CoorSv) \stackrel{\text{def}}{=} E^d_{s_1} \subset E^d_{s_2}$

• **Coordination d'états :** Pour une donnée d, on dit que deux services s_1 et s_2 ont une coordination d'états, s'il existe un état $e^o_{s_1,i}$ vu par s_1 et un état $e^o_{s_2,j}$ vu par s_2 qui sont identiques. Nous notons *CoorE* (Coordination d'états) ce type de coordination. Le prédicat utilisé pour exprimer cette coordination est défini comme suit : $DCoor(s_1, s_2, d, CoorE) \stackrel{\text{def}}{=} \exists i \in \{0..n-1\},\ \exists j \in \{0..m-1\}\ e^d_{s_1,i} = e^d_{s_2,j}$

où n dénote le nombre d'états de d vus par s_1 et m dénote le nombre d'états de d vus par s_2.

Nous pouvons utiliser ce type de coordination pour exprimer la dépendance de synchronisation parallèle de service présentée informellement ci-dessus et qui signifie que deux services s_i et s_j *"décident"* d'un point de rendez-vous où ils échangent des informations, puis chacun des services continue normalement son exécution.

$SCoor(s_1, s_2, DSP) = \forall d \in DE\ DCoor(s_1, s_2, d, CoorE)$ où DE est l'ensemble des données échangées entre les services s_1 et s_2 pendant leur point de rendez-vous.

Le type de coordination d'état peut être utilisé pour spécifier le cas général de deux services qui sont coordonnés sur un état arbitraire. Cependant, il y a plusieurs situations dans lesquelles ces coordinations

se produisent au début ou à la fin de l'un ou des deux services coordonnés. Étant donné deux services et une donnée, nous exprimons ci-dessous, les types de coordination correspondant aux situations suivantes :

– l'état final vu par un service est coordonné avec l'autre service,
– les états finaux vus par les deux services sont coordonnés,
– l'état initial vu par un service est coordonné avec l'autre service,
– les états initiaux vus par les deux services sont coordonnés,
– l'état final vu par un service est coordonné avec l'état initial vu par l'autre service,
– tous les états vus par les deux services sont coordonnés.

Dans ce qui suit, nous définissons formellement ces différents types de coordination.

• **Coordination d'un état final :** Pour un objet donné d, l'état final vu par le service s_1 est coordonné au service s_2 si cet état appartient à la séquence d'états de d vus par s_2 (*i.e.* l'état final vu par s_1 est coordonné à un état de d vu par s_2). Nous notons *CoorEF* (Coordination d'un État Final) ce type de coordination. Le prédicat utilisé pour exprimer cette règle est défini comme suit :
$DCoor(s_1, s_2, d, CoorEF) \stackrel{\text{def}}{=} e^d_{s_1,n-1} \in E^d_{s_2}$, où n dénote le nombre d'états de d vus par s_1.

• **Coordination des états finaux :** Pour une donnée d, les services s_1 et s_2 sont coordonnés sur leurs états finaux si les deux services observent le même état final de d. Nous notons ce type de coordination par *CoorEFX* (Coordination des états finals (Etats FinauX)). Le prédicat utilisé pour exprimer cette règle est défini comme suit : $DCoor(s_1, s_2, d, CoorEFX) \stackrel{\text{def}}{=} e^d_{s_1,n-1} = e^d_{s_2,m-1}$
où n dénote le nombre d'états de d vus par s_1 et m dénote le nombre d'états de d vus par s_2.

• **Coordination d'un état initial :** Pour une donnée d, l'état initial vu par un service s_1 est coordonné à un service s_2 si cet état appartient à la séquence d'états de d vus par s_2 (*i.e.* l'état initial vu par s_1 est coordonné à un état de d vu par s_2). Nous notons ce type de coordination par *CoorEI* (Coordination d'un État Initial). Le prédicat utilisé pour exprimer cette règle est défini comme suit :
$DCoor(s_1, s_2, d, CoorEI) \stackrel{\text{def}}{=} e^d_{s_1,0} \in E^d_{s_2}$

