NANCY
UNIVERSITE

UFR Mathématiques et Informatique **École Doctorale IAEM Lorraine**

Département de Formation Doctorale en Informatique

Quelques briques et concepts de base pour la coopération et les systèmes coopératifs

HDR

présentée et soutenue publiquement le 1/12/2004

pour l'obtention de l'

Habilitation à Diriger des Recherches de l'Université Nancy 2

(spécialité informatique)

par

Khalid Benali

Composition du jury

Rapporteurs : M. Michel E. Adiba, Professeur à l'Université Joseph Fourier, Grenoble
M. Claude Chrisment, Professeur à l'Université Paul Sabatier, Toulouse
M. Michel Schneider, Professeur à l'Université Blaise Pascal, Clermont-Ferrand

Examinateurs : M. Claude Godart Professeur à l'Université Henri Poincaré, Nancy
M. Jean-Marie Pinon, Professeur à L'INSA, Lyon
Mme Jeanine Souquières , Professeur à l'Université Nancy 2, Nancy

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria

Mis en page avec la classe thloria.

Table des matières

Première partie

Synthèse des travaux

1

Parcours professionnel, diplômes et état civil

Parcours professionnel

- **De 1990 à 1997** : **Maître de Conférences à l'Université de Metz**
  Chercheur au LRIM (Laboratoire de Recherche en Informatique de Metz) tout en étant chercheur associé au CRIN (Centre de Recherche en Informatique de Nancy)[1]
  Enseignant à l'Université de Metz
- **Depuis 1997** : **Maître de Conférences à l'Université Nancy 2**
  Chercheur au LORIA au sein du projet INRIA ECOO
  Enseignant à l'Université Nancy 2

Adresse professionnelle actuelle
LORIA-Université Nancy 2
Campus Scientifique
BP 239
54506 Vandœuvre-lès-Nancy Cedex
e-mail : Khalid.Benali@loria.fr

Diplômes et état civil

Né le 26 novembre 1961 à Rabat (Maroc), titulaire d'un **doctorat de l'université de Nancy 1 en informatique**, obtenu le 24 novembre 1989 au sein du CRIN [7], ma scolarité antérieure est résumée ci-dessous :

D.E.A Informatique :	Nancy 1, Septembre 1985, Mention AB
Maîtrise Informatique	Nancy 1, Juin 1984
Licence Informatique	Nancy 1, Juin 1983, Mention AB
DEUG SM2	Nancy 1, Juin 1981
SM1	Nancy 1, Juin 1980, Mention AB
Baccalauréat Série C	Rabat (Maroc / Académie de Bordeaux), Juin 1979, Mention AB

[1] puis au LORIA (Laboratoire lorrain de recherche en informatique et ses applications) quand celui-ci est devenu le nouveau laboratoire d'informatique de Nancy

2

Synthèse des activités pédagogiques

2.1 Activités d'enseignement

Je cite les cours donnés à l'université de Metz où j'ai été en poste à d'octobre 1990 à août 1997, puis à l'Université de Nancy 2 où je suis en poste depuis septembre 97, sachant qu'actuellement je fais mon service principalement entre l'IUP-MIAGE (Méthodes Informatiques Appliquées à la Gestion) et le DESS SID (Systèmes d'Information Distribués).

Il est à noter, par ailleurs, que la plupart des enseignements que j'ai réalisés ont été des créations (Applications Distribuées, Analyse et Conception Orientées Objet et UML, Informatique Appliquée,...). C'est à dire qu'il a fallu les penser, les décrire, "monter" l'enseignement de toutes pièces et faire un polycopié.

Les polycopiés réalisés sont de deux genres. Pour les premiers et deuxièmes cycles (comme par exemple le cours d'architecture des ordinateurs en Licence Informatique ou le cours de C en Licence d'IUP-MIAGE ou encore le cours d'analyse et de conception orientées objets et UML de maîtrise de l'IUP-MIAGE), le polycopié correspond à un document se suffisant à lui même et comprenant "l'ensemble" des informations intéressant l'étudiant. Ce polycopié est utilisé comme référence "unique" au cours. En effet, même si je continue à citer un certain nombre d'ouvrages de référence pour la matière enseignée (ouvrages achetés et mis à disposition au centre de documentation), l'expérience montre que les étudiants de 1er et 2ème cycle ne font pas une utilisation intensive (loin de là !) des livres et des centres de documentation. En ce qui concerne les troisièmes cycles et en particulier les DESS ou j'effectue un certain nombre de cours, les polycopiés mis à disposition des étudiants ne sont que les copies de mes transparents électroniques car les étudiants doivent compléter leur lecture par les références citées, que celles-ci soient des livres ou de la documentation disponible en ligne. Un budget est même prévu pour l'achat de livres de références spécifiques aux DESS. Dans le cas des étudiants de DESS, si on considère globalement les références citées, le pourcentage de lecture de celles-ci est satisfaisant. Par contre les web-références sont beaucoup plus utilisées que les références à des livres.

Dans le descriptif suivant, je liste les cours effectués et le public visé, avec, entre parenthèses, le volume horaire en cours magistral suivi du volume horaire en travaux dirigés effectués. **Un symbole $\bigoplus$ indique les cours que je fais actuellement.**

1er cycle

DEUG

- Initiation à l'algorithmique et à la progammation
  Cours et TD en 1ère année du DEUG SV (Sciences de la Vie) de Metz (16+24 heures)
- Informatique Appliquée :Architecture d'un ordinateur et assembleur, Réseau, Base de données, Intelligence artificielle.
  Cours et TD en 2ème année du DEUG MIAS (Mathématiques, Informatique et Applications aux Sciences) de Metz (18+30 heures)
  J'avais proposé et monté ce cours d'ouverture comme un "produit d'appel" pour la Licence d'Informatique dont j'étais responsable. En effet la vision "mathématiques appliquées" (vision prévalant à l'époque) ne me semblait pas la bonne façon de présenter les contours de l'informatique aux étudiants de DEUG. La majorité de ceux que cette vision "mathématiques appliquées" intéressait allaient de, toutes les façons, faire une licence de mathématiques !
- Outils de Base.
  Cours et TD en 1ère année du DEUG MISASHS (Mathématiques, Informatique, statistiques appliquées aux sciences humaines et sociales) de Nancy 2 (25 heures)
- Initiation à l'algorithmique et à la progammation
  Cours et TD en 1ère année du DEUG MISASHS de Nancy 2 (50 heures)

2ème cycle d'informatique

Licence d'informatique

- Architecture des ordinateurs
  Cours et TD en licence d'informatique de Metz (24+36 heures)
- Programmation en C
  Cours et TD en 2ème année de l'IUP-MIAGE de Nancy 2 (10+20 heures)

Maîtrise d'informatique

- Analyse et conception orientées objet
  Cours et TD en maîtrise d'informatique de Metz (15+12 heures)
- $\bigoplus$ Analyse et conception orientées objet et UML
  Cours et TD en 3ème année de l'IUP-MIAGE de Nancy 2 (32+10 heures)
- $\bigoplus$ Conception de SI en utilisant une approche objet et UML
  TD en 3ème année de l'IUP-MIAGE de Nancy 2 (24 heures)

3ème cycle d'informatique

- Conception et programmation objets (C++)
  Cours en cycle C du CNAM des universités de Nancy (26 heures)
- Conception objet (partie du module Bases de Données Orientées Objet)
  Cours et TD en DESS informatique de Nancy 1 et ESIAL 3 (6+9 heures)
- Réseaux et Systèmes d'Information.
  Cours en DESS ACSI de Nancy 2 (20 heures)
- Modèle de Transactions exotique pour la CAO (partie du module Bases de Données pour le Génie Logiciel)

Cours en DEA informatique de Nancy (6 heures)
- ⨁ Les outils du travail coopératif et la cohérence des données coopératives (partie d'un module faisant intervenir des "experts" extérieurs)
  Cours en DEA informatique (RACOR) de Troyes (3 heures)
- ⨁ Applications Distribuées.
  Cours et TD en DESS SID de Nancy 2 (20+10 heures)
- ⨁ Méthodes de Conception Orientées Objets et Patrons.
  Cours et TD en DESS SID de Nancy 2 (20+10 heures)
- ⨁ Projets industriels.
  Encadrement en DESS SID de Nancy 2 (15h)
- ⨁ Projets bibliographique et d'application.
  Encadrement en DESS ACSI de Nancy 2 (20h)

Description d'un module créé : Analyse et conception orientées objet et UML (ACOO/UML)

A mon arrivée à l'UFR MI de Nancy 2, on m'a proposé de faire une dizaine d'heures d'introduction à la démarche objet en utilisant OMT dans le cadre d'un cours de Conseption de Systèmes d'Informations de maîtrise "classique" utilisant Merise et la démarche "maison" (démarche intégratrice). Le formalisme et l'approche de conception objet associée d'OMT étant quelque peu en perte de vitesse par rapport au formalisme émergeant de l'époque, j'ai proposé de construire cette introduction en utilisant comme formalisme support UML et un procédé inspiré du RUP (Rational Unified Process). L'année suivante, un cours de maîtrise ayant été replacé dans le cursus en licence et ayant de la sorte libéré un certain nombre d'heures en maîtrise, il a été décidé de faire de cette introduction à la démarche objet un module à part entière de 32h de cours et 10h de TD. J'ai donc monté ce module dont voici un bref descriptif.

Objectifs

- Présenter aux étudiants quelques techniques d'analyse et de conception orientées objets en s'appuyant sur les formalismes d'UML qui sont présentés et expliqués.
- Présenter et utiliser un atelier d'analyse et de conception orientées objets.
- Utiliser une méthode simple permettant d'utiliser les différents formalismes UML pour l'analyse et la conception d'un système.

Mode d'évaluation

Examen final et réalisation d'un projet d'analyse ou de conception d'un SI en utilisant les formalismes d'UML à réaliser, en liaison avec le cours de Conception des Systèmes d'Information avec application des divers concepts abordés et utilisation de l'atelier Rational Rose.

Contenu

Ce cours traite de l'analyse et la conception orientées objets de logiciels ou de systèmes d'information, depuis la phase de démarrage (raison d'être et portée du projet) jusqu'à la phase de transition (test, optimisation, formation,...). Toute méthode d'analyse et/ou de conception s'appuie sur trois éléments un procédé (processus) d'analyse ou de conception, un langage de modélisation et un ou plusieurs outils support. Les choix faits dans le cadre de ce cours sont : Un procédé simple, UML comme langage de modélisation et Rational Rose comme outil support.

Dans ce cours, nous abordons successivement les points suivants :
- Introduction : Description et comparaison des méthodes d'analyse et de conception orientées objet
- Un procédé (ou processus de développement) complet, itératif et incrémental.
- Les différents formalismes d'UML et leur utilisation dans le cadre du procédé choisi.
- Conclusion et ouverture : les patrons de conception. Sensibiliser les étudiants à la notion de Design Pattern (ou patron de conception) en vue de remplacer la démarche de redécouverte d'une solution de conception classique par l'utilisation du patron de conception correspondant.

La présentation et l'utilisation de Rational Rose se fera à différents moments du cours en fonction des besoins.

Support pédagogique

Le support du cours est sous forme de transparents et d'un polycopié.
Atelier d'analyse et de conception orientées objet en UML : Rational Rose.
Ouvrages conseillés (achetés au Centre de Documentation du PLG) :
- Uml Distilled : Applying the Standard Object Modeling Language, Martin Fowler, Kendall Scott (Contributor), Addison-Wesley, 1997.
- UML Toolkit, Hans-Erik Eriksson, Magnus Penker , John Wiley & Sons, 1997.
- Object-Oriented Methods : A Foundation (Uml Edition), James Martin, James J. Odell, Prentice Hall, 1997.
- Design Patterns : Elements of Reusable Object-Oriented Software, Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Addison-Wesley, 1994.
- Analysis Patterns : Reusable Object Models, Martin Fowler, Addison-Wesley, 1996.
- The UML reference manual, J. Rumbaugh, I. Jacobson, G. Booch, Addison-Wesley, 1999.
- The UML user guide, G. Booch, J. Rumbaugh, I. Jacobson, Addison-Wesley, 1999.
- The unified software development process, I. Jacobson, G. Booch, J. Rumbaugh, Addison-Wesley, 1999.

Remarques

TD : Quelques cas d'école et une étude de cas en UML permettant de concevoir une solution en utilisant les formalismes les plus importants d'UML.

Liens avec les autres matières

- Liens avec les autres matières de l'année : Liaison avec tous les cours d'informatique et en particulier avec le cours de Conception des Systèmes d'Information (les formalismes d'UML sont utilisés pour la conception de systèmes d'information) et celui de Génie Logiciel (UML est un des langages de modélisation possibles dans le cadre de la conception de logiciels). Il est à noter que le lien avec le cours de CSI est renforcé et affirmé par le fait que je fait une partie des TDs de CSI (en fin d'année), pour appliquer les concepts vus en ACOO/UML sur une étude de cas déja traité avant de manière classique.
- Liens avec les matières de l'année suivante : La conclusion de ce cours est une introduction au cours du DESS SID de conception orientée objets entièrement construit autour des patrons. En maîtrise, on leur apprend à concevoir un système par leur propres moyens, et en DESS on leur apprend à concevoir un système en réutilisant des patrons propres et déja éprouvés par d'autres concepteurs...

2.2 Activités pédagogiques autres que l'enseignement

- Encadrement de projets en DESS CFMAO option CAO à l'université de Metz puis en DESS ACSI et SID à l'université Nancy 2,
- Suivi de stages de fin d'études en DESS CFMAO Option CAO, puis en DESS ACSI, DESS SID et en IUP-MIAGE.
- Participation à plusieurs jurys de soutenance de thèses, de DEA, de mémoires CNAM, de jurys de soutenance de stages de DESS, de stages IUP-MIAGE et de jury probatoire CNAM.
- Accueil au sein de l'équipe ECOO du LORIA et encadrement d'un stage de fin d'études de l'université de Tunis.
- Accueil au sein de l'équipe ECOO du LORIA et encadrements d'un stage de 2ème année d'études d'ingénieur de l'ENSIMAG (Ecole Nationale Supérieure d'Informatique et de Mathématiques Appliquées de Grenoble).
- Accueil au sein de l'équipe ECOO du LORIA et encadrements de nombreux (6) stages de fin d'étude d'élèves ingénieurs en provenance de l'Ecole Mohammedia des Ingénieurs de Rabat (EMI, l'école d'ingénieurs généralistes de référence au Maroc) et de l'Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes (ENSIAS, l'école d'ingénieurs en informatique de référence au Maroc).

  Ces derniers types d'encadrement relèvent autant des activités pédagogiques car ils concernent des étudiants que des activités de recherche car ces étudiants (élèves ingénieurs pour leur grande majorité) devaient, à chaque fois, réaliser du développement pour la plate-forme d'expérimentation qui nous sert à valider, de manière pratique, nos activités de recherche.

3

Synthèse des activités d'animation, d'encadrement et de transfert

3.1 Activités diverses (administratives, collectives,...)

Un symbole $\bigoplus$ indique les acivités diverses que je fais actuellement.

- Responsable de la licence d'informatique de l'université de Metz pendant 4 ans,
- Membre du conseil scientifique de l'université de Metz pendant 4 ans,
- Membre du conseil d'administration de l'université de Metz pendant 2 ans (mandat écourté pour cause de changement d'Université),
- **Directeur de l'IUP-MIAGE** de l'université Nancy 2 pendant un mandat de trois ans (novembre 1998, novembre 2001).
- Rédacteur principal du dossier de réhabilitation de l'IUP-MIAGE de l'université Nancy 2 pour le quadriennal en cours (2001-2005).
- Participation à la création de la Licence Professionnelle des métiers de l'informatique de l'université Nancy 2.
- Co-responsable administratif de la Licence Professionnelle des métiers de l'informatique de l'université Nancy 2 depuis sa création (octobre 2000)(Le déroulement de la formation se faisant chez nos collègues de l'IUT, je me dois de préciser que cette co-responsabilité au titre de l'UFR MI de Nancy 2 n'est pas une responsabilité très lourde).
- **"Créateur" du DESS SID (Systèmes d'Information Distribués)** de l'université Nancy 2 (J'étais le "porteur" de la demande d'habilitation de ce DESS)
- $\bigoplus$ **Responsable du DESS SID** (Systèmes d'Information Distribués) de l'université Nancy 2 depuis sa création (octobre 2001). Ce DESS est en cours d'habilitation comme une des deux spécialités du Master MIAGE de Nancy 2.
- Membre élu et vice-président, au titre du collège des Maître de Conférences, de la commission de spécialistes(CSE) (27 ème section) de l'université Nancy 2 (mandat terminé en décembre 2003).
- $\bigoplus$ Membre extérieur de la CSE (27 ème section) de l'universite de Metz (mandat en cours).
- $\bigoplus$ **Responsable de la spécialité IISE (Intégration et Interopérabilité des Systèmes d'Entreprise) du Master "Information numérique en Entreprise"** en cours de montage pour une co-habilitation les universités de Nancy 1 et Nancy 2.

Quelques éléments d'appréciation de la lourdeur de ces activités diverses (administratives, collectives,...)

La description des items de la liste des activités diverses précedentes pourraient, à elle seule, donner lieu à un document complet. Ceci n'est pas l'objectif de ce document mais je ne peux m'empécher de donner quelques éléments d'appréciation de ces activités.

Direction de l'IUP-MIAGE

En ce qui concerne mes activités de direction de l'IUP-MIAGE, il est à noter que l'IUP-MIAGE de Nancy regroupe environ 150 étudiants qui suivent 2000 heures de formation en 3 ans et qui sont en stage sur un période de 5 mois au minimum. Une vingtaine d'enseignant chercheurs de l'UFR MI enseignent en IUP-MIAGE et l'équipe pédagogique est complétée par une dizaine d'intervenants extérieurs.

La tâche du **directeur** de l'IUP-MIAGE comprend les **aspects pédagogique, organisationnel et stratégique**. Je n'étais, bien sûr, pas seul, mais j'étais responsable de l'ensemble.

En ce qui concerne les **aspects pédagogiques** traités, je peux évoquer une tâche menée à bien, à savoir l'introduction d'une nouvelle deuxième langue vivante. Parmi les intervenants exterieurs, se trouvent les enseignants de langue proposés par le SELV (Service d'Enseignement des Langues Vivantes) de l'université Nancy 2. Les IUP ont l'obligation statutaire d'avoir deux langues vivantes au programme. Jusqu'à mon arrivée, seul l'anglais et l'allemand était enseignés. J'ai réussi à convaincre l'équipe pédagogique en place de la nécessité d'introduire la nouvelle langue en vogue dans les lycées, à savoir l'espagnol. L'introduction se faisant à coût constant, celle-ci signifie une diminution des volumes horaires existant et donc un travail certain de discussions et d'animation pour que la réforme soit celle des enseignants de langue et non celle du directeur de l'IUP-MIAGE. D'autres aspects pédagogiques comme le remplacement de C++ par Java ont été plus simples à mettre en œuvre, dans la mesure où le changement était plus facilement accepté par le corps enseignant. Sans remettre en cause le fait que la maîtrise d'un langage de programmation objet s'impose dans le cadre du cursus d'un informaticien, le choix historique du langage C++ ne me semblait plus convenir. En effet, d'une part, bon nombre des concepts objets se trouvent implantés de manière plus élégante (et surtout plus stricte[2]) en Java, et d'autre part, la diffusion très large de Java au sein des entreprises et du secteur éducatif m'ont semblé deux arguments décisifs pour le remplacement de C++ par Java.

En ce qui concerne les **aspects organisationnels** traités, je peux citer le renouvellement des intervenants en gestion. Les enseignements de gestion, importants dans l'optique d'un diplôme à double compétence (informatique et gestion) comme l'est l'IUP-MIAGE laissaient un peu à désirer par certains aspects. Quelques intervenants en gestion accumulaient un volume horaire un peu trop conséquent et cela se ressentait sur la qualité de certains de leurs modules. Après avoir identifiés, les différents modules "à problème" et avoir compris l'objectif pédagogique de ces modules -tâche non triviale pour un non-gestionnaire-, j'ai "prospecté" sur la région pour trouver plusieurs intervenants extérieurs de qualité qui ont rajeuni et raffraichi les enseignements de gestion. Ces intervenants extérieurs sont toujours en place, leurs enseignements sont de qualité et ce sont des collègues appréciés.

Et enfin d'un **point de vue stratégique**, je peux citer la création du DESS SID (Systèmes d'Information Distribués). La filière IUP en général et IUP-MIAGE en particulier avaient du mal à se situer dans le schéma 3/5/8 de l'époque (devenu depuis LMD). Le montage et la création d'un

[2]Le fait que C++ offre la liaison dynamique simplement comme une "option" possible m'a toujours semblé antinomique de la vision objet de la programmation

DESS avec des objectifs pédagogiques complémentaires à ceux de l'IUP-MIAGE permettaient, d'une part, de compléter la formation des meilleurs Miagistes et, d'autre part, de les "positionner" à bac+5. La spécialité retenue concerne les Systèmes d'Information distribués (ou SI répartis ou encore SI en réseau). En effet, avec la montée en puissance des réseaux et des systèmes de télécommunications, les systèmes distribués en général et les Systèmes d'Information distribués deviennent la norme, aussi bien dans le milieu universitaire que dans le milieu des entreprises. Ce DESS SID a été aussi pensé de manière à compléter et non à concurrencer les différents DESS informatiques déjà présents sur Nancy et regroupés au sein de l'ISIAL (DESS Audit et Conception des Systèmes d'Information, DESS Informatique (option Ingénierie du logiciel et option Ingénierie des Réseaux et des Systèmes). Le DESS SID fait maintenant partie de l'ISIAL et propose une "poursuite" naturelle d'études pour les Miagiste et une alternative supplémentaire pour les autres maîtrises.

Par ailleurs, l'organisation des concours d'accès à l'IUP-MIAGE ou l'élaboration des emplois du temps ou l'organisation des stages ou l'organisation des interventions d'industriels ou des rapports avec ceux-ci (entre autres, dans le cadre du conseil de perfectionnement -instance de l'IUP regroupant universitaires et industriels-) ou encore la rédaction du dossier de réhabilitation de l'IUP-MIAGE de l'université Nancy 2 pour le quadriennal en cours (2001-2005), même si elles n'ont pas été de mon ressort exclusif (j'avais une équipe de direction) ont du être gérées, contrôlées et assumées par moi.

Création du DESS SID (Systèmes d'Information Distribués)

Si la gestion d'un diplôme existant est lourde et prenante, la création d'un diplôme est, à mon avis, plus compliquée et nécessite absolument une vision à long terme.

Le besoin stratégique ayant été identifié et les objectifs pédagogiques déterminés (cf paragraphe précédent), le montage du DESS SID a nécessité bon nombre d'heures de travail, de concertation et d'interactions avec les instances de l'université (voire des universités) et du ministère.

En synthèse, la quatrième promotion a entamé ses cours début octobre 2004 au sein du DESS SID de l'ISIAL et de l'université Nancy 2 avec une innovation pédagogique intéressante : un jour par semaine est consacré à un projet industriel mené individuellement par un étudiant au sein d'une entreprise sur un "vrai" problème de l'entreprise.

Outre les aspects pédagogiques lourds inhérents au montage d'un nouveau diplôme (complémentarité des différents cours, "recrutement" d'enseignants pour les cours, demande de subventions pour les aspects matériels, demande de salle, montage d'une salle de TPs/Projets avec des droits spécifiques pour les étudiants du DESS SID,...), il faut aussi citer les interactions et l'implication forte d'un certain nombre d'entreprises partenaires. En effet, outre le projet industriel réalisé au sein des entreprises, celles-ci (W4, Intech, CIC-Développement, Syntegra, BDC Multimédia,...) nous offrent des cours et des séminaires, d'une part, pour une vision "industrielle" du métier et, d'autre part, pour compléter le volume horaire restreint alloué de manière standard par Nancy 2 à tous ses DESS.

En outre, et à titre personnel, j'ai du préparer deux nouveaux modules (Applications Distribuées, Méthodes de Conception Orientées Objets et Patrons) lors de l'ouverture du DESS. Le DESS a été habilité en fin d'année scolaire (!) et les étudiants recrutés "dans la foulée". Il a alors fallu que je prépare les nouveaux cours pour la rentrée.

Ce DESS est actuellement en passe de devenir une des deux spécialités du Master MIAGE de Nancy 2.

3.2 Activités d'encadrement de recherche

NB : Les description des travaux et les enchaînements scientifiques entre ces différents travaux sont décrits dans le chapitre de synthèse des travaux de recherche. Un symbole $\bigoplus$ indique l'acivité en cours.

• **Co-Encadrement (avec Claude Godart) de 2 thèses en informatique** :
- Une architecture pour intégrer des composants de contrôle de la coopération dans un atelier distribué (M. Munier, Thèse en Informatique de l'UHP, Nancy 1, 15 janvier 1999.)
- Un modèle orienté services procédés pour la coopération et l'interconnexion de procédés d'entreprises (K. Baïna, Thèse en Informatique de l'UHP, Nancy 1, 16 mai 2003.)

• **Encadrement de 6 DEA d'informatique** :
- Mini atelier de génie logiciel basé sur un modèle (D. Philippi, Nancy, septembre 1991)
- Analyse, conception, codage, test : vers un outil unique, graphique et intelligent (J-C. Colson, Nancy, septembre 1992)
- Implantation d'un outil graphique de conception du dialogue et de l'interface du système de CAO SACADO (J. Lahyane, Nancy, septembre 1995)
- Formalisation et réalisation d'un service de négociation en ingénierie concourante (K. Baïna, INPG - Université Joseph Fourier de Grenoble, juin 1999)
- Interconnexion de procédés par négociation et échanges de services (W. Gaaloul, Nancy, juillet 2002)
- Patron de coopération (U. Yildiz, Nancy, juin 2004)

• **Encadrement d'un DRT SIO** de Nancy 2 :
- Conception d'un système de gestion des emplois et des compétences en contexte intranet / internet (M. Bellaj, Université Nancy 2, BDC-Nancy, juin 2001),

• **Co-Encadrement avec Dalila Chiadmi (EMI-Université Mohammed V de Rabat) d'un DESA d'informatique de l'Université Mohammed V de Rabat** [3] **:**
- Un modèle de tactiques et d'agents réactifs d'aide à la décision lors de la négociation de service (A. Zellou, DESA Informatique de l'Université Mohammed V de Rabat, décembre 2001)

[3] Le DESA (Diplôme d'Etudes Supérieures Approfondies) est "l'équivalent" Marocain du DEA Français.

3.3 Activités d'animation de recherche

Un symbole $\bigoplus$ indique les acivités actuelles.

• **Membre de trois comités de programme**
- INFORSID'2000, Lyon
- INFORSID'2001, Martigny, Suisse
- MCSEAI'2002 (Maghrebian Conférence on Software Engineering and Artificial Intelligence), Annaba, Algérie

• **Président du comité d'organisation d'INFORSID'2003, Nancy, du 3 au 6 juin 2003**.

Je vais donner ci-dessous quelques chiffres et faits pour évaluer une telle tâche.

Tout d'abord, il a fallu, bien sûr, gérer toute la logistique de la conférence (amphithéatre, salles de réunion, repas, hébergement, soirée de gala,...) avec l'incertitude "normale" pour toute conférence, à savoir le nombre exact de participants. Je ne détaillerai pas cet aspect logistique, nécessaire, mais sans visibilité dans la mesure ou tout fonctionne correctement (ce qui a été, heureusement, le cas).

La conférence commençait par une journée de tutoriels à ma charge pour laquelle j'ai choisi les sujets et les intervenants. L'objectif, à travers cette journée, était, d'une part, de présenter quelques une des tendances actuelles d'ouverture des Systèmes d'Information vers le web et, d'autre part d'avoir des intervenants provenant d'horizons divers (enseignants/chercheurs, chercheurs, industriels). Les exposés finalement retenus ont été les suivant :
- De la technologie BD à la technologie Web (Nacer Boudjlida, LORIA, Pr UHP-Nancy 1)
- Vers la gestion de services Web (Marie Christine Fauvet, IMAG, Pr UJF-Grenoble)
- Extraction, gestion de connaissance et Web (Amedeo Napoli, LORIA, DR CNRS)
- Le Workflow (Philippe Betschart, W4 S.A., W4 Team Manager)

Par ailleurs, nous avons réalisé des affiches et un site web (http ://inforsid2003.loria.fr) pour la promotion de la conférence.

Les actes ont été publiés en couleur et comprenaient 23 articles après une selection, effectuée par le comité de programme, sur 64 soumissions provenant d'Algérie, du Canada, de France, du Maroc, de Suisse, du Qatar et de Tunisie.

Un dernier point que j'évoquerai est la réalisation de nombreux dossiers de demande de subventions qui ont permis à différents acteurs économiques et sociaux de la région de nous apporter leur soutien (Région Lorraine, Communauté Urbaine du Grand Nancy,...) .

En conclusion et pour synthétiser nous pouvons dire que la conférence a été un succès avec la présence de 80 participants.

• Membre du comité d'organisation des Assises du GdR I3, Nancy décembre 2002

• $\bigoplus$ Relecteur ("reviewer") d'articles et président de quelques sessions pour un certain nombre de conférences (Inforsid, International Workshop on The Many Facets of Process Engineering, Conférence Africaine de Recherche en Informatique, Maghrebian Conférence on Software Engineering and Artificial Intelligence, CITE, TES, SEKE ...)

• Membre du comité de pilotage de la Deuxième École de Modélisation d'Entreprise (Nîmes, Mars 2004), dont l'objectif est de dresser un bilan des concepts et pratiques de la modélisation des systèmes d'information pour l'entreprise et des liens avec la modélisation d'entreprise. Cette école doit contribuer à une meilleure connaissance des rapports entre modèles d'entreprise et modèles de système d'information.

• membre de MIPS/Maroc (Maghrebian Information Processsing Society) et du réseau marocain du TOKTEN (Transfer Of Knowledge Through Expatriate Nationals) financé par le PNUD (Programme des Nations Unis pour le Développement),

• $\bigoplus$ membre du comité exécutif de l'association INFORSID,

• $\bigoplus$ **membre élu du Conseil National des Universités(CNU)** (27 ème section) depuis octobre 1999. J'ai terminé un premier mandat en 2003 et j'ai, à nouveau, été élu en octobre 2003 pour un second mandat.

Quelques chiffres et faits pour évaluer une telle tâche qui est une tâche très lourde et très prenante pendant 2 mois dans l'année :

– deux sessions par an (qualifications et promotions),
– chaque membre du CNU rapporte sur un nombre très conséquent de dossiers (une quarantaine en moyenne !) à chaque session. Ces rapports servent de base à la décision de qualification (ou promotion) prise par le CNU.

Cette activité au sein du CNU m'a permis de travailler avec des enseignants-chercheurs traitant différentes thématiques de recherche et dont les environnements de travail sont extrémement variés. Cette situation de mixité n'est pas si courante dans le travail quotidien et permet de s'enrichir et d'avoir une ouverture scientifique non négligeable. Par ailleurs, le fait de rapporter sur des dossiers dont la thématique, tout en étant proche de ma thématique de recherche, n'est pas la mienne m'a obligé à apprendre de nombreuses choses extrémement intéressantes.

• $\bigoplus$ Membre du Groupe de Travail Collecticiels du GdR I3.

• $\bigoplus$ **Co-créateur et Co-animateur du Groupe de Travail ECI (systèmes d'information, Entreprise Communicante, Interopérabilité des systèmes d'entreprise) dans le cadre des GdR I3 et MACS du CNRS**. L'objectif de ce groupe de travail, à caractère transversal et commun aux GdR I3 (Information - Interaction - Intelligence) et au GdR MACS (Modélisation, Analyse et Conduite des Systèmes dynamiques), est d'identifier les difficultés liées à l'interopérabilité des systèmes d'information et des modèles d'entreprise et de proposer des solutions génériques permettant de résoudre ces problèmes. Ce groupe de travail traite de la même problématique scientifique que le réseau thématique européen UEML et que le réseau d'excellence européen INTEROP (cf. 3.4).

3.4 Activités de transfert

Participation à plusieurs projets de recherches/développements "industriels" ou de transfert (le dernier de ces projets (INTEROP) est en cours et signalé par un $\bigoplus$).

• **ALF (projet Esprit I) (1987-1991)**
Ce projet Européen, auquel j'ai participé, dans le cadre de ma thèse d'université, avait pour objectif de construire un Atelier de Génie Logiciel de 3ème Génération. L'objectif précis était la conception et la réalisation d'une structure d'accueil permettant le développement d'Ateliers de Génie Logiciel à capacités d'initiative. Pour ce faire, une nouvelle génération d'interfaces compatibles avec PCTE a été proposée. Les participants au projet étaient le GIE Emeraude regroupant Bull, Eurosoft et Syseca (Paris, France), CIG-INTERSYS S.A. (Bruxelles, Belgique), Computer Technologies Co.-CTC (Athènes, Grèce), Grupo de Mecánica del Vuelo, S.A. (Madrid, Espagne), International Computers Limited (Reading, Grande Bretagne), l'université de Nancy-CRIN (Nancy, France), l'université de Dortmund-Informatik X (Dortmund, RFA), Cerilor (Nancy, France), l'université Catholique de Louvain (Louvain-la-Neuve, Belgique), et l'université de Dijon-CRID (Dijon, France). Ma participation au projet était importante dans la mesure où ma thèse a été réalisée dans ce cadre et ma dernière année de thèse a été financé directement par ce projet. De nombreuses publications ont été faites dans ce cadre. En particulier [1] propose une synthèse des activités de recherche menées dans le cadre de ce projet.

• **Cocao (projet CTI-CNET) (1998-2001)**
Ma participation à ce projet Français impliquant le CNET, le LORIA et le CRAI (Centre de Recherche en Architecture et Ingénierie) de Nancy, a concerné la première année du projet et plus particulièrement l'analyse des usages en vigueur pour les projets architecturaux nécessitant une coopération entre les différents acteurs du domaine. Cette participation a donné lieu à deux articles [20] et [23] écrits avec des chercheurs du CRAI.

• **MTI (Management Territorial Interactif) (2000-2001)**
Ce projet Nancéen était porté par des gestionnaires du GREFIGE (Groupe de Recherche en Economie Financière et Gestion des Entreprises), laboratoire de recherche du Pôle Lorrain de Gestion (Université Nancy 2) et concernait l'introduction des NTI et l'impact de celles-ci dans le Management Territorial. Mon rôle était celui de "l'expert informaticien". Ma participation m'a permis d'appréhender la vision des gestionnaires et des chercheurs en gestion sur les Systèmes d'Information.

• **UEML (Projet IST) (2002-2003)**
L'objectif du projet Européen UEML était la création d'un Langage Unifié de Modélisation d'Enterprise afin de faciliter l'interopérabilité entre systèmes de modélisation d'entreprises et, de là, l'interoperabilité entre entrepises. UEML regroupait huit partenaires en provenance du monde de la recherche et de l'industrie : Computas (Norvège), Graisoft (France), CIMOSA (Allemagne), IPK (Fraunhofer Institute for Production Systems and Design Technology - Allemagne), le LORIA (France), l'université de Namur (Belgique), l'université de Turin (Italie) et l'université polytechnique de Valencia (Espagne). Ma participation a principalement porté sur l'état de l'art en modélisation de systèmes et sur la définition des constructeurs de base du langage visé, ces constructeurs de base ayant été déduit d'une méta-modélisation des modèles sous-hacents aux trois outils de modélisation utilisés dans le cadre du projet. [38] propose une synthèse des activités de recherche menées dans le cadre de ce projet.

• $\bigoplus$ **INTEROP (Projet NOE de 36 mois ayant démarré le 1er novembre 2003)**
Le projet UEML a donné lieu à une "suite" par la soumission d'un projet de Réseau d'ex-

cellence Européen (NoE ou "Network of Excellence") autour de l'interoperabilité (INTEROP ou "INTEROPerability research for networked enterprises applications and software") dans le cadre du 6ème PCRD européen (sixth framework programme (FP6)). Le NOE INTEROP a été accepté (et financé) et à démarré début novembre 2003[4]. Je participe au travail dans plusieurs "Work Packages" (WP) et je suis, par ailleurs, co-responsable d'une tâche dans un WP, depuis le démarrage du projet. Le travail dans ce WP a pour but la réalisation d'une carte de connaissances des recherches "communes" au sein d'INTEROP ("Knowledge map for Enterprise Modelling, Ontology, Architecture and Platform for interoperability researches inside INTEROP"). Outre cette implication dans ce WP, ma participation continuera dans l'immédiat, dans un autre WP traitant de la modélisation d'entreprise et qui est la suite logique du projet UEML (Common Enterprise Modelling Framework (CEMF)).

Tous ces projets ont été l'occasion de Recherche/Développement et ont donné lieu à de nombreux rapports et à des publications (cf., entre autres, les références citées dans ce paragraphe).

[4]Nous avons également participé à l'élaboration d'un projet intégré ("Integrated Project") autour des entreprise virtuelles (Global Network Management) qui n'a finalement pas été retenu dans le cadre du FP6

4

Synthèse des travaux de recherche

Je fais une présentation de mes différentes phases de recherche, sachant que ma recherche actuelle est représentée par un $\bigoplus$ et correspond à mon activité depuis 1997, date de mon arrivée à l'université Nancy 2 et de mon intégration complète au LORIA. **Le cadre général du projet de recherche que je mène est la coopération que j'aborde sous deux principaux aspects, à savoir l'échange de documents ou de services et la négociation de ces échanges.**

Modélisation de procédés

Ma thèse d'université a eu pour cadre général le Génie Logiciel et a porté sur les "Procédés de développement" de logiciels. Elle a été réalisée au Centre de Recherche en Informatique de Nancy (CRIN) au sein de l'équipe Génie Logiciel, et en partie dans le cadre du projet Esprit ALF.

La rédaction de ma thèse a eu pour objectif de présenter les deux aspects fondamentaux traités au cours de mon travail de thèse, à savoir le partage d'informations communes et cohérentes sur les objets manipulés lors de la production de logiciels et la modélisation de ce processus de production [7]. Une maquette a été réalisée avec comme objectif de permettre de se rendre compte "de visu" de certains aspects du comportement d'un système basé sur un tel modèle.

J'ai, par ailleurs, participé dans le cadre du programme ESPRIT I, au projet de recherche ALF d'ont l'objectif était la conception et la réalisation d'une structure d'accueil permettant le développement d'Ateliers de Génie Logiciel à capacités d'initiative (Bruxelles, octobre 87). Pour situer en quelques mots le projet ALF, disons que son objectif était de proposer une nouvelle génération d'interfaces compatibles avec PCTE .

La culture modélisation des aspects statiques acquise dans ces travaux anciens a été exploitée dans les travaux autour de la manipulation et l'échanges d'objets ou de documents dans les systèmes coopératifs et la culture modélisation de processus a été exploitée dans les travaux autour de la dynamique des système coopératifs et, en particulier, celle des systèmes de workflow.

Conception Assistée par Ordinateur

A mon arrivée à l'université de Metz, après ma nomination en tant que Maître de conférences, je me suis intégré au Laboratoire de Recherche en Informatique de Metz (LRIM), tout en restant chercheur associé au CRIN dans le cadre de l'équipe Génie Logiciel (GL) puis dans celui de l'équipe Environnements pour la COOpération (Projet INRIA ECOO).

Mon implication au sein du LRIM m'a amené à adapter et réorienter mon activité de recherche vers le thème du laboratoire qui était la Conception Assistée par Ordinateur (CAO) et je me suis donc attelé à la modélisation des interactions entre le système et ses utilisateurs et à celle des interactions entre les différents développeurs du système.

Ces interactions nécéssitant un langage ou un formalisme de description, nous avons développé un formalisme de description simple s'appuyant sur une représentation graphique.

Ce formalisme graphique décrit grace au Langage de Description de Formalismes (LDF) [17] et l'outil associé nous permettait de créer et de manipuler des schémas de description de procédés de développement respectant cette syntaxe graphique. Le LDF et l'outil associé nous ont permis de jouer deux rôles différents : celui de concepteur de formalisme pour concevoir un formalisme et celui d'utilisateur de formalisme pour créer des schémas respectant ce formalisme [18,19].

⊕ Coopération

Un aspect que nous n'avions qu'effleuré au LRIM concernait l'aspect coopératif de la conception. L'outil développé à Metz l'a été suivant un modèle client/serveur et permettait de ce fait à plusieurs clients ou utilisateurs de coopérer à un même travail, le serveur servant à stocker et à gérer les données communes et chaque client prenant en charge la gestion de ses traitements particuliers. Cependant les problèmes spécifiques liés à cette coopération implicite entre les différents utilisateurs n'ont pas été traités de manière exhaustive.

Le cadre général du projet de recherche que je mène est la coopération (thématique du projet INRIA ECOO (Environnements pour la COOpération) du LORIA). J'aborde, à titre personnel, cette coopération sous deux angles principaux, à savoir l'échange de documents ou de services et la négociation de ces échanges.

⊕ Coopération et échange de documents

Pour travailler ensemble et coopérer, un certain nombre d'utilisateurs de profils différents poursuivent un but de conception commun et doivent, de ce fait, coopérer, échanger un certain nombre d'informations et percevoir, éventuellement sous des angles et avec des points de vue différents, différents artefacts communs. Chaque utilisateur manipule de manière sous-jacente le même objet qu'il faudra donc partager. Dans le contexte initial du projet INRIA ECOO, le problème de la coopération est abordé selon une approche transactionnelle. Un modèle spécifique de transactions, les COO-transactions, ainsi qu'un nouveau critère de correction, la COO-sérialisabilité avaient été définis au sein de l'équipe[Mol96].

Dans le but de valider cette démarche et d'expérimenter dans un cadre plus général le protocole de synchronisation respectant le critère de COO-sérialisabilité pour les applications coopératives dans le domaine du génie logiciel développé par l'équipe ECOO, nous avons démarré une collaboration avec le Centre de Recherche en Architecture et Ingénierie (CRAI) de Nancy. En effet dans le domaine du BTP (Bâtiments et Travaux Publics), les outils de CAO sont largement utilisés et la conception-réalisation d'un projet architectural implique un grand nombre d'acteurs de métiers différents (architecte, thermicien, conducteur d'opérations, maître d'ouvrage,...) coopérant dans le projet.

Notre activité avec le CRAI nous a permis de faire une proposition dans le cadre des consultations thématiques du CNET pour 1997 portant sur les *"Modèles de coopération en Co-conception"*, proposition acceptée et ayant donné lieu à une CTI CNET. Cette activité a donné lieu à un certain nombre de publications avec les architectes du CRAI [20,23].

Par ailleurs et d'un point de vue plus fondamental, les COO-transactions coopèrent en s'échangeant des données par l'intermédiaire d'un référentiel commun (architecture centralisée). Or dans le domaine des applications qui nous intéressent, cette centralisation est un obstacle à l'autonomie des activités. Il est plus adéquat que chaque activité du système possède sa propre base de données locale et coordonne elle-même ses échanges avec les autres activités.

Si les différents modèles de transactions et critères de correction existants permettent effectivement de prendre en compte certains aspects de la coopération, aucun ne répond simultanément à nos besoins de coopération (les transactions ne doivent pas être isolées les unes des autres), de distribution (chaque transaction a sa propre copie des objets qu'elle manipule) et d'autonomie (pas de site central ; le contrôle des interactions est réalisé localement par chaque transaction). Ce sont les raisons pour lesquelles nous avons décidé de définir un nouveau modèle de transactions étendu qui satisfasse ces différents besoins de la coopération. Ce nouveau modèle de transactions avancé supportant les exécutions de transactions distribués géographiquement et coopérant par échange de résultats intermédiaires a été défini et présenté dans [24,25,26,28] ainsi que par Manuel Munier dans son rapport de thèse[Mun99]. De nouveaux critères de correction locaux ont été définis (la D-sérialisabilité et la DisCOO-sérialisabilité) qui sont assurés localement au niveau de chaque transaction et garantissent les mêmes propriétés au niveau du système complet que leurs homologues "globaux" (sérialisabilité et COO-sérialisabilité).

⊕ Coopération et négociation des échanges

Cependant dans le cadre des applications coopératives ditribuées, toutes les transactions ne sont pas forcément guidés par les mêmes règles pour coopérer avec leur partenaires. Etant donné que les transactions de notre modèle n'intéragissent pas via des accès concurrents à un référentiel commun mais par l'intermédiaire d'échanges de données explicites avec les autre transactions nous avons choisi de leur permettre de négocier les schémas de coopération qu'elles utilisent.

Nous avons donc proposé un modèle de négociation pour les applications distribuées coopératives (télétraitement, travail coopératif, vidéoconférence, commerce électronique,...), modèle que nous voulons générique et flexible. Les utilisateurs d'environnements distribués coopératifs ont besoin de mécanismes supports de la négociation afin d'évaluer et d'étudier les différentes alternatives possibles lors d'une prise de décision collective. Nous avons formalisé la négociation suivant trois points de vue : les informations échangées entre les agents afin de négocier (le langage), la façon dont ces informations sont échangées (le protocole) et le comportement interne d'un agent (la tactique). Outre une séparation des problèmes liés à chacune de ces trois facettes de la négociation, cela permet une grande flexibilité. Nous avons choisi une approche transactionnelle basée sur les actes de langage pour modéliser la négociation. Le modèle de négociation obtenu a été implanté comme un service de négociation au sein notre environnement distribué pour la coopération et présenté dans [27,29] ainsi que par Karim Baina dans son rapport de DEA[Baï99].

Il est a noter que la définition de ce modèle de négociation nous a permis d'introduire un nouveau paradigme de coopération (la prise de tour) qui est venu s'ajouter à ceux définis dans le cadre des modèles transactionnels COO et DisCOO (client/serveur, rédacteur/relecteur et écriture coopérative).

Cette approche de la négociation a été, en particulier, mise en œuvre pour permettre de choisir des schémas de coopération et de définir des politiques de coopérations entre acteurs. Mais ce n'est là qu'un aspect des possibilités offertes par notre modèle de négociation.