• **Coordination des états initiaux :** Pour une donnée d, les services s_1 et s_2 sont coordonnés sur leurs états initiaux pour une donnée d si les deux services observent le même état initial de d. Nous notons ce type de coordination par *CoorEIX* (Coordination des États InitiauX). Le prédicat utilisé pour exprimer cette règle est défini comme suit : $Coor(s_1, s_2, d, CoorEIX) \stackrel{\text{def}}{=} e^d_{s_1,0} = e^d_{s_2,0}$

• **Coordination séquentielle :** Pour une donnée d, deux services s_1 et s_2 sont séquentiellement coordonnés si l'état final vu par l'un est l'état initial vu par l'autre. Nous notons *CoorSq* (Coordination Séquentielle) ce type de coordination. Le prédicat utilisé pour exprimer cette règle est défini comme suit :
$DCoor(s_1, s_2, d, CoorSq) \stackrel{\text{def}}{=} e^d_{s_1,n-1} = e^d_{s_2,0}$

• **Coordination totale :** Pour une donnée d, deux services s_1 et s_2 sont totalement coordonnés s'ils observent exactement les mêmes états de la donnée dans le même ordre. Les états de la donnée sont produits par l'un, l'autre ou un tiers service. Nous notons *CoorTot* (Coordination Totale) ce type de coordination. Le prédicat utilisé pour exprimer cette règle est défini comme suit :
$DCoor(s_1, s_2, d, CoorTot) \stackrel{\text{def}}{=} n = m \wedge \forall$ i $\in$ {0..n-1}, $e^d_{s_1,i} = e^d_{s_2,i}$
où n dénote le nombre d'états de d vus par s_1 et m dénote le nombre d'états de d vus par s_2.

L'ensemble de contrats suivant montre un exemple de contrats de coordination basée sur les états des données de services procédés de la section 2.3 :
$DCoor(s_1,\ s_{11},\ d_1,\ CoorTot)$: signifie que les services s_1 (service requis composé de développement et d'hébergement du portail) et s_{11} (service fourni de développement de portail) sont en coordination totale sur la donnée d_1 (contrat juridique du développement du portail),
et $DCoor(s_{11},\ s_{22},\ d_3,\ CoorEIX)$: signifie que les services s_{11} et s_{22} (service fourni d'hébergement de portail) sont coordonnés sur leurs états initiaux pour la donnée d_3 (charte graphique de la réalisation du portail).

5 Modèle d'interaction de services procédés

Dans cette section, nous proposons un modèle d'interaction qui intègre le support de partage d'information et la coordination entre services coopérants. Ainsi, ce modèle peut décrire les besoins de gestion de rôle et la coordination de services dans la coopération.

Formellement le modèle d'interaction de services est représenté par le triplet (S, D, CL) où S est un ensemble de services, D est un ensemble de données partagées et CL (Contract List) est un ensemble de contrats de coopération. Chaque contrat relie un ensemble de services participants à la coopération par :

• *DAL* (*Data Access List*), un ensemble de droits d'accès sur les données communes.

• *MAL* (*Method Access List*), un ensemble de droits d'exécution des méthodes rendues accessibles par les services.

• *EAL* (*Event Access List*), un ensemble de droits de réception des événements émis par les services.

• *DCL* (*Data Coordination List*), un ensemble de règles de coordination basées sur les états des données communes.

• *SCL* (*Service Coordination List*), un ensemble de règles de coordination basées sur les états des services.

Un exemple de contrat d'interaction peut être le triplet (S, D, CL) tels que :

$S = S_{frn} \cup S_{req}$: union des services fournis et des services requis suivants : $S_{frn} = \{s_{11}$: service de développement de portail, s_{22} : service d'hébergement de portail, s_{32} : service de collection de contenu$\}$ et $S_{req} = \{s_1 := (s_{11},\ s_{22})$ (service composé), $s_4 := s_{32}$ (service atomique)$\}$

$D = \{\ d_1$: contrat juridique du développement du portail, d_2 : contrat juridique de l'hébergement du portail, d_3 : charte graphique de la réalisation du portail, d_4 : prototype du portail, d_5 : cahier des charges du module de FOAD,...$\}$

et CL :

$DAL = \{Acces(s_1,\ d_1,\ lecture),\ Acces(s_1, d_1,\ écriture),\ Acces(s_{22},\ d_3,\ lecture),\ ...\}$,
$MAL = \{Exec(s_1,\ M'_{s_{11}} \cup M'_{s_{22}}), Exec(s_4,\ M'_{s_{32}}),\ ...\}$,
$EAL = \{Recept(s_1,\ E'_{s_{11}} \cup E'_{s_{22}}), Recept(s_4,\ E'_{s_{32}}),\ ...\}$,
$DCL = \{DCoor(s_1,\ s_{11},\ d_1,\ CoorTot),\ DCoor(s_{11},\ s_{22},\ d_3,\ CoorEIX),\ ...\}$ et
$SCL = \{SCoor(s_{11},\ s_1,\ DI),\ SCoor(s_{22},\ s_1,\ DI),\ SCoor(s_{22},\ s_{11},\ DSP),\ ...\}$.

avec

$M'_{s_{11}}$ = {getPortalDevelopmentAllocatedHumanResources(), getPortalDevelopmentStatus(), checkinDocument(), checkoutDocument()},
$M'_{s_{22}}$ = {getPortalHostingStatus(), checkinDocument(), checkoutDocument()},
$M'_{s_{32}}$ = {getProcessInstanceAchievedModuleDuration(), checkinDocument(), checkoutDocument()},
$E'_{s_{11}}$ = {ASK_FOR_MORE_SPECIFICATION, END_PORTAL_DEVELOPMENT},
$E'_{s_{22}}$ = {BEGIN_PORTAL_HOSTING, ASK_FOR_MORE_SPECIFICATION} et
$E'_{s_{32}}$ = {END_MODULE_COLLECTION, PRODUCED_NEW_MODULE_VERSION}

Le modèle d'interaction définit des contrats de coopération. Ces contrats, modélisent les règles de travail entre les services procédés. Nous nous proposons ici de vérifier la cohérence des règles définies dans les contrats de coopérations. Cette vérification nous permet d'éviter la contradiction entre les contrats agréés.

Les règles de coopération définissent explicitement des opérations permises, des opérations refusées et des opérations obligatoires. Les règles de partage d'informations, qu'on peut définir au sein d'un contrat de coopération, attribuent des droits (positifs ou négatifs) sur des données partagées, sur les méthodes visibles et/ou sur des événements visibles. Chaque droit d'accès positif autorise le service qui le détient à exécuter un ensemble d'opérations sur les données partagées ou privées ou de recevoir des événements reflétant les états des données ou des autres services. A titre d'exemple, le droit d'écrire un répertoire autorise de créer, déplacer ou supprimer une entrée du répertoire tandis que le droit d'écrire un objet fichier permet d'écrire ou de lire le contenu de ce fichier.

Les règles de partage d'informations qui attribue à un service procédé un droit d'accès négatif lui refuse l'exécution d'un ensemble d'opérations sur les données, l'exécution de certaines méthodes ou la réception de certains événements.

Pour respecter certaines règles de coordination, les services procédés dans un contrat de coopération doivent voir des états des données partagées. En effet, s'il existe une règle de coordination de type $DCoor(s_1, s_2, d, coortype)$ alors s doit voir au moins une fois la donnée d. Pour ce faire, il doit exécuter au moins une opération lui permettant de voir l'état de d. S'il existe une règle de coordination de type $DCoor(s_1, s_2, d, coortype)$ alors pour respecter la politique de coopération, s_2 doit voir au moins un état de d si le type de coordination $coortype$ appartient à $\{CoorEIX,\ CoorEFX,\ CoorTot,\ CoorSq\}$.

Pour un service procédé s participant à un contrat de coopération on peut définir par conséquent un ensemble d'opérations que s peut exécuter, un ensemble d'opérations que s ne peut pas exécuter et

un ensemble d'opérations que s doit exécuter. Un service s peut respecter un contrat de coopération si toutes les opérations obligatoires sont permises et toutes les opérations obligatoires ne sont pas refusées. Un contrat de coopération est, alors, cohérent s'il peut être respecté par tous ses services.

6 Conclusion

Nous avons montré dans cet article comment nous pouvions interconnecter des procédés par le biais des interactions de services procédés. Notre modèle d'interaction permet à différents services de communiquer et de se coordonner pour réaliser une coopération inter-procédés et donc inter-organisationnelle. En effet, notre modèle d'interaction intègre le support de partage d'information et la coordination entre services coopérants. Il peut, ainsi, décrire les besoins de gestion de rôle et la coordination de services dans la coopération suivant un ensemble de droits d'accès sur les données communes, un ensemble de droits d'exécution des méthodes rendues accessibles par les services, un ensemble de droits de réception des événements émis par les services, un ensemble de règles de coordination basées sur les états des données communes, et un ensemble de règles de coordination basées sur les états des services.