⊕ Coopération et échanges de services

Nous considérions, dans l'approche suivie jusqu'alors, que les acteurs travaillaient sans contrôle sur leur activité propre, et que seules étaient contrôlées leurs interactions avec les autres acteurs. Par ailleurs ces interactions se résumaient à l'"échange" de documents. C'est une vision un peu réductrice de la coopération. En fait chaque acteur suit un processus individuel et les phases d'échanges correspondent simplement à une phase spécifique de "rendez-vous" entre processus individuels. Nous avons donc été amené, "naturellement", à nous intéresser à la possibilité de définir des processus individuels que doivent suivre les acteurs (description de procédés grâce à des systèmes de workflow ou autres) et d'interconnecter ces procédés en négociant des "compromis" entre différents procédés.

Notre objectif est donc devenu de développer un modèle d'interconnexion de procédés qui soit flexible, qui dépasse les problèmes d'hétérogénéité et de fermeture des systèmes de gestion de procédés et qui puisse supporter les ressources des procédés à interconnecter et les fonctionnalités nécessaires à leur interconnexion. Pour ce faire, nous avons choisi l'approche qui nous semblait la plus prometteuse, à savoir « *une approche orientée services procédés* ». Notre modèle orienté services procédés permet aux entreprises d'interconnecter leurs procédés par la proposition des services procédés qu'elles peuvent réaliser et par la demande des services procédés qu'elles veulent externaliser. Ainsi, un service procédé fourni par une entreprise représente un procédé englobant les compétences et le savoir-faire de cette entreprise. Un service procédé requis par une entreprise, quant à lui, représente une activité que cette dernière a besoin de réaliser mais qu'elle ne peut faire elle-même par manque de compétences, de temps, etc.

Notre modèle formalise et développe le paradigme de coopération par échange de services procédés. Ce paradigme se concrétise par la définition, la publication, la recherche, la négociation et l'interconnexion de services procédés. Ainsi, chaque entreprise dispose de référentiels de définition et de publication de services procédés qui lui permettent d'exprimer ses compétences et ses besoins en termes de procédés-métiers. Chaque entreprise, souhaitant entrer en coopération, recherche les services procédés qui lui conviennent, négocie des contrats d'interconnexion de ses services procédés avec ceux de ses partenaires et peut alors commencer à coopérer.

Ce nouveau modèle d'interconnexion de procédés d'entreprises a été défini et présenté dans [29,30,35,36] ainsi que par Karim Baina dans son rapport de thèse[Baï03].

⊕ Vers un patron de coopération

Après cette description succinte de mes activités de recherche passées, je vais présenter en quelques mots, l'un des aspects de mon projet actuel.

La coopération et l'interconnexion de procédés d'entreprises telles que nous les avons présentés précédemment peuvent être vues comme un exemple d'implantation pour la coopération et l'interconnexion de composants particuliers que sont les procédés d'entreprises.

J'essaye, actuellement, d'apporter une solution générique au problème d'interconnexion et de coopération de composants. L'objectif est d'exprimer la généricité sous-jacente au concept d'interconnexion de composants distincts et hétérogènes (workflows, bases de données, processus, agents, etc.) sous forme d'un patron de coopération que l'on nommera « Service ». Le patron « Service » exprimera un schéma d'organisation structurelle et fondamentale pour des composants logiciels hétérogènes et coopératifs.

Notre patron «Service» essaye d'apporter une solution abstraite au besoin d'interconnexion pour la coopération de composants dans un contexte de coopération générique et globale. En isolant les principes fondamentaux, en identifiant la structure générale et en décrivant les collabo-

rations et l'implantation des composants de notre patron, nous avons pour objectif de réaliser un moyen utile à l'élaboration d'une conception orientée objet réutilisable, modulaire et facilement maintenable répondant au besoin de coopération.

La premier exemple d'application de ce patron « Service » est, bien sûr, l'interconnexion de services liés aux procédés d'entreprises. Notre proposition est une première itération pour fournir un haut niveau d'abstraction pour la conception, la construction et la maintenance d'outils d'interconnexion et de coopération entre composants. Des travaux sont en cours pour confirmer le bien-fondé de nos choix conceptuels avec un autre exemple d'application découvert et quelques pistes pour la recherche et l'analyse d'autres cas d'utilisation qui valideraient plus globalement nos travaux.

En ce qui concerne les publications sur ce projet, nous avons commencé à publier [37,39], mais nous devons encore valider l'approche par une ou plusieurs publications dans une conférence majeure du domaine.

Par ailleurs, nous cherchons dans nos travaux appliqués menés dans le cadre de la modélisation d'entreprise (réseau thématique UEML, GT ECI des GdR I3 et MACS, réseau d'excellence INTEROP), à valider notre patron par la découverte et la modélisation de cas concrets de coopération inter-entreprise.

5

Liste des publications personnelles

5.1 Liste des publications personnelles classées par ordre chronologique

Je ne détaille pas les rapports internes ou les rapports dans le cadre des projets auquels j'ai participé ou participe actuellement (ALF (Esprit I), COCAO (CTI-CNET), MTI, UEML (IST), INTEROP (NoE)). Un $\bigoplus$ signale les publications des quatre dernières années, c.a.d. les publications numérotés de 27 à 39.

[1] C. Godart, K. Benali et J.C. Derniame. Propositions pour un système de gestion d'objets. In *Actes CGL3 (3ème Colloque-Exposition de Génie Logiciel)*, Versailles, Mai 1986.

[2] K. Benali, J.C. Derniame et C. Godart. Système d'information et génie logiciel. In *Actes Congrès INFORSID'87*, Lyon, Juin 1987.

[3] C. Godart, K. Benali, N. Boudjlida, F. Charoy et J.C. Derniame. Les bases de données sur le chemin du génie logiciel. In *Actes Journées d'Etude AFCET, Sophia-Antipolis.* **Serge Miranda, éditeur, Des bases de données aux bases de connaissances, ediTESTS, Paris, 1987** .

[4] N. Boudjlida, C. Godart, J.C. Derniame, K. Benali et O. Gervaise. Vers des ateliers de logiciel supportant des méthodes. In *Actes Conférence JISI'88 (Journées Internationales des Sciences de l'Informatique)*, Tunis, Tunisie, Avril 1988.

[5] J.C. Derniame, H. A. Bahsoun, K. Benali, N. Boudjlida et C. Godart. Towards Assisted Software Processes. In *Proceedings CASE'88 (International Workshop on Computer Aided Software Engineering)*, Cambridge (Massachussetts), USA, July 1988.

[6] K. Benali, N. Boudjlida, F. Charoy et J.C. Derniame. A Model for Assisted Software Processes. In *Proceedings ICCI'89 (International Conference on Computing and Information)*, Toronto, Canada, R. Janicki et W.W. Koczkodaj, éditeurs, Canadian Scholars Press Inc., Toronto, 1989

[7] **K. Benali. Assistance et pilotage dans le développement de logiciel - Vers un modèle de description. Thèse de Doctorat de l'université de Nancy 1, Novembre 1989.**

[8] K. Benali, N. Boudjlida, F. Charoy, J.C. Derniame, C. Godart, Ph. Griffiths, V. Gruhn, Ph. Jamart, A. Legait, D.E. Oldfied et F. Oquendo. Presentation of the ALF Project. In *Proceedings of the first International Conference on Software Development Environments and Factories (SDE&F1)*, Berlin, RFA, **N. Madhavji, W. Shafer and H. Weber, éditeurs, Pitman Publishing, Londres, Grande Bretagne, 1990** .

[9] K. Benali, J. Lonchamp, C. Godart, et J.C. Derniame. La modélisation des procédés

de fabrication : une voie vers l'assistance intelligente en production de logiciel. In *Actes ERGO-IA'90 (Ergonomie et Informatique Avancée)*, Biarritz, Septembre 1990.

[10] K. Benali. The roles cooperating within a model driven IPSE. In *DRAFT No 1*, Chipot, éditeur, Metz, 1990.

[11] K. Benali. L'assistance et le pilotage en production de logiciel grace à la modélisation des procédés de fabrication. In *Actes DI'90 (Conférence des docteurs en Informatique)*, Dijon,Octobre 1990.

[12] J. Lonchamp, K. Benali, C. Godart et J.C. Derniame. Modeling and Enacting Software Processes : an Analysis. In *Proceedings IEEE COMPSAC'90 (International Computer Software & Applications Conference)*, Chicago, USA, Novembre 1990.

[13] J.C. Derniame, K. Benali, N. Boudjlida, C. Godart et J. Lonchamp. Roles cooperation through software process instantiation. In *Proceedings ISPW-6 (6th International Software Process Workshop)*, Hakodate, Japon, IEEE computer society press, Los Alamitos, USA, 1991.

[14] K. Benali et J.C. Derniame. Assistance and Guidance in Software Production through Software Process Modeling. In *Proceedings SEKE'91 (International Conference on Software Engineering & Knowledge Engineering)*, Skokie, USA, Mai 1991.

[15] J. Lonchamp, K. Benali, J.C. Derniame et C. Godart. Towards assisted Software Engineering Environments. **In la revue : Information and Software Technology**, Vol 33, No 8, Butterworth Scientific Limited, Londres, Grande Bretagne, Octobre 1991.

[16] K. Benali et J.C. Derniame. Software Processes Modeling :What, Who, and When. In *Proceedings Second European Workshop Software Process Technology*, Trondheim, Norvège, **J.-C. Derniame, éditeur, Lecture Notes in Computer Science, 635, Springer Verlag, 1992**.

[17] K. Benali et J.C. Colson. Analyse, conception, codage, test : vers un outil unique, graphique et intelligent. In *Actes (Conférence internationale du génie logiciel et ses applications)*, Toulouse , Décembre 1992.

[18] K. Benali, J.C. Colson. Modélisation et assistance au développement en CAO. In *Actes (Conférence internationale du génie logiciel et ses applications)*, Paris, Novembre 1995.

[19] K. Benali, J.C. Colson et J. Lahyane Formalism modelling and visualizing : an experimentation. In Proceedings Conference on Visual Data Exploration and Analysis III (SPIE'96), San José (Californie, USA), Janvier 1996.

[20] J.-C. Bignon, G. Halin, D. Léonard, O. Malcurat, K. Benali et C. Godart. Evolution de la maîtrise d'œuvre, pratiques coopératives et informatique répartie, In Mieux produire ensemble, Nancy, France, avril 1998

[21] K. Benali, M. Munier et C. Godart Cooperation Models in Co-Design In Proceedings International Conference on Agile Manufacturing, Minneapolis, USA, Juin 1998.

[22] K. Benali, G. Canals, C. Godart et S. Tata. An Approach for Developing Cooperation in Project-Enterprises, In Proceedings 3th International Conference on the Design of Cooperative Systems, Cannes, France, mai 1998

[23] J.-C. Bignon, G. Halin, K. Benali et C. Godart. Cooperation models in co-design : application to architectural design, In Proceedings International Conference on Design and Decision Support Systems - ICD&DSS'98, Maastricht, Hollande, juillet 1998.

[24] K. Benali, M. Munier, et C. Godart. Cooperation models in co-design, **In la revue :International Journal of Agile Manufacturing - IJAM**, Vol 2 N° 2, Published by th International Society of Agile Manufacturing, 1999

[25] M. Munier, K. Benali, C. Godart. A transactional approach for cross-organizational cooperation, In Proceedings Globecom (Global telecommunications conference), Rio de

Janeiro, Brésil, Décembre 1999

[26] M. Munier, K. Benali, C. Godart. DisCOO, a really distributed system for cooperation, **In la revue : Networking and Information Systems Journal**, Vol.2 N° 5-6, pp605-637, 1999, Hermes Science Publishing Limited, Oxford

⊕ [27] M. Munier, K. Baïna, K. Benali. A négociation model for CSCW , In Proceedings 5th International Conference on **Cooperative Information Systems, CoopIS'00** , Eilat, Israel, **Etzion/Scheuermann, editors, Lecture Notes in Computer Science, 1901, Springer Verlag, 2000**.

⊕ [28] M. Munier, K. Benali, C. Godart Un système coopératif basé sur les transactions , In Proceedings INFORSID'01, mai 2001, Martigny, Suisse

⊕ [29] K. Baïna, K. Benali, C. Godart A process service model for dynamic enterprise process interconnection, In Proceedings 6th International Conference on **Cooperative Information Systems, CoopIS'01** , Trento, Italie **Giunchiglia/Batini, editors, Lecture Notes in Computer Science, Springer Verlag, 2001**.

⊕ [30] K. Baïna, K. Benali, C. Godart, Les services procédés, une solution pour l'interconnexion des procédés d'entreprises, **In la revue : Ingénierie des Systèmes d'Information**, Numéro spécial Interopérabilité des Systèmes d'Information (ISI'01), Vol. 6, Num. 3, pp 145-181, 2001, Hermès Science Publications, Paris

⊕ [31] K. Baïna, S. Tata, K. Benali, Un modèle d'interaction de services pour la coopération des procédés, In Proceedings 7ème Conférence Maghrébine des Sciences Informatiques (MCSEAI'02),Vol. 2 pp 189-201, mai 2002, Annaba Algérie.

⊕ [32] K. Baïna, S. Tata, K. Benali, A Model for Process Service Interaction, Short paper and poster In Proceedings On the Move to Meaningful Internet Systems 2002 : CoopIS, DOA, and ODBASE, Confederated International Conferences CoopIS, DOA, and ODBASE 2002, October 2002, Irvine, USA, **R. Meersman, Z. Tari, et al., editors, Lecture Notes in Computer Science, 2519, Springer Verlag, 2002**.

⊕ [33] K. Benali, G. Bourguin, B. David, A. Derycke, C. Ferraris Collaboration / Coopération, In Proceedings Assises du GdR I3, Décembre 2002, Nancy, France, **J. Le Maitre, éditeur, CEPADUES éditions, 2002**.

⊕ [34] K. Baïna, S. Tata, K. Benali, A Model for Process Service Interaction, In Proceedings Business Process Managemen (BPM 2003 June 2003, Eindhoven, The Netherlands, **Van Der Alst et al., editors, Lecture Notes in Computer Science, LNCS 2678, pp 261-275, Springer Verlag, 2003**.

⊕ [35] K. Baïna, K. Benali, C. Godart, Dynamic Interconnection of Enterprise Workflow Processes, In Proceedings Concurrent Engineering 2003, July 2003, Madeira, Portugal, **R. Jardim-Gonçalves et al., editors, A.A. Balkema Publishers Verlag, 2003**.

⊕ [36] K. Baïna, K. Benali, C. Godart, Dynamic Interconnection of Heterogeneous Workflow Processes Through Services, In Proceedings On the Move to Meaningful Internet Systems 2003 : CoopIS, DOA, and ODBASE, Confederated International Conferences CoopIS, DOA, and ODBASE 2003, November 2003, Catania, Italy **R. Meersman, Z. Tari, D. Schmidt et al., editors, Lecture Notes in Computer Science, LNCS 2888, pp 444-461, Springer Verlag, 2003**.

⊕ [37] Walid Gaaloul, Karim Baïna, Khalid Benali, Claude Godart, Un patron pour l'interconnexion de composants distribués, In Proceedings CoPSTIC'03, Conférence en Sciences et Techniques de l'Information et de la Communication, Décembre, 2003, Rabat, Maroc

⊕ [38] Hervé Panetto, Giuseppe Berio, Khalid Benali, Nacer Boudjlida, Michaël Petit, A Unified Enterprise Modelling Language For Enhanced Interoperability Of Enterprise Models, In Proceedings INCOM2004, 11th IFAC Symposium on Information Control Problems

in Manufacturing, April, 2004, Salvador, Brasil

⊕ [39] Walid Gaaloul, Karim Baïna, Khalid Benali, Claude Godart, A Pattern for Interconnecting Distributed Components, Short paper and Poster In ICEIS'04, International Conference on Enterprise Information Systems, April 2004, Porto, Portugal

5.2 Liste des publications personnelles classées par type

Les publications sont numérotées (de 1 à 39) par ordre chronologique comme dans la liste précédente. Je ne détaille pas les rapports internes ou les rapports dans le cadre des projets auquels j'ai participé ou participe actuellement (ALF (Esprit I), COCAO (CTI-CNET), MTI, UEML (IST), INTEROP (NoE)) Un ⊕ signale les publications des quatre dernières années, c.a.d. les publications numérotés de 27 à 39.

Articles dans des revues internationales

[15] J. Lonchamp, K. Benali, J.C. Derniame et C. Godart. Towards assisted Software Engineering Environments. **In la revue : Information and Software Technology**, Vol 33, No 8, **Butterworth Scientific Limited, Londres, Grande Bretagne, Octobre 1991**.

[24] K. Benali, M. Munier, et C. Godart. Cooperation models in co-design, **In la revue :International Journal of Agile Manufacturing - IJAM**, Vol 2 N° 2, Published by th International Society of Agile Manufacturing, 1999

[26] M. Munier, K. Benali, C. Godart. DisCOO, a really distributed system for cooperation, **In la revue : Networking and Information Systems Journal,** Vol.2 N° 5-6, pp605-637, 1999, HERMES Science Publishing Limited, Oxford

Articles dans des revues nationales

⊕ [30] K. Baïna, K. Benali, C. Godart, Les services procédés, une solution pour l'interconnexion des procédés d'entreprises, **In la revue : Ingénierie des Systèmes d'Information, Numéro spécial Interopérabilité des Systèmes d'Information (ISI'01)**, Vol. 6, Num. 3, pp 145-181, 2001, Hermès Science Publications, Paris

Articles dans des conférences internationales avec comité de selection

[6] K. Benali, N. Boudjlida, F. Charoy et J.C. Derniame. A Model for Assisted Software Processes. In *Proceedings ICCI'89 (International Conference on Computing and Information)*, Toronto, Canada, R. Janicki et W.W. Koczkodaj, éditeurs, Canadian Scholars Press Inc., Toronto, 1989

[8] K. Benali, N. Boudjlida, F. Charoy, J.C. Derniame, C. Godart, Ph. Griffiths, V. Gruhn, Ph. Jamart, A. Legait, D.E. Oldfied et F. Oquendo. Presentation of the ALF Project. In *Proceedings of the first International Conference on Software Development Environments and Factories (SDE&F1)*, Berlin, RFA, **N. Madhavji, W. Shafer and H. Weber, éditeurs, Pitman Publishing, Londres, Grande Bretagne, 1990** .

[12] J. Lonchamp, K. Benali, C. Godart et J.C. Derniame. Modeling and Enacting Software Processes : an Analysis. In *Proceedings IEEE COMPSAC'90 (International Computer Software & Applications Conference)*, Chicago, USA, Novembre 1990.

[14] K. Benali et J.C. Derniame. Assistance and Guidance in Software Production through Software Process Modeling. In *Proceedings SEKE'91 (International Conference on Software Engineering & Knowledge Engineering)*, Skokie, USA, Mai 1991.

[19] K. Benali, J.C. Colson et J. Lahyane Formalism modelling and visualizing : an experimentation. In Proceedings Conference on Visual Data Exploration and Analysis III (SPIE'96), San José (Californie, USA), Janvier 1996.

[21] K. Benali, M. Munier et C. Godart Cooperation Models in Co-Design In Proceedings International Conference on Agile Manufacturing, Minneapolis, USA, Juin 1998.

[22] K. Benali, G. Canals, C. Godart et S. Tata. An Approach for Developing Cooperation in Project-Enterprises, In Proceedings 3th International Conference on the Design of Cooperative Systems, Cannes, France, mai 1998

[23] J.-C. Bignon, G. Halin, K. Benali et C. Godart. Cooperation models in co-design : application to architectural design, In Proceedings International Conference on Design and Decision Support Systems - ICD&DSS'98, Maastricht, Hollande, juillet 1998.

[25] M. Munier, K. Benali, C. Godart. A transactional approach for cross-organizational cooperation, In Proceedings Globecom (Global telecommunications conference), Rio de Janeiro, Brésil, Décembre 1999

⊕ [27] M. Munier, K. Baïna, K. Benali. A négociation model for CSCW , In Proceedings 5th International Conference on **Cooperative Information Systems, CoopIS'00** , Eilat, Israel, **Etzion/Scheuermann, editors, Lecture Notes in Computer Science, 1901, Springer Verlag, 2000**.

⊕ [29] K. Baïna, K. Benali, C. Godart A process service model for dynamic enterprise process interconnection, In Proceedings 6th International Conference on **Cooperative Information Systems, CoopIS'01** , Trento, Italie **Giunchiglia/Batini, editors, Lecture Notes in Computer Science, Springer Verlag, 2001**.

⊕ [34] K. Baïna, S. Tata, K. Benali, A Model for Process Service Interaction, In Proceedings Business Process Managemen (BPM 2003 June 2003, Eindhoven, The Netherlands, **Van Der Alst et al., editors, Lecture Notes in Computer Science, LNCS 2678, pp 261-275, Springer Verlag, 2003**.

⊕ [35] K. Baïna, K. Benali, C. Godart, Dynamic Interconnection of Enterprise Workflow Processes, In Proceedings Concurrent Engineering 2003, July 2003, Madeira, Portugal, **R. Jardim-Gonçalves et al., editors, A.A. Balkema Publishers Verlag, 2003**.

⊕ [36] K. Baïna, K. Benali, C. Godart, Dynamic Interconnection of Heterogeneous Workflow Processes Through Services, In Proceedings On the Move to Meaningful Internet Systems 2003 : CoopIS, DOA, and ODBASE, Confederated International Conferences CoopIS, DOA, and ODBASE 2003, November 2003, Catania, Italy **R. Meersman, Z. Tari, D. Schmidt et al., editors, Lecture Notes in Computer Science, LNCS 2888, pp 444-461, Springer Verlag, 2003**.

⊕ [38] Hervé Panetto, Giuseppe Berio, Khalid Benali, Nacer Boudjlida, Michaël Petit, A Unified Enterprise Modelling Language For Enhanced Interoperability Of Enterprise Models, In Proceedings INCOM2004, 11th IFAC Symposium on Information Control Problems in Manufacturing, April, 2004, Salvador, Brasil

Articles dans des conférences nationales avec comité de selection

J'ai mis dans cette catégories, en plus des conférences nationales, quelques conférences "internationales" mais dont la portée reste francophone ou régionale (Maghrébine par exemple).

[1] C. Godart, K. Benali et J.C. Derniame. Propositions pour un système de gestion d'objets.

In *Actes CGL3 (3ème Colloque-Exposition de Génie Logiciel)*, Versailles, Mai 1986.

[2] K. Benali, J.C. Derniame et C. Godart. Système d'information et génie logiciel. In *Actes Congrès INFORSID'87*, Lyon, Juin 1987.

[3] C. Godart, K. Benali, N. Boudjlida, F. Charoy et J.C. Derniame. Les bases de données sur le chemin du génie logiciel. In *Actes Journées d'Etude AFCET, Sophia-Antipolis.* **Serge Miranda, éditeur, Des bases de données aux bases de connaissances, ediTESTS, Paris, 1987** .

[4] N. Boudjlida, C. Godart, J.C. Derniame, K. Benali et O. Gervaise. Vers des ateliers de logiciel supportant des méthodes. In *Actes Conférence JISI'88 (Journées Internationales des Sciences de l'Informatique)*, Tunis, Tunisie, Avril 1988.

[9] K. Benali, J. Lonchamp, C. Godart, et J.C. Derniame. La modélisation des procédés de fabrication : une voie vers l'assistance intelligente en production de logiciel. In *Actes ERGO-IA'90 (Ergonomie et Informatique Avancée)*, Biarritz, Septembre 1990.

[17] K. Benali et J.C. Colson. Analyse, conception, codage, test : vers un outil unique, graphique et intelligent. In *Actes (Conférence internationale du génie logiciel et ses applications)*, Toulouse , Décembre 1992.

[18] K. Benali, J.C. Colson. Modélisation et assistance au développement en CAO. In *Actes (Conférence internationale du génie logiciel et ses applications)*, Paris, Novembre 1995.

⊕ [28] M. Munier, K. Benali, C. Godart Un système coopératif basé sur les transactions , In Proceedings INFORSID'01, mai 2001, Martigny, Suisse

⊕ [31] K. Baïna, S. Tata, K. Benali, Un modèle d'interaction de services pour la coopération des procédés, In Proceedings 7ème Conférence Maghrébine des Sciences Informatiques (MCSEAI'02),Vol. 2 pp 189-201, mai 2002, Annaba Algérie.

⊕ [33] K. Benali, G. Bourguin, B. David, A. Derycke, C. Ferraris Collaboration / Coopération, In Proceedings Assises du GdR I3, Décembre 2002, Nancy, France, **J. Le Maitre, éditeur, CEPADUES éditions, 2002**.

⊕ [37] Walid Gaaloul, Karim Baïna, Khalid Benali, Claude Godart, Un patron pour l'interconnexion de composants distribués, In Proceedings CoPSTIC'03, Conférence en Sciences et Techniques de l'Information et de la Communication, Décembre, 2003, Rabat, Maroc

Articles dans des workshops internationaux avec comité de selection

[5] J.C. Derniame, H. A. Bahsoun, K. Benali, N. Boudjlida et C. Godart. Towards Assisted Software Processes. In *Proceedings CASE'88 (International Workshop on Computer Aided Software Engineering)*, Cambridge (Massachussetts), USA, July 1988.

[13] J.C. Derniame, K. Benali, N. Boudjlida, C. Godart et J. Lonchamp. Roles cooperation through software process instantiation. In *Proceedings ISPW-6 (6th International Software Process Workshop)*, Hakodate, Japon, IEEE computer society press, Los Alamitos, USA, 1991.

[16] K. Benali et J.C. Derniame. Software Processes Modeling :What, Who, and When. In *Proceedings Second European Workshop Software Process Technology*, Trondheim, Norvège, **J.-C. Derniame, éditeur, Lecture Notes in Computer Science, 635, Springer Verlag, 1992**.

Articles dans des workshops nationaux avec comité de selection

[11] K. Benali. L'assistance et le pilotage en production de logiciel grace à la modélisation des procédés de fabrication. In *Actes DI'90 (Conférence des docteurs en Informatique)*,

Dijon,Octobre 1990.

[20] J.-C. Bignon, G. Halin, D. Léonard, O. Malcurat, K. Benali et C. Godart. Evolution de la maîtrise d'œuvre, pratiques coopératives et informatique répartie, In Mieux produire ensemble, Nancy, France, avril 1998

Les "inclassables"

[7] **K. Benali. Assistance et pilotage dans le développement de logiciel - Vers un modèle de description. Thèse de Doctorat de l'université de Nancy 1, Novembre 1989.**

[10] K. Benali. The roles cooperating within a model driven IPSE. In *DRAFT No 1*, Chipot, éditeur, Metz, 1990.

⊕ [32] K. Baïna, S. Tata, K. Benali, A Model for Process Service Interaction, Short paper and poster In Proceedings On the Move to Meaningful Internet Systems 2002 : CoopIS, DOA, and ODBASE, Confederated International Conferences CoopIS, DOA, and ODBASE 2002, October 2002, Irvine, USA, **R. Meersman, Z. Tari, et al., editors, Lecture Notes in Computer Science, 2519, Springer Verlag, 2002**.

⊕ [39] Walid Gaaloul, Karim Baïna, Khalid Benali, Claude Godart, A Pattern for Interconnecting Distributed Components, Short paper and Poster In ICEIS'04, International Conference on Enterprise Information Systems, April 2004, Porto, Portugal

Deuxième partie

Descriptions des activités de recherche récentes

Afin de différencier les articles auquels j'ai participé et qui servent de base à ces travaux de la bibliographie des différents domaines étudiés, mes propres références seront notées [i] avec i $\in$ [1..39] et se trouvent, à nouveau, dans une bibliographie précédant la bibliographie générale en fin de document

6

Coopération et échange de documents

6.1 Introduction

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. En effet les compétences nécessaires sont diverses et variées. Si nous prenons le domaine des Bâtiments et Travaux Publics qui va nous fournir l'exemple support de présentation de notre approche, les compétences pour réaliser une simple maison vont de la conception des volumes par l'architecte à la spécification des éléments structurels par l'ingénieur-structure en passant par la climatisation faite par le thermicien.

Dire que ces différents corps de métier sont amenés à travailler sur le même projet ne suffit pas, car outre le fait qu'ils travaillent tous, les différents acteurs coopérent et collaborent. En effet la réalisation d'un bâtiment ou de tout projet n'est pas faite en une succession d'étapes menées par un seul acteur mais par un ensemble d'acteurs coopérant à la réalisation d'un but commun. C'est le concept même d'ingénierie concourante qui permet un travail en synergie des différents acteurs autorisant une réduction des délais de production et une meilleure prise en compte, tout au long du projet, des spécificités et des compétences de chacun des acteurs.

Cette ingénierie concourante nécessite une coopération entre les différents acteurs et un échange des données produites par chacun d'entre eux. L'échange de données existe depuis longtemps sous des formes simples (courrier, fax, échange de disquettes,...) dans les entreprises. Dans le contexte actuel, avec la démocratisation d'internet, de plus en plus d'entreprises utilisent ce medium pour l'échange de données. Les différents acteurs ne sont alors plus soumis à l'obligation de travailler dans un même lieu géographique. On parle alors d'entreprise virtuelle ou d'entreprise-projet si le travail en commun dure le temps d'un projet (par exemple un chantier en BTP).

Cependant, cet échange non contrôlé ne permet pas une réelle synergie entre les différents acteurs. Il ne suffit pas simplement d'échanger des données pour travailler ensemble, il faut aussi contrôler et gérer ces échanges. Pour revenir à notre exemple support, il ne suffit pas simplement d'envoyer une version du plan de l'architecte au thermicien pour résoudre les problèmes liés à leur collaboration. En effet le plan de l'architecte envoyé ne correspond qu'à une version à un moment donné. La collaboration impliquant une certaine concourance dans le travail des deux acteurs, la production de différentes versions des documents échangés implique un contrôle des échanges. Ce contrôle des échanges est actuellement possible dans les environnements d'aide à la coopération, mais avec un contrôle en général centralisé. Dans le BTP, des systèmes tels que les armoires à plans (informatisés et centralisés) répondent partiellement au contrôle des échanges,

en centralisant ce contrôle et en gérant des droits spécifiques sur cette armoire à plan centrale.

L'objectif de ce chapitre est la description de notre modèle de coopération et d'échange de documents. Nous commençons par présenter, dans le paragraphe 6.2, les différents types d'environnement d'aide à la coopération. Nous présentons ensuite, dans le paragraphe 6.3, l'exemple support qui va nous permettre de présenter notre démarche et la formalisation du système. Les paragrahes 6.4 et 6.5 montrent la philosophie globale de notre système, à savoir, la distribution du contrôle des échanges et l'accès aux données échangées de manière standard. Le paragraphe 6.6 présente succintement la formalisation du contrôle des échanges. Nous commençons par expliciter notre choix d'une approche transactionnelle pour la réalisation de notre système de coopération. Puis nous présentons la formalisation en ACTA des différents éléments de notre système transactionnel coopératif (histoire locale, base locale, opération de transfert,...) et de notre critère de correction distribué (*DisCOO*-sérialisabilité). La formalisation complète est exposée dans [26] ainsi que dans le rapport de thèse de Manuel Munier[5].

Nous allons donc présenter les différents types d'environnement d'aide à la coopération, puis la la philosophie globale du modèle visé (c.a.d. la distribution du contrôle des échanges et l'accès aux données échangées de manière standard), et enfin nous expliciterons notre choix d'une approche transactionnelle pour la modélisation du système de coopération visé.

6.2 Environnements d'aide à la coopération existants

Dans le cas d'une application relativement complexe, il n'existe généralement pas d'acteur qui possède à lui seul la maîtrise et la connaissance intégrale de l'activité globale exécutée. Il est donc extrêmement difficile pour un acteur quelconque d'appréhender, "manuellement", toutes les conséquences d'une modification apportée à un document de l'application. Ce sont les raisons pour lesquelles il est indispensable d'utiliser des environnements mettant en œuvre des mécanismes de coordination et de communication suffisament sophistiqués, permettant ainsi de notifier et de propager les changements aux acteurs qui sont concernés et de s'assurer que les efforts de chacun des acteurs du projet sont coordonnés de manière à réduire l'impact d'une modification d'un document sur l'activité globale.

Ces environnements peuvent être classés en quatre catégories selon leur approche de la coopération. Nous avons tout d'abord les **gestionnaires de configurations** dont l'objectif est de gérer les versions et les configurations successives des différentes données partagées (cohérence des données). Viennent ensuite les **environnements centrés procédés** qui permettent de contrôler les états successifs des données partagées et/ou l'enchaînement des différentes activités. Dans le domaine des bases de données, les **systèmes transactionnels** garantissent que l'exécution en parallèle de plusieurs activités (encapsulées dans des transactions) n'introduit pas d'inconsistance au niveau des résultats produits par ces activités. La dernière catégorie, les **outils d'aide au travail coopératif**, est plus orientée vers les aspects communication et relations humaines de la coopération.

Afin de contrôler les mises-à-jour concurrentes effectuées par les différentes activités d'un système distribué, il est possible d'utiliser un outil de gestion de configurations (RCS [TW89], ClearCase [Atr94], Continuus, Adèle [BE94]). Son rôle est d'assurer le stockage des données partagées, appelées ressources, tout en gardant une trace de leur évolution (généralement sous la forme de leurs versions successives) et en contrôlant les accès concurrents effectués par les activités. Par exemple, si deux activités modifient en parallèle une même ressource, elles vont

[5] M. Munier. *Une architecture pour intégrer des composants de contrôle de la coopération dans un atelier distribué.* Thèse en Informatique de l'UHP, Nancy 1, 15 janvier 1999.

chacune développer, à partir d'une version initiale de cette ressource, ce que l'on appelle une branche de versions. De cette façon, chacune des activités travaille sur sa propre copie de la ressource, sans être pertubée par les modifications effectuées par l'autre activité. Lorsqu'elles auront terminé leur travail, une activité (éventuellement différente) sera chargée de fusionner ces deux branches afin de ne produire qu'une seule nouvelle version, i.e. que les modifications d'une des activités n'écraseront pas les modifications de l'autre. Le rôle du gestionnaire de configurations sera alors de mémoriser le fait que cette nouvelle version est dérivée des deux précédentes.

Toutefois, la plupart des gestionnaires de configurations reposent sur une architecture client/serveur (référentiel centralisé, éventuellement répliqué et/ou partitionné sur plusieurs serveurs). Ils ne conviennent donc pas à nos exigences de distribution et d'autonomie des activités. En outre, les gestionnaires de configurations sont essentiellement concernés par les problèmes de concurrence d'accès à un référentiel : gestion des versions et des configurations de ressources partagées. Ils ne définissent aucun contrôle de la coopération sur les échanges entre activités.

A la différence des gestionnaires de configurations, les outils de gestion de procédés sont principalement orientés vers la description des exécutions correctes en termes d'états successifs d'une ressource ou d'enchaînement des différentes activités : modèles à flots de tâches ou modèles de workflow [Coa97, AAAM96] (modèle des contrats [WR92]), règles événement/condition/action [Bar92a] (Adèle-Tempo [BE94], MARVEL [Bar92a, Bar92b]). Cette approche nécessite généralement de décrire l'application complète, i.e. en tenant compte de toutes les activités et de toutes les ressources. Si l'on intègre en plus les problèmes liés à la synchronisation des activités distribuées, le modèle obtenu devient alors rapidement complexe du fait de la complexité inhérente du contexte à modéliser.

Contrairement aux outils de gestion de procédés existants, notre objectif n'est pas de décrire comment les activités doivent travailler pour pouvoir coopérer, mais simplement de définir de quelle façon doivent se dérouler les échanges entre ces activités. Nous voulons imposer des contrôles sur les échanges de résultats entre activités, mais pas sur les activités elles-même ni sur la manière dont elles produisent ces résultats.

Le travail de groupe assisté par ordinateur (*"Computer Supported Cooperative Work"*) a pour objectif de permettre à des groupes d'usagers de collaborer à des buts communs au moyen d'un système informatique, appelé système collaboratif ou collecticiel. Toutefois, contrairement aux gestionnaires de configurations, aux outils de gestion de procédés ou aux systèmes transactionnels, un environnement CSCW n'est pas uniquement orienté vers le maintien de la cohérence des objets partagés (gestion des accès concurrents). Un tel environnement prend également en compte des aspects plus "humains" de la coopération tels que la gestion d'un groupe de personnes, les mécanismes de notification, les techniques de communication (messagerie électronique, vidéoconférences, ...). BSCW [BAB+97, BHT97] (partage d'informations au travers d'un référentiel centralisé), Wiki [6] (rédaction collective de documents via le Web) et Microsoft NetMeeting [7] (vidéo/audio conférence, partage d'applications, tableau blanc, forums de discussion synchrones) sont des exemples de tels collecticiels.

En ce qui concerne le contrôle de la cohérence, les collecticiels reposent sur les mécanismes développés dans les systèmes distribués ou les gestionnaires de configurations : verrouillage des objets, passage de jeton (ou "prise de tour"), détection des dépendances (conflits résolus par les utilisateurs). A la différence des systèmes transactionnels, ces techniques ont pour objectif d'assurer la cohérence des objets partagés et non de coordonner les activités qui coopèrent.

Dans la cas d'un système transactionnel [AA90, BN97], chaque activité est encapsulée dans

[6] Wiki : `http ://wiki.lri.fr :8080/scoop/scoop.wiki`
[7] NetMeeting : `http ://www.microsoft.com/netmeeting/features/`

une transaction dont l'exécution représente la séquence des opérations (lectures et écritures par exemple) invoquées par cette activité sur les objets partagés. Une transaction constitue l'unité d'exécution élementaire : soit la transaction se termine totalement et elle a les effets désirés sur les objets auxquels elle a accédé (la transaction est dite "validée"), soit elle est interrompue et elle n'a aucun effet (la transaction est dite "annulée"). Cette règle du "tout ou rien" (encore appelée propriété d'atomicité des transactions) permet d'assurer l'atomicité aux défaillances des transactions.

L'idée de base d'un système transactionnel est de garantir que si chaque transaction, prise individuellement, s'exécute correctement, alors leur exécution entremêlée (due à leurs accès concurrents aux objets partagés) devra être "correcte". Ceci est assuré par un critère de correction défini comme étant un ensemble de propriétés caractérisant l'histoire des exécutions considérées comme correctes. L'histoire de l'exécution entremêlée de plusieurs transactions est représentée par la séquence des opérations (lecture, écriture, ...) invoquées, concurremment, sur les objets partagés. Le critère de correction le plus répandu dans le domaine des applications traditionnelles (administration, banque, ...) est la "sérialisabilité". Celui-ci considère que l'exécution entremêlée de plusieurs transactions est correcte si elle produit un résultat équivalent à une exécution en série de ces transactions (l'exécution est dite sérialisable).

Ce critère est cependant trop strict puisqu'il garantit l'isolation des transactions (atomicité à la concurrence) : les états intermédiaires d'une transaction, en termes de valeurs des objets qu'elle manipule, ne sont pas visibles par d'autres transactions. La sérialisabilité ne supporte donc pas la coopération telle que nous l'avons définie, à savoir la possibilité offerte aux transactions de s'échanger des résultats intermédiaires au cours de leur exécution. De nouveaux modèles de transactions et critères de correction ont toutefois été définis pour relâcher l'isolation entre les transactions : transactions emboîtées [Mos81], transactions multiniveaux [Bee88], sagas [GMS87], ...

6.3 Présentation de l'exemple support

L'exemple qui va nous servir à illustrer nos concepts est celui que nous avons développé dans [23,24,25] (lui-même inspiré de celui présenté dans [RG96]). L'application considérée a pour objectif la conception d'un appartement sur un niveau ayant une salle de séjour et dont l'un des côtés est constitué intégralement d'une baie vitrée. Plusieurs partenaires interviennent lors de cette conception, formant ainsi une entreprise-projet. Ceux-ci travaillent sur trois documents qu'ils sont amenés à s'échanger : le **plan** rédigé conjointement par l'architecte et l'ingénieur-structure, l'**avis** de l'urbaniste et, dans une moindre mesure sur cet exemple, les spécifications de la **baie** vitrée définies par le thermicien. Dans cet exemple, pour des raisons de simplification, nous représenterons chaque partenaire par une activité

- l'**architecte** : Il s'occupe de la conception du bâtiment en termes de disposition des murs (disposition et taille des pièces), d'emplacements de fenêtres (luminosité), ..., c'est-à-dire de dessiner le plan du bâtiment en ne considérant que les aspects volume, espace et luminosité des appartements.
- l'**ingénieur-structure** : Son activité consiste à spécifier les éléments structurels de l'appartement et à garantir la stabilité de la construction. De tels éléments (murs porteurs, poutres ou colonnes de soutènement, ...) seront choisis de manière à respecter le plus possible l'harmonie et les choix de l'architecte.
- le **thermicien** : Il est en charge de la climatisation du bâtiment (chauffage, isolation thermique, ...). Nous limitons son intervention au choix du type de baie vitrée (matériaux,

épaisseur) en fonction de l'exposition et de la surface vitrée.
- l'**urbaniste** : Il contrôle le plan produit par l'architecte et émet un avis sur son travail en fonction de critères d'intégration urbaine du bâtiment. L'architecte doit alors tenir compte de cet avis pour rédiger un nouveau plan acceptable, et accepté, par l'urbaniste.
- le **pompier** : Avant d'émettre son avis, l'urbaniste peut demander conseil à un pompier pour vérifier que l'appartement est conforme aux normes incendie en vigueur (emplacement des issues de secours, des trappes de désenfumage, ...).

Les différents échanges de données entre les partenaires ne sont toutefois pas soumis aux mêmes règles. S'il est souhaitable, dans certains cas, d'encourager une coopération maximale entre les partenaires, dans d'autres il peut être nécessaire de fixer certaines contraintes. C'est le cas par exemple de l'urbaniste et de l'architecte : ils se partagent certains documents, mais seulement en lecture, chacun ne pouvant modifier les documents de l'autre.
- **client/serveur** : L'architecte (le "serveur") fournit différentes versions successives du plan au thermicien (le "client"). Les échanges se font donc uniquement de l'architecte vers le thermicien.
- **rédacteur/relecteur** : L'urbaniste (le "relecteur") lit, mais ne modifie pas, le plan qui lui est fourni par l'architecte (le "rédacteur"). Il rédige alors son avis qu'il transmet à l'architecte qui le lit, mais ne le modifie pas, pour mettre à jour son plan, et ainsi de suite.
- **écriture coopérative** : L'architecte et l'ingénieur-structure travaillent tous les deux à la rédaction du plan (même objet logique). Durant toute la durée de l'activité de conception ils le modifient, éventuellement simultanément, intégrent les modifications effectuées par l'un ou l'autre, dans le but de fournir au final une version de ce plan qui les satisfait tous les deux.

Notre objectif est donc de concevoir un modèle pour **coordonner**, via l'utilisation de différents **schémas de coopération**, les **échanges de données** entre différentes activités. Il nous faut pour cela pouvoir stocker des données au niveau d'une activité, définir des protocoles d'échange, puis appliquer ces protocoles pour contrôler les interactions entre les activités.

6.4 Contrôle des échanges

Lorsqu'une activité veut partager un objet donné avec une autre activité, il y a tout d'abord une phase de négociation pour définir le schéma de coopération qui contrôlera les échanges concernant cet objet entre ces deux activités. Il s'agit, au minimum, de s'assurer que le schéma que l'une des activités désire utiliser est connu de l'autre activité. Le résultat de cette négociation est un **contrat** passé entre les deux activités et fixant les règles de coopération à respecter pour le partage de l'objet concerné. Par exemple $Contract[archi, therm, \{plan\}, client\_server]$ représente le contrat passé entre l'architecte et le thermicien pour partager l'objet `plan` selon le mode de coopération "client/serveur". Nous obtenons ainsi pour chaque activité du système une **table de coopération** contenant tous les contrats signés par cette activité.

Lors d'un échange de données entre deux activités, ce qui constitue une opération de transfert, chacune de ces deux activités contrôlera localement (i.e. à partir des informations contenues dans sa table de coopération) que cet échange est correct par rapport au contrat qu'elles auront négocié toutes les deux. Si l'une ou l'autre détecte une violation du contrat (ou plus exactement du schéma de coopération "figurant" sur le contrat) cette opération d'échange sera refusée. Une activité comporte donc deux composants dédiés au contrôle de ses échanges avec les autres activités : un **protocole** chargé de la gestion de la table de coopération de l'activité ainsi que de l'évaluation des schémas de coopération figurant dans cette table ; un **coordinateur** pour

Figure 6.1 Exemple de tables de Coopération

activité	partenaire	objets	schéma	rôle
architecte	inge.-structure	{plan}	écriture coopérative	∼
architecte	thermicien	{plan}	client/serveur	serveur
architecte	urbaniste	{{plan},{avis}}	rédacteur/relecteur	∼

activité	partenaire	objets	schéma	rôle
thermicien	architecte	{plan}	client/serveur	client

contrôler que tous les accès réalisés sur l'espace de coopération respectent le protocole.

6.5 Accès aux données

Le référentiel local d'une activité est composé de deux parties : une partie publique, nommée **espace de coopération**, et une partie privée, nommée **espace de travail**. L'espace de coopération contient les versions des objets rendues publiques par l'activité, c'est-à-dire les versions pouvant être importées par d'autres activités, ainsi que les relations entre ces différentes versions le cas échéant. Les échanges de données entre activités seront réalisés entre leurs espaces de coopération respectifs. L'espace de travail permet quant à lui de présenter à l'utilisateur les objets de l'espace de coopération sous une forme utilisable par ses applications existantes (fichiers `.DXF` pour Autocad ou `.DOC` pour Word par exemple). C'est l'endroit où les applications vont effectivement manipuler les données.

L'utilisateur devra tout d'abord importer depuis une autre activité le document (sous forme d'objet avec ses "informations de contrôle") qu'il désire utiliser. Ce document sera alors stocké dans son espace de coopération. Afin de pouvoir le manipuler (sous forme de fichier par exemple) avec ses applications courantes, il devra ensuite transférer (opération `Check_Out`) ce document dans son espace de travail. Les modifications qu'il effectuera alors sur ce document ne seront pas visibles aux autres activités (l'espace de travail est une zone privée). Quand il jugera son travail terminé, il publiera (opération `Check_In`) la nouvelle version de ce document, ce qui aura pour effet de mettre à jour l'objet document stocké dans son espace de coopération. A partir de cet instant, ce document (sous forme d'objet) pourra être à son tour importé par d'autres activités.