Même si les détails d'implantation ne sont pas l'objectif de cet article, nous pouvons signaler que nous développons notre modèle d'interaction de services procédés au sein de notre environnement coopératif d'échange de services procédés *DISCOBOLE* (*DIStributed CO-operation and Business prOcess on LinE*), la nouvelle génération de *DisCOO* [23, 24]. *DISCOBOLE* est un environnement de travail coopératif qui permet à des acteurs distribués d'échanger des documents selon des règles de coopération et d'interconnecter leurs procédés (workflows) selon le paradigme d'échanges de services. En effet, *DISCOBOLE* permet aux acteurs de coopérer en interconnectant leurs procédés de production (workflows). Ces procédés peuvent être gérés au sein d'outils de workflows hétérogènes. Pour coopérer, chaque acteur décrit ses besoins et compétences en termes de services procédés. Ces derniers sont publiés dans des référentiels partagés qui sont interrogeables selon des algorithmes de sélection dynamique (*matching*). Après la recherche des services procédés correspondant à leurs besoins (respectivement compétences), les acteurs négocient un contrat d'interconnexion liant leurs services requis et fournis, avant de commencer la coopération effective.

Un service procédé est une entité qui encapsule un procédé pour ne rendre visible depuis l'extérieur que les éléments nécessaires à son interconnexion. En effet, on peut voir la structure de service procédé comme la réalisation d'un patron de conception (*design pattern)* [15] pour la coopération. Un service procédé s'apparente aux patrons de conception "proxy" et "adaptateur" de procédé. En fait, si le proxy limite l'accès aux ressources du procédé (droits d'accès aux données $Acces(s,\ d,\ dr)$, visibilité de méthodes $Exec(s,\ M)$, visibilité d'événements $Recept(s,\ E)$), l'adaptateur procure une nouvelle interface au procédé qu'il adapte (possibilité d'échange de données, possibilité de se coordonner, possibilité de contrôler la cohérence des interactions). *DISCOBOLE* est développé en Java sur un bus CORBA.

Cette présentation se voulait synthétique et n'a pas englobé l'ensemble de nos travaux. En particulier, nous n'avons pas abordé un aspect qui nous semble important, mais qui a fait l'objet d'autres publications et que nous n'avons pas voulu développer dans cet article, à savoir la dynamicité des interconnexions. Au lieu d'une interconnexion statique et planifiée préalablement, l'interconnexion flexible et dynamique de deux procédés désigne le fait que tout paramètre nécessaire à la coopération entre ces deux procédés se décide à l'exécution et rien n'est planifié à l'avance (*e.g.* choix des partenaires, des services à interconnecter, des protocoles de communication à adopter (*e.g.* SOAP/HTTP, CORBA/IIOP, etc.), du contrat d'interconnexion des services, du contrat de partage de données, etc.). Plus généralement, on crée, on fournit, on demande un service, on négocie le contrat de visibilité d'un service, on lie contractuellement deux services, on active un service et on échange des données entre services dynamiquement. En effet, on ne peut pas prévoir, préalablement à une interconnexion de procédés, les possibilités de coopération dont on disposera plus tard. Pour résoudre ces problèmes de co-décision d'interconnexion entre les procédés, des mécanismes de négociation ont été créés et utilisés [22]. Cette application de notre modèle de négociation à l'interconnexion de procédés n'est qu'une des voies de recherche que nous menons actuellement autour des services procédés. D'autres problèmes comme la recherche et le *matching* de services ont été traités et décrits dans [3, 4]. D'autres problèmes concernent l'application de notre modèle à des "services Web" et donc l'adaptation de nos critères de protection et de coordination au monde Internet.

Nous avons montré par cet article l'intérêt de l'approche orientée service et interaction de services pour l'interconnexion de procédé et indiqué par cette conclusion "ouverte" les voies prometteuses à suivre dans la même optique.

Bibliographie

[1] G. Alonso, D. Agrawal, A. El Abbadi, and C. Mohan. Functionality and limitations of current workflow management systems. *IEEE Expert - Special Issue on Cooperative Information Systems*, 12(5), 1997.