Le référentiel local d'une activité est ainsi représenté par deux composants distincts : une zone d'échange, l'espace de coopération, accessible aux autres activités et dans laquelle sont stockés les objets partagés ; une zone privée, l'espace de travail, sur les données de laquelle l'activité peut travailler (appels d'outils existants) pour accomplir sa tâche. Les transferts entre ces deux zones sont réalisés à l'initiative de l'activité (et donc de l'utilisateur). En d'autres termes, c'est l'utilisateur qui décide du moment où il publie ses résultats (intermédiaires ou finaux) ainsi que du moment où il intègre, au niveau de son espace de travail, les modifications effectuées sur les objets partagés par les autres activités (préalablement importées dans son espace de coopération).

Notre approche ayant pour objectif de permettre la cooperation et le travail collaboratif sans changer les outils et les modes de production individuels, nous ne nous intéressons et ne gérons que les données de l'espace de coopération. L'espace de travail ne nous intéresse que comme destination d'un transfert depuis l'espace de cooopération ou comme source d'une publication vers celui-ci. Du point de vue des acteurs du système, c'est-à-dire les utilisateurs, il est nécessaire qu'ils puissent utiliser leurs applications courantes (Autocad, Word, emacs, gcc, ...) sur les données partagées. Ces données doivent donc être accessibles dans leur format natif, i.e. des fichiers `.DXF` ou `.DOC`, voire même des tuples dans une base de données ou bien des composants

dans un environnement orienté composants (ex : servant CORBA, composants EJB, services web)

6.6 Formalisation du contrôle des échanges

6.6.1 Approche transactionnelle

Dans le cas des applications coopératives distribuées et hétérogènes, la programmation explicite des interactions est généralement difficile à maîtriser puisqu'elle nécessite de prévoir toutes les interactions possibles (problème combinatoire). En outre, cette approche "programmation concurrente" est également source d'erreurs (interactions non prévues ou incompatibles, verrous mortels,...). A l'inverse, une approche "contrôle de la concurrence d'accès" a justement pour objectif de masquer cette complexité et de décharger au maximum les programmeurs d'applications des problèmes liés aux interactions entre activités concurrentes : un protocole de contrôle de la concurrence assure que le fonctionnement global d'un ensemble d'activités concurrentes est correct, à supposer bien entendu que chaque activité soit individuellement correcte. Nous avons donc choisi une **approche transactionnelle** plutôt qu'une approche basée sur les gestionnaires de configurations, les environnements centrés procédés ou les outils de CSCW. Le contrôle de la concurrence est ainsi réalisé par un critère de correction qui définit les exécutions (concurrentes) considérées comme étant correctes.

Une application coopérative est donc vue comme un ensemble de transactions qui accèdent "en même temps" à un ensemble d'objets. Chaque transaction encapsule une activité (supposée correcte) de l'application afin de lui cacher les problèmes liés à la concurrence d'accès. Une transaction peut ainsi être représentée par la séquence des opérations invoquées sur les objets manipulés. Mais le problème de la synchronisation de ces transactions est plus complexe que celui de la synchronisation des transactions dans des contextes applicatifs traditionnels (administration, banque, ...) [GR93] où les activités sont essentiellement concurrentes, i.e. pouvant s'exécuter de manière isolée en s'ignorant complètement l'une de l'autre. Dans notre cas, il serait donc plus exact de parler d'une approche "contrôle de la coopération" que d'une approche "contrôle de la concurrence". Nous aborderons le problème de la coopération dans les applications coopératives distribuées et hétérogènes sous l'angle des **systèmes transactionnels coopératifs**.

De nouveaux modèles de transactions dits "étendus" ont été développés afin de relâcher ou d'assouplir une ou plusieurs des propriétés ACID[8] des transactions classiques, d'organiser les transactions selon les spécificités de l'application (parallèlisme local, décomposition d'une transaction en sous-transactions, ...) ou de permettre l'utilisation de critères de corrections moins contraignants que la sérialisabilité.

Dans le modèle de transactions emboîtées (*"nested transactions"*) introduit dans [Mos81], une transaction peut être décomposée en plusieurs transactions, appelées sous-transactions[9], et ainsi de suite. Les transactions sont donc organisées de manière hiérarchique, la transaction se trouvant à la racine de la hiérarchie représentant la transaction globale. Ces relations structurelles entre les transactions nous permettent alors de définir des règles de contrôle de la concurrence moins strictes que dans les modèles de transactions classiques.

Le modèle des transactions multiniveaux (*"multi-level transactions"*) introduit dans [Bee88] est dérivé du modèle des transactions emboîtées pour permettre de prendre en compte la sémantique des objets manipulés (en exploitant les propriétés de commutativité de leurs opérations)

[8] ACID : **A**tomicité, **C**ohérence, **I**solation, **D**urabilité

[9] Une sous-transaction doit démarrer après sa transaction mère et se terminer avant elle.

ainsi que la manière dont ils sont construits[10] (en se basant sur la propriété d'indépendance entre les objets[11] [CF90]).

Le modèle des sagas introduit dans [GMS87] permet de résoudre le problème de l'isolation dans le cas de transactions longues. Une telle transaction, appelée saga, sera décomposée en plusieurs sous-transactions ACID, à la manière d'un modèle de transactions emboîtées à deux niveaux. Les règles de contrôle de la concurrence ne sont toutefois pas les mêmes. En premier lieu, une saga (transaction racine) ne possède pas la propriété d'isolation alors que ses sous-transactions l'ont. Par conséquent, une sous-transaction d'une saga peut voir les résultats partiels d'une autre saga. Deuxièmement, contrairement au modèle des transactions emboîtées, la validation d'une sous-transaction n'est pas conditionnée par la validation de la saga dont elle fait partie. Finalement, l'abandon d'une sous-transaction entraîne l'abandon de la saga correspondante.

Les modèles de transactions que nous avons étudiés jusqu'à présent sont tous indépendants des applications. En effet, qu'il s'agisse des transactions plates, emboîtées, multiniveaux ou encore des sagas, les notions définies par ces modèles ne sont pas liées à la sémantique des applications visées. Afin de relâcher les propriétés ACID des transactions (notamment en ce qui concerne la contrainte d'isolation), une autre solution consiste à définir des modèles de transactions spécifiques à des domaines particuliers.

C'est le cas par exemple des transactions coopératives introduites dans [NRZ92]. Le contrôle de la concurrence est réalisé sur la base d'une grammaire (dont les terminaux sont les opérations) qui caractérise les exécutions correctes. Une part importante du contrôle de la cohérence est ainsi reportée sur les utilisateurs qui devront écrire une grammaire correcte. Il est en outre difficile dans ce cas de garantir des propriétés générales sur les exécutions.

Une autre solution (travaux réalisés dans le cadre des langages concurrents à objets) consiste à intégrer le concept de transaction directement au niveau des objets en concevant des mécanismes permettant d'assurer la sérialisabilité. Ces mécanismes ont pour rôle de synchroniser les invocations des objets, appelés **objets atomiques** [Wei89, Wei84], concernés par la sérialisabilité (objets comptes bancaires par exemple). Chaque objet devient ainsi responsable de la gestion de ses accès concurrents effectués par les différentes transactions qui le manipulent. Un protocole de sérialisation (identique pour tous les objets atomiques) permet ensuite de garantir que tous les objets atomiques manipulés par une transaction définissent le même ordre de sérialisation pour cette transaction par rapport aux transactions concurrentes. En ce qui concerne la détection et le traitement des conflits, chaque objet atomique est libre de définir ses propres mécanismes. La prise en compte de la sémantique des objets dans ces mécanismes peut d'ailleurs permettre d'augmenter la concurrence entre les transactions [CFR89].

Si les différents modèles de transactions et critères de correction existants permettent effectivement de prendre en compte certains aspects de la coopération, aucun ne répond simultanément à nos besoins de **coopération** (les transactions ne doivent pas être isolées les unes des autres), de **distribution** (chaque transaction a sa propre copie des objets qu'elle manipule) et d'**autonomie** (pas de site central ; le contrôle des interactions est réalisé localement par chaque transaction). Ce sont les raisons pour lesquelles nous avons décidé de définir un nouveau modèle de transactions étendu qui satisfasse ces différents besoins de la coopération.

[10] Un objet X manipulé par une application est construit par niveaux d'abstraction successifs à partir d'objets existants. Ainsi, à partir d'objets primitifs (ex : pages, segments, ...) manipulables par des opérations élémentaires (lire, écrire), sont obtenus de proche en proche des objets typés plus élaborés, manipulables par des opérations appelées méthodes.

[11] Deux objets sont dits indépendants quand une modification de l'un n'a pas de conséquence sur la représentation de l'autre.

6.6.2 Bases locales, histoires locales et opérations de transfert

Pour définir notre modèle de transactions, la première étape consiste à formaliser l'aspect distribution de ces transactions. Il s'agit en fait d'attribuer un référentiel local à chaque activité du système, cette activité s'exécutant dans une transaction longue. Les objets manipulés par les transactions ne seront donc plus stockés dans un seul et unique référentiel, mais dans les référentiels locaux associés aux différentes activités. Celles-ci possèderont donc chacune leur propre copie des objets qu'elles manipulent. Par conséquent, contrairement aux objets des modèles de transactions classiques, un même "objet logique" de notre modèle aura donc plusieurs "instances" qu'il nous faudra distinguer. Un second point est la définition de l'histoire locale d'une transaction, i.e. la liste des événements du système qui seront journalisés pour une transaction.

Puisque chaque transaction possèdera sa propre copie des objets qu'elle manipule, les échanges de données entre transactions ne seront donc plus **implicites** (via des accès concurrents à un référentiel commun) mais **explicites** via l'utilisation d'**opérations de transfert** entre les référentiels respectifs de ces transactions. Le rôle de ces opérations sera de synchroniser, entre plusieurs transactions, les valeurs des différentes instances d'un même objet logique. Ces notions de bases/histoires locales et d'opérations de transfert nous permettront par la suite de définir des **règles de coopération locales** pour le partage des objets entre deux transactions, i.e. des règles qui ne sont définies que sur des **informations locales** à ces deux transactions.

Nous étudierons ensuite l'impact de ces changements sur des critères de correction "globaux" puisque basés sur l'histoire globale du système (ex : détection de cycles de dépendances entre activités). Nous formulerons notre modèle de transactions en utilisant les notations nouvellement introduites de manière à obtenir des critères de correction "locaux", c'est-à-dire se basant uniquement sur les histoires locales des transactions. L'idée sous-jacente est que le fait d'assurer la correction de l'exécution de chaque transaction par rapport aux transactions avec lesquelles elle interagit directement doit permettre d'assurer, implicitement, la correction de l'exécution globale du système.

Base Locale d'une Transaction Notre premier objectif est de représenter explicitement, au niveau du modèle de transactions, le fait qu'il n'existe plus un seul et unique référentiel commun à toutes les transactions du système, mais que celles-ci possèdent chacune leur propre base de données locale. Comme nous l'avons précédemment expliqué, cela signifie que si un objet est partagé (utilisations simultanées) entre plusieurs transactions, chacune d'entre elles possèdera sa propre copie de cet objet dans sa base locale. Par conséquent, un même **objet logique** aura donc plusieurs **instances** (une instance par transaction qui le manipule) qu'il nous faudra distinguer au niveau de notre modèle. Il ne s'agit pas simplement de réplicas (synchronisés automatiquement par le système) mais bien de copies indépendantes dont les transactions s'échangeront explicitement les valeurs. En outre, ces différentes instances pourront éventuellement avoir des valeurs différentes à un instant donné lors de l'exécution.

Lorsqu'une transaction t exécutera une opération op sur un objet ob (ce qui est noté $op_t[ob]$ dans le formalisme ACTA [CK94]), il sera donc nécessaire de préciser sur quelle instance de cet objet ob sera effectuée cette opération op. Nous utiliserons la notation $op_t[ob_{t'}]$ pour représenter le fait que l'opération $op_t[ob]$ est réalisée sur l'objet ob de la transaction t'. Ce que nous appelons la **base locale** d'une transaction t sera donc l'ensemble des objets ob_t.

Par exemple, la lecture par le thermicien du plan produit par l'architecte dans la base locale de celui-ci sera représentée par l'opération $read_{therm}[plan_{archi}]$.

Histoire Locale d'une Transaction Comme nous pouvons le constater, cette notation nous permet de représenter le fait qu'une transaction puisse effectuer des opérations sur des objets se trouvant non seulement dans sa propre base locale mais également dans les bases locales d'autres transactions (exemples : $op_{t_i}[ob_{t_j}]$ ou encore $read_{therm}[plan_{archi}]$). Ce sont précisément ces opérations qui nous permettront par la suite de représenter les **interactions** entre transactions, et en particulier les transferts de données entre transactions.

En utilisant cette nouvelle notation nous pouvons maintenant définir la notion d'histoire locale d'une transaction t. Il s'agit en fait d'identifier les événements du système qui concernent la transaction t. En ACTA [CK94], ces événements sont de deux types : les événements dits significatifs (`Begin`, `Commit`, `Abort`, ...) et les événements relatifs aux objets (invocations d'opérations sur les objets). Ce sont principalement ces derniers qui nous intéressent puisqu'ils vont nous permettre de contrôler la visibilité des objets entre les différentes transactions via la notion de "vue d'une transaction". En ACTA, la vue d'une transaction t, notée $View_t$, indique les objets et les états de ces objets visibles par la transaction t à un instant donné. En d'autres termes, la vue d'une transaction détermine quelles sont les opérations dont les effets (sur les objets) sont visibles par cette transaction. En outre, une vue étant une projection de l'histoire globale courante (notée H_{ct}), l'ordre partiel entre les opérations est conservé. En ce qui concerne notre modèle de transactions, les opérations dont les effets sont visibles par une transaction t sont :

1. les opérations invoquées par la transaction t elle-même, quels que soient les objets sur lesquels ces opérations ont été invoquées : $\{p_t[ob_{t'}] \in H_{ct}\}$
2. les opérations invoquées par des transactions t' (différentes de t) sur des objets se trouvant dans la base locale de la transaction t : $\{p_{t'}[ob_t] \in H_{ct}\}$

Par exemple, la lecture par le thermicien du plan produit par l'architecte dans la base locale de celui-ci, représentée par l'opération $read_{therm}[plan_{archi}]$, sera journalisée à la fois par les transactions *therm* et *archi*. Le thermicien la journalise puisqu'il s'agit d'une opération qu'il a lui même invoquée (cf. point 1). L'architecte la journalise car cette opération est invoquée sur un objet de sa propre base locale (cf. point 2).

La vue d'une transaction t de notre modèle, notée $View_t$, est définie de la manière suivante : $View_t = \{p_t[ob_{t'}] \in H_{ct}\} \cup \{p_{t'}[ob_t] \in H_{ct}\}$. Ceci determine quelles sont, parmi toutes les opérations de l'histoire globale, celles dont la transaction t a connaissance, et donc ainsi son **histoire locale** que nous noterons $H_{ct/t}$. A titre d'exemple, une certaine exécution de l'exemple donnera lieu à la journalisation des opérations suivantes par chacune des transactions *archi* et *inge* [12] (cf. figure 6.2).

Figure 6.2 Exemple de journalisation pour archi et inge

archi	**inge**
$Contract[archi, inge, \{plan\}, C/W]$	$Contract[archi, inge, \{plan\}, C/W]$
$write_{archi}[plan_{archi}]$,	
$read_{inge}[plan_{archi}]$,	$read_{inge}[plan_{archi}]$,
	$write_{inge}[plan_{inge}]$,
	$write_{inge}[plan_{inge}]$,
$read_{archi}[plan_{inge}]$,	$read_{archi}[plan_{inge}]$,
$write_{archi}[plan_{archi}]$.	
	$write_{inge}[plan_{inge}]$.

[12] NB : C/W est la notation utilisée pour l'écriture coopérative (Cooperative/Write).

Une histoire H (locale ou globale) est donc une succession d'invocations d'opération (*"object events"*). Nous noterons $H^{(ob)}$ la projection de l'histoire H par rapport à un objet ob particulier. Cette histoire $H^{(ob)} = p_1 \circ p_2 \circ \ldots \circ p_n$ indique à la fois l'ordre des opérations invoquées sur l'objet ob (i.e. l'opération p_i précède l'opération p_{i+1}, ce que nous noterons également $p_i \rightarrow p_{i+1}$) ainsi que la composition fonctionnelle des opérations. Ceci signifie que l'état s de l'objet ob dans l'histoire $H^{(ob)}$ est l'état produit par l'invocation successive (cf. relation d'ordre $\rightarrow$) des différentes opérations de $H^{(ob)}$ à partir d'un état initial s_0 (i.e. $s = state(s_0, H^{(ob)})$). Pour simplifier, nous noterons cet état $state(H^{(ob)})$.

En ACTA, une transaction accède et manipule des objets de la base en invoquant des opérations spécifiques aux différents objets. Chaque opération retourne une valeur et produit un état. Si s est l'état d'un objet, $return(s, p)$ renvoie le résultat produit par l'opération p. L'état de cet objet produit par l'exécution de l'opération p est noté $state(s, p)$. Deux opérations (également vues comme des appels de fonctions) sont alors dites conflictuelles pour un état $H^{(ob)}$ (ce qui sera noté $conflict(H^{(ob)}, p, q)$ ou plus simplement $conflict(p[ob], q[ob])$) si leurs effets sur cet état ou si leurs valeurs de retour ne sont pas indépendantes de leur ordre d'exécution. Deux opérations qui ne sont pas en conflit sont dites *compatibles*.

Puisque les changements d'état des objets sont observés via les valeurs de retour des opérations, nous pouvons définir une relation de dépendance entre deux opérations conflictuelles. En cas de conflit entre deux opérations (i.e. le prédicat $conflict(H^{(ob)}, p, q)$ vaut vrai), $return\_value\_independent(H^{(ob)}, p, q)$ est vrai si la valeur de retour de q est indépendante du fait que p précède q ou non, i.e., $return(H^{(ob)} \circ p,\ q) = return(H^{(ob)},\ q)$; sinon q est *"return-value dependent"* de p (noté $return\_value\_dependent(H^{(ob)}, p, q)$).

Comme nous pouvons le constater, ces deux définitions (introduites par le formalisme ACTA) ne sont pas, du point de vue de notre modèle de transactions, exprimées par rapport aux instances "physiques" des objets (ex ob_t), mais par rapport aux objets dits "logiques" (ex : ob).

En considérant l'objet logique *plan*, l'opération de lecture par le thermicien du plan produit par l'architecte, représentée par $read_{therm}[plan]$, sera en conflit avec une opération de modification de ce même plan par l'architecte $write_{archi}[plan]$. En outre, nous pouvons également dire que le prédicat suivant est vrai : $return\_value\_dependent(H^{(plan)}, write_{archi}[plan], read_{therm}[plan]$.

En fait, de la même façon que $conflict(p[ob], q[ob])$ est une abbréviation de $conflict(p_{t_i}[ob], q_{t_j}[ob])$ [13], nous pouvons considérer, dans un premier temps, que la notation $conflict(p_{t_i}[ob], q_{t_j}[ob])$ est une abbréviation de $conflict(p_{t_i}[ob_{t_k}], q_{t_j}[ob_{t_k}])$ [14]. Cela signifie que pour le moment, deux opérations ne peuvent être en conflit que si elles sont invoquées sur la même instance d'un objet logique. Par la suite, lorsque nous définirons la notion d'opérations de transfert, nous préciserons la signification de $conflict(p_{t_i}[ob_{t_{k_1}}], q_{t_j}[ob_{t_{k_n}}])$, i.e. d'un conflit entre deux opérations p_{t_i} et q_{t_j} invoquées sur des instances différentes d'un même objet logique ob. Idem pour le prédicat $return\_value\_independent(p_{t_i}[ob], q_{t_j}[ob])$.

Dans un premier temps, notre objectif est donc de pouvoir déterminer, lorsqu'une transaction t invoque une opération p sur un objet $ob_{t'}$, quelles sont les opérations, invoquées par d'autres transactions, avec lesquelles l'opération $p_t[ob_{t'}]$ peut être en conflit (ensemble noté $ConflictSet_t$ dans le formalisme ACTA). Les ensembles $View_t$ et $ConflictSet_t$ définissent ainsi les événements pouvant être invoqués. Plus précisément, les préconditions de ces événements[15] sont évaluées par rapport à ces deux ensembles. Si ses préconditions sont vérifiées, le nouvel événement est effecti-

[13] Les opérations p et q (invoquées respectivement par deux transactions t_i et t_j quelconques) sont conflictuelles au niveau de l'objet ob.

[14] Les opérations p_{t_i} et q_{t_j} sont conflictuelles quelle que soit l'instance ob_{t_k} de l'objet logique ob.

[15] Pour chaque événement, ses préconditions sont dérivées de la définition axiomatique de la transaction qui l'invoque.

vement invoqué puis journalisé dans les histoires locales des différentes transactions concernées par l'occurence de cet événement (cf. définition de la vue d'une transaction). Dans le cas contraire (i.e. au moins une des préconditions n'est pas vérifiée), l'invocation de cet événement est refusée.

En ce qui concerne notre modèle de transactions, les opérations pouvant être en conflit avec des opérations invoquées par une transaction t sont définies ci-dessous. Le prédicat $Inprogress(p)$ représente le fait que l'opération p est en cours d'exécution, i.e. qu'elle n'a pas encore été ni validée ni annulée.

- les opérations effectuées par des transactions t' ($t' \neq t$) sur des objets ob_t de la transaction t : $\{p_{t'}[ob_t] \in H_{ct} \mid Inprogress(p_{t'}[ob_t])\}$
  Par exemple, une opération invoquée par l'architecte sera potentiellement conflictuelle avec une opération $read_{therm}[plan_{archi}]$ invoquée par le thermicien sur l'objet *plan* de l'architecte.
- les opérations exécutées par des transactions t' ($t' \neq t$) sur des objets d'une tierce transaction t'' ($t'' \neq t$) sur lesquels la transaction t a précédemment invoqué des opérations :

  $$\{p_{t'}[ob_{t''}] \in H_{ct} \mid (t' \neq t) \wedge (\exists q\ q_t[ob_{t''}] \in H_{ct}) \wedge Inprogress(p_{t'}[ob_{t''}])\}$$

  Par exemple, une opération $read_{therm}[plan_{archi}]$ invoquée par le thermicien sera potentiellement en conflit avec une opération $read_{inge}[plan_{archi}]$ invoquée par l'ingénieur-structure car elles portent toutes les deux sur l'objet *plan* de l'architecte.

Par conséquent, pour une transaction t de notre modèle, l'ensemble des opérations pouvant être en conflit avec une des opérations de t est défini de la manière suivante :

$$\begin{aligned} ConflictSet_t = \quad & \{p_{t'}[ob_t] \in H_{ct} \mid Inprogress(p_{t'}[ob_t])\} \\ & \cup \{p_{t'}[ob_{t''}] \in H_{ct} \mid (t' \neq t) \wedge (\exists q\ q_t[ob_{t''}] \in H_{ct}) \wedge Inprogress(p_{t'}[ob_{t''}])\} \end{aligned}$$

Si l'on impose que tous les objets du système se trouvent dans une seule base de données commune à toutes les transactions, alors les définitions des ensembles $View_t$ et $ConflictSet_t$ deviennent équivalentes à celles fournies dans la définition axiomatique des transactions atomiques, à savoir :

- $View_t = H_{ct}$ où H_{ct} représente l'histoire globale courante du système
- $ConflictSet_t = \{p_{t'}[ob] \mid t' \neq t,\ Inprogress(p_{t'}[ob])\}$

Par conséquent, les notions de base locale (différentes instances ob_t d'un même objet logique *ob*) et d'histoire locale (chaque transaction n'a qu'une vision locale du système) ne sont qu'une extension apportée au formalisme ACTA. Elles vont nous permettre de modéliser les échanges de données **explicites** entre les transactions.

Opérations de transfert Chaque transaction de notre modèle possède dorénavant sa propre base de données locale dans laquelle elle stocke une copie physique (une instance) de tous les objets qu'elle partage. Par conséquent, si deux transaction *archi* et *therm* partage un même objet logique *plan*, chacune d'elles aura sa propre instance (respectivement $plan_{archi}$ et $plan_{therm}$) dans sa base locale. De quelle manière les transactions *archi* et *therm* vont-elles s'échanger les valeurs de cet objet *plan* ?

En fait, nous avons vu, lors de la présentation de notre nouvelle notation pour désigner les objets, qu'une transaction n'était pas limitée à l'invocation d'opérations uniquement sur "ses" objets. Par exemple, une transaction *therm* peut en effet exécuter une opération *read* sur l'objet *plan* se trouvant dans la base locale de la transaction *archi* : cette opération est noté $read_{therm}[plan_{archi}]$. Ce sont donc de telles opérations dites "de transfert" qui vont permettre aux activités de coopérer par échanges de données.

Dans un modèle de transactions classique, une interaction entre des transactions t_i et t_j au niveau d'un objet *ob* est représentée de la façon suivante : $(p_{t_i}[ob] \rightarrow q_{t_j}[ob]) \ \wedge \ conflict(p_{t_i}[ob], q_{t_j}[ob])$. Ceci signifie que les transactions t_i et t_j ont invoqué des opérations $p_{t_i}[ob]$ et $q_{t_j}[ob]$ sur un même objet *ob*, que l'opération $p_{t_i}[ob]$ précède l'opération $q_{t_j}[ob]$ (selon l'ordre partiel défini par l'histoire globale H_{ct}), et que ces deux opérations sont conflictuelles. En utilisant notre nouvelle notation pour désigner les objets, quelle est maintenant la signification de l'expression $(p_{t_i}[ob_{t_{k_1}}] \rightarrow q_{t_j}[ob_{t_{k_n}}]) \ \wedge \ conflict(p_{t_i}[ob_{t_{k_1}}], q_{t_j}[ob_{t_{k_n}}])$? Il existe en fait deux possibilités :

- soit il s'agit de la même instance ob_{t_k} de l'objet logique *ob*, ce qui nous ramène alors au cas $(p_{t_i}[ob_{t_k}] \rightarrow q_{t_j}[ob_{t_k}]) \ \wedge \ conflict(p_{t_i}[ob_{t_k}], q_{t_j}[ob_{t_k}])$
- soit il s'agit de deux instances différentes du même objet logique *ob*, i.e. $(p_{t_i}[ob_{t_{k_1}}] \rightarrow q_{t_j}[ob_{t_{k_n}}]) \ \wedge \ conflict(p_{t_i}[ob_{t_{k_1}}], q_{t_j}[ob_{t_{k_n}}])$ avec $t_{k_1} \neq t_{k_n}$

En ce qui concerne le premier cas, cela ne pose pas de problème particulier puisque nous pouvons utiliser les définitions classiques de la relation $\rightarrow$ de précédence entre opérations et du prédicat $conflict$. C'est évidemment le deuxième cas de figure qui nous intéresse, et plus particulièrement la signification de $conflict(p_{t_i}[ob_{t_{k_1}}], q_{t_j}[ob_{t_{k_n}}])$. Intuitivement, cela signifie que *"la valeur de l'objet ob de la transaction* t_{k_n} *(i.e.* $ob_{t_{k_n}}$*) dépend de la valeur de l'objet ob de la transaction* t_{k_1} *(i.e.* $ob_{t_{k_1}}$*)"*. En d'autres termes, nous avons un enchaînement d'opérations dans lequel les transactions t_{l_i} nous ont permis de "propager", de proche en proche, la valeur de l'objet $ob_{t_{k_1}}$ vers la transaction t_{k_n}.

De façon plus formelle, nous pouvons définir la notion d'opération de transfert de la manière suivante : une opération de transfert est une **opération virtuelle** qui est en fait la succession de deux opérations invoquées par une même transaction t_l, l'une pour accéder à la valeur de l'objet $ob_{t_{k_i}}$ (ex : $read_{t_l}[ob_{t_{k_i}}]$), l'autre pour modifier la valeur de l'objet $ob_{t_{k_j}}$ (ex : $write_{t_l}[ob_{t_{k_j}}]$), de manière à transférer la valeur de l'instance $ob_{t_{k_i}}$ de l'objet *ob* vers une autre de ses instances $ob_{t_{k_j}}$. Une telle opération sera notée $transfer_{t_l}[ob_{t_{k_i}}, ob_{t_{k_j}}]$ et correspondra à un couple d'opérations, i.e. $transfer_{t_l}[ob_{t_{k_i}}, ob_{t_{k_j}}] = (q_{t_l}[ob_{t_{k_i}}], p_{t_l}[ob_{t_{k_j}}])$, telles que $(q \in Read^{(ob)}) \wedge (p \in Write^{(ob)}) \wedge (q_{t_l}[ob] \ \rightarrow \ p_{t_l}[ob])$ avec $Read^{(ob)}$ représentant l'ensemble des opérations permettant d'accéder, sans la modifier, à la valeur de l'objet *ob*, et $Write^{(ob)}$ représentant l'ensemble des opérations permettant de modifier la valeur de l'objet *ob*.

Ce sont ces opérations qui vont nous permettre de "synchroniser" les histoires locales des différentes transactions. En effet, les relations *"read-from"* entre les transactions seront représentées par le fait qu'une opération $p_{t_l}[ob_{t_k}]$ sera journalisée à la fois par la transaction t_l (celle qui invoque l'opération) et par la transaction t_k (celle qui possède l'objet sur lequel est invoquée l'opération).

Nous pouvons ensuite définir la relation de dépendance sémantique $\xrightarrow{dep}$ entre objets. Cette relation nous permet de savoir, pour un objet *ob* donné, que la valeur de telle instance de cet objet a été produite à partir de la valeur de telle autre instance. La relation de dépendance sémantique $\xrightarrow{dep}$ exprime en fait au niveau des objets la relation de dépendance causale existant au niveau des événements. Des dépendances causales, on peut extraire les dépendances sémantiques entre objets. Nous pouvons ainsi définir la notion de conflit entre deux opérations invoquées sur des instances différentes d'un même objet logique :

$conflict(p_{t_i}[ob_{t_{k_i}}], q_{t_j}[ob_{t_{k_j}}]) \ \Leftrightarrow \ conflict(p_{t_i}[ob], q_{t_j}[ob]) \wedge (ob_{t_{k_i}} \xrightarrow{dep} ob_{t_{k_j}})$

Afin d'illustrer ceci, revenons à notre exemple support.

1. L'ingénieur-structure modifie le plan dans sa base locale : opération $write_{inge}[plan_{inge}]$.
2. L'architecte importe cette nouvelle version du plan : opération $transfer_{archi}[plan_{inge}, plan_{archi}]$, c'est-à-dire les opérations $read_{archi}[plan_{inge}]$ puis $write_{archi}[plan_{archi}]$. Ainsi, l'objet

$plan_{archi}$ dépend de l'objet $plan_{inge}$, i.e. $plan_{inge} \xrightarrow{dep} plan_{archi}$.

3. L'urbaniste importe à son tour cette nouvelle version du plan maintenant disponible chez l'architecte : opération $transfer_{urba}[plan_{archi}, plan_{urba}]$, c'est-à-dire les opérations $read_{urba}[plan_{archi}]$ puis $write_{urba}[plan_{urba}]$. Ainsi, l'objet $plan_{urba}$ dépend de l'objet $plan_{archi}$, i.e. $plan_{archi} \xrightarrow{dep} plan_{urba}$.

Par conséquent on a bien $conflict(write_{inge}[plan_{inge}], read_{urba}[plan_{archi}])$ puisque l'on a à la fois $conflict(write_{inge}[plan], read_{urba}[plan])$ au niveau de l'objet logique $plan$, et $plan_{inge} \xrightarrow{dep} plan_{urba}$ au niveau des instances de cet objet.

Nous pouvons maintenant définir les opérations d'importation et d'exportation d'un objet par une transaction depuis/vers une autre transaction comme étant des opérations de transfert particulières :

- **Importation** : Si $t_{l_i} = t_{k_{i+1}}$, alors $transfer_{t_{k_{i+1}}}[ob_{t_{k_i}}, ob_{t_{k_{i+1}}}] \equiv import_{t_{k_{i+1}}}[ob_{t_{k_i}}]$, i.e. la transaction $t_{k_{i+1}}$ importe (dans son objet $ob_{t_{k_{i+1}}}$) la valeur de l'objet ob de la transaction t_{k_i} (i.e. de l'objet $ob_{t_{k_i}}$).
- **Exportation** : Si $t_{l_i} = t_{k_i}$, alors $transfer_{t_{k_i}}[ob_{t_{k_i}}, ob_{t_{k_{i+1}}}] \equiv export_{t_{k_i}}[ob_{t_{k_{i+1}}}]$, i.e. la transaction t_{k_i} exporte la valeur de l'objet ob (i.e. $ob_{t_{k_i}}$) vers la transaction $t_{k_{i+1}}$ (i.e. vers l'objet $ob_{t_{k_{i+1}}}$).

Intuitivement, nous pouvons interpréter ceci de la façon suivante : en cas d'importation *"c'est le consommateur qui va chercher, de gré, l'information chez le producteur"*, alors qu'en cas d'exportation *"c'est le producteur qui diffuse, de force, l'information chez le consommateur"*.

Par exemple, l'importation, par l'urbaniste, de la version du plan produite par l'architecte sera formalisée par l'exécution des deux opérations suivante : lecture du document plan chez l'architecte ($read_{urba}[plan_{archi}]$) puis mise à jour de sa copie du plan ($write_{urba}[plan_{urba}]$). L'exportation du plan de l'ingénieur-structure vers l'architecte est réalisée de manière similaire : $read_{inge}[plan_{inge}]$ puis $write_{inge}[plan_{archi}]$.

Si l'on impose que tous les objets du système soient stockés dans une seule et unique base de données commune à toutes les transactions (ce qui est le cas dans les modèles de transactions classiques), on retrouve alors la notation habituelle $(p_{t_i}[ob] \rightarrow q_{t_j}[ob]) \wedge conflict(p_{t_i}[ob], q_{t_j}[ob])$.

6.6.3 Contrôle distribué

En fait, dans les sections suivantes, nous n'utiliserons cette notion de conflit entre deux opérations $p_{t_i}[ob_{t_{k_i}}]$ et $q_{t_j}[ob_{t_{k_j}}]$ invoquées sur des instances $ob_{t_{k_i}}$ et $ob_{t_{k_j}}$ différentes d'un même objet logique ob que pour démontrer que les propriétés garanties par les critères de correction "distribués" sont les mêmes que celles garanties par les critères de correction "centralisés". En pratique, puisque les critères de correction distribués que nous aurons définis ne seront basés que sur des informations locales aux transactions (i.e. sur les histoires locales des transactions), ceux-ci utiliseront simplement la notion de conflit classique entre les opérations d'une même histoire locale (opérations invoquées sur une même instance d'un objet logique).

Afin de permettre aux transactions de coopérer par le biais d'échanges de valeurs intermédiaires (i.e. de résultats produits en cours d'exécution et potentiellement inconsistants vis-à-vis du système), un nouveau critère de correction, *COO*, a été défini dans [Mol96]. En brisant la propriété d'isolation des transactions, ce critère accepte un certain nombre d'exécutions non sérialisables. Il permet ainsi aux transactions de coopérer selon trois paradigmes : client/serveur, rédacteur/relecteur, écriture coopérative. Toutefois, bien que les différentes transactions du système puissent être géographiquement distribuées, le contrôle de leur interactions reste centralisé

puisque ce critère est défini sur l'histoire globale du système. Notre objectif est donc de fournir une version distribuée[16] du critère de correction *COO*.

Nous rappelons tout d'abord ci-dessous les principaux axiomes des *COO*-transactions (figure 6.3). Nous définissons ensuite un ensemble d'axiomes équivalent (i.e. assurant les mêmes propriétés au niveau global), mais en fondant ces axiomes uniquement sur les histoires locales des transactions.

Critère de correction *COO* La *COO*-sérialisabilité a été définie dans [Mol96] pour permettre à un ensemble de transactions de s'exécuter de manière coopérative en relâchant la propriété d'isolation. Cela signifie que les différentes transactions peuvent coopérer en s'échangeant, au cours de leur exécution, des données par l'intermédiaire d'un référentiel commun. Dans *COO* nous distinguons donc deux types de résultats : les résultats **intermédiaires** produits par une transaction en cours d'exécution (ces résultats préliminaires sont potentiellement inconsistants et sujets à de nouvelle modifications), et les résultats **finaux** produits par une transaction à la fin de son exécution. Ce critère de correction peut être vu comme une extension de la sérialisabilité classique pour supporter la notion de résultat intermédiaire. Intuitivement, une exécution coopérative sera considérée comme étant correcte si elle respecte les règles de synchronisation suivantes :

1. Si une transaction produit un résultat intermédiaire, alors elle doit produire le résultat final correspondant lorsqu'elle est validée.
2. Si une transaction accède à un résultat intemédiaire d'une autre transaction, alors elle doit lire le résultat final correspondant avant de pouvoir produire ses propres résultats finaux. Lorsqu'une transaction accède à un résultat intemédiaire d'une autre transaction, elle devient dépendante de cette transaction. Lorsqu'elle lit le résultat final correspondant, cette dépendance est levée.
3. En cas de cycle dans le graphe des dépendances entre transactions (échanges bidirectionnels entre deux transactions par exemple), les transactions impliquées dans le cycle sont groupées. Les transactions d'un même groupe (noté T_{coo}) ont alors l'obligation de terminer leur exécution en même temps de manière indivisible.

Critère de correction distribué *DisCOO* Le modèle des *COO*-transactions est donc un modèle de transactions classique du point de vue de la manière dont les transactions accèdent aux objets (référentiel commun centralisé) et dont le critère de correction, la *COO*-sérialisabilité, coordonne les interactions (axiomes définis sur l'histoire globale du système). Fondamentalement, cela signifie qu'il est nécessaire d'avoir connaissance de **toutes** les opérations invoquées par **toutes** les transactions sur **tous** les objets partagés pour contrôler les interactions entre ces transactions.

Notre objectif étant d'assurer ce contrôle non plus de façon centralisée mais par le biais de contrôles effectués localement par chaque transaction, nous allons maintenant présenter une version distribuée de la *COO*-sérialisabilité : la *DisCOO*-sérialisabilité. En d'autres termes, par "distribuer un critère de correction" nous voulons exprimer le fait de définir des axiomes basés uniquement sur les informations contenues dans les histoires locales des transactions et qui, en étant ainsi vérifiés localement par chaque transaction, assurent implicitement les mêmes propriétés, d'un point de vue global, que le critère de correction initial. L'idée est donc de fournir une définition axiomatique qui soit équivalente à celle présentée figure 6.3 du point de vue des

[16] *DisCOO* est l'abréviation de "Distributed *COO*".

Figure 6.3 Principaux Axiomes des *COO*-Transactions

1. Pour une séquence d'opérations donnée, seule la dernière opération a pour obligation d'être validée. Seules les opérations de la sous-trace utile d'une transaction sont donc prises en compte.
   $(\texttt{Commit}_t \in H) \Rightarrow$
   $\forall ob\ \forall Q_t[ob]\ \exists q \in Q_t\ (\forall q' \in Q_t\ ,\ q' \neq q\ ,\ q'_t[ob] \rightarrow q_t[ob])\ \wedge\ (Commit_t[q_t[ob]] \in H)$
2. Toute opération validée doit dépendre (si dépendance il y a) d'une opération validée.
   $(Commit_t[q_t[ob]] \in H) \Rightarrow$
   $\exists p\ (\ return\_value\_dep(p_{t'}[ob], q_t[ob])\ \wedge\ (p_{t'}[ob] \rightarrow q_t[ob])\ )\ \Rightarrow$
   $(Commit_{t'}[p_{t'}[ob]] \in H)$
3. Toutes les transactions d'un groupe doivent converger vers un état final unique.
   $(\texttt{Commit}_t \in H)\ \wedge\ t \in T_{coo}\ \Rightarrow$
   $\forall ob\ \forall q\ (State(H^{(ob)}) \neq State(H^{(ob)} \circ q))\ \wedge\ (Commit_{t_i}[q_{t_i}[ob]] \in H)\ \wedge\ (t_i \neq t)\ \Rightarrow$
   $\exists p\ (\ return\_value\_dep(q_{t_i}[ob], p_t[ob])\ \wedge\ (State(H^{(ob)}) = State(H^{(ob)} \circ p))\ \wedge$
   $(q_{t_i}[ob] \rightarrow p_t[ob])\ \wedge\ (Commit_t[p_t[ob]] \in H)$
   $)$
4. Dans un groupe de transactions, soit toutes les transactions sont validées (`Commit`), soit aucune ne l'est (`Abort`).
   $\forall t_i, t_j \in T_{coo}\ ,\ t_i \neq t_j\ ,\ (t_i\ \mathcal{SCD}\ t_j)\ \wedge\ (t_i\ \mathcal{AD}\ t_j)$
5. Si une opération est annulée, alors toutes les opérations qui en dépendent sont également annulées.
   $(Abort_t[p_t[ob]] \in H)\ \Rightarrow$
   $(\ return\_value\_dep(p_t[ob], q_{t_i}[ob])\ \wedge\ (p_t[ob] \rightarrow q_{t_i}[ob])\ )\ \Rightarrow\ (Abort_{t_i}[q_{t_i}[ob]] \in H)$

propriétés garanties mais dont les axiomes peuvent être vérifiés localement par chaque transaction. Par "équivalent" il s'agit essentiellement de prouver qu'assurer la *DisCOO*-sérialisabilité à chaque nœud du réseau de transactions nous assure également la *COO*-sérialisabilité entre les transactions, i.e. que toute exécution *DisCOO*-sérialisable est également *COO*-sérialisable.

Intuitivement, nous pouvons interpréter les deux premiers axiomes des *COO*-transactions de la manière suivante : une transaction ne peut être validée (`Commit`) que si *"elle est à jour"*, i.e. que les résultats lus sont bien les derniers résultats en date produits par leurs transactions respectives, et que ces résultats sont "stabilisés" (i.e. que les opérations qui les ont produits ont été validées). En d'autres termes, si une opération q est dépendante d'une opération p (i.e. $return\_value\_dep(p, q)$), alors la dernière occurence de q doit dépendre de la dernière occurence de p. Ceci sera représenté par le prédicat up_to_date défini ci-dessous[17] et représentant le fait que la transaction t_j est "à jour" par rapport à la transaction t_k en ce qui concerne les objets de l'ensemble O.

$up\_to\_date(t_j, t_k, O) \equiv \forall ob \in O \quad \forall Q_{t_j}[ob_{t_k}]$
$(\ \exists t_i\ \exists p\ rvd(p_{t_i}[ob_{t_k}], LastOcc(Q_{t_j}[ob_{t_k}]))\ \wedge\ (p_{t_i}[ob_{t_k}] \rightarrow LastOcc(Q_{t_j}[ob_{t_k}]))\ ) \Rightarrow$
$(\ \nexists p' \in Sequence(p_{t_i}[ob_{t_k}]) \quad (LastOcc(Q_{t_j}[ob_{t_k}]) \rightarrow p'_{t_i}[ob_{t_k}])\ )$

A noter que la définition du prédicat up_to_date utilise les fonctions $LastOcc$ et $Sequence$ qui nous fournissent respectivement la dernière occurence d'une opération dans une séquence d'opérations donnée et la séquence d'opérations à laquelle appartient l'occurence d'une opération donnée.

[17] Nous utiliserons l'abréviation *rvd* pour désigner le prédicat *return_value_dependent*.

Séquence d'Occurences d'une Opération : Une même opération $p_{t_i}[ob_{t_j}]$ peut être invoquée plusieurs fois au cours de l'exécution. Bien que notées de manière identique dans l'histoire, il s'agit d'occurences différentes. L'ensemble des occurences de cette opération $p_{t_i}[ob_{t_j}]$ (ordonné selon la relation de précédence $\rightarrow$) est appelé **séquence** de l'opération $p_{t_i}[ob_{t_j}]$. Cette séquence sera également notée $P_{t_i}[ob_{t_j}]$. La fonction *Sequence* nous renvoie donc l'ensemble des occurences d'une opération donnée à partir de l'une de ses occurences, i.e.

$Sequence(occ) = \{p_{t_i}[ob_{t_j}] \in H \mid occ \equiv p_{t_i}[ob_{t_j}]\}$

Si nous prenons l'exemple de l'architecte, celui-ci va modifier sa copie du plan du bâtiment à plusieurs reprises. Au niveau de la transaction représentant l'acivité `archi`, ceci sera matérialisé par l'invocation (et la journalisation) de plusieurs opérations $write_{archi}[plan_{archi}]$ successives. La liste ordonnée de ces opérations sera appelée la **séquence** de l'opération $write_{archi}[plan_{archi}]$ et sera notée $WRITE_{archi}[plan_{archi}]$.

Dernière Occurence d'une Opération : La fonction *LastOcc* nous renvoie la dernière occurence d'une séquence d'opérations, i.e.

$LastOcc(S) = occ \in S \quad$ telle que $\quad \nexists\, occ' \in S\ (occ' \neq occ) \quad occ \rightarrow occ'$

Critère *DisCOO* **:** Nous pouvons maintenant exprimer le fait que la validation (`Commit`) d'une transaction n'est possible que si cette transaction est *up_to_date* par rapport à toutes les transactions dont elle dépend et si celles-ci sont elles-même validées (et ainsi de suite de manière récursive) par les deux axiomes définis ci-après.

$(\texttt{Commit}_{t_j} \in H_{t_k}) \Rightarrow \forall ob \quad up\_to\_date(t_j, t_k, \{ob\})$

$(\texttt{Commit}_{t_j} \in H_{t_k}) \Rightarrow$
$\quad (\ \exists p, q\ (rvd(p_{t_i}[ob_{t_k}], q_{t_j}[ob_{t_k}]) \wedge (p_{t_i}[ob_{t_k}] \rightarrow q_{t_j}[ob_{t_k}]))\ ) \Rightarrow$
$\quad (\texttt{Commit}_{t_i} \in H_{t_k})$

Si la relation *return_value_dependent* introduit un cycle de dépendances entre certaines transactions, cela signifie que toutes ces transactions devront être *up_to_date* les unes par rapport aux autres, i.e. devront avoir atteint un consensus sur les valeurs des objets partagés. Ceci nous garantit donc la convergence du groupe vers un état final unique. Le deuxième axiome précise également qu'une transaction t_j dépendant d'une transaction t_i ne peut être validée ($\texttt{Commit}_{t_j} \in H_{t_k}$) que si $\texttt{Commit}_{t_i} \in H_{t_k}$. Comme nous pouvons le constater, cet axiome n'impose pas que la validation de t_i **précède** celle de t_j (i.e. $\texttt{Commit}_{t_i} \rightarrow_{H_{t_k}} \texttt{Commit}_{t_j}$). Intuitivement, cela signifie qu'en cas de cycle il n'y aura pas d'interblocage. Il suffira tout simplement que tous les `Commit` des transactions impliquées dans le cycle "apparaissent" **en même temps** dans l'histoire (i.e. que ces transactions soient validées en même temps).