[2] G. Alonso and C. Mohan. Workflow management systems: The next generation of distributed processing tools. In S. Jajodia and L. Kerschberg, editors, *Advanced Transaction Models and Architectures*, pages 35–62. Kluwer Academic Publishers, 1997.

[3] K. Baïna, K. Benali, and C. Godart. Les services procédés, une solution pour l'interconnexion des procédés d'entreprises. *Ingénierie des Systèmes d'Information, Numéro spécial Interopérabilité des Systèmes d'Information (ISI'01)*, 6(3):145–181, 2001.

[4] K. Baïna, K. Benali, and C. Godart. A process service model for dynamic enterprise process interconnection. In *9th Int. Conf. on Cooperative Information Systems, In Cooperation with VLDB 2001 (CoopIS'01)*, volume 2172 of *LNCS*, pages 239–254, Trento, Italy, September 5-7 2001. Springer-Verlag.

[5] A. P. Barros and A. H. M. Ter Hofstede. Modelling extensions for concurrent workflow coordination. In *4th IFCIS Int. Conf. on Cooperative Information Systems (CoopIS'99), IEEE Computer Society*, pages 336–347, Edinburgh, Scotland, September 1999.

[6] B. Benatallah, B. Medjahed, A. Boughettaya, A. Elmagarmid, and J. Beard. Composing and maintaining web-based virtual enterprises. In *First Workshop on Technologies for E-Services, In cooperation with VLDB 2000 (TES'00)*, Cairo, Egypt, September 14-15 2000.

[7] G. A. Bolcer and G. Kaiser. Swap: Leveraging the web to manage workflow (swap). In *IEEE Internet Computing*. IEEE Computer Society, http://computer.org/internet/, January-February 1999.

[8] Y. Breitbart, A. Deacon, H-J. Schek, A. Sheth, and G. Weikum. Merging application-centric and data-centric approaches to support transaction-oriented multi-system workflows. In *SIGMOD Record*, volume 22, pages 23–30, 1993.

[9] F. Casati and A. Discenza. Supporting workflow cooperation within and across organisations. In *15th ACM Symposium on Applied Computing (SAC'00)*, pages 19–21, Como, Italy, March 2000.

[10] F. Casati, S. Ilnicki, L. J. Jin, and M. C. Shan. eflow: an open, flexible, and configurable approach to service composition. In *Second International Workshop on Advance Issues of E-Commerce and Web-Based Information Systems (WECWIS'00)*, pages 125–132, Milpitas, California, June 8-9 2000.

[11] P. K. Chrysanthis and K. Ramamritham. ACTA: The SAGA continues. In *Database Transaction Models for Advanced Applications*, pages 349–397. 1992.

[12] P. Dewan and H-H. Shen. Flexible meta access-control for collaborative applications. In *Proceedings of ACM Conference on Computer-Supported Cooperative Work*, Primitives for Building Flexibile Groupware Systems, pages 247–256. ACM Press, 1998.

[13] W. K. Edwards. Policies and roles in collaborative applications. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pages 11–20, Boston, Massachusetts, USA, November 1996. ACM Press.

[14] C.A. Ellis, S. Gibbs, and G. Rein. Groupware: some issues and Experiences. In *Communications of the ACM, 34 (1)*, pages 38–58, 1994.

[15] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns, Elements of Reusable Object-Oriented Software*. Addition-Wesley, Massachusetts, 1994.

[16] D. Georgakopoulos, M. F. Hornick, F. Manola, M. L. Brodie, S. Heiler, F. Nayeri, and B. Hurwitz. An extended transaction environment for workflows in distributed object computing. In *IEEE Data Engineering Bulletin*, volume 16, pages 24–27, June 1993.

[17] D. Georgakopoulos, H. Schuster, A. Cichocki, and D. Baker. Managing process and service fusion in virtual enterprises. *Information Systems, Special Issue on Information Systems Support for Electronic Commerce*, 1999.

[18] P. Grefen, K. Aberer, Y. Hoffner, and H. Ludwig. Crossflow: cross-organisational workflow management in dynamic virtual enterprises. In *International Journal of Computer Systems, Science and Engineering (IJCSSE'00)*, pages 277–290, 2000.