Finalement, il nous faut également garantir que si une opération p est annulée, alors toutes les opérations q qui en dépendent (i.e. $return\_value\_dep(p, q)$) seront à leur tour être annulées (cf. axiome ci-dessous). Cette condition impose que toutes les transactions d'un groupe soit annulées si l'une d'entre elles venait à être annulée.

$(Abort_{t_j}[p_{t_j}[ob_{t_k}]] \in H_{t_k}) \Rightarrow$
$\quad rvd(p_{t_j}[ob_{t_k}], q_{t_i}[ob_{t_k}]) \wedge (p_{t_j}[ob_{t_k}] \rightarrow q_{t_i}[ob_{t_k}]) \Rightarrow$
$\quad (Abort_{t_i}[q_{t_i}[ob_{t_k}]] \in H_{t_k})$

Pour résumer, la *DisCOO*-sérialisabilité est définie par les trois axiomes de la figure 6.4. Ce critère de correction nous permet d'offrir une plus grande autonomie aux transactions de notre modèle. En effet, pour contrôler que les interactions d'une transaction avec ses transactions partenaires sont correctes, au moment de sa validation (`Commit`) par exemple, seules des **informations locales** à cette transaction sont utilisées, c'est-à-dire des informations journalisées soit dans sa propre histoire locale, soit dans les histoires locales des transactions avec lesquelles

elle a échangé des données. Cela signifie en particulier que la *DisCOO*-sérialisabilité n'est plus fondée sur la détection de cycles de dépendances entre les transactions, car déterminer de tels cycles nécessite d'avoir une vue globale du système. La *DisCOO*-sérialisabilité effectue plutôt un contrôle "de proche en proche". Par conséquent, la notion de groupe de transactions, bien que toujours sous-jacente, n'est plus utilisée explicitement par le modèle des *DisCOO*-transactions.

Figure 6.4 Axiomes de la *DisCOO*-Sérialisabilité

1. $(\texttt{Commit}_{t_j} \in H_{t_k}) \Rightarrow \forall ob \quad up\_to\_date(t_j, t_k, \{ob\})$

2. $(\texttt{Commit}_{t_j} \in H_{t_k}) \Rightarrow$
   $(\exists p, q \ (rvd(p_{t_i}[ob_{t_k}], q_{t_j}[ob_{t_k}]) \ \wedge \ (p_{t_i}[ob_{t_k}] \rightarrow q_{t_j}[ob_{t_k}]))) \Rightarrow$
   $(\texttt{Commit}_{t_i} \in H_{t_k})$

3. $(Abort_{t_j}[p_{t_j}[ob_{t_k}]] \in H_{t_k}) \Rightarrow$
   $rvd(p_{t_j}[ob_{t_k}], q_{t_i}[ob_{t_k}]) \ \wedge \ (p_{t_j}[ob_{t_k}] \rightarrow q_{t_i}[ob_{t_k}]) \Rightarrow$
   $(Abort_{t_i}[q_{t_i}[ob_{t_k}]] \in H_{t_k})$

La plate-forme d'expérimentation nous a permis de réaliser et de valider les quatre services essentiels pour la coopération que nous avons identifiés : un service d'**espace de coopération** pour gérer les ressources de la base de données locale d'une activité ; un service d'**espace de travail** qui permet à l'utilisateur de manipuler les ressources de son espace de coopération à l'aide de ses applications habituelles (Word, Autocad, emacs, gcc, ...) ; un service de **coordination** qui garantit que toutes les requêtes transmises au service d'espace de coopération sont validées par le service de protocole (cf. couche "contrôle des interactions") ; un service de **protocole** chargé, pour une activité donnée, du contrôle effectif des interactions de cette activités avec les autres activités du système (contrats négociés, histoire locale de l'activité, vérification des schémas de coopération par rapport à l'histoire locale courante). Ces services ont été développés par Manuel Munier en tant que services CORBA de manière à ce qu'ils puissent être à leur tour utilisés par d'autres services ou objets CORBA, au même titre que les services CORBA standard. En fait, tous les composants de l'architecture du système développé sont eux-même des objets CORBA.

Le principal composant du service de coordination est le coordinateur. Son rôle au sein de notre architecture est de "filtrer" les échanges d'une activités avec les autres activités du système, c'est-à-dire de refuser tout échange qui ne vérifierait pas le(s) schéma(s) de coopération négocié(s). Sinon la requête est transmise à l'espace de coopération de l'activité.

Le cœur du prototype *DisCOO* est le service de protocole, et plus particulièrement les composants chargés de la gestion de l'histoire locale de cette activité et de la vérification d'un schéma de coopération donné par rapport à cette histoire locale. Au niveau formel, nous avons défini un schéma de coopération comme étant un ensemble de règles de coopération. Ces règles sont elles-même des prédicats destinés à être évalués par rapport à l'histoire locale courante de l'activité. Par exemple, le schéma de coopération "écriture coopérative" est représenté par le prédicat *DisCOO* (qui utilise le prédicat *up_to_date*) :

$$up\_to\_date(t_j, t_k, O) \equiv \forall ob \in O \quad \forall Q_{t_j}[ob_{t_k}]$$
$$(\exists t_i \ \exists p \ rvd(p_{t_i}[ob_{t_k}], LastOcc(Q_{t_j}[ob_{t_k}])) \ \wedge \ (p_{t_i}[ob_{t_k}] \rightarrow LastOcc(Q_{t_j}[ob_{t_k}]))) \Rightarrow$$
$$(\nexists p' \in Sequence(p_{t_i}[ob_{t_k}]) \quad (LastOcc(Q_{t_j}[ob_{t_k}]) \rightarrow p'_{t_i}[ob_{t_k}]))$$

$$DisCOO(t_i,t_j,O) \equiv \begin{cases} \forall t_1,t_2\ (t_1 \in \{t_i,t_j\},\ t_2 \in \{t_i,t_j\} - t_1) \\ \quad (\texttt{Commit}_{t_1} \in H_{t_2}) \ \Rightarrow\ up\_to\_date(t_1,t_2,O) \\ \\ \forall ob \in O \quad \forall t_1,t_2\ (t_1 \in \{t_i,t_j\},\ t_2 \in \{t_i,t_j\} - t_1) \quad \forall t_k\ (t_k \neq t_1) \\ \quad (\texttt{Commit}_{t_1}[q_{t_1}[ob_{t_2}]] \in H_{t_2}) \ \Rightarrow \\ \qquad rvd(p_{t_k}[ob_{t_2}], q_{t_1}[ob_{t_2}]) \ \wedge\ (p_{t_k}[ob_{t_2}] \rightarrow_{H_{t_2}} q_{t_1}[ob_{t_2}]) \ \Rightarrow \\ \qquad (\texttt{Commit}_{t_k}[p_{t_k}[ob_{t_2}]] \in H_{t_2}) \\ \\ \forall ob \in O \quad \forall t_1,t_2\ (t_1 \in \{t_i,t_j\},\ t_2 \in \{t_i,t_j\} - t_1) \quad \forall t_k\ (t_k \neq t_1) \\ \quad (Abort_{t_1}[p_{t_1}[ob_{t_2}]] \in H_{t_2}) \ \Rightarrow \\ \qquad rvd(p_{t_1}[ob_{t_2}], q_{t_k}[ob_{t_2}]) \ \wedge\ (p_{t_1}[ob_{t_2}] \rightarrow_{H_{t_2}} q_{t_k}[ob_{t_2}]) \Rightarrow \\ \qquad (Abort_{t_k}[q_{t_k}[ob_{t_2}]] \in H_{t_2}) \end{cases}$$

Plutôt que de reprogrammer des algorithmes représentant ces différentes règles de coopération, celles-ci ont été implantées directement sous la forme de prédicats Prolog. En procédant de cette manière, les schémas et règles de coopération (formalisés en ACTA) sont réellement implantés sous la forme de propriétés (les prédicats Prolog) évaluées sur les histoires locales des activités. Du point de vue de la programmation cela nous permet de créer et de modifier les schémas de coopération de façon "naturelle" par rapport à leur définition en ACTA. L'ajout de nouveaux schémas de coopération au prototype *DisCOO* en est ainsi grandement facilitée.

Nous verrons plus tard comment cette approche nous a permis d'introduire de nouveaux schémas de coopération permettant de modéliser la négociation. Pour conclure la présentation du prototype *DisCOO* signalons que tous les composants ont été programmés en Java au-dessus d'un ORB lui-même développé en Java (JacORB). Quant à l'interpréteur Prolog, il est lui aussi développé en Java. Le fait d'utiliser le langage Java nous permet ainsi de déployer *DisCOO* dans tout environnement (Windows 98/NT, Unix, MacOS, ...) pour lequel une machine virtuelle Java est disponible. Le prototype *DisCOO* est donc complètement découplé de l'environnement d'exécution, tant du point de vue du système d'exploitation (bytecode Java) que de l'infrastructure de communication (CORBA). A titre d'exemple *DisCOO* a été testé sur un réseau local hétérogène composé à la fois de PC fonctionnant sous Windows NT, de stations de travail Solaris et de PC sous Linux.

6.7 Conclusion

Les travaux décrits dans ce chapitre et détaillés dans [26] ont donné lieu à deux résultats. D'un point de vue formel tout d'abord, il s'agit de la définition d'un nouveau **modèle de transactions avancé**, les *DisCOO*-transactions, ainsi que d'un **critère de correction distribué**, la *DisCOO*-sérialisabilité (avec trois schémas de coopération : "écriture coopérative", "client/serveur" et "rédacteur/relecteur"). Le second résultat est la définition d'une **architecture** nous permettant de construire de telles applications en "connectant", une fois qu'un schéma de coopération aura été négocié, les différentes activités distribuées. Cette architecture a ensuite été mise en œuvre sous forme de **services de base pour la coopération** au sein du prototype *DisCOO* développé par Manuel Munier.

Comme nous l'avons vu dans la section 6.2, les environnements d'aide à la coopération actuels ne sont pas adaptés aux applications distribuées. La plupart d'entre eux sont basés sur une architecture client/serveur dans laquelle, bien que les différentes activités puissent s'exécuter sur des sites géographiquement distribués, le contrôle de la coopération (cohérence des données partagées, enchaînement des activités,...) reste centralisé sur un référentiel commun (éventuellement répliqué et/ou partitionné sur plusieurs serveurs). La centralisation facilite la synchronisation et le contrôle de l'activité globale, mais son principal inconvénient est la nécessité, pour les activités

clientes, d'être connectées en permanence à ce serveur, ou du moins de devoir s'y connecter pour échanger des données avec une autre activité.

Un second inconvénient de ces environnements d'aide à la coopération concerne les techniques mises en œuvre pour coordonner les différentes activités. Les gestionnaires de versions et de configurations se limitent à assurer le stockage des données partagées tout en gardant une trace de leur évolution (sous la forme d'un graphe des versions successives) et en contrôlant les accès concurrents effectués par les activités. De leur côté, les environnements centrés procédés nécessitent généralement de décrire toute l'application, c'est-à-dire de prévoir toutes les interactions possibles entre toutes les activités, une tâche qui peut rapidement devenir très complexe. Quant aux systèmes transactionnels, ils permettent de garantir que si chaque activité (encapsulée dans une transaction), prise individuellement, s'exécute correctement, alors leur exécution entremêlée (due à leurs accès concurrents aux objets partagés) sera conforme à un ensemble de propriétés appelé critère de correction. Les critères de correction existants sont toutefois mieux adaptés aux transactions de courte durée et plutôt isolées les unes des autres qu'aux activités coopératives distribuées qui nous intéressent.

Partant des travaux réalisés dans *COO* en ce qui concerne la correction syntaxique des interactions coopératives, la première étape de notre travail consistait donc à passer d'un contrôle des accès concurrents (interactions implicites) à un contrôle des échanges de données entre transactions (interactions explicites). Nous avons pour cela défini les notions de **référentiel local** d'une transaction, d'**histoire locale** d'une transaction, et d'**opération de transfert** entre transactions. L'idée est que chaque transaction posséde sa propre copie des objets auxquels elle accède et coopère avec les autres transactions en échangeant des valeurs de leurs copies respectives. Ce sont ces opérations de transfert qui nous permettent de synchroniser les histoires locales des différentes transactions. C'était l'étape *"autonomie pour l'accès aux données"* (section 6.6.2). La deuxième étape concernait la définition de nouveaux **critères de correction** qui soient eux-mêmes **distribués**. L'idée était en effet de permettre à chaque transaction du système de coordonner elle-même ses propres échanges de données avec les autres transactions. A l'inverse d'un critère de correction "classique" qui est défini sur l'histoire globale, un critère de correction dit "distribué" est destiné à être vérifié par chaque nœud du système (un nœud représentant l'exécution d'une transaction) en n'ayant accès qu'aux informations journalisées localement par ce nœud (l'histoire locale de la transaction). Nous avons ainsi défini un nouveau critère de correction local (la *DisCOO*-**sérialisabilité**) qui, lorsqu'il est assuré au niveau de chaque transaction, garantit les mêmes propriétés au niveau du système complet que son homologue "global" classique (la *COO*-sérialisabilité). C'était l'étape *"autonomie pour le contrôle des échanges"* (section 6.6.3).

Les transactions de notre modèle sont ainsi organisées en réseau, les nœuds réprésentant les transactions et les arcs réprésentant les schémas de coopération négociés entre les transactions. Il s'agit d'une architecture d'égal-à-égal dans laquelle chaque transaction est elle-même responsable du contrôle de ses propres interactions avec les autres transactions du système.

Lorsque nous avons formalisé le contrôle des échanges entre activités (section 6.6), nous ne nous sommes en fait intéressés qu'au résultat de la négociation d'un schéma de coopération entre deux transactions, c'est-à-dire à l'événement $Contract[t_i, t_j, O, s]$ qui est journalisé dans l'histoire locale de chaque transaction. Nous verrons dans le chapitre suivant comment introduire des possiblités de négociation en considérant les opération de négociation et de re-négociation comme étant des opérations classiques (au même titre que les opérations *read* et *write* par exemple) pouvant être invoquées sur des "objets de négociation" qui seraient porteurs d'informations spécifiques : liste des schémas proposés par chaque partenaire, liste des schémas convenant aux deux partenaires, proposition d'un autre schéma en cas de refus de celui demandé par le partenaire, ...

7

Coopération et négociation des échanges

7.1 Introduction

L'environnement de travail coopératif formalisé et présenté dans le chapitre précédent permet à différents agents (des activités pilotées par des opérateurs humains) géographiquement éloignés de coopérer via le partage de ressources (essentiellement des documents). Chaque agent disposant de sa propre copie des ressources partagées, la coopération entre agents s'effectue au travers d'importations et d'exportations de ressources. Des règles de coopération pour partager les ressources (schémas de coopération) ont été définies et chaque agent est responsable du contrôle de la correction de ses échanges. Ont ainsi été définis le mode "écriture-coopérative", le mode "client/serveur" et le mode "rédacteur/relecteur" . Etant fondés sur le même critère de correction, ces modes de coopération peuvent cohabiter au sein d'une même communauté d'agents, voire au sein d'un même agent lorsque celui-ci coopère de manière différente avec plusieurs partenaires. Ainsi, quand deux agents se connectent pour partager une ressource, ceux-ci doivent en premier lieu "signer un contrat" indiquant le mode de coopération qu'ils ont choisi. D'où l'idée de permettre aux agents de négocier ce mode de coopération en fonction de leurs contraintes respectives. D'autre part, il peut être utile de renégocier un mode de coopération en cours d'exécution, que ce soit pour résoudre un conflit entre deux modes ou pour assouplir les relations entre deux agents. Finalement, ces mêmes mécanismes de négociation peuvent également servir à assister les agents lors de la fusion de deux versions d'une même ressource, typiquement dans la phase finale d'une relation écriture-coopérative.

Par ailleurs nous n'avons pas voulu un modèle de négociation ad hoc. Nous avons préféré suivre une approche de modélisation complète qui nous a permis de concevoir un modèle de négociation générique, utilisable dans de nombreux contextes. En effet, l'explosion actuelle des applications distribuées, telles le télétravail, le travail coopératif, la téléconférence, et le commerce électronique, a donné une importance accrue aux systèmes de décision assistée par ordinateur. En effet, la prise de décision de groupe s'avère incontournable dans le travail quotidien (ex : la décision des plans d'action et des moyens à déployer, l'évaluation de l'état d'avancement, l'attribution et la coordination des tâches etc.). Nous nous intéressons, en particulier, à la formalisation et l'implantation d'un modèle générique de négociation au sein des applications de travail coopératif. En effet, afin d'étudier les alternatives possibles pour l'établissement des choix décisionnels communs, les utilisateurs d'un environnement de travail coopératif, pouvant être géographiquement éloignés, ont naturellement besoin de mécanismes d'aide à la négociation. L'objectif poursuivi

par l'intégration d'un service de négociation à un système d'aide au travail coopératif est de permettre à chaque acteur de coopérer avec la possibilité d'exprimer ses propositions décisionnelles au sein du groupe. Chaque coopérant participe au processus de négociation au travers de son rôle, sa responsabilité et sa spécialité. Un tel service de négociation pourrait offrir un cadre confortable assistant le groupe dans diverses situations décisionnelles. La négociation a été traitée dans le cadre de diverses thématiques de recherche : la psychosociologie, la théorie des jeux, l'économie, l'intelligence artificielle distribuée, etc. Nous décrivons cet état de l'art des modèles de négociation existants dans la section 7.2, puis le modèle de négociation que nous proposons dans la section 7.3.

7.2 Les différents modèles de négociation

La négociation est la science des observations précises, des prétentions réalistes, de l'analyse cohérente des faits, des conclusions logiques, des comportements planifiés et de la prévention optimale contre tout changement de la situation [Spe82]. La négociation peut être divisée en deux catégories, douce ou dure [JF89]. La négociation douce est caractérisée par une situation où les négociateurs ont plusieurs solutions possibles qu'ils adaptent selon leurs contraintes et leurs objectifs afin d'atteindre un état de satisfaction commun (win-win). Quant à la négociation dure, elle est caractérisée par des ressources fixées et limitées, des négociateurs dont les objectifs sont en conflit direct et qui visent la maximisation de leur part des ressources sans faire de concession (win-lose). Il n'existe pas de modèle universel de négociation qui puisse prévaloir pour chaque problème de négociation, à cause de paramètres mettant en jeux la culture des différents négociateurs, le langage et le vocabulaire qu'ils emploient pour négocier, le domaine de la négociation, le médium de la communication, sans parler des problèmes spécifiques à la situation de négociation. Le processus de négociation a été analysé selon deux approches différentes. L'approche descriptive analyse les interactions et les échanges entre les négociateurs. L'approche perspective s'intéresse, quant à elle, à l'explication des intérêts et des objectifs de chaque partie prenante de la négociation. Dans ce qui suit, le processus de négociation sera analysé selon l'Intelligence Artificielle Distribuée (IAD), la psychosociologie, la théorie des jeux, la théorie des actes de langages et les systèmes d'aide à la décision de groupe. En Intelligence Artificielle Distribuée [18] , les techniques de négociation importent aussi bien dans les situations de résolution de conflits que dans les situations d'échange de services. En effet, lorsque des conflits apparaissent, il est important de pouvoir en limiter les effets en établissant des compromis ou en dépassant la nature des conflits. La spécification de telles techniques suivant un protocole de négociation entre agents (communicants) utilisera un langage de communication dont le rôle est d'assurer les interactions par le biais de transmissions d'informations et de demandes mutuelles de renseignements et de services. Dans le contexte d'échange de service, un agent (client), ayant besoin qu'un service lui soit fourni par un autre agent (serveur), doit entrer en négociation avec ce dernier afin de déterminer un contrat sous certains termes et conditions. L'application de la négociation de services guidées par des agents intelligents peut concerner en l'occurrence : le diagnostic d'une défaillance, l'achat d'un produit en ligne [SHH00], l'allocation d'une bande passante pour la transmission, l'organisation de projet [KPPL00], la distribution de tâches entre agents [Koo88]. Pour la théorie des jeux, la négociation, impliquant un ensemble de joueurs, consiste à choisir une possibilité parmi un ensemble d'alternatives. Les choix peuvent mener les joueurs à des situations conflictuelles sans pour autant leur interdire de coopérer [JF89]. La théorie des jeux apporte des notions impor-

[18] IAD : discipline qui a pour objectif de réaliser des organisations de systèmes (ou agents) capables de résoudre des problèmes par le biais d'un raisonnement généralement basé sur une manipulation symbolique. [Fer95]

tantes pour le processus de négociation telles la prise de tour (ou "turn taking"), les tactiques, les stratégies, les situations d'équilibre, etc. La théorie des jeux fait une analyse perspective de la négociation très limitée si l'on considère les interactions possibles entre négociateurs [LB93]. La psychosociologie analyse l'état psychologique du négociateur selon trois types de forces R-forces [19] , A-forces [20] et C-forces [21] qui influencent son comportement vis-à-vis des parties opposées [MS77]. Elle souligne que la convergence de la négociation résulte de l'équilibre de ces trois forces. D'autres travaux relatifs à la communication entre agents ont également été réalisés du point de vue du langage et des informations induites par les mots utilisés pour négocier. Bien que les utilisations du langage naturel soient potentiellement infinies, la thèse défendue par la théorie des actes du langage [Aus62, Sea75] est qu'il est possible de regrouper ces termes en catégories pour obtenir un nombre fini d'actions verbales : les actes de langage (ou speech acts). Selon cette théorie, quelqu'un qui prononce une phrase ne fait pas que dire des mots, mais, il essaie de changer l'état du monde [Lev83]. En effet, pour une certaine phrase, cette théorie distingue entre trois aspects, locutoires (contenu du message), illocutoires (effet attendu du message (l'acte)) et perlocutoires (effet réel sur l'auditeur). Par ailleurs, [Sin93] et [Fer95] proposent une classification des actes illocutoires en sept catégories.

Catégorie	**Fonction (Exemple)**
Assertif	Donne une information sur le monde en affirmant quelque chose (il fait beau)
Déclaratif	Accomplit un acte par le fait même de prononcer l'énoncé (je déclare la séance ouverte)
Directif	Donne des directives au destinataire (Quelle est la valeur de la troisiéme décimale de ?)
Expressif	Donne au destinataire des indications concernant l'état mental du locuteur (je m'excuse pour hier)
Permissif	Annonce une permission (tu peux ouvrir la fenêtre)
Promessif	Engage le locuteur à accomplir certains actes dans l'avenir (je viendrai à la réunion de 5 heures)
Prohibitif	Interdit l'accomplissement d'une action (tu n'a pas le droit de voir le contenu de ce document)

Les processus de négociation basés sur cette théorie ont été modélisés soit à l'aide d'un diagramme états/transitions dont les transitions sont étiquetées par les actes de langage, soit à l'aide d'un ensemble de règles événement/condition/action où un événement correspond à la réception d'un acte de langage et où une action désigne la réaction de l'agent à cet acte (émission d'un autre acte de langage) (cf. [MLRR89, CW94]). D'autres travaux utilisent les actes de langages pour décrire la coopération entre agents comme un procédé (workflow) itératif se composant de quatre étapes préparation-négociation-accomplissement-satisfaction [MMWFF92, Den92]. Finalement, les systèmes d'aide à la négociation (NSS ou Negotiation Support System), qui forment une classe particulière des GDSS (Group Decision Support System) regroupant les technologies et les méthodologies assistant un groupe de personnes dans la formulation et la résolution de problèmes non structurés [JF89], combinent la communication, les techniques informatiques, et les techniques d'aide à la décision afin d'assister les gens dans leurs tâches de négociation. Dans

[19] R-forces : représentent son lien à la position de la coalition de négociateurs à laquelle il appartient,

[20] A-forces : indiquent le degré de compassion du négociateur envers la position de la coalition opposée,

[21] C-forces : régissent la capacité du système organisationnel ou social, où coexistent les coalitions, à imposer une issue finale au conflit,

les situations de conflits, les NSS peuvent être utilisés comme un langage commun de négociation. Ils aident dans le processus de négociation sans pour autant modéliser les aspects de prise de décision [BZ98]. Les NSS peuvent fournir une assistance aussi bien automatisée que semi-automatisée pour exprimer les problèmes d'une manière transparente et structurée. Par ailleurs, deux types de services peuvent être assurés par les environnements GDSS/NSS : les services synchrones et asynchrones. Les services synchrones mettent en place des mécanismes de rendez-vous, nécessitant "la présence" des différents acteurs avant que la négociation ne puisse débuter (ex : la téléconférence, les systèmes de réunion électronique (EMS ou Electronic Meeting System)). Les services asynchrones, quant à eux, offrent un cadre plus confortable de négociation où les acteurs peuvent interagir librement dans le temps tout en effectuant leur travail habituel (ex : la messagerie électronique ou la communication homme-homme médiatisée (CMC ou Computer-Mediated Communication)). Il ne s'agit là que de quelques exemples parmi tant d'autres modèles de négociation développés (cf. la négociation par apprentissage (Bazaar [ZS97]), la médiation et l'arbitrage en négociation (Persuader [Syc89]), la négociation par argumentation [GK97, KTP97, KSE98, PSJ98],...). Les modèles étudiés proposent un cadre très riche pour la modélisation du processus de négociation, ils présentent cependant de nombreux défauts par rapport à l'objectif de nos travaux sur la négociation. Les modèles de négociation selon la théorie des jeux et la psychosociologie reposent sur la définition de fonctions heuristiques de force ou d'utilité permettant à chaque agent d'évaluer l'état de la négociation au vu des propositions qui lui sont faites. Si cette approche permet d'éviter les divergences et de garantir la terminaison de la négociation (chaque proposition est meilleure que la précédente, négociation par raffinements successifs), elle nécessite toutefois de pouvoir comparer les différentes propositions et de définir un ordre (même partiel) sur l'ensemble des propositions envisageables. Cette hypothèse est loin d'être possible dans de nombreux champs d'applications. Quant aux modèles émanant de l'IAD, ils traitent d'une classe très particulières de négociation : "la négociation automatisée entre agents artificiels". Cependant, notre objectif est de modéliser le service de négociation pour les applications distribuées coopératives pouvant interagir avec des agents humains qui n'admettent pas de modèles prévoyant tous les comportements comme cela peut être le cas pour les agents artificiels. De plus, les protocoles de négociation en IAD sont souvent basés sur des algorithmes numériques ou logiques d'évaluation de l'état courant de l'agent, ce qui est irréaliste dans une situation de négociation entre humains. D'autre part, la plupart des services de négociation ne distinguent pas l'objectif de la négociation du protocole de négociation. En d'autres termes, les mécanismes de négociation sont généralement "codés en dur" parmi d'autres mécanismes dépendant du domaine d'application, ce qui ne permet aucune réutilisabilité du service de négociation. Quant aux GDSS/NSS, ils fournissent de riches notions pour analyser et modéliser le processus de négociation [AE94, BLR95, BS96], cependant, ils ne peuvent pas être facilement intégrés dans une application distribuée coopérative. En effet, soit ils offrent un cadre informel d'expression du problème dont la sémantique est difficilement exploitable (ex : messagerie électronique, téléconférence, communication homme-homme médiatisée, SANP [CW94]), soit ils sont adaptés à la résolution d'une classe spécifique de problèmes (ex : Information lens [MLRR89]). Contrairement aux modèles étudiés, notre approche consiste à appréhender séparément différents mécanismes inhérents à la négociation : structuration des données à échanger entre les agents (le langage), règles régissant la communication entre agents (le protocole) et le comportement vis-à-vis de ceux-ci (les tactiques). Une telle approche assure une flexibilité importante si l'on considère la possibilité d'adapter les mécanismes de la négociation selon l'objectif de la négociation, les rôles des agents,...De plus, les interactions entre agents peuvent se dérouler aussi bien en mode synchrone qu'asynchrone ce qui permet de gérer différentes situations de négociation au sein de l'application distribuée coopérative.

7.3 Notre modèle de négociation : une approche transactionnelle

Nous nous plaçons dans le cadre d'un système de travail coopératif composé d'agents autonomes. Un agent peut être soit actif (il réagit automatiquement ou semi-automatiquement aux messages qui lui sont envoyés par les autres agents) soit passif (il agit sous le contrôle d'un utilisateur qui déclenche l'exécution de méthodes sur cet agent). La négociation peut alors être vue de la manière suivante : "Un agent pose un problème à un autre agent, et ces deux agents dialoguent pour trouver la meilleure solution au problème". L'objectif de nos travaux est de construire un modèle de négociation qui soit indépendant de tout domaine d'application particulier. Il s'agit donc réellement de définir des mécanismes de négociation génériques, utilisables tant pour négocier les droits d'accès à un service particulier que pour définir les règles de coopération à respecter lors du partage d'une ressource entre deux agents par exemple.

7.3.1 Vue d'ensemble

Du point de vue de la négociation, les deux agents impliqués jouent des rôles différents : client et serveur. Le client est l'agent du système ayant un problème à résoudre. Pour cela, il effectue une requête auprès d'un de ses partenaires (appelé serveur) pour trouver une solution à son problème. C'est le client qui initie la négociation avec le serveur. La requête est la description du problème que le client veut résoudre. Selon le domaine d'application, il peut s'agir de l'utilisation d'un service, du partage d'une ressource, de la sélection d'une ressource parmi celles disponibles, etc. La solution, quant à elle, désigne l'alternative que le serveur propose à un client pour satisfaire la requête que ce dernier lui a soumise. Là encore, selon le domaine d'application, la notion de solution peut désigner différents concepts : droits d'accès pour l'utilisation d'un service, règles de coopération pour contrôler le partage d'une ressource, désignation d'une ressource conforme au(x) critère(s) de sélection négocié(s), etc. Une solution correspond à un accord (ou contrat) possible entre les deux agents. Comme nous pouvons le constater, les termes client et serveur ne désignent que des rôles joués par les différents agents au cours d'une négociation donnée, et plus particulièrement lors de leur connexion. Nous sommes en fait dans une organisation d'égal-à-égal, c'est-à-dire qu'un agent sera à la fois client et serveur selon qu'il envoie une requête à un agent ou qu'il répond à la requête d'un de ses partenaires (i.e. il propose une solution). Nous pouvons décomposer le processus de négociation bilatérale en trois étapes.

1. Le client contacte tout d'abord le serveur et lui fait part de sa requête (phase de sollicitation).
2. Se déroule ensuite la phase de négociation proprement dite : le client et le serveur s'échangent des messages pour construire l'ensemble des solutions acceptables par les deux agents.
3. Finalement, le client choisit une des solutions parmi celles résultant de la phase de négociation, puis en informe le serveur qui peut alors prendre les décisions qui s'imposent (allocation de la ressource demandée, ouverture des droits d'accès pour accéder à un service ou pour partager des données,...). C'est la phase de sélection.

C'est sur la formalisation de la phase de négociation que portent nos travaux. Pour cela, nous avons choisi d'aborder la négociation selon trois aspects : les informations échangées entre les agents pour négocier (le langage), la manière dont elles sont échangées (le protocole), et le comportement "interne" d'un agent (les tactiques). En d'autres termes, nous distinguons le langage de représentation des décisions prises par les agents, le protocole permettant aux agents de se communiquer leurs décisions, et la manière dont un agent prend ses décisions (objectifs personnels, contraintes à respecter, réactions semi-automatiques aux décisions du partenaire,...).

Outre une séparation des problèmes liés à chacune de ces trois facettes de la négociation, cette approche nous autorise également une plus grande souplesse que les systèmes classiques (basés sur des protocoles de négociation codés en dur) pour combiner différents langages, protocoles et tactiques au sein d'une même communauté d'agents, voire d'un même agent s'il mène simultanément plusieurs négociations avec plusieurs partenaires.

7.3.2 Utilisation des actes de langage pour négocier

Comme nous l'avons déjà expliqué, nous ne voulons pas élaborer un système propriétaire supplémentaire. Il s'agit au contraire de définir une infrastructure que l'on puisse paramétrer pour supporter différentes formes de négociation. Pour atteindre cet objectif, nous avons choisi d'utiliser une approche transactionnelle basée sur des actes du langage : une négociation est vue comme étant une transaction possédant un début (solicit), une fin (accept, confirm ou kill), et devant respecter certains critères. Nous allons journaliser un certain nombre d'événements (notamment des actes du langage via l'invocation d'opérations assert[speech_act]) au niveau de chaque agent (notion d'histoire locale), puis nous définirons des propriétés qui devront être vérifiées sur ces histoires (notion de critère de correction). Certaines de ces propriétés seront dédiées à la coordination des événements entre deux agents (protocole de négociation), alors que d'autres concerneront le contrôle des décisions prises par un agent (tactiques mises en œuvre par cet agent). A noter que, dans cette infrastructure, chaque agent possède sa propre histoire locale dans laquelle il journalise uniquement les événements qui le concernent, à savoir ses propres opérations ainsi que ses interactions avec les autres agents du système. Une fois les propriétés définies, chaque agent pourra donc les évaluer localement par rapport à sa propre histoire. Ainsi, lors d'une interaction entre deux agents (ex : transfert d'un fichier), les deux agents impliqués doivent donner leur accord (par rapport à leur histoire locale et aux propriétés qu'ils doivent respecter) pour que cette interaction puisse effectivement avoir lieu. L'originalité de notre approche est que les contrôles réalisés par le protocole et les tactiques sur l'enchaînement des décisions ne sont pas fondés sur des scénarios de négociation prédéfinis (cf. diagramme états/transitions) mais sur la définition et la vérification d'invariants caractérisant les séquences de décisions qualifiées de "correctes". Pour reprendre le vocabulaire des systèmes transactionnels, on parle de critères de correction des négociations. Afin d'illustrer notre modèle nous allons donner des exemples représentant un langage, un protocole et une tactique de négociation. Bien que ces exemples aient été formellement spécifiés, nous n'en donnerons ici que les grandes lignes, l'objectif étant la présentation de notre approche plutôt qu'une définition formelle et exhaustive des opérations et axiomes du modèle lui-même. (Pour plus de détails, voir [27])

Un exemple de langage de négociation

Pour que deux agents puissent négocier, il faut qu'ils disposent d'un langage de négociation commun leur permettant d'exprimer leurs propositions mutuelles. L'exemple de langage de négociation suivant permet de décrire les informations nécessaires à la décision (attributs x_i dont on spécifie le domaine de valeurs D_i), mais, il permet aussi de mieux préciser le sens de la décision à l'aide d'une force illocutoire f (un verbe prohibitif, permissif, assertif, directif, promessif, expressif ou déclaratif décrivant l'effet attendu par la décision). Un acte de langage de négociation que l'on note speech_act est alors représenté par le couple $(f, ((x_1, v_1 \in D_1), \ldots, (x_n, v_n \in D_n)))$ où v_i représente une valeur de D_i instanciant l'attribut x_i lors de la négociation. L'originalité de ce type de langage est qu'il est adapté aussi bien à la négociation entre agents humains ("chaque négociateur va se donner une idée subjective du comportement de son interlocuteur à travers

les actes de langages") qu'à la négociation entre agents artificiels ("l'agent pourra évaluer les chances de convergence du processus de négociation à travers une sémantique particulière de la force illocutoire"). Par exemple un processus de négociation entre un thermicien et un architecte pour le partage d'un document plan pourra être basé sur un attribut x_1 représentant le mode de coopération ($x_1 \in D_1$ avec D_1 = {écriture-coopérative, client/serveur, rédacteur/relecteur}). Chaque négociateur pondérera sa décision par f, une force illocutoire qui pourra être un verbe assertif (proposer, contre-proposer), un verbe prohibitif (refuser), un verbe expressif (dire, penser), un verbe directif (demander, insister, revendiquer), etc.

Un exemple de protocole de négociation

L'objectif d'un protocole de négociation est de contrôler l'entrelacement des opérations assert invoquées par les deux agents. Cela peut se résumer à un problème de concurrence d'accès sur le canal de communication ou de gestion de droit de parole. Les règles d'enchaînement de ces décisions, les décisions possibles selon le contexte (histoire courante), les actions à entreprendre en fonction des décisions prises par le partenaire, etc, ne concernent pas le protocole de négociation mais les tactiques de négociation. Un exemple de protocole de négociation entre deux agents est le turn-taking (ou tour de rôle). Celui-ci garantit qu'un agent ne pourra prendre une décision, i.e. invoquer une opération assert[speech_act], que s'il a au préalable acquis le droit de parole (événement getTurn). Outre les différentes opérations assert qu'il a invoquées ou que son partenaire a invoquées à son égard, un agent journalisera donc également dans son histoire locale d'autres événements tel que getTurn. Le principal axiome du protocole turn-taking peut alors être schématisé par l'invariant suivant sur l'histoire locale de chacun des deux agents : "Un agent A ne peut prendre une décision ($assert_A[speech\_act]$) que s'il a acquis le droit de parole ($getTurn_A$ apparaît dans son histoire locale) et qu'aucun autre agent B $\neq$ A ne l'a repris depuis (i.e. aucun événement $getTurn_B$ n'a été journalisé depuis la dernière occurrence de l'opération $getTurn_A$)". Bien que l'objectif de ce chapitre ne soit pas axé sur les détails de la formalisation, on donne à titre d'exemple la transcription de l'axiome précédent .

$$(assert_A[speech\_act] \in H_A) \Rightarrow \left\{ \begin{array}{ll} & getTurn_A \in H_A \\ \wedge & \forall\, B \neq A \quad \nexists\, getTurn_B \in H_A \\ & \qquad\qquad \text{such that} \quad LastOcc(getTurn_A) \rightarrow getTurn_B \end{array} \right.$$

Un exemple de tactique de négociation

L'objectif d'une tactique de négociation est de définir la manière dont un agent va prendre ses décisions et/ou choisir les solutions (opérations assert, accept/reserve, confirm, kill,...) en termes de propriétés sur son histoire locale. Elle va nous permettre de décrire, entre autres : le comportement contextuel de l'agent (i.e. décisions envisagées par rapport aux décisions des autres agents), le comportement réactif de l'agent (i.e. décisions envisagées en réaction aux décisions des autres agents), la coordination des différentes négociations réalisées par un agent (négociations avec plusieurs partenaires en imposant des contraintes entre les solutions négociées), et finalement, les objectifs de l'agent (i.e. les contraintes sur ses décisions fonction de l'utilité/de l'intérêt/ou du coût/...). Un exemple de tactique de négociation est la négociation atomique. Un agent client participe simultanément à plusieurs négociations avec la contrainte suivante : soit toutes les négociations aboutissent chacune à la mise en œuvre d'une solution, soit aucune action n'est entreprise. Ce comportement est formalisé par deux axiomes. Le premier indique qu' "une négociation atomique engagée par un agent A est en fait composée de plusieurs négociations élémentaires avec des agents B_1,...,B_n". Le second axiome affirme que "si l'agent A a confirmé

une solution pour une des requêtes (avec un agent B_i), alors il doit également avoir confirmé une solution pour chacune des autres requêtes (avec des agents B_j)." Cette tactique correspond en quelque sorte au mécanisme de verrouillage à deux phases (two phases locking) que l'on peut trouver dans les systèmes transactionnels classiques : une transaction nécessitant l'utilisation de plusieurs ressources doit tout d'abord acquérir un verrou sur chacune de ces ressources (première phase) avant de pouvoir y accéder (deuxième phase). Si, lors de la première phase, une des ressources ne peut pas être verrouillée, alors tous les verrous déjà acquis sont relâchés et la transaction est relancée ultérieurement.

7.4 Conclusion

Dans ce chapitre, nous avons succinctement décrit l'état courant des modèles et des applications dans le domaine de la négociation. Une analyse des approches et des systèmes de négociation actuels nous a conduit à la conclusion qu'ils sont "rigides", manquent de toute généricité permettant de les appliquer à différents domaines d'application et ne répondent pas de manière satisfaisante aux besoins en négociation des applications de travail coopératif. La non adéquation des solutions/approches actuelles à ces besoins nous a amené à définir une solution originale d'un modèle de négociation générique pour les applications de travail coopératif. Ce modèle formalise la négociation selon trois points de vue : le langage (les informations pour négocier), le protocole (la façon dont ces informations sont échangées) et la tactique (la façon dont un négociateur prend ses décisions). D'un point de vue conceptuel, notre approche propose une alternative intéressante aux modèles de négociation actuels et autorise une plus grande flexibilité que les systèmes traditionnels. Pour obtenir une modélisation axiomatique de notre modèle de négociation, nous l'avons formalisé suivant une approche transactionnelle basée sur les actes de langage. Dans notre modèle, chaque information concernant la négociation est représentée comme un acte de langage et la négociation est vue comme une transaction devant respecter certains critères. Chaque agent négociateur journalise (log) les événements relatifs à ses propres opérations et à ses interactions avec les autres agents. Nous avons défini et présenté les contraintes que doivent respecter ces histoires, certaines de ces contraintes servant au contrôle du protocole de négociation et d'autres au contrôle des décisions prises. Le fait que nous utilisions des axiomes pour définir les propriétés de la négociation nous permet d'enrichir notre modèle de négociation avec de nouveaux axiomes permettant de répondre à des besoins supplémentaires classiques en négociation tels que la gestion des alternatives ou encore de la délégation. Cet enrichissement du modèle peut être poursuivi tant que la cohérence globale de l'ensemble des axiomes est préservée et peut être une continuation intéressante de notre travail. Nous avons validé notre approche en utilisant les concepts de base de notre modèle, à savoir le langage de négociation et le protocole de négociation pour le développement d'un exemple de négociation dans le cadre de notre modèle coopératif transactionnel.

Un exemple de scénario de négociation peut se dérouler entre un thermicien et un architecte pour le partage d'un document plan. Un tel processus de négociation pourra être basé sur un attribut x_1 représentant le mode de coopération ($x_1 \in D_1$ avec $D_1 = \{$écriture-coopérative, client/serveur, rédacteur/relecteur$\}$). Outre l'affectation d'une valeur à l'attribut x_1, chaque acte du langage est pondéré par une force illocutoire qui permet à chaque négociateur de mieux préciser le sens de sa décision.

Nous intéressant principalement à des agents humains, nous avons quelque peu négligé les tactiques de négociation en supposant que l'agent humain se chargera, seul, de cet aspect de la négociation. Néanmoins la prise en compte de cet aspect de la négociation et son implantation

nous a permis de faire programmer des agents réactifs qui pourront assister les négociateurs humain dans la mesure où leur tactique aura été formellement décrite. Ce travail est décrit par A. Zellou dans son rapport de DESA [22][Zel01]. Une autre direction de recherche dans laquelle nous avons exploité nos résultats dans la modélisation de la négociation est le domaine de l'interconnexion des procédés métiers et les workflows coopératifs. Afin de permettre la coopération et, de là, la coordination de différents workflows, diverses négociations devront être menées entre des activités de workflows hétérogènes. Ces aspects seront décrits dans le chapitre suivant.

[22]Un modèle de tactiques et d'agents réactifs d'aide à la décision lors de la négociation de service (A. Zellou, DESA Informatique de l'Université Mohammed V de Rabat, décembre 2001)

8

Coopération et échanges de services

8.1 Introduction

Nous avons vu, dans le chapitre 6 , l'une des formes les plus "simples" de coopération, à savoir la coopération par le biais de l'"échange de documents". Si cette coopération traite correctement, à mon sens, le problème du "produit" (le document), elle ne s'intéresse pas au processus menant à ce produit. En outre, ce processus étant constitué par l'interconnexion des différents processus mis en œuvre par chacun des acteurs intervenant dans cette production, l'interconnexion des processus nous est apparue comme une suite logique au travail réalisé.

Les processus mis en œuvre par chacun des acteurs étant, en outre, de plus en plus souvent le résultat de procédés d'entreprise automatisés, 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 interentreprises. Concernant les systèmes de gestion de workflows (SGWF : Systèmes de Gestion de Workflows), 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 (BPDL : Business Process Definition Language), plateformes particulières de gestion de procédés,...). Afin d'améliorer le support générique d'interconnexion de procédés au sein des SGWFs existants, de nouveaux modèles d'interconnexion sont en cours de développement. Ces modèles traitent de la conscience de groupe et de la formalisation du flux de données et de contrôle entre les procédés coopérants (ex : modèles de gestion du partage de données [RD97], modèles de réplication de données [AAAM97], mécanismes de communication par envoi de messages [AM97, BH99], paradigme d'abonnement/notification d'événements [HA99, vdABEW00, CD00], invocation d'objets distants [WFM96, OMG00], extension de protocoles de transfert [BK99], ...), ou encore du contrôle du cycle de vie des procédés coopérants (ex : protocoles d'échanges transactionnels [BDS+93, GHM+93, RS95], ...). Ces modèles se limitent aux protocoles de communication de données et font abstraction de la structure et de la sémantique de la coopération de procédés [GSCB99]. Par ailleurs, pour assurer l'interoperabilité et l'interconnexion des procédés d'entreprises, les standards et environnements d'échange de données XML [W3C98] entre procédés métiers se focalisent sur les formats des messages XML, les règles de transformation entre types de messages et sur les protocoles d'échanges de ces messages (*e.g.* : W*f*XML [WFM00], BizTalk [Mic00]). Afin de pallier cette limite, de nombreuses initiatives ont été lancées pour construire des modèles et environnements autour de schémas d'intéropérabilité de haut niveau d'abstraction

qui peuvent être réutilisables dans différentes situations de coopération inter-entreprises (*e.g.* : ebXML : electronic business XML [UO00], HP e-speak [HP01]). Cependant, poussées par différents organismes ou éditeurs de logiciels, ces plateformes, conçues pour assurer l'intéropérabilité et l'intégration des applications, se confrontent elles mêmes au problème de leur hétérogénéité.

Un modèle d'interconnexion de procédés voulant dépasser les limites des SGWFs et des cadres d'interconnexion de procédés existants, doit prendre en considération les problèmes et compromis suivants :

- L'ouverture *versus* la confidentialité : Bien que les entreprises aient besoin de passerelles d'interconnexion entre leurs procédés, elles peuvent être concurrentes directement ou indirectement. De ce fait, tout modèle d'interconnexion doit prendre en compte les aspects de confidentialité des procédés (i.e. distinction entre informations privées et informations partagées, etc).
- La coopération planifiée *versus* la coopération dynamique : Chaque acteur, au sein d'un système, ne résout pas toujours les problèmes qu'il rencontre de la même manière. Ceci se répercute par conséquent à l'échelle de la coopération interorganisationnelle. En effet, dans les mêmes conditions de travail, deux entreprises ne coopèrent pas de la même manière durant deux projets de même type. Il s'avère donc difficile de décrire un scénario de coopération qui définisse à l'avance toutes les possibilités d'interactions entre deux procédés organisationnels (i.e. explosion combinatoire) [GSCB99].
- L'autonomie *versus* la cohérence : Bien que les entreprises travaillent ensemble dans le cadre de projets communs, elles conservent leur entière autonomie et peuvent travailler isolément entre chaque réunion commune. Cependant, ceci ne doit pas aller à l'encontre des termes du contrat signé et donc de la cohérence globale du projet. Par conséquent tout modèle d'interconnexion doit en assurer la cohérence globale tout en permettant le travail asynchrone [AAAM97].

A la suite de notre modèle de négociation pour les plateformes de travail coopératif présenté dans le chapitre précédant et dans [27], ce chapitre présente de manière détaillée notre modèle supportant l'interconnexion dynamique des procédés d'entreprises à travers la négociation de services procédés. Ce chapitre est structuré comme suit : la section 8.2 situe notre approche orientée service procédé ; puis les sections 8.4, 8.5 et 8.6 formalisent, repectivement, notre modèle de procédés, notre approche orientée services procédés et notre modèle d'interconnexion de services procédés ; finalement la section 8.7 conclut ce chapitre.

8.2 L'approche service pour l'interconnexion de procédés d'entreprises

Notre objectif est de développer un environnement supportant la coopération et l'interconnexion dynamique des procédés organisationnels. Nous désignons par "dynamique" le fait qu'aucun paramètre de cette interconnexion ne doit être prédéfini ou planifié préalablement et que tout se décide en cours d'exécution (ex : primitives de communication, rendez-vous d'échange de données). Ceci signifie qu'une organisation voulant interconnecter son procédé de travail avec le procédé d'une autre organisation (ex : pour l'externalisation du développement d'un logiciel, pour la commande d'un service procédé en ligne, pour un échange d'information dans une entreprise virtuelle, etc.), doit co-décider d'un contrat d'interconnexion à l'exécution. Afin de permettre un tel type d'interconnexion entre les procédés, premièrement, des mécanismes de négociation sont, entre autres, nécessaires pour l'aide aux décisions communes entre procédés (ex : service procédé à échanger, rendez-vous, traités, contrats, résultats finals, etc.). Pour ce faire, nous avons

déployé notre modèle générique de négociation [27], développé pour supporter une négociation bilatérale ne dépendant ni du domaine d'application, ni des éléments négociés. Par ailleurs, afin de développer notre modèle dynamique d'interconnexion de procédés, 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. Nous avons, donc, opté pour une approche orientée service procédé. **Pour résumer notre approche, nous visons le développement d'un modèle d'interconnexion des procédés d'entreprises à travers la négociation de services procédés.**

Le concept de service a été défini dans différents domaines de recherche : paradigmes objets [OMG97], systèmes distribués [Kut98, BMB+00], modélisation de procédés [SBMW96, GSCB99, GPS99, KWA99, CIJS00, GAHL00, vdHHP01, ZBN01], etc. Dans le domaine des procédés et des workflows, un service procédé peut être vu comme 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 procédé 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 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 procédé [GSCB99], sélection d'une sous-partie d'un service procédé [KWA99], configuration dynamique des activités d'un service procédé [CIJS00], niveaux abstraits de contrôle de flux d'un service procédé [GAHL00], adaptation des méthodes et événements de deux services procédés [BMB+00], etc. Un service procédé se comporte comme un patron architectural (architectural pattern) [BMR+95] qui supporte, d'une manière pertinente, la coopération et l'interconnexion dynamique entre procédés d'entreprises. Il existe déjà des plateformes d'échange de services web entre applications. Les plus importantes sont CORBA [GGM97, HV99], Jini de Sun [Sun01], les services web (IBM WSTK [IBM00], Microsoft .NET [Mic01]), etc. Si l'on considère notre problématique d'interconnexion de procédés d'entreprises, ces plate-formes présentent, cependant, un certain nombre de limites dont certaines sont détaillées ci-dessous. Au sein de ces plateformes, pour un service donné, l'entreprise qui le fournit doit décider, durant la phase de son développement, du contrat d'utilisation de ce service (i.e. : fonctionnalités qu'elle veut exposer sous forme de méthodes web). Par conséquent, un utilisateur du service ne peut pas modifier ou négocier ce contrat (ex : demander moins de fonctionnalités pour payer moins de droits d'utilisation). De la même manière, si le fournisseur veut fournir un contrat plus restrictif ou plus riche, il faut qu'il publie un nouveau service sur le web. Ces plateformes existantes **manquent de dynamicité**. Par ailleurs, il n'existe pas encore de format universel de services, ce qui rend difficile leur orchestration au sein d'un procédé ou workflow. Si les services ont été conçus pour diminuer les efforts nécessaires à leur intégration au sein de systèmes complexes, des lacunes persistent au niveau de leur interopérabilité. En effet, les nombreux vendeurs de plate-formes orientées services essaient de promouvoir leurs visions du modèle de service universel sans se préoccuper de l'intégration avec les modèles de services existants. Les plateformes orientées services existantes **manquent d'interopérabilité**. Par ailleurs, les architectures orientées services ne sont pas encore assez mûres. En effet, elles se basent sur un paradigme asymétrique de publication, de recherche et d'interconnexion de services. En d'autres termes, seuls les fournisseurs de services peuvent décrire leurs offres et il n'existe aucun moyen de décrire et publier les requêtes des demandeurs sous formes de services requis. Ceci ne permet pas aux fournisseurs de services de rechercher les demandes potentielles de leurs services. Par conséquent, cela diminue la conscience de groupe entre entreprises coopérantes et complique l'interconnexion de leurs procédés. D'autre part, les mécanismes de localisation de services se

limitent à la recherche contextuelle basée sur des liens structurels et/ou textuels (ex : serveur de nommage de CORBA, JNDI de Jini, UDDI [UDD00]). Même s'il existe d'autres mécanismes plus riches au sein des architectures orientées services (ex : recherche de services par contraintes dans le service vendeur de CORBA), celles-ci ne sont pas encore très évoluées et de nombreux efforts restent à déployer. Ces plateformes montrent une certaine **immaturité**.

Toutes ces lacunes des plateformes existantes nous ont amenés à proposer un nouveau modèle d'interconnexion de services procédés. Notre modèle d'interconnexion de services procédés permettra aux entreprises coopérantes de structurer, classifier (définition de profil de service procédé) et comparer leurs services procédés (mise en correspondance (matching) des services procédés), sélectionner dynamiquement un service procédé fourni parmi ceux qui correspondent (au sens du matching) à un service procédé requis donné, négocier des services procédés selon leur profil ou API (Application Programming Interface), et finalement rendre possible la coopération entre leurs procédés métiers à travers la mise en liaison contractuelle (wrapping) des services procédés. Par exemple, un service procédé peut concerner aussi bien une longue transaction électronique (ex : l'externalisation du développement d'un composant logiciel, l'inscription en ligne à une session de FOAD (Formation Ouverte et à Distance), etc.) qu'une courte transaction électronique (ex : la commande en ligne d'un livre, l'activation d'un procédé administratif, l'échange de données planifié dans une entreprise virtuelle, etc.).

8.3 Présentation de l'exemple support

Afin d'illustrer notre approche, nous utiliserons tout au long de ce chapitre un exemple dans le contexte de la Formation Ouverte et A Distance (FOAD).

Soit une entreprise de FOAD (R) qui a comme mission de développer des bases de connaissances réunissant des sessions de formations dans différents domaines. Elle possède divers procédés métiers qui lui servent à organiser et à maîtriser l'exécution de ses activités et ses projets. Ces procédés combinent des tâches et activités internes ou externes. Les activités internes de l'entreprise sont totalement supportées par le personnel de celle-ci, alors que pour assurer les activités externes, l'entreprise fait appel à des entreprises tierces. Une activité peut être externalisée faute de temps, de compétence, de maîtrise de coûts etc. Nous appellerons ces activités à externaliser des services procédés requis. De la même manière, nous appellerons les activités représentant des compétences fournies par une entreprises, des services procédés fournis. Un service procédé fourni encapsule un procédé ou une partie d'un procédé métier de l'entreprise qui le fournit. Ce procédé décrit les étapes essentielles de déroulement du service procédé fourni. L'entreprise doit donc décrire ses besoins et ses compétences et les communiquer aux acteurs externes. Notre modèle permet donc la **définition et publication de services procédés**. Pour ce faire, l'entreprise possède deux référentiels, le premier sert à présenter les services procédés qu'elle demande (espace de services procédés requis) et le deuxième sert à présenter les services procédés qu'elle fournit (espace de services procédés fournis). Afin de constituer ses bases de connaissances fédérant les sessions de formations ouvertes et à distance, l'entreprise R effectue un certain nombre d'activités internes, mais elle externalise également certaines activités par manque de compétences dans les domaines traités. Dans notre exemple, l'entreprise R requiert, entre autres, un service procédé de collecte de modules de formation particuliers (service procédé requis s_4). Pour ce faire, elle fait appel à des fournisseurs de contenu FOAD. Plusieurs entreprises ont pour mission de collecter du contenu de FOAD, en l'occurrence, le fournisseur de contenu (P_3) et l'agence multimédia (P_4). L'entreprise P_3 fournit un service procédé de collecte de contenu de FOAD (s_{32}) et l'entreprise P_4 fournit un service procédé de réalisation de contenu multimédia (s_{41}). Plusieurs services procédés

fournis peuvent répondre au besoin de l'entreprise R, entre autres, les services procédés fournis s_{32} et s_{41}.

La sélection d'un service procédé fourni, répondant à un service procédé requis, se base sur des algorithmes de calcul de correspondance (*matching*) et de voisinage. Ces derniers permettent d'aider les entreprises à raffiner leur recherche selon leurs besoins. Notre modèle permet donc la **recherche et la sélection de services procédés**.

Une fois que l'entreprise qui a besoin d'une compétence externe aboutit à un ensemble de services procédés fournis qui correspondent à son besoin, une série de négociations peut commencer pour aboutir à un contrat qui satisfait le demandeur et le fournisseur du service procédé et donc à éliminer les autres services procédés éligibles. Notre modèle permet donc la **négociation de services procédés**.

Dans notre exemple, c'est l'entreprise P_3 qui réussit à répondre aux besoins de l'entreprise R et à satisfaire ses demandes à travers son service procédé fourni s_{32}. Désormais, les deux services procédés s_4 et s_{32} sont unis par un contrat qui dictera les règles de coopération entre les entreprises R et P_3.

8.4 Modèle de procédés

Notre modèle de procédés est initialement basé sur la définition du modèle de procédés de F. Leymann et D. Roller [LR00]. Si ce modèle peut être aisément appliqué pour des workflows traditionnels (au sein d'une même entreprise), il ne considère pas explicitement l'interconnexion de procédés. Notre objectif est d'enrichir ce modèle par de nouveaux concepts et définitions afin de supporter les aspects les plus pertinents, à notre sens, de l'interconnexion des procédés. La plupart des modèles d'interconnexion étudiés se limitent à la définition du flux de données et de contrôle entre les procédés sans se préoccuper de deux points d'accès aux procédés importants : méthodes d'instance de procédés et événements d'instance de procédés. Afin d'éviter l'hétérogénéité des SGWFs actuels, nous considérerons, dans ce chapitre, que les SGWFs sont compatibles avec les standards de la W*f*MC, ou au moins partagent une API commune basée sur l'Interface 2 de la W*f*MC [WFM98]. Pour ce faire, nous développons et définissons les notions de procédé, de graphe de procédé et d'API de procédé, puis les concepts de méthodes et d'événements d'instance de procédé.

Un procédé représente tout outil permettant de décrire explicitement les grandes étapes du projet et assurant le suivi de déroulement de ses étapes. Le déroulement de ces étapes peut être automatique, semi-automatique ou manuel. Un procédé est responsable de l'offre d'un ensemble de primitives de consultation et de mise à jour de l'état de déroulement de ses étapes à la demande (méthodes) et d'un ensemble de signaux informant de l'état d'avancement des étapes du procédé (événements). Nous représentons un procédé, que l'on note P, par un graphe représentant les étapes de déroulement du procédé (i.e. la structure du flux de contrôle d'un procédé [LR00]) et une API définissant les primitives d'accès (méthodes) et les signaux (événements) du procédé :

Définition 1 (P : Procédé, $G(N,\ E)$: Graphe de procédé et API de procédé)
On définit un procédé P comme une paire $P = (G(N,\ E),\ ((Methods,\ \leq),\ (Events,\ \leq)))$, où $G(N,\ E)$ est le graphe de procédé et $((Methods,\ \leq),\ (Events,\ \leq))$ l'API du procédé.
Un graphe de procédé est un couple $G(N,\ E)$, où N est l'ensemble des nœuds (activités) et E est l'ensemble des connexions de contrôle (arcs orientés) du graphe ; $(a \in N,\ b \in N,\ c \in C) \in E$ signifie qu'il y a une connexion de contrôle entre a et b pondérée par une condition c. C est l'ensemble de toutes les conditions (règles métier : booléens ou prédicats).

On appelle API de procédé, que l'on note Process API, la paire combinant les ensembles $Methods$ et $Events$ liés à un procédé : Process API $= ((Methods, \leq), (Events, \leq))$.

Définition 2 ($Methods$: Méthodes de procédé)
Soit $Methods$ l'ensemble de toutes les procédures et fonctions de l'API de procédé. Plus précisément, $Methods$ est l'union des méthodes de consultation d'instance de procédé ($R-methods$) et des méthodes de mise à jour d'instance de procédé ($U-methods$). $Methods = R-methods \cup U-methods$.

Une instance de procédé est une vue (ou réalisation) du procédé à un instant t. Un procédé peut avoir plusieurs instances. Ces instances peuvent se trouver dans divers états. Les méthodes de consultation d'instance de procédé $R-methods$ sont des primitives qui permettent de lire (sans altération) l'instance d'un procédé (ou l'un de ses composants) durant son exécution (ex : getProcessInstanceStatus(), getProcessInstanceAchievedDurationRate(), getProcessInstanceAllocatedHumanResources()).

Quant aux méthodes de mise à jour d'une instance de procédé $U-methods$, ce sont des primitives qui permettent de modifier une instance de procédé (ou l'un de ses composants) durant son exécution (ex : setProcessInstanceStatus(..), setProcessInstanceAllocatedHumanResources(..)).

Un des éléments primordiaux de la visibilité de l'API du procédé, à travers un service procédé (cf. section 8.5.2), est l'établissement d'une relation d'ordre total $\leq$ dans $Methods$. Cette dernière rendra possible la comparaison totale et la sélection des services procédés selon leur API, suivant une tactique de négociation (cf. section 8.6.2). W étant un ensemble fini, on note $|W|$ son cardinal. Pour construire un ensemble totalement ordonné $(Methods, \leq) = \{m_1, .., m_{|Methods|}\}$, on munit l'ensemble $Methods$ d'une relation d'ordre total $\leq$ qui exprime la confidentialité d'une méthode, de l'ensemble $Methods$, liée à une instance de procédé telle que :

$$\leq : Methods \times Methods \rightarrow Boolean$$
$$(m_1 \leq m_2) \Leftrightarrow (m_1 \text{ est aussi confidentielle que } m_2 \vee$$
$$m_1 \text{ est moins confidentielle que } m_2) \text{ pour un observateur externe.}$$

Afin d'aider le fournisseur du service procédé dans la définition d'une telle relation d'ordre, on peut classer $Methods$ en k classes de visibilité $(C_i)_{i=1,..,k}$ telles que :

$$1 \leq k \leq |Methods| \wedge$$
$$\forall\, i \in \{1, .., k\},\ C_i \subseteq Methods \wedge$$
$$\forall\, m_1,\ m_2 \in C_i,\ m_1 \text{ est aussi confidentielle que } m_2 \wedge$$
$$\forall\, m \in C_i,\ m' \in C_{i' > i}, \text{ on a } m \leq m'$$

Si l'on considère dans notre exemple support le service procédé fourni s_{32} de collection de contenu de FOAD, il est représenté par un procédé dont le graphe est composé de quatre activités séquentielles et d'une API composée de dix méthodes et de six événements. Nous illustrons ci-dessous l'approche utilisée pour ordonner l'ensemble $Methods$ des méthodes du procédé encapsulé par s_{32}, et de la même manière nous ordonnerons l'ensemble $Events$ de ses événements. Soit l'ensemble $Methods$ suivant tel que $|Methods| = 10$ et soit k le nombre de classes de visibilité, telle que k est choisi dans l'intervalle $1 \leq k \leq |Methods|$. Par exemple, soit $k = 5$ comme suit :

Classes de visibilité	*Methods*	
C_1	m_1	getProcessInstanceStatus()
	m_2	getProcessInstanceEffectiveAchievedDurationInDays()
C_2	m_3	getProcessInstanceAchievedDurationRate()
	m_4	getProcessInstanceAchievedModuleDuration()
C_3	m_5	getProcessInstanceAchievedModuleDurationRate()
	m_6	getProcessInstanceAllocatedHumanResources()
C_4	m_7	setProcessInstanceStatus(..)
	m_8	setProcessInstanceEffectiveAchievedDurationInDays(..)
	m_9	setProcessInstanceAchievedModuleDuration(..)
C_5	m_{10}	setProcessInstanceAllocatedHumanResources(..)

Dans cette configuration de $(Methods,\ \leq)$, on a : $(m_1\ \leq\ m_2)$ car $m_1,\ m_2\ \in\ C_1$ et on a $(m_1\ \leq\ m_3)$ car $m_1\ \in\ C_1$ et $m_3\ \in\ C_{2\ >\ 1}$.

Comme introduit précédemment, l'API d'un procédé (Process API) est composée des méthodes et événements liés à une instance de procédé. De la même manière que nous avons détaillé les méthodes d'une instance de procédé, nous détaillons ci-dessous, les événements d'une instance de procédé. Les événements liés à une instance de procédé sont des messages de notifications émanant d'une instance de procédé (ou l'un de ses composants) durant son exécution :

Définition 3 (*Events* : Événements d'une instance de procédé)
Soit *Events* l'ensemble de tous les événements d'une instance de procédé (i.e. les événements gérant l'état d'exécution d'une instance de procédé et de ses composants (ex : NEW_MODULE_COLLECTION_STATUS_PI_EVENT) et les événements gérant l'état de la production des données par une instance de procédé (ex : PRODUCED_NEW_MODULE_VERSION_PI_EVENT).

Pour construire un ensemble totalement ordonné $(Events,\ \leq) = \{e_1, .., e_{|Events|}\}$, on le munit d'une relation d'ordre totale mesurant la confidentialité d'un événement lié une instance de procédé au sein de l'ensemble *Events*.

$$\leq :\ Events\ \times\ Events\ \rightarrow\ Boolean$$
$$(e_1\ \leq\ e_2)\ \Leftrightarrow\ (e_1 \text{ est aussi confidentiel que } e_2\ \vee$$
$$e_1 \text{ est moins confidentiel } e_2) \text{ pour un observateur externe.}$$

Une approche similaire à celle utilisée pour ordonner totalement *Methods* peut être utilisée par le fournisseur de service pour ordonner totalement l'ensemble *Events* en $(C_i^{'})_{i=1,..,k'}$ classes de visibilité.

De la même manière que précédemment, nous illustrons ci-dessous l'approche utilisée pour ordonner l'ensemble *Events* des événements du procédé encapsulé par le service procédé fourni s_{32}. Soit l'ensemble *Events* suivant tel que $|Events|\ =\ 6$ et soit $k^{'}$ le nombre de classes de visibilité, telle que $k^{'}$ est choisi dans l'intervalle $1\ \leq\ k^{'}\ \leq\ |Events|$. Par exemple, soit $k^{'}\ =\ 3$:

Classes de visibilité	*Events*	
$C_1^{'}$	e_1	END_MODULE_COLLECTION_PI_EVENT
	e_2	BEGIN_MODULE_COLLECTION_PI_EVENT
$C_2^{'}$	e_3	NEW_MODULE_COLLECTION_STATUS_PI_EVENT
	e_4	ASK_FOR_MORE_SPECIFICATION_PI_EVENT
$C_3^{'}$	e_5	ASK_FOR_REMARKS_ON_MODULE_VERSION_PI_EVENT
	e_6	PRODUCED_NEW_MODULE_VERSION_PI_EVENT

Dans cette configuration de $(Events, \leq)$, on a :
$(e_1 \leq e_2)$ car $e_1, e_2 \in C_1^{'}$, $(e_1 \leq e_5)$ car $e_1 \in C_1^{'}$ et $e_5 \in C_{3\ >\ 1}^{'}$. Le modèle de procédés ayant été formellement défini, nous allons, maintenant, définir notre modèle de services procédés.

8.5 Modèle de services procédés

L'échange de services procédés 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) et de coordination (synchronisation, vérification de critères de cohérence). De ce fait, il offre une grande richesse en termes d'expressivité de son utilisation dans les situations de coopération. Notre objectif est de développer un modèle de services procédés qui devra supporter la coopération et l'interconnexion dynamique entre les procédés inter-organisationnels. De plus, un service procédé devra offrir des mécanismes pour mieux comprendre et exploiter les ressources du procédé qu'il encapsule. Principalement, un service procédé est une abstraction fonctionnelle d'un procédé : il présente les propriétés significatives (profil du procédé), le contrat de visibilité de l'API du procédé et les données du procédé. Un service procédé est classé selon une catégorie qui en définit le type et qui permet de le positionner par rapport aux autres services procédés. Il possède un nom et connaît son demandeur et son fournisseur. Un service procédé encapsule un procédé. L'exécution du service procédé est gérée par une instance (i.e. réalisation) du procédé qu'il encapsule. A chaque instant, le service procédé passe par un état selon un cycle de vie de service procédé :

Définition 4 (Service procédé)
Notons B l'alphabet ASCII, *objectClasses* l'ensemble de toutes les classes objets, et $T = (I\!R \cup Boolean \cup B^*)$([23]), soit $\overline{V}$ l'ensemble des contrats de visibilité d'API, $\mathcal{IDC}$ et $\mathcal{ODC}$ respectivement les espaces de données entrantes et sortantes du service procédé. Soit E l'ensemble des entreprises coopérantes et soit $\mathcal{Q}$ l'ensemble des états de service procédé [24]. Notons S l'ensemble des services procédés, chaque service procédé $s \in S$ est représenté par un n-uplet : $s = (nom \in B^+,$ *catégorie* $\in objectClasses, profil \in T^k, contrat\_de\_visibilité \in \overline{V},$ *données_entrantes* $\subseteq \mathcal{IDC}$, *données_sortantes* $\subseteq \mathcal{ODC}, demandeur \in E, fournisseur \in E,$ *procédé* $\in P, instance$ (une instance de *procédé*), *état* $\in \mathcal{Q})$.

Durant le procédé de construction d'une base de connaissances de FOAD (*e.g.* : en informatique), l'entreprise R identifie un besoin de développer un module de FOAD au tour d'un thème parti-

[23] B^* (respectivement B^+) représente le langage formé de toutes les chaînes ASCII (respectivement non vides).
[24] cf. [30] pour plus de détails sur le cycle de vie d'un service procédé à travers les états composant l'ensemble Q = {TO_BE_REQUESTED, TO_BE_PROVIDED, REQUESTED, PROVIDED, R-WRAPPED, P-WRAPPED, R-IN_NEGOTIATION, P-IN_NEGOTIATION, R-ENACTED, P-ENACTED, R-COMMITTED, P-COMMITTED, R-ABORTED, P-ABORTED}

culier (*e.g.* : les objets distribués). Pour ce faire, elle définit ce besoin de module de FOAD en termes de service procédé de collection de FOAD à externaliser à une entreprise tierce. L'entreprise R spécifie des propriétés qu'elle associe au service procédé à externaliser. Le développement du module de FOAD ne doit pas dépasser "2 mois" et son prix ne doit pas dépasser "1500 Euros". Par ailleurs, le contexte du module doit être du niveau "middleware", la durée du suivi du module doit être de "60 heures", le module doit s'adresser à des "professionnels" et doit traiter les aspects "pratiques" de la programmation des objets distribués. Nous illustrons, ci-dessous, ce besoin en utilisant une réalisation du service procédé requis $s_4 \in S$ (service procédé de collection de contenu FOAD). Après la négociation du profil et du contrat de visibilité de s_4 et après la liaison contractuelle (*wrapping)* de s_4 et de s_{32} (le service procédé fourni lui correspondant), nous obtenons :

s_4 = (nom = "module Objets Distribués",
catégorie = e_learning_module_collection_process_service,
profil = (service_procédé_durée_en_mois = 1 (mois),
service_procédé_prix_en_Euro = 1000 (Euro),
module_contexte = 5 [middleware],
module_durée = 60 (heures),
module_niveau = 4 [professionnel],
module_type = 3 [pratique]
),
contrat de visibilité = (M_2, EV_3),
données entrantes = $\emptyset$ ($\mathcal{IDC}$ initialement vide),
données sortantes = $\emptyset$ ($\mathcal{ODC}$ initialement vide),
demandeur = R (entreprise de FOAD),
fournisseur = P_3 (fournisseur de contenu FOAD),
procédé = procédé_de_collecte_de_contenu_FOAD,
instance = instance_de_procédé,
état = R-WRAPPED
)

Comme mentionné précédemment, un service procédé est principalement une abstraction fonctionnelle d'un procédé présentant les propriétés significatives du procédé (profil), le contrat de visibilité de l'API du procédé et les données du procédé. Nous détaillons ces trois éléments essentiels pour accéder à un service procédé dans les sections suivantes.

8.5.1 Profil d'un service procédé

Comme décrit précédemment, le profil d'un service procédé présente un ensemble de propriétés typées. Ces propriétés peuvent décrire différents aspects et niveaux d'interconnexion supportés par le service procédé. Le profil, dont nous ne limitons pas le champ d'application, peut concerner les protocoles de transfert, les mécanismes de sécurité, les standards de formatage de données, les paradigmes de partage de données, les critères de correction transactionnelle, les critères de qualité, les aspects liés au commerce électronique, le contexte d'application, etc.) Le profil de service permet la classification, l'indexation, la comparaison, la recherche et la négociation d'un service procédé. Pour ce faire, les entreprises, au sein d'une même communauté métier, doivent se mettre d'accord à propos de langages communs de services procédés (ex : ontologies de concepts métiers clefs, catégories de services procédés métiers, profil...) afin de définir et de comprendre des services procédés. Des travaux de recherche et de normalisation sont encore émergeants dans ce domaine prometteur [UDD00, UO00]. Dans l'exemple support, le profil du service procédé requis

s_4 dépend de quelques aspects de commerce électronique (service_procédé_durée_en_mois, service_procédé_prix_en_Euro) et de quelques aspects liés à l'application du service procédé (module_contexte, module_durée, module_niveau, module_type).

Avant de signer un contrat d'interconnexion entre un service procédé requis et un service procédé fourni, le profil du service procédé est sujet à une négociation entre le demandeur d'un côté et le fournisseur de l'autre. L'objectif de cette négociation et d'aboutir à un profil de service procédé qui satisfasse les deux parties et qui soit réalisable (cf. section 8.6).

8.5.2 Contrat de visibilité d'un service procédé

Dans les SGWFs existants, en ce qui concerne les droits d'accès à un procédé, deux alternatives seulement sont possibles : *boîte noire* (ni les appels de méthodes de procédé ni les événements de son exécution ne sont permis ou visible pour une organisation externe) ou *boîte blanche* (l'API du procédé peut être complètement accessible : méthodes et événements). Si une organisation estime que la deuxième alternative est très permissive (à l'égard de la confidentialité, de la sécurité ou du marketing, par exemple), elle présente son procédé comme étant une *boîte noire*. La sécurité de cette dernière est évidente, mais, elle n'est, cependant, pas viable du point de vue de l'interconnexion des procédés. Un des objectifs de notre modèle d'interconnexion de services procédés est de rendre possible la spécification d'un panel de droits d'accès aux API de procédé allant de la solution boîte noire à la solution boîte blanche. Pour ce faire, nous définissons les notions de couches de visibilités de méthodes et de couches de visibilités des événements pour aboutir à la notion de contrat de visibilité d'un service procédé.

Basé sur la classification de l'ensemble $Methods$ et de l'ensemble $Events$, l'ensemble des couches de visibilité structure les méthodes et les événements de procédés afin de rendre possible la négociation d'un sous ensemble particulier de $Methods$ (une couche de méthodes) et d'un sous ensemble particulier d'$Events$ (une couche d'événements) pour l'interconnexion de services procédés :

Définition 5 (M, EV et $\overline{V}$: Couches de visibilité et contrats de visibilité)
W étant un ensemble fini, on note $\wp(W)$ l'ensemble des parties de W. On définit l'ensemble des couches de visibilité de méthodes M comme suit : $M \subset \wp(Methods)$ tel que
$M = \cup_{i=0}^{|Methods|}\{M_i\}$ où $M_i \subseteq (Methods, \leq) = \{m_1, .., m_{|Methods|}\}$, $M_0 = \emptyset$ et $M_{1 \leq i \leq |Methods|} = M_{i-1} \cup \{m_i\}$.
Ainsi, $M = \{\emptyset, \{m_1\}, \{m_1, m_2\}, .., \{m_1, m_2, .., m_{|Methods|}\}\}$. NB : $M_{|Methods|} = Methods$ et $|M| = |Methods| + 1$
On définit l'ensemble des couches de visibilité des événements EV comme suit : $EV \subset \wp(Events)$ tel que $EV = \cup_{i=0}^{|Events|}\{EV_i\}$ où $EV_i \subseteq (Events, \leq) = \{e_1, .., e_{|Events|}\}$, $EV_0 = \emptyset$
et $EV_{1 \leq i \leq |Events|} = EV_{i-1} \cup \{e_i\}$.
Ainsi, $EV = \{\emptyset, \{e_1\}, \{e_1, e_2\}, .., \{e_1, e_2, .., e_{|Events|}\}\}$ NB : $EV_{|Events|} = Events$ et $|EV| = |Events| + 1$

Basé sur les concepts de couches de visibilité de méthodes et de couches de visibilité d'événements, un contrat de visibilité est une restriction d'API de procédé par rapport aux méthodes et événements permis. La boîte blanche et noire sont des contrats de visibilité particuliers : Soit $\overline{V}$, l'ensemble des contrats de visibilité, le produit cartésien de l'ensemble des couches de visibilité de méthodes (M) et de l'ensemble des couches de visibilité des événements (EV). $\overline{V} = M \times EV$, chaque élément $v \in \overline{V}$ représente une restriction (ou un droit d'accès) de l'API de procédé, puisque c'est une combinaison d'un sous ensemble de $Methods$ et d'un sous ensemble d'$Events$.

Soit $\subseteq$ une relation d'ordre partiel sur $\overline{V}$ telle que :

$$\subseteq : \overline{V} \times \overline{V} \rightarrow Boolean$$
$$v_i(M_j,\ EV_k) \subseteq vi'(M_{j'},\ EV_{k'}) \Leftrightarrow M_j \subseteq M_{j'} \wedge EV_k \subseteq EV_{k'}$$

Nous illustrons les concepts de contrat de visibilité de service procédé au sein de notre exemple support d'entreprise de FOAD. Nous avons $Methods = M_{10}$ et $Events = EV_6$. De plus, nous avons : $M_2 = \{$getProcessInstanceStatus(), getProcessInstanceEffectiveAchievedDurationInDays()$\}$ et $EV_3 =\{$END_MODULE_COLLECTION_PI_EVENT, BEGIN_MODULE_CO-LLECTION_PI_EVENT, NEW_MODULE_COLLECTION_STATUS_PI_EVENT$\}$.
Comme le montre l'exemple, R (l'entreprise de FOAD) a négocié, parmi d'autres contrats de visibilité proposés, $v_i \in \overline{V}$ tel que $v_i = (M_2, EV_3)$ pour s_{32} (le service procédé fournis par P_3 (le fournisseur de contenu FOAD) et qui correspond au service procédé s_4 requis par R) . Nous avons bien $(M_2,\ EV_3) \subseteq (M_{10},\ EV_6)$. Cela signifie que le contrat de visibilité du service procédé s_4 "cache" (au sens de l'inclusion dans $\overline{V}$) l'API du procédé qu'il encapsule.

8.5.3 Données de service procédé

Définition 6 ($\mathcal{IDC}$ et $\mathcal{ODC}$: Espaces de données d'un service procédé)
Soit D l'ensemble de toutes les données (ex : documents structurés, semi-structurés, ...), on note $\mathcal{DC}$ l'ensemble des espaces de données avec $\mathcal{DC} \subset \wp(D)$, s_{req} est un service procédé requis et s_{frn} est le service procédé fourni qui lui est interconnecté. L'espace de données entrantes $\mathcal{IDC}$ (Process Service Input Data Container) ($s \in S$) associé à un service procédé s est l'ensemble des documents lus (mais non créés) par un service procédé durant son exécution, où $\mathcal{IDC} \subset \mathcal{DC}$, $\mathcal{IDC}(s_{req} \in S) = \{d_i \in D / producteur(d_i) = fournisseur(s_{req}) \wedge consommateur(d_i) = demandeur(s_{req})\}$ et $\mathcal{IDC}(s_{frn} \in S) = \{d_i \in D / producteur(d_i) = demandeur(s_{frn}) \wedge consommateur(d_i) = fournisseur(s_{frn})\}$ (nous parlons de synchronisation sur données entrantes). L'espace de données entrantes d'un service procédé évolue tout au long du cycle de vie du service procédé (i.e. les données entrantes peuvent parvenir à des instants différents de l'exécution du service procédé.)
L'espace de données sortantes $\mathcal{ODC}$ (Process Service Output Data Container) ($s \in S$) associé à un service procédé s est l'ensemble de documents créés ou modifiés par un service procédé durant son exécution. Soit s_{req} un service procédé requis et s_{frn} est le service procédé fourni qui lui est interconnecté. On a $\mathcal{ODC} \subset \mathcal{DC}$, $\mathcal{ODC}(s_{req} \in S) = \{d_i \in D / producteur(d_i) = demandeur(s_{req}) \wedge consommateur(d_i) = fournisseur(s_{req})\}$ et $\mathcal{ODC}(s_{frn} \in S) = \{d_i \in D / producteur(d_i) = fournisseur(s_{frn}) \wedge consommateur(d_i) = demandeur(s_{frn})\}$ (nous parlons de synchronisation sur données sortantes). L'espace de données sortantes d'un service procédé évolue tout au long du cycle de vie du service procédé (i.e. les données sortantes peuvent être produites à des instants différents de l'exécution du service procédé.)

Dans notre contexte, nous utiliserons s_4 et s_{32} comme service procédé requis (respectivement fourni). Ces deux services procédés manipulent durant leur exécution un certain nombre de documents (d_1 : cahier des charges du module de FOAD, d_2 : contrat juridique du développement du module, d_3 : prototype du module, d_4 : remarques sur le prototype du module et d_5 : version finale du module). Ces différents documents sont produits à différents moments de l'exécution de s_4 et s_{32} et sont échangés entre eux selon certaines règles illustrées ci-dessous :

- Après la liaison contractuelle des deux services procédés requis s_4 et fourni s_{32}, correspondants (au sens du *matching*), le contenu de l'espace des *données_ entrantes* de s_4 (ex : d_1) est recopié vers l'espace des *données_ entrantes* de s_{32}. Il s'agit de l'initialisation des *données_ entrantes* du service procédé fourni s_{32}.

- Il se peut qu'un nouveau document externe parvienne au service procédé requis s_4 longtemps après le *wrapping* ou même après l'activation (l'*enactment*) (ex : d_2). Dans ce cas, le service procédé fourni s_{32} est notifié de l'arrivée de ce nouveau document et peut en faire une copie dans son espace de *données_ entrantes* pour le prendre en compte dans son exécution.
- Dès l'activation du service procédé fourni s_{32}, ce dernier commence à s'exécuter et produit des documents dans son espace de *données_ sortantes* (ex : d_2, d_3) le service procédé requis s_4 en est notifié et peut en faire une copie dans son espace de *données_ entrantes* pour le prendre en compte dans ses interactions ultérieures avec s_{32}. De la même manière, le service procédé requis s_4 peut produire des documents dans son espace de *données_ sortantes* (ex : d_4), le service procédé fourni s_{32} en est notifié et peut en faire une copie dans son espace de *données_ entrantes* pour le prendre en compte dans ses interactions ultérieures avec s_4. Il s'agit de la synchronisation des deux vues s_4 et s_{32} sur données entrantes et données sortantes.
- Après la fin de l'exécution (validation) du service procédé fourni s_{32}, le contenu de l'espace des *données_ sortantes* de ce dernier (ex : d_2, d_3 et d_5) est recopié vers l'espace des *données_ sortantes* du service procédé requis s_4. Ils'agit de la validation et de la persistance des *données_ sortantes* du service procédé fourni s_{32}.

Après avoir présenté les entreprises R (une entreprise de FOAD) et P_3 (une entreprise de collection de contenu de FOAD) et leurs interactions dans les exemples précédents, l'exemple suivant donne une vue plus complète de l'interconnexion du procédé de R avec les procédés de ces partenaires. L'interconnexion comme présentée se fait via une approche orientée service où les acteurs opèrent en proposant (respectivement demandant) des services procédés à réaliser (respectivement externaliser) et en négociant des contrats d'interconnexion de ces services procédés. On peut imaginer de nouveaux fournisseurs de services procédés : P_1 (une agence web) et P_2 (un hébergeur de sites web). En effet, pour réaliser son objectif de production de sessions de formation ouvertes et à distance, l'entreprise R fait appel à ses partenaires fournisseurs de services procédés. Un tel exemple montre comment notre modèle peut être appliqué dans l'interconnexion des procédés d'entreprises par le biais de la gestion de l'externalisation des services procédés.

8.6 Modèle d'interconnexion de services procédés

Notre modèle d'interconnexion de services procédés se base sur deux facettes principales, la recherche et la sélection de service procédé (cf. section 8.6.1) et la négociation de services procédés (cf. section 8.6.2). En effet, un demandeur de service procédé vise à faire faire un service procédé requis s_{req} dont il a spécifié le profil par un tiers suivant un contrat d'externalisation. Le profil et le contrat de visibilité du service procédé à externaliser vont permettre au demandeur du service procédé de trouver le meilleur service procédé fourni s_{frn} et donc de former le meilleur couple service procédé abstrait/concret (s_{req}, s_{frn}) qui réponde à ses besoins d'externalisation à travers des recherches de voisinages et des négociations de profils et de contrats de visibilités.

8.6.1 Recherche et sélection de services procédés

La définition 7 introduit le prédicat de correspondance (matching) entre services procédés et la définition 8 présente le calcul de voisinage de services procédés. Nous définissons la fonction de correspondance (*matching*) de services procédés comme étant un mécanisme permettant l'évaluation et la comparaison des services procédés. Le profil de service procédé est la clef de ces

mesures de correspondance. Le calcul de correspondance de deux services procédés améliorera la conscience d'une communauté d'entreprises (*group awarness*) à propos d'un nouveau service procédé disponible. De plus, cela facilitera l'accessibilité et l'interconnexion des procédés d'entreprises (ex : moteurs de recherche de services procédés sur le web, système d'aide à l'évaluation d'un service procédé, etc.). Les techniques de *matching* sont nécessaires pour aider les demandeurs (resp. les fournisseurs) de services procédés à trouver les services procédés qui correspondent le plus à leur services procédés requis (resp. fournis).

Définition 7 (*match* : Fonction de correspondance de services procédés)
Soient s_{req} et s_{frn} respectivement le service procédé requis et fourni et soit $dist : S \times S \rightarrow I\!R^+$ une fonction distance. On définit une fonction de correspondance (*matching*) comme un prédicat $match : S \times S \times I\!R^+ \rightarrow Boolean$ qui vérifie si deux services procédés sont adéquats entre eux (i.e. le service procédé fourni s_{frn} répond aux besoins du service procédé requis s_{req}), formellement :

$$match(s_{req} \in S,\ s_{frn} \in S,\ \Delta \in I\!R^+) = true \Leftrightarrow ((s_{req}.categorie = s_{frn}.categorie \vee s_{req}.categorie\ est\_sous-type\_de\ s_{frn}.categorie) \wedge dist(s_{req},\ s_{frn}) \leq \Delta)$$

La forme générale de la fonction distance scalaire *dist* sera :
$dist_{n \in I\!N^*, w \in I\!R^{+k}}(s_1 \in S, s_2 \in S) = \sqrt[n]{\sum_{i=1}^{k} w_i |profil(s_1, i) - profil(s_2, i)|^n}$, où $w = (w_i)_{i=1,..,k}$ sont des poids réels positifs croissant selon l'importance que l'on veut donner au i^{eme} élément des profils de service procédé à comparer (par exemple, si l'on choisit $(w_i)_{i\ =\ 1,..,k} = 1$ et $n = 1$, on obtiendra la distance de Hamming, et avec $(w_i)_{i\ =\ 1,..,k} = 1$ et $n = 2$, on obtiendra la distance Euclidienne). En plus de la distance scalaire, on a également considéré sa forme multidimensionnelle présentant les éléments de la distance sur chacun des k axes du profil de service procédé : $dist\_axis_{i \in \{1,..,k\}, w_i \in I\!R^+}(s_1 \in S, s_2 \in S) = w_i |profil(s_1, i) - profil(s_2, i)|$, avec la même sémantique des poids $(w_i)_{i\ =\ 1,..,k}$.

Dans le sens de la fonction de distance *dist*, le voisinage d'un service procédé s est l'ensemble de tous les services procédés correspondant (au sens du *matching*) à ce service procédé dans une "sphère" d'un rayon Δ. Cette fonction de voisinage est à considérer comme un mécanisme d'aide à la décision d'interconnexion de services procédés plutôt qu'un algorithme de recherche de services procédés.

Définition 8 (NH :Fonction de voisinage de service procédé)
Soit $NH(s \in S)$, le voisinage d'un service procédé s, l'ensemble de tous les services procédés correspondant à s et qui en soient proches (au sens de la distance). Formellement : $NH(s \in S) = \{NH(s,\ \Delta) \mid \Delta \in I\!R^+, min, max \in I\!N^2 \wedge min \leq |NH(s, \Delta)| \leq max\}$ avec $NH(s,\ \Delta) = \{s' \in S \mid match(s,\ s',\ \Delta)\}$

Pour calculer un voisinage significatif et pertinent (du point de vue de la distance par rapport au service procédé référence et par rapport au nombre de voisins trouvés), on procède de la manière itérative suivante : On calcule $NH(s,\ \Delta_{t=0})$ le voisinage avec une valeur initiale de $\Delta = \Delta_{t=0}$. Si $NH(s,\ \Delta_t)$ se situe dans l'intervalle d'admission $[min,\ max]$, on arrête le calcul de l'ensemble de voisins du service procédé s et on dit que $NH(s) = NH(s,\ \Delta_t)$. Sinon, on recalcule un nouveau Δ_t. Ce dernier est revu à la baisse (fonction *diminuer*) si $NH(s,\ \Delta_t) > max$ ou à la hausse (fonction *augmenter*) si $NH(s,\ \Delta_t) < min$. Il se peut, également, que l'algorithme ne converge pas (ex : $|S| = 1$, avec $S = \{s\}$). On détecte ce type de cas

quand on dépasse un nombre limite d'itérations assez grand. Auquel cas, on dit que $NH(s) = \emptyset$.

Algorithme de calcul de voisinage de service procédé :
Soient les constantes réelles suivantes : δ (la valeur initiale $\Delta_{t=0}$), l (le nombre limite d'itérations), m_1 et m_2 (les valeurs respectives de *min* et *max*), et deux fonctions *augmenter* et *diminuer* (croissantes et bijectives sur $\mathbb{R}$, $augmenter$, $diminuer : \mathbb{R} \rightarrow \mathbb{R}$, avec $\Delta < augmenter(\Delta)$, $diminuer(\Delta) < \Delta$ et
$\Delta < diminuer(augmenter(\Delta)) < augmenter(\Delta)$ [25] :
$NH(s \in S) \equiv NH(s, \delta \in \mathbb{R}^+, l \in \mathbb{N}, m_1 \in \mathbb{N}, m_2 \in \mathbb{N}) \equiv$

$t \leftarrow 0$ % le nombre d'itérations est initialement à 0
$\Delta_t \leftarrow \delta$ % la valeur initiale de Δ est à δ
% on itère au maximum l fois pour la recherche du voisinage
$itération\_limite \leftarrow l$
$min \leftarrow m_1$ % au minimum, on admet m_1 services procédés voisins
$max \leftarrow m_2$ % au maximum, on admet m_2 services procédés voisins
répéter
- % on calcule le voisinage avec la valeur de Δ_t courant
- $NH_t \leftarrow NH(s, \Delta_t)$
- **si** $|NH_t| < min$
  - **alors** { $\Delta_{t+1} \leftarrow augmenter(\Delta_t)$; % on élargit le champs de recherche du voisinage }
- **sinon si** $|NH_t| > max$
  - **alors** { $\Delta_{t+1} \leftarrow diminuer(\Delta_t)$; % on réduit le champs de recherche du voisinage }
- **fin si**
- $t \leftarrow t + 1$

tant que $((|NH_t| < min \vee |NH_t| > max) \wedge (t \leq itération\_limite))$

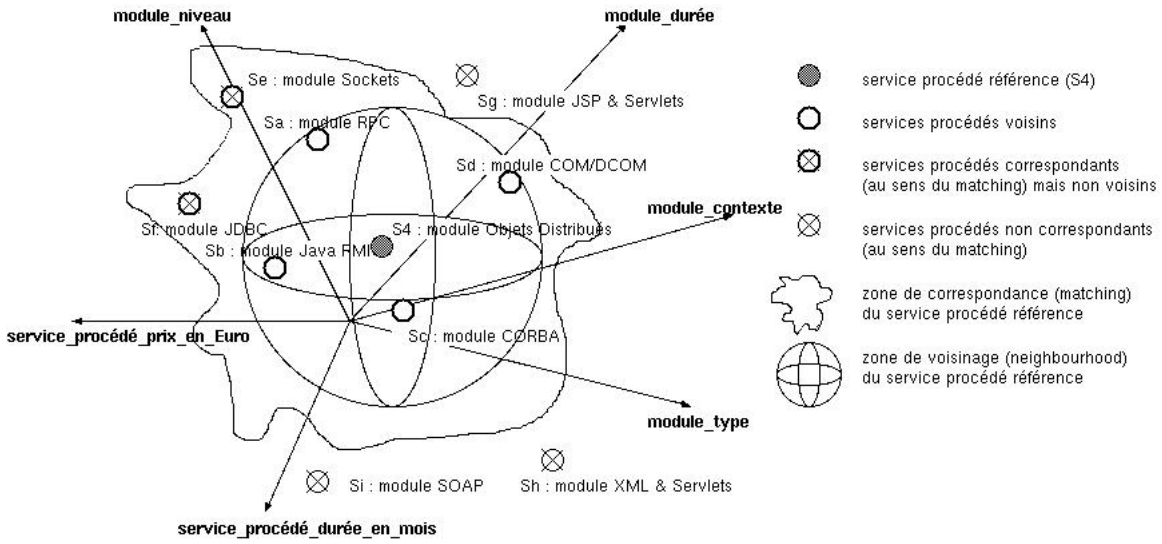
La figure 8.1 illustre le concept de calcul de correspondance et de voisinage d'une instance du service procédé requis s_4. Cette instance concerne la collection de contenu FOAD dans l'objectif de construire un "module Objets distribués". En effet, pour un service donné s_4, le calcul de correspondance permet de partitionner l'espace des services procédés en deux sous espaces : l'espace des services procédés correspondant à s_4 (s_a, s_b, s_c, s_d, s_e et s_f) et l'espace des services procédés non correspondant à s_4 (s_g, s_h et s_i). D'un autre côté, le calcul de voisinage permet de partitionner l'espace des services procédé correspondant à s_4 (au sens du matching) en deux sous espaces : l'espace des services procédés correspondant et voisins de s_4 (s_a, s_b, s_c et s_d) et l'espace des services correspondant mais non voisins de s_4 (s_e et s_f).