[19] C. Hagen and G. Alonso. Beyond the black box: Event-based inter-process communication in process support systems. In *19th International Conference on Distributed Computing Systems (ICDCS 99)*, Austin, Texas, USA, May/June 1999.

[20] J. Klingemann, J. Wasch, and K. Aberer. Adaptative outsourcing in cross organizational workflows. In *11th International Conf. On advanced Information Systems Engineering (CaiSE'99)*, Heidelberg, Germany, June 14-18 1999.

[21] L. Kutvonen. *Trading services in open distributed environments.* PhD thesis, Department of Computer Science, University of Helsinki, Finland, 1998.

[22] M. Munier, K. Baïna, and K. Benali. A negotiation model for CSCW. In O. Etzion and P. Scheuermann, editors, *5th IFCIS Int. Conf. on Cooperative Information Systems, In Cooperation with VLDB 2000 (CoopIS'00)*, volume 1901 of *LNCS*, pages 224–235, Eilat, Israel, September 6-8 2000. Springer-Verlag.

[23] M. Munier, K. Benali, and C. Godart. DisCOO, a really distributed system for cooperation. *Networking and Information Systems Journal*, 2(5-6):605–637, 1999.

[24] M. Munier, K. Benali, and C. Godart. Un système coopératif basé sur les transactions. In *INFormatique des ORganisations et Systèmes d'Information et de Décision, XIXème Congrès INFORSID, 29 Mai - 01 Juin, Genève, Suisse*, pages 163–177, May 2001.

[25] OMG. *Autonomous Decentralized Service System (ADSS) Domain Special Interest Group Whitepaper ver 1.0 (ads/97-12-01).* OMG (Object Management Group), www.omg.org, December 5 1997.

[26] OMG. *Workflow Management Facility Convenience Document combining dtc/99-07-05 dtc/2000-02-03 (WF RTF 1.3 Report).* OMG (Object Management Group), ww.omg.org, February 14 2000.

[27] G. Piccinelli. Distributed workflow management: The team model. In *3rd IFCIS Int. Conf. on Cooperative Information Systems (CoopIS'98), Sponsored by IFCIS, The Intn'l Foundation on Cooperative Information Systems*, pages 292–299, New York City, New York, USA, August 20-22 1998. IEEE-CS Press.

[28] F. Rabitti, E. Bertino, W. Kim, and D. Woelk. A model of authorization for next-generation database systems. *ACM Transactions on Database Systems*, 16(1):88–131, March 1991.

[29] M. Reichert and P. Dadam. A framework for dynamic changes in workflow management systems. In *8th International Workshop on Database and Expert Systems Applications (DEXA'97)*, Toulouse, France, September 1-2 1997.

[30] M. Rusinkiewicz and A. Sheth. Specification and execution of transactional workflows. In Won Kim, editor, *Modern Database Systems, The Object Model Interoperability and beyond*, pages 592–620. Addison Wesley, ACM Press, 1995.

[31] H. Shen and P. Dewan. Access control for collaborative environments. In *Proceedings of the Conference on Computer Supported Cooperative Work*, Toronto, Canada, November 1992. ACM Press.

[32] Samir Tata. *Outils pour la description et la mise en œuvre des interactions coopératives dans les équipes virtuelles.* Thèse en informatique, Université Henry Poincaré - Nancy I, Loria, Décembre 2000.

[33] W-J. van den Heuvel, J. Hang, and M. P. Papazoglou. Service Representation, Discovery, and Composition for E-marketplaces. In *9th Int. Conf. on Cooperative Information Systems, In Cooperation with VLDB 2001 (CoopIS'01)*, volume 2172 of *LNCS*, pages 270–284, Trento, Italy, September 5-7 2001.

[34] W.M.P. van der Aalst, P. Barthelmess, C.A. Ellis, and J. Wainer. Workflow modeling using proclets. In O. Etzion and Peter Scheuermann, editors, *5th IFCIS Int. Conf. on Cooperative Information Systems (CoopIS'00)*, volume 1901 of *LNCS*, pages 198–209, Eilat, Israel, September 6-8 2000.

[35] WFMC. *Workflow Standard - Interoperability, Abstract Specification, WFMC-TC-1012, Version 1.0.* WFMC (Workflow Management Coalition), www.wfmc.org, October 20 1996.