8.6.2 Négociation de services procédés

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 (ex : choix des partenaires, des services procédés à interconnecter, des protocoles de communication à adopter (ex : SOAP/UDDI, IDL/CORBA), du contrat d'interconnexion des services procédés, du contrat de partage de données, ...). Plus généralement, on crée, on fournit, on demande un service procédé, on négocie le profil ou le contrat de visibilité d'un service procédé, on lie contractuellement deux services procédés, on active un service procédé et on échange des données entre services procédés

[25] Pour calculer le voisinage d'une manière dichotomique, on pourrait choisir les fonctions *augmenter* et *diminuer* telles que : $augmenter(\Delta) = 2.\Delta$, et $diminuer(\Delta) = \frac{1}{2}.(\Delta + augmenter^{-1}(\Delta)) = \frac{3}{4}.\Delta$. Cela vérifie $\Delta < diminuer(augmenter(\Delta)) = \frac{3}{2}.\Delta < augmenter(\Delta)$.

Figure 8.1 un exemple de calcul de correspondance et de voisinage pour des services procédés de collection de modules de FOAD



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 s'avèrent indispensables. Pour ce faire, nous déployons notre système de négociation. Ce système a été développé de manière générique pour permettre une négociation ne dépendant ni du domaine d'application ni des éléments négociés. Il peut être, ainsi, appliqué aux problèmes de co-décision entre les demandeurs et les fournisseurs de services procédés pendant les sessions d'interconnexion des services procédés. Notre modèle de négociation a été développé suivant une approche transactionnelle utilisant la théorie des actes de langage [Sea69]. Il est basé sur trois piliers : **le langage**, **le protocole** et **la tactique** de négociation. L'objectif de ce chapitre n'est pas de détailler notre modèle de négociation qui a été développé dans [27] et présenté dans le chapitre précédent, mais d'en montrer l'application dans notre problématique d'interconnexion de procédés d'entreprises. En effet, on applique notre système d'aide à la négociation dans deux situations de négociation pour l'interconnexion de procédés comme suit :

- **Négociation de profil de service procédé :** cela concerne la décision d'un profil de service procédé p parmi les valeurs de profils possibles $p \in D = T^k$, T étant l'ensemble des attributs du profil et k la taille du profil.
- **Négociation de contrat de visibilité de service procédé :** cela concerne la décision d'un contrat de visibilité v parmi ceux déjà proposés $v \in D = i_j$ ($i_j \in I$).

Une tactique de négociation a comme objectif de définir la manière dont un acteur va prendre ses décisions et/ou choisir les solutions lors d'une négociation [27]. Ci-dessous, est donné un exemple de tactique pour assurer la convergence de la négociation de contrats de visibilité de services procédés :

Exemple de tactique pour la négociation de contrat de visibilité : Afin d'aider les fournisseur et demandeur d'un service procédé à co-décider facilement du contrat de visibilité à adopter pour leur interconnexion de procédé, on leur offrira un moyen d'exprimer un large panel structuré de contrats de visibilité d'un service procédé (cf. section 5). Pour ce faire, nous

introduisons le concept de contrat d'interconnexion qui structure l'ensemble des contrats de visibilité entre boîte noire et boîte blanche (i.e. plus flexibles que boîte noire mais moins permissifs que boîte blanche) :

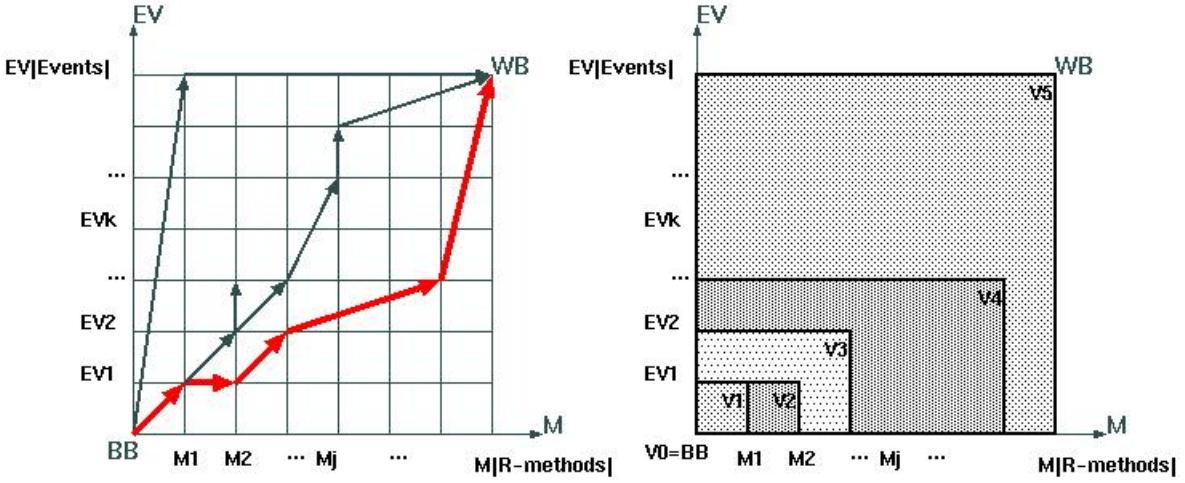
DÉFINITION 9 (I : LES CONTRATS D'INTERCONNEXION DE SERVICES PROCÉDÉS)
Soit I l'ensemble des contrats d'interconnexion de services procédés. Chaque élément de I est constitué de contrats de visibilité croissant par rapport aux méthodes et événements de l'API du service procédé. Formellement, $I \subset \wp(\overline{V})$ tel que :

$$\begin{cases} I = \{i_j \subset \overline{V} \mid i_j = \{v_0, .., v_{p_j-1}\} \text{ où } |i_j| = p_j\} \wedge \\ v_0 = (M_0,\ EV_0) \in \overline{V} \textbf{ (boîte noire)} \wedge \\ \forall\ k \in \{0, .., p_j - 2\}, v_k, v_{k+1} \in \overline{V}^2,\ v_k \subseteq v_{k+1} \wedge \\ v_{p_j-1} = (M_{|R-Methods|},\ EV_{|Events|}) \in \overline{V} \textbf{ (boîte blanche)} \end{cases}$$

Chaque $i_j \in I$ représente un ensemble d'alternatives de contrats de visibilité que le fournisseur peut proposer pour ses demandeur de service, avec p_j le nombre de ces alternatives.

Dans la figure 8.2, le diagramme de gauche montre quelques possibilités de contrats d'interconnexion, alors que le diagramme de droite présente un élément $i_j = \{v_0, .., v_{p_j-1}\}$ de I, avec $p_j = 6$. Ce dernier est le chemin mis en valeur dans le diagramme de gauche, i_j est un chemin et chacun de ses sommets est un contrat de visibilité. NB : Soit $i_j \in I$ et $p_j = |i_j|$, on a $p_j \in \{2, .., |M| + |EV| - 1\}$.
En fait, p_j peut être égal au moins à 2
(ex : $i_j = \{BB = (M_0, EV_0),\ WB = (M_{|Methods|}, EV_{|Events|})\}$) et au plus à $|M| + |EV| - 1$ ce qui représente la longueur maximale d'un chemin de source BB et de destination WB dans la grille $M \times EV$.

Figure 8.2 L'ensemble des contrats d'interconnexion de services procédés



L'ensemble des contrats d'interconnexion trouve son application comme tactique de négociation assurant le respect de convergence de la négociation d'un contrat de visibilité (**négociation par concession**). Soit $I = \{i_1, i_2, .., i_j\}$. Le fournisseur du service procédé peut choisir $i' = \{v_0, .., v_{|i'|-1}\} \in I$ tel que : s'il propose le contrat de visibilité $v_t \in i'$ (à l'instant t) et le demandeur le rejette, le fournisseur pourra proposer $v_{t+1} \in i'$ (succédant directement à v_t).

Ayant développé nos modèles de procédés et de services procédés, on peut maintenant définir notre modèle d'interconnexion de services procédés. Nous appelons interconnexion de services

procédés d'entreprises, la possibilité de lier leur procédés en les rendant conscients des services procédés requis et fournis et de leur permettre de négocier des contrats d'interconnexion avant la collaboration effective. Cela pourra se réaliser si l'on déploie nos modèles de procédés et de services procédés.

8.6.3 Démarche d'interconnexion de services procédés

Nous présentons la démarche d'interconnexion de services procédés selon la séquence d'opérations suivante : une fois défini **(étape 1)** puis publié **(étape 2)**, la définition d'un service procédé fourni est accessible à toutes les enterprises de la communauté. Chaque demandeur de services procédés peut consulter cette définition et vérifier sa correspondance à ses définitions de services procédés requis (au sens de *matching*) et peut ainsi calculer les voisinages de ses services procédés **(étape 3)**. Si le calcul de voisinage renvoie des résultats intéressants (selon le point de vue du demandeur), il peut rentrer en négociation avec les fournisseurs des services procédés voisins afin de co-décider des meilleurs profils de services procédés (au sens de la fonction distance) **(étape 4)**. Parallèlement à la décision du profil du service procédé, le contrat de visibilité du service procédé peut être négocié pour permettre l'établissement dynamique de l'API du service procédé **(étape 5)**. Pour deux services procédés voisins fournis, les fournisseurs, doivent offrir le meilleur profil et contrat de visibilité pour conclure la négociation de service procédé (on parle alors de tactique de coordination des deux négociations des étapes 4 et 5). La mise en liaison contractuelle des services procédés (*wrapping*) **(étape 6)** est l'étape finale de l'interconnexion avant l'activation du service procédé **(étape 7)**. Tandis que la mise en liaison contractuelle des services procédés traite de la validation et de la diffusion des paramètres décidés du service procédé (profil et contrat de visibilité) sur les deux vues du service procédé (la vue du demandeur et celle du fournisseur), l'activation d'un service procédé concerne l'activation de l'instance de procédé lié au service procédé. Les échanges entre le demandeur et le fournisseur du service procédé sont à gérer dorénavant par un agent de contrôle de cycle de vie du service procédé (*moniteur de service procédé*) **(étape 8)**. Le moniteur surveillera le bon usage du contrat de visibilité (i.e. invocation de méthodes de services procédés, envoi d'événements à partir de l'instance du service procédé, production de documents dans les espaces de données de service procédé).

On utilise les services procédés requis/fourni de collection de contenu de FOAD s_4/s_{32}, décrits dans les exemples précédents, afin d'illustrer notre approche d'interconnexion de services procédés à travers la description des huit étapes, préalablement définies, en termes d'interactions entre le demandeur R et les fournisseurs P_1, P_2 et P_3 des services procédés.

Nous explicitons ci-dessous, pour résumer, les importantes opérations de manipulations de services procédés :

1. **création de service procédé** : on crée un service procédé en le décrivant en termes de propriétés de son profil (cf. section 8.5.1) et son contrat de visibilité (cf. section 8.5.2). Les services procédés sont créés dans l'espace privés de services procédés, après leur publication, ils deviendront accessibles aux autres partenaires, dans les espaces des services procédés requis ou fournis,

2. **édition de service procédé** : on édite un service procédé quand on accède et modifie éventuellement son profil. On peut éditer un service procédé, changer son profil, puis en publier une instance. Si l'on répète la même opération plusieurs fois, on obtiendra différentes instances du service procédé qui évolueront indépendemment les unes des autres,

3. **demande de service procédé** : on demande un service procédé en en transférant une instance vers l'espace des services procédés requis. On peut avoir plusieurs instances du même service procédé dans l'espace des services procédés requis. Par opposition aux services procédés fournis, les services procédés requis sont des services procédés abstraits à spécifier. Cela signifie qu'ils ne sont pas liés à un procédé concret,

4. **mise à disposition de service procédé** : par similarité à la demande de service procédé, on fournit un service procédé en en transférant une instance vers l'espace des services procédés fournis. Plusieurs instances du même service procédé peuvent être publiées, à des instants différents, dans l'espace des services procédés fournis.

5. **calcul de voisinage de service procédé** : on calcule le voisinage du service procédé en examinant les services procédés des partenaires qui peuvent lui correspondre,

6. **demande de négociation du profil de service procédé** : la négociation du profil du service procédé permet de résoudre les problèmes de décision du choix du service procédé fourni correspondant le plus a un service procédé requis au sens du profil. Le demandeur d'un service procédé requis s_{req} propose de nouveaux profils aux fournisseurs des services procédés fournis s_{frn} correspondant à s_{req} (i.e. $s_{frn} \in D = NH(s_{req})$). Pour deux services procédés fournis, les fournisseurs auront à offrir le meilleur profil lors de la négociation du profil de service procédé (cf. section 8.6.2),

7. **demande de négociation du contrat de visibilité de service procédé** : la négociation du contrat de visibilité du service procédé permet d'établir, à l'exécution, un accord de visibilité de l'API de procédé entre le demandeur et le fournisseur de service procédé. Ceci assure une coopération flexible et dynamique (cf. section 8.6.2).

8. **liaison contractuelle de services procédés** : après le matching entre un service procédé requis et un service procédé fourni, et la négociation bilatérale du profil de service procédé et du contrat de visibilité à adopter, le wrapping entre les deux services procédés consiste à valider les paramètres du service procédé fourni (*catégorie*, *nom*, *profil*, *fournisseur*, *procédé*, *instance*) et à les rendre persistants sur le service procédé requis. De plus, les paramètres du service procédé requis (*demandeur* et *données_ sortantes*) sont rendus persistants sur le service procédé fourni (i.e. les *données_ sortantes* du service procédé requis sont recopiées dans l'espace des *données_ entrantes* du service procédé fourni). Nous parlons d'initialisation des *données_ entrantes* du service procédé fourni.

9. **médiation de service procédé** : un demandeur de service procédé peut le fournir à un autre demandeur. Nous parlons alors de médiateur de services procédés. Le médiateur de service procédé n'est qu'un intermédiaire qui consomme le service procédé indirectement en le refournissant à un autre partenaire. On peut, alors, aboutir à une longue chaîne de responsabilités entre le fournisseur initial, la série des médiateurs et le demandeur final.

10. **activation de service procédé** : l'activation d'un service procédé requis, précédemment lié (au sens du wrapping) à un service procédé fourni, consiste à créer et activer une instance du procédé (*process*) encapsulé par le service procédé fourni. L'interconnexion des procédés est désormais validée, commencent alors l'exécution et le contrôle du cycle de vie de service procédé.

11. **validation de service procédé** : la validation d'un service procédé actif est un événement significatif qui désigne que l'exécution du service procédé s'est bien déroulée et que le

procédé de travail lié à ce service procédé est terminé. La validation se matérialise par la vérification du bon déroulement du service procédé procédé (contrôle de cohérence), auquel cas les *données_ sortantes* du service procédé fourni sont rendues persistantes dans l'espace des *données_ sortantes* du service procédé requis. Nous parlons alors de validation et persistance des *données_ sortantes* du service procédé fourni.

12. **annulation de service procédé** : L'annulation d'un service procédé actif est un événement significatif (au sens des transactions) qui désigne que le travail effectué par le service procédé est caduc et que les effets générés par ce service procédé (ex : documents intermédiaires produits, événements, résultats de méthodes) ne sont plus valides et doivent être annulés ou compensés. L'annulation de service procédé est un problème très complexe si l'on considère que les acteurs liés au service procédé annulé sont nombreux (ex : cas de la chaîne de responsabilités des médiateurs d'un service procédé).

Ayant détaillé le cadre d'utilisation et de définition d'un service procédé, nous avons utilisé une modélisation objet pour décrire ces services procédés [35,36]. Nous nous sommes basés sur le formalisme UML pour le faire. Une description complète ainsi que la modélisation de l'implantation a été faite par Karim Baïna dans sa thèse[Baï03].

Notre modèle d'interconnexion de procédés d'entreprises a été défini et présenté dans [29,30,32,35,36] ainsi que par Karim Baina dans son rapport de thèse [Baï03].

Il a, par ailleurs, été implanté 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* [26,28], développé en Java sur un bus CORBA par Karim Baïna.

Un exemple de coopération entre entreprises possédant des systèmes de workflows différents a été traité au sein de *DISCOBOLE* . Les entreprises impliquées ont utilisé ces trois systèmes de gestion de workflows Breeze[DST01], Renew[SSA00] et WorkCoordinator[Hit02] pour la gestion de leurs propres procédés qu'elles désirent interconnecter. Ces trois systèmes de gestion de procédés ont été choisis pour leur souplesse et pour leur interface de gestion de workflow (WAPI).

Enfin notons que, parmi d'autres concepts du modèle, nous avons développé les différents espaces de définition et de publication de services procédés et instancié notre modèle de négociation décrit dans le chapitre précédent et en [27] pour la négociation du profil et du contrat de visibilité de service procédé. En outre, nous avons développé le modèle de services procédés selon une approche de patrons architecturaux (architectural pattern), approche que nous synthétiserons dans le chapitre suivant.

8.7 Conclusion

L'interconnexion des procédés métiers devient très importante dans l'économie d'Internet. En effet, les mécanismes d'interconnexion de procédés s'avèrent nécessaires pour renforcer la conscience de groupe au sein des entreprises virtuelles, pour faciliter les transactions électroniques et pour ouvrir les procédés des entreprises sur les places de marché internet. Ayant été développés pour répondre aux besoins internes des entreprises, la plupart des systèmes de gestion de procédés, ne sont pas adaptés à la coopération inter-enterprise. De plus, les modèles et solutions existantes, sont en général, propriétaires (i.e. basés sur : des langages spécifiques de définition de procédés métiers (BPDLs), des plateformes SGWFs particulières, des formats privés d'échanges de données,...), ou traitent des aspects limités aux formats de messages, aux règles de leur transformation et aux protocoles de communication sans pour autant répondre au problème d'interconnexion des procédés. Par ailleurs, les plate-formes orientées services existantes manquent de dynamicité, d'interopérabilité et restent immatures pour résoudre tous les

problèmes d'interconnexion des procédés d'entreprises. Notre travail est une contribution dans le domaine de l'interconnexion des services procédés. Il fournit un cadre dédié à la définition, à l'échange et à l'interconnexion de services procédés. En effet, afin d'ouvrir les procédés coopératifs entre eux sans négliger les aspects de confidentialité des procédés, nous avons traité l'interconnexion de procédés d'entreprises selon deux angles complémentaires : la "visibilité" et la "dynamicité". Pour modéliser notre modèle d'interconnexion de procédés d'entreprises, nous avons élaboré notre modèle de procédés et notre modèle de services procédés suivant une approche orientée service procédé. Pour synthétiser, un service procédé est une entité logicielle, ou un patron, capable de présenter les particularités et les objectifs d'un procédé sans en révéler la structure. Il joue le rôle de "proxy" et "d'adaptateur" de procédé. En fait, si le proxy se limite à fournir la même interface (ou un sous-ensemble de l'interface) de son sujet, l'adaptateur procure une nouvelle interface à l'objet qu'il adapte. Un service procédé contrôle dynamiquement l'accès aux objets procédés et enrichit son interface par de nouvelles fonctionnalités telles le calcul de correspondance (*matching*) de services procédés, l'exploration du voisinage de services procédés et la négociation de services procédés. Il rend ainsi l'interconnexion de services procédés flexible et dynamique plutôt que statique et planifiée ce qui est très important si l'on veut assurer l'interopérabilité et l'interconnexion des procédés d'entreprises sur Internet.

Si notre modèle nous a permis d'apporter une réponse intéressante au problème d'interconnexion des procédés, il n'en reste pas moins que notre travail de recherche n'est pas encore terminé. En effet, dans nos travaux actuels, nous n'avons fait qu'effleurer le problème de la cohérence de l'interconnexion des procédés : comment rester cohérent si un des services impliqués dans l'interconnexion est bloqué (ex : à cause d'une panne ou d'une exception) ou annulé ? Comment rester cohérent si un des services fournis est lui même sous-traité à une partie externe à l'interconnexion (i.e. cohérence de la médiation de services) ? Comment rester cohérent si le service fourni est, non pas un service atomique, mais une composition de services (i.e. cohérence de la composition de services) ? etc. Par ailleurs, ce service procédé que nous avons introduit et formalisé n'est qu'une étape vers un patron architectural de coopération que nous synthétisons dans le chapitre suivant.

9

Vers un patron architectural de coopération

9.1 Introduction

Après avoir présenté, dans le chapitre précédent, un modèle dynamique d'interconnexion et de coopération de procédés d'entreprises, qui se base sur une architecture orientée service (voir également [36]), nous allons aller plus loin en proposant un patron de coopération. Au-delà de la spécificité du contexte d'interconnexion et de coopération de procédés d'entreprises, notre patron de coopération «Service» essaye d'apporter une solution abstraite au besoin d'interconnexion pour la coopération de composants dans un contexte de coopération générique et globale. En isolant les principes fondamentaux, en identifiant la structure générale et en décrivant les collaborations et l'implantation des composants de notre patron, nous avons pour objectif de proposer un moyen utile à l'élaboration d'une conception orientée objet réutilisable, modulaire et facilement maintenable répondant aux besoins de coopération émergeant. En effet, les patrons facilitent la réutilisation de solutions de conception et d'architectures efficaces. En clair, ils aident le concepteur à obtenir plus rapidement une conception plus ajustée à un problème bien identifié. Ils documentent les connaissances existantes et nous aident à trouver une solution appropriée pour les problèmes conceptuels. En conclusion, ils fournissent une boite à outils mentale aidant à construire un logiciel qui satisfait à la fois les exigences fonctionnelles et non fonctionnelles de l'application. Ce travail est dans la même veine que les travaux sur les patrons d'interconnexion de services de [vdABtHK00], [BDFR03] et [FDBP01].

Comme nous l'avons vu dans le chapitre précédent, ce patron a fait l'objet d'une première implantation pour réaliser un modèle orienté services pour la coopération et l'interconnexion de procédés d'entreprises. L'interconnexion de services procédés d'entreprises est, comme nous l'avons vu dans le chapitre précédent, la possibilité de lier des procédés d'entreprises en rendant les entreprises conscientes des services procédés requis et fournis et de leur permettre de négocier des contrats d'interconnexion avant une collaboration effective. Le modèle d'interconnexion de services procédés permet ainsi aux entreprises coopérantes de structurer, classifier et comparer leurs services procédés (mise en correspondance -*matching*-), sélectionner dynamiquement un service procédé fourni parmi ceux qui correspondent (au sens du matching) à un service procédé requis donné, négocier des services procédés selon leur signature (profil + API), et finalement rendre possible la coopération entre leurs procédés métiers à travers une mise en liaison contractuelle *wrapping*.

Notre objectif, maintenant, est d'exprimer la généricité sous-jacente au concept d'intercon-

nexion de services procédés sous forme d'un patron architectural pour l'interconnexion de composants distincts et hétérogènes (workflows, bases de données, processus, agents, etc.) que l'on nommera «Service». Le patron architectural «Service» exprimera un schéma d'organisation structurelle et fondamentale pour des systèmes logiciels hétérogènes coopératifs. Pour ce faire, nous allons suivre une approche basée sur les patrons architecturaux *Architectural Patterns*[BMR+95]. En effet, nous allons essayer d'abstraire un couple problème/solution (*i.e* besoin de coopération / service) en relevant les facteurs communs, ce qui va nous permettre d'en synthétiser un modèle de solution générique. Notre patron décrit un problème conceptuel particulier et récurrent qui surgit dans un contexte conceptuel spécifique qui est celui de la coopération, et présente un schéma générique et sûr comme solution. Cette solution est spécifiée à travers la description de ses composants, leur responsabilités et la manière dont ils collaborent.

9.2 Le patron Service

Le patron «Service» partage certains aspects architecturaux avec d'autres patrons de type *From Mud to structure.* En particulier, il nous aide à éviter un «océan» de composants en supportant la décomposition d'un système coopératif en sous-tâches qui traitent des aspects structuraux et organisationnels tel que les rôles, et des aspects comportementaux tel que les droits d'accès, la sécurité, l'authentification, etc. Notre patron fournit, entre autres, un schéma conceptuel pour les systèmes distribués. En effet, il traite des besoins de coopération dynamique qu'expriment des composants géographiquement éloignés, en spécifiant le cœur structurel et fonctionnel de l'application solution. Il découple ainsi les acteurs des traitements nécessaires à la coopération.

9.2.1 Exemple et contexte

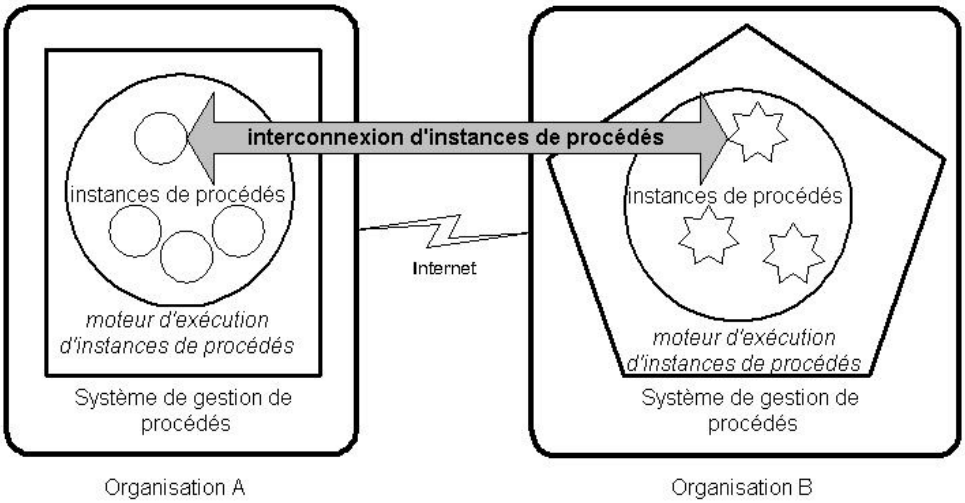
Comme nous l'avons présenté dans le chapitre précédent, nous utilisons, pour supporter la coopération des procédés d'entreprises, l'approche orientée services. Cette approche consiste à définir, publier, négocier et lier des services procédés distants qui représentent les procédés des entreprises. Un service procédé fourni par une entreprise spécifie la quantité de travail qu'elle promet de réaliser sous un contrat de coopération. Il encapsule par ce fait un procédé ou une partie d'un procédé représentant le savoir-faire de cette entreprise. Symétriquement à un service procédé fourni, un service procédé requis par une entreprise représente une activité de procédé dont cette dernière a besoin et ne peut pas réaliser elle-même et que nous synthétisons par la figure 9.1.

Cet exemple n'est en fait qu'une instance de problème ou s'exprime un besoin de coopération dans un environnement de composants hétérogènes. Dans ce contexte de coopération dans un environnement hétérogène, les applications manipulent un nombre considérable de composants distincts et hétérogènes (workflows, bases de données, processus, agents, etc.). Ces composants sont souvent amenés à coopérer entre eux pour réaliser des traitements communs à ces applications. Nous décrivons ci-dessous les éléments de problème que le patron service est amené à résoudre :

1. L'hétérogénéité des composants :

   (a) l'hétérogénéité de leurs infrastructures informatiques : matériels, réseaux, systèmes d'exploitations, etc ;

Figure 9.1 Interconnexion de procédés inter-entreprises



(b) l'hétérogénéité de leurs environnements d'exécution : langages et syntaxes de définition des composants, sémantiques des langages de définition des composants, structures des données utilisées par les composants, etc.

2. La fermeture des composants :

   (a) soit les composants ne fournissent pas d'interface (API -Application Programmable Interface-) ;

   (b) soit les composants fournissent des interfaces propriétaires et donc différentes ;

   (c) soit ces interfaces ne sont pas accessibles pour les applications clientes distantes (par exemple via le web).

9.2.2 Approche pour un patron architectural

Le service est vu comme étant une entité logicielle capable de présenter les particularités et les objectifs d'un composant sans en révéler ni la structure ni l'implantation (par exemple, dans une base de données, un système de gestion de workflow, etc.). En effet, un service offre une abstraction fonctionnelle d'un composant (ou d'une partie d'un composant) fourni par une application. Un service spécifie la quantité de travail qu'une application promet de réaliser sous un contrat de coopération. De plus, il spécifie les parties du composant qu'il couvre et les moyens d'y accéder.

Pour mieux supporter la coopération de leurs composants sans pour autant toucher à leur implantation, les applications ont besoin de gérer des méta-données sur leurs composants. Ces méta-données permettront d'effectuer, entre autres, les opérations de gestion classiques des composants (par exemple, recherche des composants selon des propriétés, calcul de correspondance entre composants (gestion et contrôle de versions -*match, diff, like, cmp, extends*-, journalisation des histoires des composants, etc.). De plus, ces méta-données permettront de personnaliser l'accès à ces composants (par exemple, pour des raisons de sécurité, de confidentialité, de gestion de licences, etc.). Cette personnalisation peut être statique (par exemple, la définition de rôles de coopération spécifiques à chaque composant client, etc.) ou dynamique (par exemple, la négociation de rôle de coopération ou de contrat de visibilité et d'accès aux composants, etc.). Le patron

«Service» vise à répondre à un besoin de coopération entre applications dans un environnement de composants hétérogènes. Pour ce faire il gére les méta-données rassemblant le profil et le contexte d'exécution des composants afin d'en assurer la recherche (le calcul de correspondance, de voisinage, etc.) ou la négociation.

Les caractéristiques du patron peuvent être résumées par les capacités offertes suivantes et ceci devant être fait sans modifier l'implantation des composants d'une application :

- Gérer des méta-données sur les composants d'une application : propriétés, méthodes, événements, etc ;
- Créer différentes instances de composants d'une application avec différentes valeurs des propriétés ;
- Pouvoir définir des droits d'invocation des méthodes et de réception des événements des composants d'une application ;
- Pouvoir effectuer des recherches sur les composants d'une application ;
- Pouvoir mesurer la correspondance entre les composants d'une application ;
- Pouvoir négocier avec une application les valeurs des propriétés de quelques-uns uns de ces composants ;
- Pouvoir contrôler l'accès aux composants d'une application selon les rôles (par exemple, pour assurer de la confidentialité, la gestion de licence, la sécurité, etc.) ;

9.3 Structure

Notre patron «Service» se décompose en deux structures :

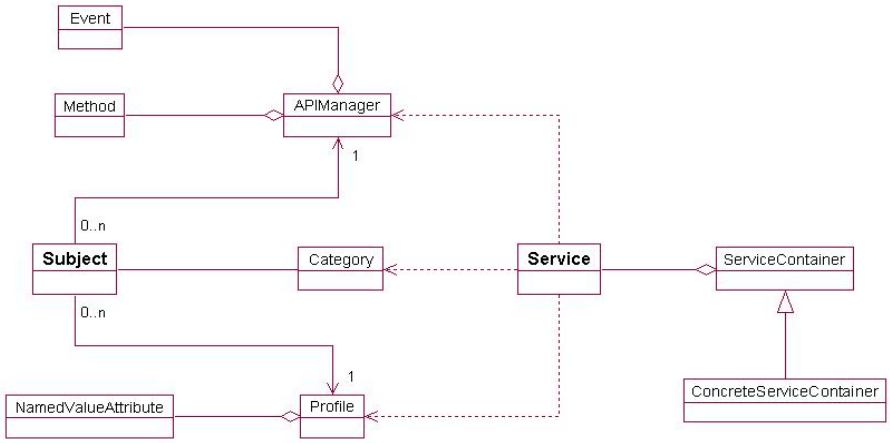
1. structure de définition d'un service : permet de définir et de personnaliser l'accès aux composants et de gérer les masses de composants ainsi définis ;
2. structure de recherche et de négociation : permet d'effectuer des opérations de gestion classiques telle que la recherche des composants d'une application selon des propriétés, la personnalisation dynamique des propriétés ou l'accès à un composant d'application.

9.3.1 Définition d'un service

Le diagramme de la figure 9.2 isole le rôle d'abstraction et d'adaptation que porte notre patron afin de personnaliser l'accès aux composants. Ainsi le «Service» est vu comme une entité logicielle qui encapsule un composant pour ne rendre visible à l'extérieur que les éléments nécessaires à sa coopération avec les composants des autres applications.

- `Subject`
  - Définit une interface objet spécifique à un composant particulier.
- `Service`
  - Gère des méta-données qui rendent possibles la recherche, la négociation et l'accès contrôlé au composant ;
  - Joue le rôle d'un proxy (ou subject par «procuration») :
    - Contrôle dynamiquement l'accès au composant qu'il encapsule ;
    - A l'aptitude d'extraire dynamiquement l'API de son composant, afin d'assurer le contrôle du droit d'invoquer une méthode et d'être notifié d'un événement émanant de ce composant ;
    - N'est pas un simple proxy, car il ne se limite pas à fournir la même interface (ou un sous-ensemble de l'interface) du composant qu'il encapsule, mais il offre également des mécanismes de recherche et de négociation du composant qu'il présente ;

Figure 9.2 Structure de définition d'un service

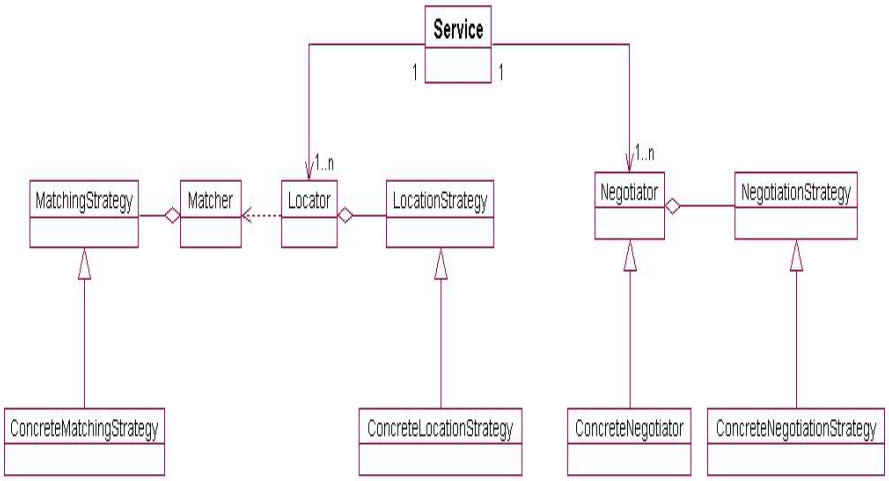


  - Joue le rôle d'un adaptateur vu qu'il procure une nouvelle interface du composant qu'il adapte :
    - Offre des fonctionnalités complémentaires à celles du composant adapté (par exemple, la négociation et la recherche) ;
    - A l'aptitude d'extraire dynamiquement la catégorie et les attributs du composant adapté pour en fabriquer un profil permettant la recherche et la comparaison aux autres composant adaptés ;
    - Un service n'est pas un simple adaptateur du fait qu'il assure également le contrôle d'accès à son composant selon les résultats de la négociation.
- `Category`
  - Permet de catégoriser les objets «`Subject`» et de les classer à l'aide d'une taxonomie spécifique (objet, XML, mots clés, indexations, etc.) au domaine d'application.
- `Profile`
  - Décrit des méta-données de l'objet «`Subject`» sous la forme de couples d'attributs (nom, valeur) spécifiques au domaine d'application.
- `APIManager`
  - Stocke les informations sur l'interface de l'objet «`Subject`» en termes de méthodes («`Method`») et événements («`Event`») ;
  - Permet de contrôler l'accès à l'interface de l'objet «`Subject`» négociée auparavant.
- `ServiceContainer`
  - Définit un référentiel des objets «`Service`» nécessaire à leurs recherches, négociations et accès ;
  - Permet d'appliquer des traitements globaux aux services de même type (par exemple analyse, fouille, transformation, vérification etc.) ;
  - Rassemble une vue des objets «`Service`» selon des critères particuliers.
- `ConcreteServiceContainer`
  - Définit un référentiel de services «`ServiceContainer`» réel (par exemple, `RequestedServiceContainer` : espace des services requis, `ProvidedServiceContainer` : espace des services fournis, `WrappedServiceContainer` : espace des services dont la signature de contrat de coopération a déjà été établie).

9.3.2 Recherche et négociation d'un service

Le diagramme de la figure 9.3 isole les opérations de recherche des composants (par exemple, calcul de correspondance ou de voisinage entre composants) et de négociation de composants (par exemple, de rôle ou de contrat de visibilité d'APIs et d'accès aux composants). Ces opérations sont les outils d'une coopération dynamique et efficace (répondant au mieux aux besoins des acteurs). En effet, d'une part les outils de définition et de recherche ont pour but de situer et positionner l'ensemble de composants fournis/requis. D'autre part, la négociation permet d'éliminer les composants qui correspondent partiellement au sens de calcul de correspondance (*matching*).

Figure 9.3 Recherche et négociation de service



- `Locator`
  - Gère la localisation de l'objet «`Service`» ;
  - Permet de rechercher les réalisations de l'objet «`Subject`» selon une interface fournie (méthodes, évenements) ;
  - Est composé de stratégies «`LocationStrategy`» de localisation d'objets «`Service`» ;
  - Dépend de l'objet «`Matcher`» qui assure le calcul de correspondance nécessaire à toute localisation de service.
- `LocationStrategy`
  - Gère dynamiquement la famille d'algorithmes de recherche des objets «`Service`» ;
  - Déclare une interface commune à tous les algorithmes de recherche des objets «`Service`» ;
  - Permet de découpler et de rendre ces algorithmes de recherche interchangeables et indépendants ;
  - Peut être réalisé par plusieurs objets «`ConcreteLocationStrategy`» (par exemple «`NeighbourhoodLocationStrategy`» qui se regroupe les algorithmes basés sur des modèles de calcul de voisinage)
- `Matcher`
  - Gère le calcul de correspondance de l'objet «`Service`» ;
  - Est composé de stratégies «`MatchingStrategy`» de calcul de correspondance d'objets «`Service`».

- `MatchingStrategy`
  - Déclare une interface commune à tous les algorithmes de calcul de correspondance des objets «`Service`»
  - Permet de gérer des algorithmes de calcul de correspondance, de les raffiner ;
  - Permet de choisir durant le temps d'éxécution (*runtime*) l'algorithme de calcul de correspondance à exécuter dynamiquement ;
  - Peut être réalisé par plusieurs objets «`ConcreteMatchingStrategy`» (par exemple, «`EuclidianMatchingStrategy`» basé sur la distance euclidienne et «`HammingMatchingStrategy`» basé sur la distance de hamming).
- `Negotiator`
  - Gère la négociation de l'objet «`Service`» ;
  - Permet de négocier l'accès à l'interface de l'objet «`Subject`» ;
  - Connaît l'ensemble des négociations concernant l'objet «`Service`» ;
  - Permet d'appliquer des traitements globaux aux négociations de même type (par exemple, analyse, tactique de décision/de coordination) ;
  - Est composé de stratégies de négociation «`NegotiationStrategy`» d'objets «`Service`».
- `ConcreteNegotiator`
  - Définit un objet «`Negotiator`» réel (par exemple, pour négocier l'API du service ou son profil).
- `NegotiationStrategy`
  - Déclare une interface commune à tous les algorithmes de négociations des objets «`Service`» ;
  - Peut être réalisé par plusieurs objets «`ConcreteNegotiationStrategy`» (par exemple, « `LanguageProtocolTacticsNegotiationStrategy`») [27].

9.4 Dynamique

Les scénarios suivants explicitent le comportement dynamique de notre patron «Service». Pour simplifier, nous traitons la dynamique avant et après la signature du contrat de coopération en deux scénarios distincts. La signature du contrat de coopération intervient après les phases de publication, de recherche et de négociation de services (*cf.* scénario 1). Ensuite, commence la coopération effective des composants *demandeur* et *fournisseur* de services (*cf.* scénario 2).

9.4.1 Scénario 1 : Publication, Recherche et Négociation des services

Ce premier scénario désigne les états par lesquels passe un composant *demandeur* en vue de trouver un objet composant *fournisseur* qui répond à ses besoins d'externalisation. Notre scénario se décompose en trois phases principales : définition/publication, recherche et négociation.

- Le composant *demandeur* `localComponent` définit le service requis qu'ils souhaite externaliser. L'objet `localSrv`, ainsi définit, décrit des méta-données du composant *demandeur* sous la forme de couples d'attributs (nom, valeur) spécifiques au domaine d'application et stocke les informations sur l'interface du composant requis en termes de méthodes `Method` et événements `Event`. Ceci permet de catégoriser les composants et de les classer à l'aide d'une taxonomie spécifique au domaine d'application. Parmi les travaux de recherche et de normalisation dans ce domaine nous pouvons citer[UO00] et [UDD00]. Le service `localSrv` est ensuite publié dans le référentiel `RequestedServiceContainer` (espace des services requis) ;

- Symétriquement, un ou plusieurs composants *fournisseur*s définissent les services fournis qu'ils souhaitent exporter. Ainsi, les objets `remoteSrv1i` et `remoteSrv2j`, qui sont de la même catégorie que le service `localSrv`, sont publiés dans le référentiel `ProvidedServiceContainer` (espace des services fournis) ;
- Il faut préciser que les services d'une même communauté doivent a priori s'accorder sur un langage commun définissant un ensemble de descriptions ou signatures. Ces signatures détaillent les catégories (types et classification) en fonction du profil et de l'API des services. Il est à noter que la coopération ne sera possible qu'entre composants de même type ;
- Après la première phase de définition, le composant *demandeur* `localComponent` sollicite, en choisissant une stratégie, sa localisation à l'objet `Locator`. Le but est de se situer et positionner l'ensemble de services fournis d'une même catégorie par rapport à l'objet `localSrv` référence, en vue de trouver ceux qui répondent au mieux à ses besoins d'externalisation. L'objet `localSrv` invoque alors la requête `localize_service` sur l'objet `Locator` qui suivra la stratégie passée en paramètre par le composant *demandeur* ;
- L'objet `Locator` invoque la requête `find` sur le référentiel `ProvidedServiceContainer` (espace des services fournis) pour chercher les réalisations possibles de composants fournis de même catégorie ou de catégorie proche de la taxonomie ;
- A la récupération de ces réalisations, l'objet `Locator` calcule en sollicitant l'objet `Matcher` leurs distances respectives au service requis de référence. Ce calcul de correspondance, qui suit une stratégie en relation avec la stratégie de localisation, dépend des profils adoptés par chaque composant ;
- Le composant *demandeur* `localComponent` sollicite l'objet `Negotiator` pour négocier avec les services fournis résultats de la phase de recherche. Le but est d'obtenir un profil et un contrat de visibilité d'API (un sous ensemble négocié des méthodes et évenememnt du composant *fournisseur)* correspondant au mieux aux besoins du composant *demandeur*. Le `Negotiator` choisit une stratégie de négociation `NegotiationStrategy` qui permet de raffiner l'exploration et le choix des services trouvés en sélectionnant ceux qui font le plus de compromis. Ainsi, la négociation permet de distinguer d'une manière plus fine entre les services fournis d'un même voisinage et ainsi résoudre un certain indeterminisme de choix de service ;
- La négociation se caractérise par un échange de messages (comprenant le profil et l'API négocié ou proposé) entre le `localNegociator` et le `remoteNegotiator`. Cet échange peut suivre une ou plusieurs tactiques qui peuvent être dynamiques ou statiques et où l'intervention des partenaires (demandeur/fournisseur) peut varier selon la stratégie de négociation `NegotiationStrategy` adoptée ;
- A la fin de la phase de négociation le composant *demandeur* `localComponent` peut choisir un composant *fournisseur* `remoteComponent` avec lequel il va valider un contrat de coopération qui est le résiultat d'un accord établi lors de leur négociation ;
- Les deux composants *demandeur* `localComponent` et *fournisseur* `remoteComponent` représentent des vues distribuées du même service. La coopération sera effective dés que le composant *demandeur* active, à travers le service `localSrv`, le composant *fournisseur* par l'appel de méthode `wrap`. L'appel de cette méthode conduit a la publication du couple de services dans le référentiel `wrappedServicesContainer` (espace des services dont le contrat de coopération a déjà été signé). Cette coopération peut être très variée : elle peut consister en de simples invocations de méthodes ou en l'envoi de messages faisant partie des événements au cours de l'exécution du service (*cf.* le scénario suivant pour plus de détails).

9.4.2 Scénario 2 : Coopération effective des services

Dès que les `composants` *demandeur* `localComponent` et *fournisseur* de services `remoteComponent` parviennent à signer un contrat de coopération, le composant *demandeur* peut entamer l'exécution effective du service requis `localWrappedService`. Le scénario suivant nous montre une séquence du mécanisme d'invocation de méthodes et de notification d'événements lors de la coopération.

- Le composant *demandeur* `localComponent` invoque la requête `find` sur le référentiel `wrappedServices` (espace des services dont le contrat de coopération a déjà été signé). Si le service requis `localWrappedService` a été sujet d'un contrat de coopération, le composant `localComponent` en récupère la référence ;
- Le composant *demandeur* `localComponent` active le composant *fournisseur* `remoteComponent` indirectement à travers les services requis `localWrappedService` et fourni `remoteWrappedService`.
- Le composant *demandeur* `localComponent` peut alors invoquer les méthodes négociées du service. Il doit préalablement interroger l'objet responsable de la vérification du respect du contrat de coopération `localContract` pour contrôler que ces méthodes figurent dans l'API négociée ;
- Le composant *fournisseur* `remoteComponent` effectue le traitement interne de la méthode invoquée ;
- Si un événement vient à se produire au sein du composant *fournisseur* `remoteComponent`, ce dernier notifie le composant *demandeur* `localcomponent` de cet événement. Une vérification préalable à cette notification doit être effectuée par l'objet remoteContract responsable de la vérification du respect du contrat de coopération chez le fournisseur.

9.5 Application du patron à l'interconnexion de procédés

Reprenons l'exemple défini dans la section 9.2.1. En appliquant au workflow d'entreprise la première structure de «Définition de Service» de notre patron , on définit une interface d'accès générique et standard au workflow (*cf.* figure 9.4). L'abstraction du workflow d'entreprise sous la forme d'un «Service Procédé» étant faite, le patron «Recherche et Négociation de Service» prendra en charge l'interconnexion de services procédés d'entreprises (*cf.* figure 9.5) et la possibilité de lier leurs procédés en les rendant conscients des services procédés requis et fournis. Ceci permettra aux entreprises de coopérer en interconnectant leurs workflows. En effet, les entreprises réalisent leur coopération en sélectionnant dynamiquement un service procédé fourni parmi ceux qui correspondent (au sens du matching) à un service procédé requis donné et en négociant des services procédés selon leur profil et/ou API.

L'application de notre patron «Service» se fait sur deux phases principales distinctes mais étroitement liées.

9.5.1 Définition et publication de services procédés

Nous définissons un service procédé comme étant une abstraction significative, une présentation sémantique et fonctionnelle d'un workflow d'entreprise(*cf.* figure 9.4). Il est une entité logicielle capable de présenter les particularités et les objectifs d'un workflow ou d'une partie

Figure 9.4 «service procédé» : le patron «Service» appliqué au procédé

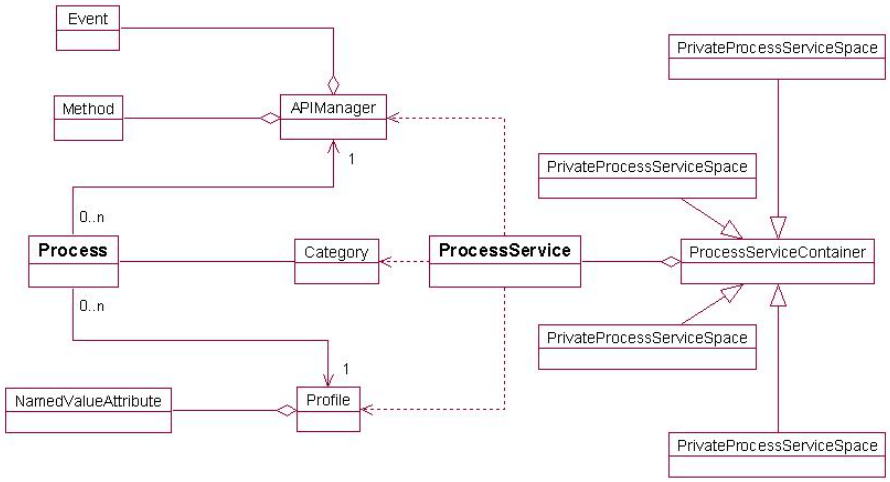
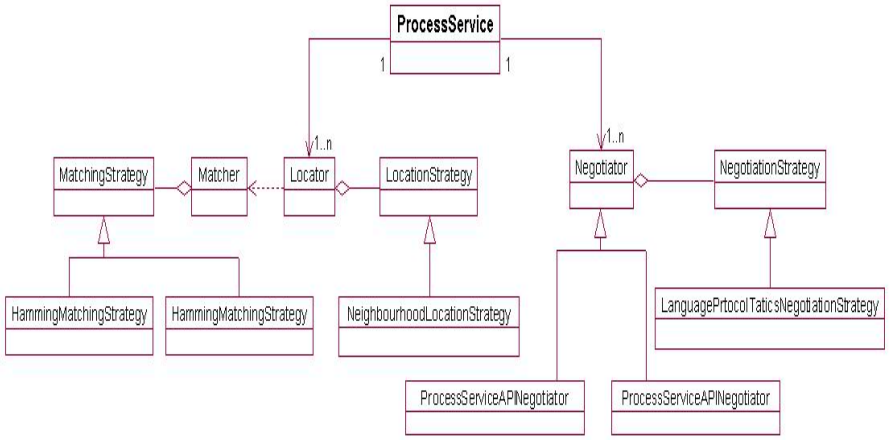


Figure 9.5 Stratégies de recherche et de négociation lié a un service



sans en révéler ni l'environnement d'exécution (par exemple, logiciel, matériel, réseaux, etc.) ni les détails d'implantation de son schéma d'exécution. Il présente les propriétés significatives (`Profil`) et l'API du procédé (`Method, Event`). Un service procédé est classé selon une catégorie (`Category`) qui en définit le type et permet de le positionner. Pour ce faire, les entreprises, au sein d'une même communauté métier, se mettent d'accord à propos d'un langage commun de services procédés (par exemple, ontologies de concepts métiers clefs, taxonomies de services procédés métiers, etc.) afin de définir et de comprendre les services procédés. Il est, par la suite, publié selon son état (privé, requis, founis, sous contrat de coopération) dans un des référentiels de services (`PrivateProcessServiceSpace, RequestedProcessServiceSpace, ProvidedProcessServiceSpace, WrappedProcessServiceSpace)`.

Recherche, sélection, négociation et interconnexion de services procédés

La sélection d'un service procédé fourni répondant à un service procédé requis se base sur des algorithmes de calcul de correspondance (`EuclidianMatchingStrategy, HammingMatching-Strategy`) et de recherche de voisinage `(NeighbourhoodLocationStrategy)` , qui sont pris en charge par les objets «`Matcher`» et «`Locator`» (*cf.* figure 9.4). L'objectif de ces algorithmes n'est pas d'automatiser totalement la phase de recherche de services procédés. Ils sont à considérer comme des outils d'aide à la décision et à la recherche des besoins de l'entreprise. Dans ce contexte, le demandeur de service procédé visant à faire faire un service procédé requis par une entreprise tierce spécifie la signature de son service procédé. Cette signature représente un contrat d'interconnexion initial qui sera à la base des interactions préalables à la coopération. La recherche peut être également faite par le fournisseur pour confronter ses offres de services procédés aux besoins des autres entreprises. Entre autres, la signature du service procédé (profil et API) à externaliser va permettre au demandeur du service de trouver le meilleur couple abstrait(requis)/concret(fourni) qui va répondre à ses besoins d'externalisation.

Une fois que l'entreprise ayant besoin d'une compétence externe a trouvé un ensemble de services procédés fournis qui correspondent à son besoin, une série de négociations entre le demandeur et les fournisseurs de service procédés doivent être menées avant la collaboration effective. Ces négociations, contrôlées et gérées par l'objet «`Negociator`», concernent la co-décision d'un contrat de coopération et peuvent être initiées indifféremment par le demandeur ou le fournisseur du service procédé. Le processus de négociation permet d'éliminer les services procédés qui correspondent partiellement au sens de calcul de correspondance (*matching*) mais qui n'offrent pas un contrat d'interconnexion de services procédés satisfaisant.

Un service procédé est ainsi lié à un objet de recherche «`Locator`» qui permet d'appliquer des stratégies de recherche combinées à des stratégies de calcul de correspondance fournies par l'objet «`Matcher`» pour retrouver pertinemment les services procédés voisins. L'objet «`Negociator`» s'occupe de la gestion des processus de négociation Il se charge d'appliquer des stratégies de négociations aux situations de négociation (par exemple, la négociation de profil de service procédé, ou la négociation de contrat de visibilité, etc.). Ces phases de recherche, sélection, et négociation utilisent un ensemble de stratégies qui peuvent être choisies dynamiquement lors d'un *runtime*.

Enfin, la contractualisation du lien entre le couple de services procédés requis/fourni permet d'expliciter les règles qui vont gérer la coopération des workflows d'entreprises. Par conséquent, il devient possible d'envisager une coopération à la fois non planifiée mais maîtrisée entre ces procédés métiers pendant une durée de vie préalablement agréée. Cette coopération peut être très variée, elle peut consister en de simples invocations de méthodes ou en l'envoi de messages faisant partie des événements au cours de l'exécution du service.

Comme nous l'avons vu au chapitre précédent, ce modèle d'interconnexion de services procédés

à été développé et exprimenté au sein de la plate forme *DISCOBOLE* (*DIStributed CO-operation and business prOcess on LinE*), environnement de travail coopératif qui permet à des acteurs distribués de coopérer en interconnectant leurs workflows selon le paradigme d'échange de services procédés. Chaque acteur possède des procédés qui peuvent être gérés par des systèmes de gestion de workflows hétérogènes et au sein d'environnements d'exécution indépendants les uns des autres. 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. Après la recherche des services procédés correspondant à leurs besoins (respectivement compétences), les acteurs négocient des contrats d'interconnexion liant leurs services procédés requis et fournis, puis peuvent commencer à coopérer.

Un exemple de coopération entre entreprises possédant des systèmes de workflows différents a été traité au sein de *DISCOBOLE*. Les entreprises impliquées ont utilisé trois systèmes de gestion de workflows (Breeze[DST01], Renew[SSA00] et WorkCoordinator[Hit02]) pour la gestion de leurs procédés qu'elles désirent interconnecter. Ces trois systèmes de gestion de procédés sont souples et facilement intégrables aux systèmes existants. De plus, ils proposent une interface de gestion de workflow (WAPI) et supportent des facilités développement de programmes associés aux workflows.

9.6 Générécité/Généralisation

Nous sommes convaincus qu'il existe de nombreuses utilisations possibles du patron service pour gérer des entités logicielles, matérielles ou réseaux qui présentent un certain nombre d'opérations, de notifications, qui doivent supporter des mécanismes de recherche, de négociation et de contrôle d'accès. Outre l'utilisation de ce patron dans le cadre de l'interconnexion de services liés aux procédés d'entreprises qui découle logiquement et naturellement de notre approche, nous pouvons citer comme autre exemple d'utilisation du patron service, l'impression sur internet (IPP). Nous allons détailler le service d'impression sur Internet IPP (*Internet Printing Protocol*) qui reflète une utilisation remarquable du patron architectural "service".

Le service d'impression tel décrit par le RFC [IET00a] du protocole d'impression par Internet IPP possède les caractéristiques suivantes :

- un *profil* : (*printer-name*, *printer-uri-supported*, *printer-location*, *printer-state*, *color-supported*, *multiple-document-jobs-supported*, *document-format-supported*, *printer-resolution*, etc.) [IET00b] ;
- des *opérations* : (*Print-Job*, *Get-Jobs*, *Pause-Printer*, *Purge-Jobs*, *Get-Printer-Attributes*, etc.) [IET00b] ;
- des *événements* : (*printer-is-accepting-jobs*, *job-state*, *job-impression-completed*, etc.) ;
- des besoins en mécanismes de *recherche* de service d'impression : dans un réseau d'imprimantes par rapport au nom de l'imprimante *printer-name*, ou par rapport aux coordonnées stockées dans l'attribut *printer-uri-supported* (ensemble des URI identifiant l'imprimante), ou par rapport aux informations stockées dans l'attribut *printer-location* (chaîne de 127 octets au maximum décrivant la localisation d'une imprimante en langage naturel -par exemple, «Salle B201, 2^{me} étage, bâtiment B»-), ou par rapport aux valeurs des autres attributs [IET00b] ;
- des besoins en mécanismes de *négociation* : de format de contenu à supporter par l'imprimante ou en général des valeurs des attributs de l'imprimante [IET01, IET99] ;
- et des besoins en mécanismes de définition de *rôle* et de *contrôle d'accès* : en impression, en supervision totale, etc.[IET00b].

9.7 Conclusion

La définition du patron service rappelle et s'appuie sur les patrons de conception proxy et adaptateur [GHJV94] mais avec des différences et des spécificités notables :

- **Proxy** : Certes, un service joue le rôle d'un proxy (ou composant de «procuration») car il contrôle dynamiquement l'accès au sujet qu'il encapsule. En effet, un service a l'aptitude d'extraire dynamiquement l'API de son sujet, afin d'assurer le contrôle du droit d'invoquer une méthode et d'être notifié d'un événement émanant de ce sujet. Cependant, un service n'est pas qu'un simple proxy, car il ne se limite pas à fournir la même interface (ou un sous-ensemble de l'interface) du sujet qu'il encapsule mais il offre également des mécanismes de recherche et de négociation du sujet qu'il présente ;
- **Adaptateur** : Certes, un service joue le rôle d'un adaptateur vu qu'il procure une nouvelle interface du sujet qu'il adapte. En effet, un service offre des fonctionnalités nécessaires à l'interconnexion qui sont complémentaires à celles du sujet adapté (par exemple, le calcul de correspondance (*matching*), la négociation, la recherche, etc.). Pour ce faire, un service a l'aptitude d'extraire dynamiquement la catégorie et les attributs du sujet adapté pour en fabriquer un profil permettant la recherche et la comparaison aux autres sujets adaptés. Cependant, un service n'est pas un simple adaptateur du fait qu'il assure également le contrôle d'accès à son sujet selon les résultats de la négociation.
- **Stratégies** : L'utilisation du patron `Stratégies` permet de gérer dynamiquement la famille d'algorithmes de calcul de distance, de mise en correspondance, de calcul de voisinage ou de négociation. En effet, il spécifie une interface qui encapsule ces stratégies dans des classes séparées et permet ainsi de les découpler et les rendre interchangeables et indépendantes. Ceci permet de choisir et de modifier des algorithmes et d'en rajouter de nouveaux dynamiquement. Ainsi, ce patron appliqué aux composants «`MatcherStrategy`», «`DistanceStrategy`» ou «`NegociationStrategy`» permet de mieux gérer l'ensemble de ces stratégies. Ce qui fait que pour chacune des stratégies précédemment citées on aura les moyens de les implanter différemment et de choisir dynamiquement l'algorithme à exécuter.

A travers ce dernier chapitre, nous avons décrit un patron architectural qui propose une solution conceptuelle au besoin d'interconnexion et de coopération dans un contexte générique et global. Notre patron «Service» aide à l'élaboration d'une conception orientée objet réutilisable, modulaire et facilement maintenable répondant aux besoins de coopération entre composants souvent hétérogènes. Notre patron «Service» exprime une architecture des systèmes logiciels hétérogènes coopératifs. Il procède en deux étapes :

- Il définit, personnalise l'accès aux composants et spécifie un référentiel qui permet de gérer les masses de composants sujets de coopération ;
- Il spécifie les opérations de gestion pour l'interconnexion telle que la recherche des composants ou la personnalisation dynamique des propriétés des composants par la négociation.

Nous avons présenté dans ce chapitre deux exemples d'applications de notre patron « Service » : l'un concernant l'interconnexion de services liés aux procédés d'entreprises, et l'autre concernant la coopération de services d'impression sur Internet. Notre proposition est une première itération pour fournir un haut niveau d'abstraction pour la conception, la construction et la maintenance d'outils d'interconnexion et de coopération entre composants. Pour confirmer le bien-fondé de nos choix conceptuels, la recherche et l'analyse d'autres cas d'utilisation seraient souhaitables.

10

Quelques pistes de recherche

Comme nous l'avons vu dans ce document, j'ai abordé la coopération sous deux angles principaux, à savoir l'échange de documents ou de services et la négociation de ces échanges. Ces deux angles "d'attaque" offrent de nombreuses possibilités de recherches futures. Tout au long de mon travail, je me suis évertué à travailler par étapes successives, et à apporter un premier élément de réponse au problème étudié avant de passer au problème suivant. L'objectif de ce court chapitre est de présenter quelques unes des pistes de recherche que ce travail pourrait initier, en oubliant l'aspect incrémental utilisé jusqu'a présent dans ma démarche. Certaines de ces pistes de recherches pourraient être menées de front.

L'échange de documents

Si le nouveau modèle de transactions avancé que nous avons défini et qui supporte les exécutions de transactions distribués géographiquement et coopérant par échange de résultats intermédiaires [24,25,26,28] apporte une réponse satisfaisante au niveau de certains des besoins que nous avons exprimés, il n'en reste pas moins que de nouveaux modèles de transaction peuvent avoir leur intêret. La preuve en est que lorsque nous avons abordé le problème de la négociation, la définition de ce modèle de négociation nous a fait introduire un nouveau paradigme de coopération (ou critère de correction des négociations) (la prise de tour) qui est venu s'ajouter à ceux définis dans le cadre des modèles transactionnels COO et DisCOO (client/serveur, rédacteur/relecteur et écriture coopérative).

Cette approche transactionnelle pour les échanges de document et la négociation de ces échanges, très intéressante par son cadre formel et les nombreuses possibilités apportées peut servir de base à de nombreux développement et approches (systèmes transactionnels spécifiques développés pour telle ou telle activité (par exemple la CAO), systèmes transactionnels génériques avec tel ou tel critère de correction spécifique,...). Il n'en demeure pas moins que cette richesse en constitue aussi le principal inconvénient. Comment comparer deux modèles de transactions voisins ? Les avantages induits par l'utilisation d'un nouveau modèle ne pâtissent ils pas de la perte des critères classiques des systèmes transactionnels classiques ? Comment utiliser un modèle développé pour une activité spécifique dans le cadre d'applications différentes ? Par ailleurs, les utilisateurs de systèmes transactionnels avancés peuvent avoir l'impression de perdre le contrôle de leur activité dans la mesure où de tels systèmes ayant pour objectif principal la consistance des données échangées brideront les interactions "sociales" utilisées classiquement dans le cadre des échanges informels entre acteurs coopérant. Une direction de recherche intéressante consisterait à offrir des mécanismes de base permettant la prise en compte de modèles transactionnels dans

lesquels les utilisateurs pourraient injecter de nouveaux critères liés à leurs besoins spécifiques. C'est ce que nous avons fait lors de l'intégration des aspects liés à la négociation dans notre système. Cependant la façon dont sont pris en compte les critères de correction dans *DisCOO* fait que cette maléabilité de notre système est loin d'être accessible à l'utilisateur standard. De ce point de vue, notre plate-forme d'expérimentation *DisCOO* doit plus être vue comme une plate-forme d'expérimentation de travail coopératif que comme une réelle plate-forme support aux modèles transactionnels comme peuvent l'être des plate-formes comme ASSET[BDG+94], TSME [GHF96] ou CAGISTrans[RN04]. Transformer *DisCOO* en une réelle plate-forme support aux modèles transactionnels pourrait être un travail intéressant mais de longue haleine dans la mesure ou il faudrait remplacer un certain nombre de programmes écrits "en dur" par des possibilités offertes à l'utilisateur de spécifier ses critères de correction par le biais d'une interface utilisateur adaptée. La gestion de plusieurs types de spécifications (spécifications transactionnelles de "base" et spécifications de l'utilisateur) ne peut, par ailleurs, se faire sans l'utilisation d'outils formels de vérification et de validation des modèles transactionnels résultant. Une autre voie de continuation "naturelle" de ces travaux de recherche pourrait être la prise en compte de la mobilité dans notre approche. En effet notre approche distribuée pourrait se placer aussi dans le cadre d'environnements mobiles dans lesquels certains des acteurs ne seraient pas connectés en permanence. De nombreux travaux ont été menés pour étendre certains modèles de transactions avancés afin de pouvoir les utiliser dans des environnements mobiles, comme Moflex [KK00] étandant les transactions Flexibles ou Kangourou [DHB97] utilisant les concepts de transactions emboîtées ouvertes et des transactions split, ou encore HiCoMo [LH02]. Le lecteur intéressé pourra se reporter avec intérêt à [SA04] qui présente un état de l'art très complet sur les transactions dans les environnements mobiles, allant des travaux de recherche aux produit commerciaux, avant de proposer AMT, un nouveau modèle de transactions mobiles adaptables.

L'échanges de services

Au delà de la coopération par échange de documents telle que nous l'avons définie et présentée dans un premier temps, nous avons introduit et présenté un modèle de coopération par échange de services procédés. Ce modèle formalise et développe le paradigme de coopération par échange de services procédés et se concrétise par la définition, la publication, la recherche, la négociation et l'interconnexion de services procédés. Ainsi, chaque entreprise dispose de référentiels de définition et de publication de services procédés qui lui permettent d'exprimer ses compétences et ses besoins en termes de procédés-métiers. Chaque entreprise, souhaitant entrer en coopération, recherche les services procédés qui lui conviennent, négocie des contrats d'interconnexion de ses services procédés avec ceux de ses partenaires et peut alors commencer à coopérer. Ce nouveau modèle d'interconnexion de procédés d'entreprises a été défini et présenté dans [29,30,35,36].

Cette approche consistant en la possibilité de définir des processus individuels que doivent suivre les acteurs (description de procédés grâce à des systèmes de workflow ou autres) et d'interconnecter ces procédés en négociant des "compromis" entre différents procédés est en fait une mine dont nous n'avons exploité qu'une des veines. En effet, d'une part, nous n'avons traité que de l'échange de services procédés et, d'autre part, nous avons considérés ces services échangés comme atomiques. Concernant les procédés et leur modélisation (BPM ou Business Process Modeling), des pistes de recherches nombreuses sont possibles et toute une communauté est active dans le domaine. Nous pouvons, en particulier, citer les travaux autour de BPMN (Business Process Modeling Notation) [Whi04]. La modélisation de procédés est une thématique de recherche à part entière et nous n'avons pas la prétention de résoudre tous les problèmes ni de détailler

toutes les pistes de recherche possible. Nous allons nous limiter à ce que nous avons traité, à savoir l'interconnexion de procédé. Dans ce cadre, l'atomicité des services, si elle nous a permis de répondre partiellement aux besoins d'interconnexion de procédés, n'en demeure pas moins une limitation sérieuse à l'utilisation de tels services, dans un cadre réel, avec la possibilité de combiner des services existant, tout en ayant un contrôle sur le service composé obtenu. Deux approches de composition de services électroniques existent : la première s'appuyant sur les modèles transactionnels (WSCA -Web Service Composition Action- [TIRL02], BTP -Business Transaction Protocol- [Com04], XLANG [Tha01] , etc.) et la deuxième sur les modèles de workflows (ebXML [UO00], WSCI -Web Service Choreography Interface- [SISM02], WSFL -Web Service Flow Language- [Ley01], etc.). Cependant, aucune des solutions existantes n'a atteint un stade de formalisation définitif. Une piste intéressante de recherche serait la définition d'un cadre plus formel qui s'appuierait sur les protocoles d'orchestration et de chorégraphie de services. L'objectif étant de pouvoir composer facilement des services, animer cette composition et suivre son évolution à travers la vérification de critères de cohérences que doit respecter le service composé d'un côté et les services le composant de l'autre.

Les patrons de coopération

Notre contribution au domaine des patrons et plus particulièrement des patrons de coopération n'est qu'une première étape dans ce domaine florissant. En particulier les composants de notre patron architectural doivent être confrontés et comparés aux patrons présentés dans [BDF+02] et [BDFR03]. En effet le niveau de granularité de ces derniers est plus fin (service discovery, service monitoring, ...) et ils peuvent par conséquent être manipulés individuellement ou en un nombre restreint et peuvent de ce fait être plus facilement applicables que notre patron architectural global.

Par ailleurs, l'applicabilité de notre patron dans le domaine des services étant faite, reste à apporter une autre vraie applicabilité de ce patron. Nous verrons dans le paragraphe suivant une piste à creuser en ce sens.

La coopération en entreprise

En parallèle avec ces différentes pistes de recherche que je qualifierais de fondamentales, il existe de nombreuses pistes de recherche appliquées. En effet si les modèles formels présentés et/ou en cours d'étude ont l'intérêt sous-jacent à tout modèle formel, c.a.d. la possibilité de les comparer les uns aux autres et de les valider par la théorie, il n'en reste pas moins que les travaux appliqués menés dans le cadre de la modélisation d'entreprise (réseau thématique UEML, GT ECI des GdR I3 et MACS, NoE INTEROP) nous ont amené à nous confronter à des problèmes dont la résolution est fortement contrainte par l'environnement industriel existant. Dans le cadre du projet thématique UEML dont l'objectif était la création d'un Langage Unifié de Modélisation d'Enterprise afin de faciliter l'interopérabilité entre systèmes de modélisation d'entreprises et, de là, l'interoperabilité entre entrepises, nous avons très vite été amené à définir, non pas, un langage unifié de modélisation d'entreprise, mais un ensemble de constructeurs de base du langage.

L'approche choisie dans le projet UEML a été de construire un meta modèle sur la base de trois Langages de Modélisation d'Entreprise (LME ou EML (pour Enterprise Modelling Languages)), à savoir IEM[MJ99], EEML[eem98] et GRAI[DVC98]. Un scénario de travail a été défini pour permettre de représenter la même situation en utilisant différents modèles (et les

outils associés à chacun de ces modèles). Tout d'abord, ce scénario a été représenté en IEM, puis des représentations "sémantiquement équivalentes" ont été faites en GRAI et EEML.

Ensuite chaque représentation (ou modèle) du scénario a été utilisée pour obtenir une méta modélisation décrivant chacun des modèles. On a donc obtenu des méta modélisations explicitant les contructeurs de base de IEM, EEML et GRAI. En parallèle, l'utilisation de chaque concept des méta modèles a été explicité au niveau du scénario de travail afin d'illustrer comment un même concept du monde réel est modélisé différemment par les trois différents modèles utilisés.

Ayant comme objectif d'avoir un méta modèle commun pour les constructeurs de base des différents modèles, nous avons comparé et unifié les trois méta modèles en utilisant une approche incrémentale. Nous avons comparé deux à deux les trois meta modèles pour établir les correspondances entre un concept d'un meta modèle et le concept (identique, similaire, englobant, inclus,...) d'un autre méta modèle.

Une fois les correspondances (et l'absence de correspondance) deux à deux établies, un ensemble de concepts communs a été identifé et et a donné lieu à la première version du méta modèle d'UEML.

Ce meta modèle représente les concepts communs sous-jacents aux trois Langages de Modélisation d'Entreprise initiaux et a été obtenu en essayant de se situer à un meilleur niveau d'abstraction que celui utilisé pour les méta modèles originaux et en "corrigeant" les anomalies détectées dans ceux-ci. Ce méta modèle a ensuite été validé en explicitant toutes les correspondances entre celui-ci et les méta modèles originaux et celles entre celui-ci et le sous ensemble du scénario utilisant les constructeurs de base méta modélisés.

Une synthèse des activités de recherche menées dans le cadre de ce projet a été présentée dans [38].

Ce travail pourrait continuer,

- soit en étoffant le nombre de constructeurs de base d'UEML avec comme objectif d'obtenir un LME "complet",
- soit en étoffant le nombre de constructeurs de base d'UEML avec comme objectif d'obtenir, non plus un LME "complet", mais les éléments nécessaires pour le "mappage" du patron de coopération présenté dans le chapitre précédent

Dans la mesure ou il serait illusoire de penser arriver à un LME complet, meilleur que ceux existant et faisant abstraction des outils et méthodes développés pour les LMEs actuels, la seconde alternative nous paraît la plus judicieuse et nous permettrait, par ailleurs, de valider notre patron par la la modélisation de cas concrets de coopération inter-entreprise.

Ces différentes pistes de recherche sont quelques unes des possibilités offertes pour une continuation dans la même optique que celle suivie jusqu'à présent, à savoir un travail de recherche effectué en allers-retours constants entre nos modèles validés théoriquement (et par des publications) et une expérimentation, que cette experimentation se fasse sur notre plate-forme d'expérimentation ou bien sur des applications "concrètes" issues du monde de l'entreprise.

Bibliographie personnelle

[1] C. Godart, K. Benali et J.C. Derniame. Propositions pour un système de gestion d'objets. In *Actes CGL3 (3ème Colloque-Exposition de Génie Logiciel)*, Versailles, Mai 1986.

[2] K. Benali, J.C. Derniame et C. Godart. Système d'information et génie logiciel. In *Actes Congrès INFORSID'87*, Lyon, Juin 1987.

[3] C. Godart, K. Benali, N. Boudjlida, F. Charoy et J.C. Derniame. Les bases de données sur le chemin du génie logiciel. In *Actes Journées d'Etude AFCET, Sophia-Antipolis.* **Serge Miranda, éditeur, Des bases de données aux bases de connaissances, ediTESTS, Paris, 1987** .

[4] N. Boudjlida, C. Godart, J.C. Derniame, K. Benali et O. Gervaise. Vers des ateliers de logiciel supportant des méthodes. In *Actes Conférence JISI'88 (Journées Internationales des Sciences de l'Informatique)*, Tunis, Tunisie, Avril 1988.

[5] J.C. Derniame, H. A. Bahsoun, K. Benali, N. Boudjlida et C. Godart. Towards Assisted Software Processes. In *Proceedings CASE'88 (International Workshop on Computer Aided Software Engineering)*, Cambridge (Massachussetts), USA, July 1988.

[6] K. Benali, N. Boudjlida, F. Charoy et J.C. Derniame. A Model for Assisted Software Processes. In *Proceedings ICCI'89 (International Conference on Computing and Information)*, Toronto, Canada, R. Janicki et W.W. Koczkodaj, éditeurs, Canadian Scholars Press Inc., Toronto, 1989

[7] **K. Benali. Assistance et pilotage dans le développement de logiciel - Vers un modèle de description. Thèse de Doctorat de l'université de Nancy 1, Novembre 1989.**

[8] K. Benali, N. Boudjlida, F. Charoy, J.C. Derniame, C. Godart, Ph. Griffiths, V. Gruhn, Ph. Jamart, A. Legait, D.E. Oldfied et F. Oquendo. Presentation of the ALF Project. In *Proceedings of the first International Conference on Software Development Environments and Factories (SDE&F1)*, Berlin, RFA, **N. Madhavji, W. Shafer and H. Weber, éditeurs, Pitman Publishing, Londres, Grande Bretagne, 1990** .

[9] K. Benali, J. Lonchamp, C. Godart, et J.C. Derniame. La modélisation des procédés de fabrication : une voie vers l'assistance intelligente en production de logiciel. In *Actes ERGO-IA'90 (Ergonomie et Informatique Avancée)*, Biarritz, Septembre 1990.

[10] K. Benali. The roles cooperating within a model driven IPSE. In *DRAFT No 1*, Chipot, éditeur, Metz, 1990.

[11] K. Benali. L'assistance et le pilotage en production de logiciel grace à la modélisation des procédés de fabrication. In *Actes DI'90 (Conférence des docteurs en Informatique)*, Dijon,Octobre 1990.

[12] J. Lonchamp, K. Benali, C. Godart et J.C. Derniame. Modeling and Enacting Software Processes : an Analysis. In *Proceedings IEEE COMPSAC'90 (International Computer Software & Applications Conference)*, Chicago, USA, Novembre 1990.

[13] J.C. Derniame, K. Benali, N. Boudjlida, C. Godart et J. Lonchamp. Roles cooperation

through software process instantiation. In *Proceedings ISPW-6 (6th International Software Process Workshop)*, Hakodate, Japon, IEEE computer society press, Los Alamitos, USA, 1991.

[14] K. Benali et J.C. Derniame. Assistance and Guidance in Software Production through Software Process Modeling. In *Proceedings SEKE'91 (International Conference on Software Engineering & Knowledge Engineering)*, Skokie, USA, Mai 1991.

[15] J. Lonchamp, K. Benali, J.C. Derniame et C. Godart. Towards assisted Software Engineering Environments. **In la revue : Information and Software Technology**, Vol 33, No 8, Butterworth Scientific Limited, Londres, Grande Bretagne, Octobre 1991.

[16] K. Benali et J.C. Derniame. Software Processes Modeling :What, Who, and When. In *Proceedings Second European Workshop Software Process Technology*, Trondheim, Norvège, **J.-C. Derniame, éditeur, Lecture Notes in Computer Science, 635, Springer Verlag, 1992**.

[17] K. Benali et J.C. Colson. Analyse, conception, codage, test : vers un outil unique, graphique et intelligent. In *Actes (Conférence internationale du génie logiciel et ses applications)*, Toulouse , Décembre 1992.

[18] K. Benali, J.C. Colson. Modélisation et assistance au développement en CAO. In *Actes (Conférence internationale du génie logiciel et ses applications)*, Paris, Novembre 1995.

[19] K. Benali, J.C. Colson et J. Lahyane Formalism modelling and visualizing : an experimentation. In Proceedings Conference on Visual Data Exploration and Analysis III (SPIE'96), San José (Californie, USA), Janvier 1996.

[20] J.-C. Bignon, G. Halin, D. Léonard, O. Malcurat, K. Benali et C. Godart. Evolution de la maîtrise d'œuvre, pratiques coopératives et informatique répartie, In Mieux produire ensemble, Nancy, France, avril 1998

[21] K. Benali, M. Munier et C. Godart Cooperation Models in Co-Design In Proceedings International Conference on Agile Manufacturing, Minneapolis, USA, Juin 1998.

[22] K. Benali, G. Canals, C. Godart et S. Tata. An Approach for Developing Cooperation in Project-Enterprises, In Proceedings 3th International Conference on the Design of Cooperative Systems, Cannes, France, mai 1998

[23] J.-C. Bignon, G. Halin, K. Benali et C. Godart. Cooperation models in co-design : application to architectural design, In Proceedings International Conference on Design and Decision Support Systems - ICD&DSS'98, Maastricht, Hollande, juillet 1998.

[24] K. Benali, M. Munier, et C. Godart. Cooperation models in co-design, **In la revue :International Journal of Agile Manufacturing - IJAM**, Vol 2 N° 2, Published by th International Society of Agile Manufacturing, 1999

[25] M. Munier, K. Benali, C. Godart. A transactional approach for cross-organizational cooperation, In Proceedings Globecom (Global telecommunications conference), Rio de Janeiro, Brésil, Décembre 1999

[26] M. Munier, K. Benali, C. Godart. DisCOO, a really distributed system for cooperation, **In la revue : Networking and Information Systems Journal**, Vol.2 N° 5-6, pp605-637, 1999, Hermes Science Publishing Limited, Oxford

[27] M. Munier, K. Baïna, K. Benali. A négociation model for CSCW , In Proceedings 5th International Conference on **Cooperative Information Systems, CoopIS'00** , Eilat, Israel, **Etzion/Scheuermann, editors, Lecture Notes in Computer Science, 1901, Springer Verlag, 2000**.

[28] M. Munier, K. Benali, C. Godart Un système coopératif basé sur les transactions , In Proceedings INFORSID'01, mai 2001, Martigny, Suisse

[29] K. Baïna, K. Benali, C. Godart A process service model for dynamic enterprise process

interconnection, In Proceedings 6th International Conference on **Cooperative Information Systems, CoopIS'01** , Trento, Italie **Giunchiglia/Batini, editors, Lecture Notes in Computer Science, Springer Verlag, 2001**.

[30] K. Baïna, K. Benali, C. Godart, Les services procédés, une solution pour l'interconnexion des procédés d'entreprises, **In la revue : Ingénierie des Systèmes d'Information**, Numéro spécial Interopérabilité des Systèmes d'Information (ISI'01), Vol. 6, Num. 3, pp 145-181, 2001, Hermès Science Publications, Paris

[31] K. Baïna, S. Tata, K. Benali, Un modèle d'interaction de services pour la coopération des procédés, In Proceedings 7ème Conférence Maghrébine des Sciences Informatiques (MCSEAI'02),Vol. 2 pp 189-201, mai 2002, Annaba Algérie.

[32] K. Baïna, S. Tata, K. Benali, A Model for Process Service Interaction, Short paper and poster In Proceedings On the Move to Meaningful Internet Systems 2002 : CoopIS, DOA, and ODBASE, Confederated International Conferences CoopIS, DOA, and ODBASE 2002, October 2002, Irvine, USA, **R. Meersman, Z. Tari, et al., editors, Lecture Notes in Computer Science, 2519, Springer Verlag, 2002**.

[33] K. Benali, G. Bourguin, B. David, A. Derycke, C. Ferraris Collaboration / Coopération, In Proceedings Assises du GdR I3, Décembre 2002, Nancy, France, **J. Le Maitre, éditeur, CEPADUES éditions, 2002**.

[34] K. Baïna, S. Tata, K. Benali, A Model for Process Service Interaction, In Proceedings Business Process Managemen (BPM 2003 June 2003, Eindhoven, The Netherlands, **Van Der Alst et al., editors, Lecture Notes in Computer Science, LNCS 2678, pp 261-275, Springer Verlag, 2003**.

[35] K. Baïna, K. Benali, C. Godart, Dynamic Interconnection of Enterprise Workflow Processes, In Proceedings Concurrent Engineering 2003, July 2003, Madeira, Portugal, **R. Jardim-Gonçalves et al., editors, A.A. Balkema Publishers Verlag, 2003**.

[36] K. Baïna, K. Benali, C. Godart, Dynamic Interconnection of Heterogeneous Workflow Processes Through Services, In Proceedings On the Move to Meaningful Internet Systems 2003 : CoopIS, DOA, and ODBASE, Confederated International Conferences CoopIS, DOA, and ODBASE 2003, November 2003, Catania, Italy **R. Meersman, Z. Tari, D. Schmidt et al., editors, Lecture Notes in Computer Science, LNCS 2888, pp 444-461, Springer Verlag, 2003**.

[37] Walid Gaaloul, Karim Baïna, Khalid Benali, Claude Godart, Un patron pour l'interconnexion de composants distribués, In Proceedings CoPSTIC'03, Conférence en Sciences et Techniques de l'Information et de la Communication, Décembre, 2003, Rabat, Maroc

[38] Hervé Panetto, Giuseppe Berio, Khalid Benali, Nacer Boudjlida, Michaël Petit, A Unified Enterprise Modelling Language For Enhanced Interoperability Of Enterprise Models, In Proceedings INCOM2004, 11th IFAC Symposium on Information Control Problems in Manufacturing, April, 2004, Salvador, Brasil

[39] Walid Gaaloul, Karim Baïna, Khalid Benali, Claude Godart, A Pattern for Interconnecting Distributed Components, Short paper and Poster In ICEIS'04, International Conference on Enterprise Information Systems, April 2004, Porto, Portugal

Bibliographie

[AA90] D. Agrawal and A. El. Abbadi. Transaction Management in Database Systems. In A. K. Elmagarmid, editor, *Database transaction models for advanced applications*. Morgan Kauffman, 1990.

[AAAM96] Gustavo Alonso, Divyakant Agrawal, Amr El Abbadi, and C. Mohan. Functionality and limitations of current workflow management systems. *IEEE Expert Journal*, 1996.

[AAAM97] 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.

[AE94] F. Ackerman and C. Eden. Issues in computer and non-computer supported GDSSs. *Decision Support Systems*, 12(336) :381–390, 1994.

[AM97] 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.

[Atr94] Atria Software Inc. ClearCase product summary. Technical report, Atria Software Inc., 24 Prime park Way, Natick, Massachusetts 01760, 1994.

[Aus62] J. L. Austin. *How to Do Things with Words*. Oxford University Press, Oxford, England, 1962.

[Baï99] Karim Baïna. *Formalisation et Réalisation d'un Service de Négociation en Ingénierie Concourante*. Dea informatique systèmes et communication, université joseph fourier à grenoble - ensimag, Université Joseph Fourier à Grenoble, 30 Juin 1999.

[Baï03] K. Baïna. *Un modèle orienté services procédés pour la coopération et l'interconnexion de procédés d'entreprises*. Thèse en informatique, Université Henri Poincaré - Nancy 1, Loria, 16 Mai 2003.

[BAB+97] R. Bentley, W. Appelt, U. Busbach, E. Hinrichs, D. Kerr, K. Sikkel, J. Trevor, and G. Woetzel. Basic Support for Cooperative Work on the World Wide Web. *International Journal of Human-Computer Studies : Special issue on Innovative Applications of the World Wide Web*, 46(6) :827–846, June 1997.

[Bar92a] N. Barghouti. *Concurrency Control in Rule-Based Software Development Environments*. PhD thesis, Columbia University, 1992. Technical Report CUCS-001-92.

[Bar92b] N. Barghouti. Supporting Cooperation in the MARVEL Process-Centered SDE. *Proceedings of the Fifth ACM SIGSOFT Symposium on Software Development Environments*, 17(5) :21–31, December 1992.

[BDF+02] B. Benatallah, M. Dumas, M.-C. Fauvet, F. A. Rabhi, and Quan Z. Sheng. Overview of Some Patterns for Architecting and Managing Composite Web Services. *ACM SIGecom Exchanges*, 3(3), 2002.

[BDFR03] B. Benatallah, M. Dumas, M.-C. Fauvet, and F. A. Rabhi. *Patterns and Skeletons for Parallel and Distributed Computing*, chapter Towards Patterns of Web Services Composition, pages 265–296. Springer-Verlag, 2003.

[BDG+94] A. Biliris, S. Dar, N.H. Gehani, H.V. Jagadish, and K. Ramamritham. ASSET : A system for supporting extended transaction. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 94)*, May 1994.

[BDS+93] 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.

[BE94] N. Belkhatir and J. Estublier. ADELE–TEMPO : An Environment to Support Process Modelling and Enaction. In A. Finkelstein, J. Kramer, and B. Nuseibeh, editors, *Software Process Modelling and Technology*. Research Study Press, 1994.

[Bee88] D. Beech. A Foundation for Evolution from Relational to Object Databases. In *Proceedings of the International Conference on Extending Database Technology*, pages 251–267, venise, March 1988.

[BH99] 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)*, pages 336–347, Edinburgh, Scotland, September 2-4, 1999. IEEE Computer Society Press.

[BHT97] R. Bentley, T. Horstmann, and J. Trevor. The World Wide Web as enabling technology for CSCW : The case of BSCW. In *Computer-Supported Cooperative Work : Special issue on CSCW and the Web*, volume 6. Academic Press, 1997.

[BK99] 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.

[BLR95] I. Benbasat, F.J. Lim, and V.S. Rao. A framework for communication support in group work with special reference to negotiation systems. *Group Decision and Negotiation*, 4(371) :135–158, 1995.

[BMB+00] B. Benatallah, B. Medjahed, A. Boughettaya, A. Elmagarmid, and J. Beard. Composing and Maintaining Web-based Virtual Enterprises. In *1st Workshop on Technologies for E-Services, In Cooperation with VLDB'2000 (TES'00)*, Cairo, Egypt, September 14-15, 2000.

[BMR+95] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture, Volume 1 : A System of Patterns*. Wiley, John and Sons, November 1995.

[BN97] P.A. Bernstein and E. Newcomer. *Principles of Transaction Processing*. Morgan Kaufmann, 1997.

[BS96] T. X. Bui and M. F. Shakun. Negotiation pocesses, evolutionary systems design, and NEGOTIATOR. *Group Decision and Negotiation*, 5(417) :339–353, 1996.

[BZ98] E. Bellucci and J. Zeleznikow. A comparative study of negotiation decision support systems. In *Proceedings of the 31st Hawaii International Conference on*

System Sciences (HICSS'98). the IEEE Computer Society. Copyright (c) 1998 Institute of Electrical and Electronics Engineers, Inc., 1998.

[CD00] 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.

[CF90] M. Cart and J. Ferrié. Integrating Concurrency Control into an Object-Oriented Database System. In *Proceedings of Advances in Database Technology - EDBT'90, LNCS-416*, pages 363–377, march 1990.

[CFR89] M. Cart, J. Ferrié, and H. Richy. Contrôle de l'exécution de transactions concurrentes. *Techniques et Sciences Informatiques*, (16) :225–240, March 1989.

[CIJS00] F. Casati, S. Ilnicki, L. J. Jin, and M. C. Shan. eFlow : an Open, Flexible, and Configurable Approach to Service Composition. In *2nd International Workshop on Advance Issues of E-Commerce and Web-Based Information Systems (WECWIS'00)*, pages 125–132, Milpitas, California, June 8-9, 2000.

[CK94] P.K. Chrysanthis and K.Ramamritham. Synthesis of Extended Transaction Models. *ACM Transactions on Database Systems*, 19(3) :451–491, september 1994.

[Coa97] Workflow Management Coalition. Terminology & Glossary. Technical Report WFMC-TC-1011, Issue 2.0, Workflow Management Coalition, June 1997.

[Com04] OASIS Committee. BTP 1.0, Business Transaction Protocol. May 2004.

[CW94] M. K. Chang and C. C. Woo. A speech-Act-Based Negotiation Protocol : Design, Implementation and Test Use. *ACM Transactions on Information Systems*, 12(4) :360–382, October 1994.

[Den92] P. Denning. Work is a closed loop process. In *American Scientist*, volume 80, pages 314–317, July-August 1992.

[DHB97] M. H. Dunham, A. Helal, and S. Balakrishnan. A mobile transaction model that capture both the data and the movement behavior. *ACM/ Baltzer Journal on special topics in mobile networks and applications*, 2(2), 1997.

[DST01] DSTC. *Breeze : workflow with ease*. DSTC (Distributed Systems Technology Centre), www.dstc.edu.au/Research/Projects/Pegamento/Breeze/breeze.html, March 30, 2001.

[DVC98] G. Doumeingts, B. Vallespir, and D. Chen. *Decision modelling GRAI grid, Chapter in Handbook on architecture for Information Systems, P. Bernus, K. Mertins, G. Schmidt (Eds)*. Springer-Verlag, 1998.

[eem98] *Action port model : A mixed paradigm conceptual workflow modeling language*. IEEE Computer Society, 1998.

[FDBP01] M.-C. Fauvet, M. Dumas, B. Benatallah, and H. Paik. Peer-To-Peer Traced Execution of Composite Services. In F. Casati, D. Georgakopoulos, and M-C. Shan, editors, *2nd Workshop on Technologies for E-Services, In Cooperation with VLDB'2001 (TES'01)*, number 2193 in LNCS, pages 103–117, Rome, Italy, September 14-15, 2001. Springer-Verlag.

[Fer95] J. Ferber. *Les Systèmes multi-agents : Vers une intelligence collective*. InterEditions, Paris, 1995.

[GAHL00] 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.

[GGM97] J-M. Geib, C. Gransart, and P. Merle. *CORBA : des concepts à la pratique*. Editions MASSON, Paris, France, 1997.

[GHF96] D. Georgakopoulos, M.F. Hornick, and F.Manola. Customizing transaction models and mechanisms in a programmable environment supporting reliable workflow automation. *IEEE Transactions on Knowledge and Data Engineering*, 8(4) :630–649, August 1996.

[GHJV94] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns, Elements of Reusable Object-Oriented Software*. Addition-Wesley, Massachusetts, 1994.

[GHM+93] 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.

[GK97] T. F. Gordon and N. Karacapilidis. The Zeno Argumentation Framework. In *Proc. Int. Conf. On AI and Law (ICAIL)*, Melbourne, 1997.

[GMS87] H. Garcia-Molina and K. Salem. Sagas. In *Proceedings of the 12th Annual ACM Conference on the Management of Data*, pages 249–259, San Francisco, California, May 1987.

[GPS99] C. Godart, O. Perrin, and H. Skaf. COO : a workflow operator to improve cooperative modelling in virtual processes. In *9th International Workshop on Research Issues on Data Engineering : Information Technology For Virtual Enterprises (RIDE-VE'99), Sponsored by the IEEE Computer Society*, Sydney, Australia, March 23-24, 1999.

[GR93] J. Gray and A. Reuter. *Transaction Processing : Concepts and Techniques*. Morgan Kaufmann, 1993.

[GSCB99] 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*, 24(6), 1999.

[HA99] 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.

[Hit02] Hitachi. *WorkCoordinator Workflow System*. Hitachi Ltd., www.hitachi.co.jp/Prod/comp/soft1/wco/eng/, 2002.

[HP01] HP. *Architectural Specification, Release A.0, www.e-speak.net*. Hewlett Packard, www.e-speak.net, January 2001.

[HV99] M. Henning and S. Vinoski. *Advanced CORBA Programming with C++*. Addison-Wesley Professional Computing Series, 1999.

[IBM00] IBM. *Web Services architectrure overview*. www-106.ibm.com/developerworks/webservices/, September 2000.

[IET99] IETF. *Protocol-independent Content Negotiation Framework*. IETF (Internet Enegineering Task Force), www.ietf.org/rfc/rfc2703.txt, September 1999.

[IET00a] IETF. *Internet Printing Protocol - IPP/1.1 : Model and Semantics*. IETF (Internet Enegineering Task Force), www.ietf.org/rfc/rfc2911.txt, September 2000.

[IET00b] IETF. *Internet Printing Protocol (IPP) : Requirements for Job, Printer, and Device Administrative Operations.* IETF (Internet Enegineering Task Force), www.ietf.org/rfc/rfc3239.txt, September 2000.

[IET01] IETF. *Internet Printing Protocol (IPP) : The 'indp' Delivery Method for Event Notifications and Protocol/1.0, draft.* IETF (Internet Enegineering Task Force), www.ietf.org/internet-drafts/draft-ietf-ipp-indp-method-06.txt, July 17, 2001.

[JF89] M. T. Jelassi and A. Forroughi. Negotiation support systems : An overview of design issues and existing sorftware. *Decison Support Systems*, 5 :167–181, 1989.

[KK00] K. Ku and Y. Kim. Moflex transaction model for mobile heterogeneous multi-databse systems. *IEEE workshop on research issues in Data Engineering, Sand Diego USA*, February, 2000.

[Koo88] C. C. Koo. A commitment-based communication model for distributed office environments. In *proc. of Conference on Office Information System*, pages 291–298, New York, 1988. ACM Press.

[KPPL00] K. Kim, B. C. Paulson, C. J. Petrie, and V. R. Lesser. *Compensatory Negotiation for Agent-Based Project Schedule Coordination.* http ://www.stanford.edu/ ks-kim/wp55.pdf, January 2000.

[KSE98] S. Kraus, K. Sycara, and A. Evenchik. Reaching agreements through argumentation : a logical model and implementation. *Artificial Intelligence*, 104(1-2) :1–69, 1998.

[KTP97] N. I. Karacapilidis, B. Trousse, and D. Papadias. Using Case-Based Reasoning for Argumentation with Multiple Viewpoints. In *Case-Based Reasoning Research and Development, Proceedings of the 2nd Int. Conference on Case-Based Reasoning (ICCBR-97)*, number 1266 in Lecture Notes in AI, pages 541–552, Providence, Rhode, Island, July 25-27, 1997. Springer-Verlag.

[Kut98] L. Kutvonen. *Trading services in open distributed environments.* Thèse en informatique, Department of Computer Science, University of Helsinki, Finland, 1998.

[KWA99] 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.

[LB93] L. H. Lim and I. Benbasat. A theoretical perspective of negotiation support systems. *Journal of management Information Systems*, 9(3), 1993.

[Lev83] S. C. Levinson. *Pragmatics.* Cambridge University Press, New York, 1983.

[Ley01] F. Leymann. WSFL 1.0 (Web Services Flow Language). In *IBM Software Group*, May 2001.

[LH02] M. Lee and S. Helal. HiCoMo : High Commit Mobie Transactions. *Kluwer Academic Publishers, Distributed and Parallel Databases (DAPD)*, 11(1), 2002.

[LR00] F. Leymann and D. Roller. *Production Workflow, Concepts and Techniques.* Prentice-Hall, Inc., 2000.

[Mic00] Microsoft. *Microsoft BizTalk Server : BizTalk Framework 2.0 : Document and Message Specification.* www.biztalk.org, December 2000.

[Mic01] Microsoft. *Microsoft .NET.* www.microsoft.com/net/, 2001.

[MJ99] Kai Mertins and Roland Jochem. *Quality-Oriented Design of Business Processes.* Kluwer Academic Publishers, Boston/Dordrecht/London, 1999.

[MLRR89] T. W. Malone, K. R. Grant K. Lai, R. Rao, and D. A. Rosenblitt. The Information Lens : An Intelligent System for Information Sharing and Coordination. *Technical Support for Work Group Collaboration*, pages 65–88, 1989.

[MMWFF92] R. Medina-Mora, T. Winograd, R. Flores, and F. Flores. The ActionWorkflow approach to workflow management technology. In *Computer-Supported Coooperative Work (CSCW'92)*, Toronto, Canada, November 1992.

[Mol96] Pascal Molli. *Environnements de Développement Coopératifs.* Thèse en informatique, Université de Nancy I – Centre de Recherche en Informatique de Nancy, 1996.

[Mos81] J. Elliot Moss. *Nested Transactions : An Approach to Reliable Distributed Computing,.* PhD thesis, MIT, 1981.

[MS77] I. Morley and G. Stephensen. *The Social Psychology of Bargaining.* Allen and Unwin, London, 1977.

[Mun99] Manuel Munier. *Une architecture pour intégrer des composants de contrôle de la coopération dans un atelier distribué.* Thèse en informatique, Université Henri Poincaré - Nancy 1, Loria, Janvier 1999.

[NRZ92] M.H. Nodine, S. Ramaswamy, and S.B. Zdonik. A Cooperative Transaction Model for Design Databases. In A.K. Elmagarmid, editor, *Database transaction models for advanced applications.* Morgan Kauffman, 1992.

[OMG97] 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.

[OMG00] OMG. *Workflow Management Facility Convenience Document combining dtc/99-07-05 dtc/2000-02-03 (WF RTF 1.3 Report).* OMG (Object Management Group), www.omg.org, February 14, 2000.

[PSJ98] S. Parsons, C. Sierra, and N. R. Jennings. Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8(3) :261–292, 1998.

[RD97] 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.

[RG96] M.A. Rosenman and J.S. Gero. Modelling multiple views of design objects in a collaborative CAD environment. *Computer-Aided Design*, 28(3) :193–205, 1996.

[RN04] Heri Ramampiaro and Mads Nygård. CAGISTrans : Providing Adaptable Transactional Support for Cooperative Work - an Extended Treatment . *Information Technology and Management*, 5(1-2) :23–64, January-April 2004.

[RS95] 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.

[SA04] Patricia Serrano-Alavadero. *Transactions adaptables pour les environnements mobiles.* Thèse en informatique, Université Joseph Fourier - Grenoble, LSR-IMAG, Février 2004.

[SBMW96] W. Schulze, M. Böhm, and K. Meyer-Wegener. Services of Workflow Objects and Workflow Meta-Objects in OMG-compliant Environments. In D. Patel, J. Shtherland, and J. Miller, editors, *2nd Workshop on Business Object and Implementation (OOPSLA'96)*, pages 118–125, San José/California, USA, 1996. Springer-Verlag.

[Sea69] J. R. Searle. *Speech Acts : An Essay in the Philosophy of Language*. Cambridge University Press, New York, 1969.

[Sea75] J.R. Searle. A taxonomy of illocutionary acts. In K. Gunderson, editor, *Language, Mind and Knowledge*, volume 7 of *Minnesota Studies in the Philosophy of Science*, pages 344–369, Minneapolis, 1975. University of Minnesota Press.

[SHH00] S. Y. W. Su, C. Huang, and J. Hammer. A replicable web-based negotiation server for e-commerce. In *Thirty-third Hawaii International Conference on System Sciences (HICSS-33), IEEE*, Hawaii, 2000.

[Sin93] M. P. Singh. A Semantics for Speech Acts. *Annals of Mathematics and Artificial Intelligence*, 8(I-II) :47–71, 1993.

[SISM02] BEA Systems, Intalio, SAP, and Sun Microsystems. WSCI 1.0 (Web Service Choreography Interface. In *W3C Note 8*, August 2002.

[Spe82] P. Sperber. *Fail-Safe Business Negotiating : Strategies and Tactics for Success*. Prentice Hall, December 1982.

[SSA00] H. Schuldt, H.J. Schek, and G. Alonso. Renew - the reference net workshop tool demonstrations, 21st international. In *Proceedings of the 21st International Conf. On Application and Theory of Petri Nets (ICATPN'00)*, Aarhus, Denmark, June 28-30 2000.

[Sun01] Sun. *Jini Network Technology*. Sun microsystems, www.sun.com/jini/, December 2001.

[Syc89] K. Sycara. Multiagent compromise via negotiation. *In Distributed Artificial Intelligence*, 2 :119–137, 1989.

[Tha01] Satish Thatte. XLANG, Web Services for Business Process Design. In *Microsoft Corporation*, 2001.

[TIRL02] F. Tartanoglu, V. Issarny, A. Romanovsky, and N. Levy. Dependability in the Web Service Architecture, . In *ICSE 2002 Workshop on Architecting Dependable Systems*, 2002.

[TW89] Tichy and F. Walter. RCS — A system for version control. *Software — Practice and Experience*, 15(7) :637–654, July 1989.

[UDD00] UDDI.Org. *Universal Description, Discovery and Integration (UDDI) Technical White Paper*. www.uddi.org, September 2000.

[UO00] UNCEFACT and OASIS. *ebXML Technical Architecture Specification*. www.ebXML.org, October 17, 2000.

[vdABEW00] 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)*, number 1901 in LNCS, pages 198–209, Eilat, Israel, September 6-8, 2000.

[vdABtHK00] W. M. P. van der Aalst, A. P. Barros, A. H. M. ter Hofstede, and B. Kiepuszewski. Advanced Workflow Patterns. In O. Etzion and Peter Scheuermann, editors, *5th IFCIS Int. Conf. on Cooperative Information Systems (CoopIS'00)*, number 1901 in LNCS, pages 18–29, Eilat, Israel, September 6-8, 2000.

[vdHHP01] 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)*, number 2172 in LNCS, pages 270–284, Trento, Italy, September 5-7, 2001.

[W3C98] W3C. *Extensible Markup Language (XML) version 1.0*. W3C (World Wide Web Consortium), www.w3.org/TR/1998/REC-xml-19980210, 1998.

[Wei84] W.E. Weihl. *Specification and Implementation of Atomic Data Types*. PhD thesis, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA, March 1984.

[Wei89] W.E. Weihl. Local Atomicity Properties : Modular Concurrency Control for Abstract Data Types. *ACM Transactions on Programming Languages and Systems*, 11(2) :249–282, April 1989.

[WFM96] WFMC. *Workflow Standard - Interoperability, Abstract Specification, WFMC-TC-1012, V. 1.0*. WFMC (Workflow Management Coalition), www.wfmc.org, October 20, 1996.

[WFM98] WFMC. *Workflow Management Application Programming Interface (Interface 2 and 3) Specification, WFMC-TC-1009, V. 2.0*. WFMC (Workflow Management Coalition), www.wfmc.org, July 1998.

[WFM00] WFMC. *Workflow Standard - Interoperability, Wf-XML Binding, WFMC-TC-1023, V. 1.0*. WFMC (Workflow Management Coalition), www.wfmc.org, May 1, 2000.

[Whi04] Stephen A. White. *Introduction to BPMN*. IBM Corporation, 2004.

[WR92] H. Wachter and A. Reuter. The ConTract Model. In A.K. Elmagarmid, editor, *Database Transaction Models for Advanced Applications*, chapter 7, pages 219–258. Morgan Kauffmann, 1992.

[ZBN01] L. Zeng, B. Benatallah, and A. H. H. Ngu. On Demand Business-to-Business Integration. In *9th Int. Conf. on Cooperative Information Systems, In Cooperation with VLDB'2001 (CoopIS'01)*, number 2172 in LNCS, pages 403–417, Trento, Italy, September 5-7, 2001.

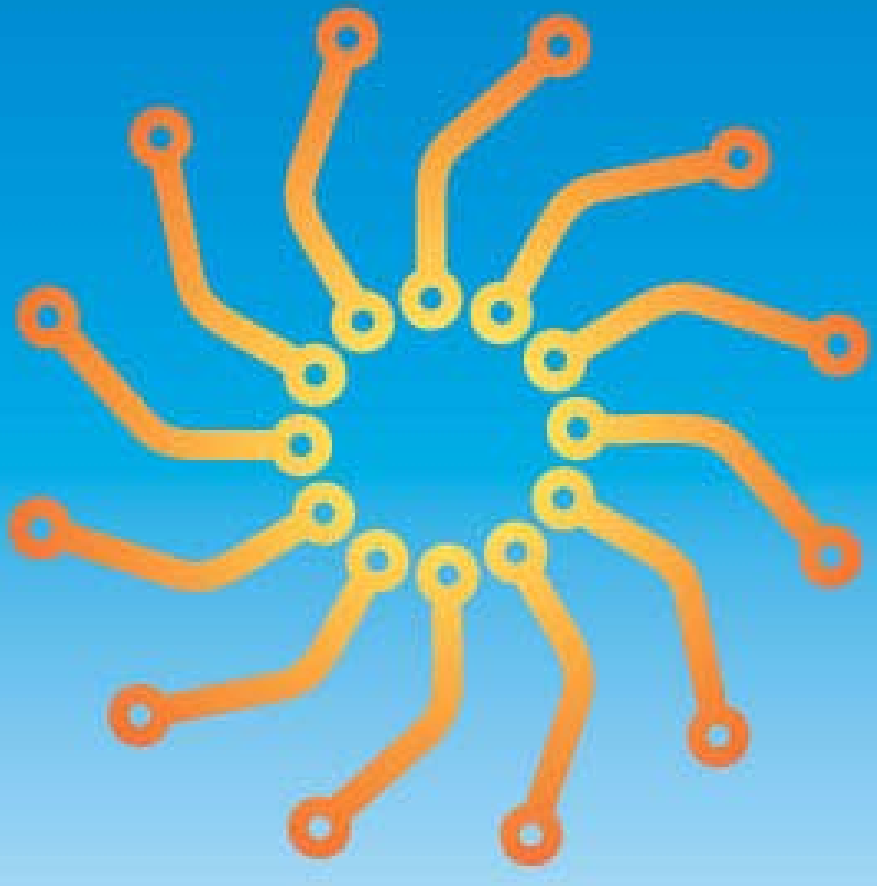
[Zel01] Ahmed Zellou. *Un modèle de tactiques et d'agents réactifs d'aide à la décision lors de la négociation de service*. DESA Informatique, Université Mohammed V de Rabat, Décembre 2001.

[ZS97] D. Zeng and K. Sycara. Benefits of Learning in Negotiation. In *Proceedings of AAAI-97*, 1997.

Troisième partie

Quelques publications récentes et significatives

INCOM 2004



11th IFAC Symposium on Information Control Problems in Manufacturing

Salvador da Bahia – Brazil, April 5 – 7 2004

A UNIFIED ENTERPRISE MODELLING LANGUAGE FOR ENHANCED INTEROPERABILITY OF ENTERPRISE MODELS

Hervé Panetto[1], Giuseppe Berio[2], Khalid Benali[3], Nacer Boudjlida[3], Michaël Petit[4]

[1]*CRAN UMR 7039, University Henri Poincaré Nancy I, F-54506 Vandoeuvre-les-Nancy Cedex*
Herve.Panetto@cran.uhp-nancy.fr
[2]*Dipartimento di Informatica, Università di Torino, Torino, Italy*
berio@di.unito.it
[3]*LORIA UMR 7503, F-54506 Vandoeuvre-les-Nancy Cedex*
{Khalid.Benali, Nacer.Boudjlida}@loria.fr}
[4]*Institut d'Informatique, Facultés Universitaires Notre-Dame de la Paix, Namur, Belgium*
mpe@info.fundp.ac.be

Abstract: There is a serious backwardness in awareness, acceptance and wide use of the Enterprise Modelling (EM) technology in industry because enterprises cannot capitalise from previous modelling efforts. This situation hinders true enterprise integration, interoperability, and enterprise knowledge sharing. A Unified Enterprise Modelling Language, based on meta-modelling of existing EM Languages, would serve as an Interlingua between EM tools providing the business community with a common visual, template based language to be used on top of most commercial enterprise modelling and workflow software tools.

Keywords: enterprise modelling, meta-modelling, interoperability, enterprise system

1. INTRODUCTION

Today's "business trends are clearly towards the need for managing organizational and operational changes within companies in order to face global competition and fluctuating market conditions" (Vernadat, 1996). This situation poses a number of integration (and interoperability) problems that enterprises have to tackle: integration of markets, integration between several development and manufacturing sites, integration between suppliers and manufacturers, integration of design and manufacturing, integration of multi-vendor hardware and software components. Enterprise Integration is therefore, not anymore only a problem of interconnecting physical and software applications but it also requires global business integration, aiming at the use of the existing or new enterprise resources in order to better achieve the overall business objectives. "Things to be integrated and coordinated need to be modelled. Thus, Enterprise Modelling (EM) is clearly a prerequisite for enterprise integration".

According to Vernadat (1996), *enterprise modelling* is the set of activities or processes used to develop the various parts of an enterprise model to address some desired modelling finality. It can also be defined as the art of "externalising" enterprise knowledge, i.e. representing the enterprise in terms of its organisation and operations (e.g. processes, behaviour, activities, information, object and material flows, resources and organisation units, and system infrastructure and architectures). The finality is to make explicit facts and knowledge that add value to the enterprises or can be shared by business applications and users. The prime goal of enterprise modelling is not only to be applied for better enterprise(s) integration but also to support analysis of an enterprise, and more specifically, to represent and understand how the enterprise works, to capitalize acquired knowledge and know-how for later reuse, to design (or redesign) a part of the enterprise, to analyse some aspects of the enterprise (by e.g. economic analysis, organization analysis, qualitative or quantitative analysis,...), to simulate

the behaviour of (some part of) the enterprise, to make better decisions about enterprise operations and organization, or to control, coordinate and monitor some parts of the enterprise.

Within the initiative on Computer Integrated Manufacturing (CIM), Enterprise Modelling was born in the United States at the beginning of the 80's and emerged through large CIM projects (e.g. ICAM or CAM-I). In the mid-80's, Europe launched several projects on Enterprise Modelling giving birth to several enterprise modelling languages (including notably GRAI (Doumeingts, *et al.*, 1998) and CIMOSA (AMICE, 1997; Kosanke, *et al.*, 1999)). As a result, in the 90's many commercial tools dealing with EM or business process modelling appeared on the marketplace, e.g. ARIS ToolSet, FirstSTEP, METIS, Enterprise Modeller, KBSI, CimTool, MO^{2}GO, e-MAGIM and many others.

Because most resources in modern enterprise were computerised systems, the problem of enterprise integration was also been approached with *workflow systems*, each one with its own modelling environment (Action Workflow, COSA, FlowMark, Lotus Notes, Teamware Flow, Ensemble, WorkParty, ...) and, in the late 90's, with EAI (Enterprise Applications Integration) tools. In computerised systems, the situation is currently worse than before. This is mainly due to the "Internet based systems" which, while offering very powerful, low cost and open infrastructures, fall short of integration. In fact, the many related standards (e.g., XML and its customisations) do not directly address the "problem of meaning".

This intensive production of tools has led to a *Tower of Babel situation* in which the many tools, while offering powerful and distinct functionalities, are unable **to interoperate** and can hardly or not at all communicate and exchange models. This is a serious drawback for awareness, acceptance and wide use of the EM technology since enterprises cannot capitalise from previous modelling efforts. This situation hinders true **enterprise integration, interoperability, and sharing enterprise knowledge**.

2. ENTERPRISE INTEGRATION AND ENTERPRISE MODELLING

Enterprise Modelling is an *engineering discipline* closely related to computerised systems. As such, it requires the combined use of Enterprise Modelling Software Environments (EMSE), Enterprise Modelling Languages (EML), and Enterprise Engineering Methodologies (EEM).
According to this point of view, there exists a lot of fragmented approaches to enterprise modelling (including Methodologies, Languages and Tools).
They cover different subsets of the different components of the enterprise engineering world. For instance, the ENV12204 (CEN, 1995) standard of CEN provides an Enterprise Modelling Language, but does not address any of the other components (no tool, methodology or meta-modelling capacity)); GERAM (GERAM, 1997) and ISO/IEC 15288 define methodological guidelines for Enterprise Engineering but without any modelling language; the Workflow Management Coalition (WfMC) provides a modelling language (WPDL) but without methodological support for using this language for modelling processes; the same happens with other EAI tools (such as the so called Integration Brokers).

They also cover different parts of the *enterprise life-cycle*. For instance, ebXML and WfMC focus on software design while approaches like CIMOSA mainly concentrate on enterprise requirements and design. Additionally, some approaches link enterprise models to Enterprise Operation Tools (EOT). These links allow the produced models to be used, for instance, for process *enactment and control* (e.g. in the WfMC approach, WPDL models are used by a workflow engine to control the execution of ongoing work). Models may be also linked to enterprise software applications like ERP (Enterprise Resource Planning) systems (e.g. ARIS and mySAP.com). Some other approaches only aim at modelling for understanding and analysing and they do not provide explicit links to operational systems or to other models in the life-cycle (e.g. ENV12204).

Enterprise modelling approaches may also have very different objectives and needs. As a meaningful example, we may compare the aims of IEM and GRAI. IEM allows representing business processes and it provides specific concepts adapted for assessing quality management procedures, but it cannot directly be employed for an operational implementation of the business processes. In fact, for quality management, it is not necessary to fully define an implemented process. The description has only to be sufficient to enable determining whether the process steps conform to defined quality procedures. This later objective requires, for example, representing outcomes of each process step and their usage as inputs for other process steps. On the other hand, the main objective of GRAI modelling is to define the control system of an enterprise. This requires a very good understanding of the relationships between the business processes at the various control levels (operational, tactical, and strategic). Quality procedures do not guarantee that an enterprise has good performance but only that its products conform to some quality criteria, whereas enterprise control tries guarantee that an enterprise takes into account market data and reflects them in the enterprise internal behaviours.

In enterprise modelling, there seems to be a tendency for approaches combining, in a more or less integrated way, *several sub-languages* or views (see e.g. CIMOSA, GRAI, and ARIS). A

combining approach allows taking advantage of the strengths of each of the sub-languages and offers the advantage that the resulting combined method or methodology offers the modeller the capacity to meet more modelling objectives. Models built with the distinct languages have to be related in some way and the *languages have to be integrated*. This need for integrating distinct modelling languages and relating models has also been recognised in domains like *software formal methods* for achieving effective and "practical" solutions to complex problems.

However, the integration of several sub-languages (often called *views*) is currently always performed within a single tool (i.e. in a single approach which creates many overlapping with other existing approaches). No tool (at least in the enterprise modelling domain) currently supports the combination of its own models with models created with a language supported by another tool.

A *completely integrated language* allowing the creation of models combining all needed aspects of the reality is probably unachievable and the supporting tools for that language would be too complex to build. Therefore, the only reasonable approach seems to be to create a modelling environment from "legacy" enterprise modelling tools (and languages) allowing to reuse existing models and to leverage these existing models and tools into an *integrated environment*. This integrated environment should also be complemented with a process for extending, in a controlled way, a set of limited constructs belonging to a *core language*. As we will see in the remainder (section 3.3), a sample of this process has been defined during the UEML project.

3. THE UEML

The UEML project[1] was set up in an attempt to contribute to the solving of the problems of multiple EMLs. The long term objective of UEML is the definition of a core language called Unified Enterprise Modelling Language, which would serve as an *Interlingua* between EM tools. This language will:

- Provide the business community with a common visual, template based language to be used on top of most commercial enterprise modelling and workflow software tools;
- Provide standardised mechanisms for sharing and exchanging enterprise models among projects, overcoming tool dependencies;
- Support the implementation of open and evolutionary enterprise model repositories to leverage enterprise knowledge engineering services and capabilities.

[1] UEML IST-TN 2001 34229, www.ueml.org

In order to prepare this long term objective, the UEML project was initiated with the objective to create and manage a working group aiming to:

- Create a European Consensus on a core set of modelling constructs and facilitating interoperability in the frame of on-going standardisation efforts in this domain.
- Build a demonstrator portal with services and contents to support and promote, testing, industrial validation, and to collect comments.

The first objective of the project was to analyse the market potential of a UEML, to accurately define the specifications of an embryo of such a language and to demonstrate and disseminate the concepts.

3.1 The need for UEML

Two main efforts related to the definition of a common core language for enterprise modelling are PSL (PSL, 2002) and ENV12204[2] that however do not currently provide a satisfactory answer to critical and also practical problems. PSL is a *logic-based approach* that does not clearly address the problem of the basic *mapping* between a generic EML and PSL itself. This mapping should e.g. state, for instance, that the concept of *Activity* belonging to an EML corresponds to the concept of *Activity* in PSL. Being a *declarative language*, PSL allows discovering *inconsistencies* between distinct models provided in distinct EMLs. However, it neither prevents nor *solves* these inconsistencies. ENV12204 is merely a set of useful concepts for understanding the *domain of enterprise modelling* (or even the set of things that need to be represented by any enterprise modelling language). However, its *syntax* is not well defined and therefore it cannot be used as an *exchange format* between distinct tools. It also does not define mappings between existing EMLs and itself.

The usefulness of a UEML would therefore reside in the availability of a *well-defined syntax* and *well-defined mappings* (possibly *standardised*) between various EMLs and UEML. We believe that the definition of mappings between languages and UEML is important but quite *independent* from the UEML definition itself. Thought, they should be precisely defined and shared (through, for instance, standardisation), they should base on *reasonable hypotheses* and will never be fully (and formally) provable.

Other approaches which attempt to solve the *problems of exchange and interoperability* between computerised systems do not deal clearly with the enterprise modelling area. They can be classified as ways for enabling *business level communication* between distinct *computer-based systems* and therefore as bottom-up approaches. For instance, ebXML, WPDL, EAI techniques are useful for defining communication at *business level* among

[2] A new version of ENV 12204, EN ISO 19440 is expected in early 2004.

enterprise software. These approaches are really useful for *programming software layers* such as *middleware* (e.g. CORBA) and *customising software architectures*. This description of purely software aspects can be considered as a kind of *high level programming* that anyway requires *enterprise modelling as a prerequisite*.

However, another prerequisite for the exchange of models (or to make models interoperable) through a common language to be meaningful is to clearly understand the *semantic links* existing between the models themselves. This understanding is fundamental because without it, an exchanged model could be understood in a totally different way by the receiving tool, and thus misinterpreted.

3.2 The need for meta-modelling

In order to understand the links between distinct languages, *meta-modelling* is an important issue. Meta-modelling allows defining the *syntax of a language*[3]. The product of meta-modelling is usually called a *meta-model*.

Meta-models need to be described by using *meta-modelling techniques* (i.e. languages for making meta-models). Approaches like XML (DTDs and Schemas), MOF, Telos, can be used. These techniques are *content-independent* (applicable for the definition of any language). Other meta-modelling techniques are *content-dependent (*and sometimes *domain-specific)*: for instance, XMI is an exchange format based on the meta-model of UML (UML, 2003) in XML designed for enabling exchange of UML models.

Accordingly, a *UEML could be defined as a content-dependent domain-specific meta-model* through a content-independent meta-model. The UEML might just use content-independent meta-modelling techniques[4] as a *way for its definition*.

Currently, several Meta-Modelling Languages (and also tools) exist but none of them are specifically targeted for the definition of EMLs. The reason is that these Meta-Modelling Languages were often developed to design and implement *Information Systems, Knowledge Base Systems and computer-based infrastructures* (environments) allowing to program meta-models.

3.3 The UEML approach

In the UEML project, a UEML meta-model was built on the basis of three existing languages (namely IEM, EEML and GRAI). An illustrative scenario was defined in which models of a common situation are stored in distinct software tools and exchanged. First, models of this scenario where elaborated in the three distinct languages and. the exchange was performed manually by specialists of these languages. More precisely, given a first model in IEM, specialists of GRAI and EEML provided the "semantically equivalent" model in their own languages. Afterward, IEM, EEML and GRAI constructs have been meta-modelled using UML class diagram. This resulted in three so-called "original meta-models". At the same time, the links between the concepts of every original meta-model and their use in the models of the Scenario were defined to illustrate how a unique real-world phenomenon is modelled with the three original meta-models.

With the aim to define a common meta-model for core constructs, we compared and "unified" the three meta-models through an incremental approach. We compared the three meta-models peer-to-peer to find any correspondence between a concept in one meta-model and a concept in another one.

Once the peer-to-peer correspondences (and absence of correspondences) had been defined, a set of common concepts were identified (Table 1) and further elaborated into the first version of the UEML meta-model 1.0 (Fig. 1). This meta-model represents the common concepts underlying the three original EMLs. This meta-model being remarkably different from the three meta-models by the use of an appropriate higher level of abstraction and considering some discrepancies among the three original meta-models, we informally re-defined new correspondences between the UEML meta-model and each original meta-model. Finally, the UEML meta-model and the new correspondences were validated against a subset of the Scenario.

3.4 Defining mappings among EMLs

The clear definition of the meta-models of existing EMLs and of UEML with meta-modelling techniques is necessary but not sufficient to achieve a meaningful exchange of models. The correspondences among constructs between two distinct languages have to be precisely defined by comparing semantics of these constructs. However, this is a difficult task because:

- EMLs are often based on informal semantics, i.e. some natural descriptions of constructs meaning.
- The way in which EMLs are used in specific context and situations may change.

Therefore, as suggested in Sect. 3.1, mappings between languages should rely on reasonable hypotheses should be clearly stated and become the base for building the language, and possibly be standardised further.

---

[3] A meta-model may also be used to define part or all of the semantics of the language. But this is often not recommended.

[4] It should be noted that the notion of meta-modelling technique is *relative*. In fact, it is often true that a language can be used as a meta-modelling technique for another (sometimes, the same) language.

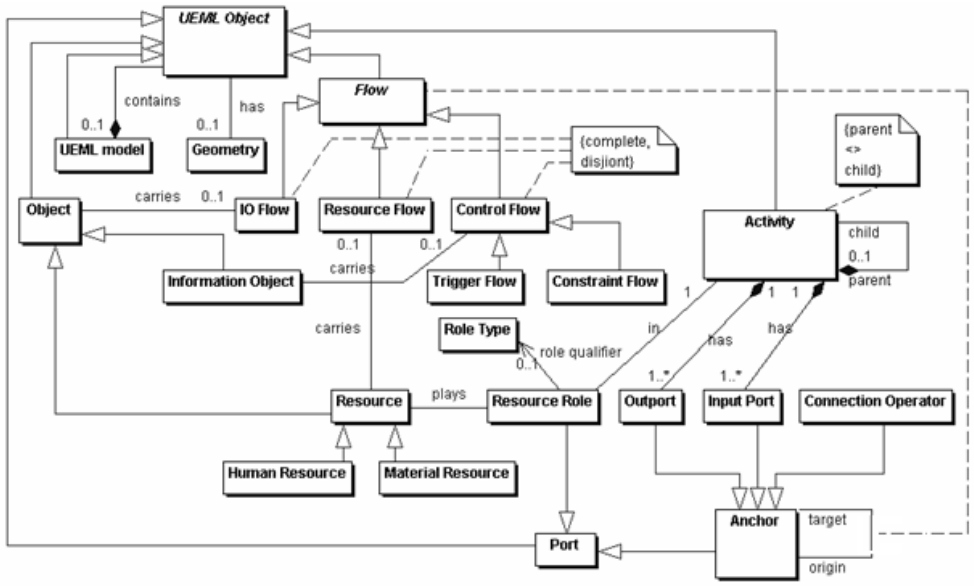


Fig. 1: The UEML 1.0 meta-model

Table 1: Extract of the common concepts of UEML

COMMON CONCEPT	GRAI	IEM	EEML
ACTIVITY	Extended activity	Action state	Task
ROLE	Not explicit	IEM Object state	Role
RESOURCE	Resource	Resource class	Resource
INPUT/OUTPUT FLOW	Input/Flow Ouput/Flow	Successor/ProcessElement	Flow (control flow= false)
CONSTRAINT FLOW	Control/Flow (trigger=false)	No direct	Flow (control flow= false and linked to Role)
CONTROL FLOW	Control/Flow (trigger=true)	ControlSuccessor/ProcessElement	Flow (control flow= true)
RESOURCE FLOW	Resource/Flow (trigger=false)	ResourceSuccessor/Resource State	Role (linked to Task)
CONNECTION OPERATOR	Logical operator	Connection Element State	DecisionPoint (not (Inport or Outport))
PORT	Connector	Port	Decisionpoint (Inport or Outport)

Mappings can be defined, more or less precisely, in various languages. For example, they can be expressed informally in natural language or through the use of a meta-modelling language. From a technical viewpoint, XSLT is a proposal to define transformation of XML documents based on a set of transformation rules expressed on the basis of XML schemas. The advantage of this approach is that software tools are already operational to interpret these mappings and apply them on XML documents. This approach could be considered as a means of implementing correspondences among EMLs, provided that these have XML syntax. Defining relationships at the language level can also be done in an "a priori" manner when new methodologies and methods are under definition. Therefore, a UEML can be a good starting base for placing under control the process of defining new methods and methodologies as well as the rules applied in a specific methodology.

4. CONCLUSION AND OUTLOOK

4.1 Future perspective for enterprise modelling

This analysis of the state of the art in enterprise modelling (Petit, *and al.,* 2002) demonstrates the need to define and develop a UEML approach to solve the current problems faced by workers in the enterprise modelling domain.

But such UEML approach can only be successful and effective under two conditions:

- Providing a global approach of interoperability among enterprise modelling software going further than only providing a common format of exchange;
- Making clear and effective the link between enterprise modelling and Enterprise Applications and Software.

4.2 UEML as a global approach to enterprise modelling

As stated earlier, a common exchange format, if deemed successful, cannot be described independently of mappings to and from existing EMLs. Furthermore, this requires the explicit definition of meta-models of the involved languages and of the mappings among their concepts. However, in order to avoid that UEML, as a common format, becomes yet another language among the large set of existing ones, it requires a larger view of interoperability among EM tools. The UEML language and approach must be flexible to be able to cope with future proprietary emerging languages and with the evolution of existing EMLs. The long term objectives of a

UEML approach would then be to provide the necessary concepts and tools to achieve the following:

- *Interoperability* between already existing supporting *tools as well as newly developed tools*,
- Well-founded integration basis between distinct *enterprise modelling languages*,
- Consistent global models on which also distinct *methodologies* can be integrated,
- Improvement of *existing methodologies* and definition of *new methodologies*.

These objectives pose a number of requirements on the UEML approach:

- The availability of meta-modelling concepts, methods and tools to properly define EMLs (existing ones, new emerging ones, UEML, their extensions and particularisations for specific purposes or applications);
- The availability of concepts, methods and tools to properly define relationships among distinct EMLs and a UEML together with relationships between models created with different EMLs and UEML;
- The concepts and tools to properly define methodologies associated with EMLs, including UEML;
- The specification of an *open architecture* in which all these elements can be implemented to provide an evolutionary multi-language platform for enterprise modelling centred on UEML.

This platform would allow creating *coherent, global and logically centralised* (integrated) models of the enterprise but which may be distributed within different enterprise modelling applications at a *physical level*.

4.3 From enterprise modelling to enterprise systems

We consider that Enterprise Engineering or Enterprise Modelling make sense provided that we are able to link the tools developed in this field with the Enterprise Applications and Software. Enterprise Modelling must be considered as the way to design the *architecture* and the *model* of the enterprise independently from existing or future Enterprise Applications and Software. We see the various levels as shown in the Fig. 2.

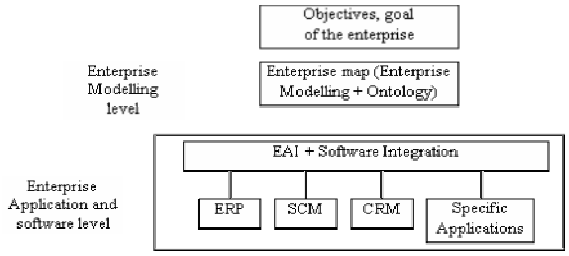


Fig. 2: Enterprise Modelling and Enterprise Application Levels

To our opinion, the future within ten years will be capabilities to generate, from the enterprise modelling level, the specifications allowing to choose or customize Enterprise Applications and Software (EAS) or to derive specifications of software applications for the enterprise models. Moreover, we believe that the EAS such as ERP, SCM or CRM will no longer have a centralised structure. Rather, they will have a modular and distributed structure. Each module will be chosen and customized according to specifications generated from the enterprise models. On the one hand, such a structure might *decrease* the time and cost of EAS development, and *increase* the adequacy of EAS to the needs of the enterprise. This should result in an increased acceptance of EAS by end-users, higher *adaptability* of EAS to the enterprise structure and improved *Return on Investment* of EAS development.

REFERENCES

AMICE. (1993). CIMOSA: Open System Architecture for CIM, *Research Reports ESPRIT*, **1**, Project 688/5288. Springer-Verlag.

Bernus P., Mertins K., and Schmidt G. (Eds) (1998). *Handbook on Architectures of Information Systems*. Springer Verlag, ISBN 3-540 64 453 9.

CEN (1995), CEN/CENELEC ENV 12204. *Advanced Manufacturing Technology, Systems Architecture, Constructs for Enterprise Modelling*, CEN, Brussels.

Doumeingts G., Vallespir B. and Chen D. (1998). Decision modelling GRAI grid, Chapter in: P. Bernus, K. Mertins, G. Schmidt (Eds.) *Handbook on architecture for Information Systems*, Springer-Verlag.

GERAM (1997). Generalized Enterprise Reference Architecture and Methodology Version 1.5, *IFAC-IFIP Task Force on Enterprise Integration*.

Kosanke K., Vernadat F.B., and Zelm M. (Eds.) (1999). CIMOSA: CIM open systems architecture - evolution and application in enterprise engineering and integration. *Computers in Industry, special issue*, **40(2-3)**.

Petit M, *and al.* (2002). *D1.1: State of the Art in Enterprise Modelling*, UEML TN IST 2001 34229, www.ueml.org

PSL (2002). Industrial automation system and integration - *Process specification language: Part 11: PSL-Core*. ISO/CD 18629-11.

UML (2002). *UML 1.5 specification*, OMG.

Vernadat F.B. (1996). *Enterprise modelling and integration: principles and applications*. Chapman & Hall.

ACKNOWLEDGEMENT

The authors would like to thank all the UEML core members for their scientific contribution to this work. This work was funded by the European Commission IST 5th framework programme.

Robert Meersman
Zahir Tari
Douglas C. Schmidt et al. (Eds.)

LNCS 2888

On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE

OTM Confederated International Conferences
CoopIS, DOA, and ODBASE 2003
Catania, Sicily, Italy, November 2003, Proceedings

DOA

Springer

Dynamic Interconnection of Heterogeneous Workflow Processes through Services

Karim Baïna[1,2], Khalid Benali[1], and Claude Godart[1]

[1] LORIA – INRIA – CNRS (UMR 7503)
BP 239, F-54506 Vandœuvre-lès-Nancy Cedex, France
{benali,godart}@loria.fr
[2] School of Computer Science and Engineering,
The University of New South Wales, Sydney NSW 2052, Australia
kbaina@cse.unsw.edu.au

Abstract. Process interconnection mechanisms are necessary to co-ordinate geographically distributed business processes in order to strength awareness inside virtual enterprises, to facilitate multinational e-transactions, etc. Actually, existing business process modelling and enactment systems (workflow systems, project management tools, shared agendas, to do lists, etc.) have been mainly developed to suit enterprise internal needs. Thus, most of these systems are not adapted to inter-enterprise co-operation. As we are interested in workflow processes, we aim, through this paper, to provide a model supporting dynamic heterogeneous workflow process interconnection. We consider the interconnection of enterprise workflow processes as the management of a "*workflow of workflows*" in which several heterogeneous workflow systems coexist. This paper introduces our process interconnection model, its implementation, and its validation through an experimentation.

Keywords: Multi-workflow systems, business process mediators and wrappers, negotiation, matchmaking, and brokering, service based workflow integration, out-sourcing based workflow interconnection.

1 Introduction

Our aim is to provide a framework to support dynamic interconnection of enterprise workflow processes. By *interconnection of enterprise workflow processes*, we mean the management of a "*workflow of workflows*" in which several heterogeneous workflow management systems coexist. By *dynamics* of enterprise workflow process interconnection, we mean that process interconnection does not consider neither predetermined communication primitives, nor scheduled points of rendezvous. In other terms, an enterprise, aiming to interconnect its workflow process with another organisation workflow process has to discover and co-decide an interconnection contract at run-time. In fact, we have transformed the problem of interconnection of two workflow processes into the problem of dynamic out-sourcing between these processes. To be interconnected with other processes,

R. Meersman et al. (Eds.): CoopIS/DOA/ODBASE 2003, LNCS 2888, pp. 444–461, 2003.

a workflow process out-sources dynamically parts of it to the other workflow processes. This enables interactions resulting from workflow interconnection to be limited in the time (*i.e.* to the out-sourcing period) and then to be well managed and controlled. Our process service interconnection model contribution consists of enriching SOA (Service Oriented Approach) with new paradigms and applying it to resolve heterogeneous workflow process interconnection problem. We propose a generic model for workflow process interconnection problem and validate this model on heterogeneous workflow management systems. Our paper is structured as follows: After a short introduction, section 2 presents the process interconnection problematics and state of the art, section 3 formalises our process service interconnection model, section 4 presents an implementation of our model, and gives some hints on our system experimentation. Finally, a short conclusion ends this paper.

2 Process Interconnection

Due to business process automation development, process interconnection becomes an important matter. Although a wide spectrum of tools for process modelling and enactment exists (workflow systems, project management tools, shared agendas, to do lists, etc.), they have been developed to suit the intern needs of enterprises, and thus, are not adapted to inter-enterprise interconnection. Compared to other enterprise process systems, workflow processes are the most mature and operational. Meanwhile, they still have many drawbacks when considering enterprise process interconnection. In spite of WFMS normalisation efforts achieved by the W*f*MC (*Workflow Management Coalition*), the OMG (*Object Management Group*) and the IETF (*Internet Engineering Task Force*), existing workflow management systems are: (1) *heterogeneous*: considering their definition and execution environments (disparate syntax and semantics of business process models and definition languages -BPDL-, ad-hoc process instance management), and their access means (non standard compliant API) ; (2) and *monolithic*: considering the absence or the poorness of their API, and the black box process instance encapsulation. Beside these problems related to workflow process entities, interconnection of these workflow processes implies several difficulties, among which, we can mention: (1) *process presentation* (how to present in a homogeneous manner workflow processes that have heterogeneous definition models ?), (2) *dynamic process interconnection* (which model to use for composing processes at run-time ?), and (3) *composed process enactment* (how to be able to execute a process interconnecting workflow processes that have heterogeneous execution models ?).

Because of heterogeneous and monolithic aspects of workflow management systems, developing generic models for enterprise workflow process interconnection is a big deal. Among several approaches for interconnecting enterprise processes we highlight the most important: (1) *Process message oriented communication* ([1], [2], and BizTalk describe several techniques for workflow process communication through asynchronous typed message passing, and adapt paradigms

like subscribe-notify, push, pull to workflow processes ;) (2) *Process event synchronisation* ([3], ICN [4], OPERA [5], WfMC [6], and WF-nets [7] upgrade process message communication paradigms with event coordination languages and algebras for synchronising interleaving workflow processes ;) (3) *Process data and interface interoperability* (Wf-XML, PIP, and e-speak establish interoperability frameworks for workflow data structures and interfaces ;) (4) *Process data concurrency and access control* ([3], [8], and IETF WebDAV & SWAP go beyond simple data interoperability to control access within shared workflow dataspaces ;) (5) *Process transactional exchange control* (COO [9], TRANSCOOP [10], WISE [11], and MQSeries [12] consider workflow processes as advanced transactions, and propose transactional models for workflow execution and data management ;) and (6) *Process service exchange* (Service concept has been defined in many research fields: Object Oriented research, Process Modelling research [1,13,14,15], Distributed System research [16,17], etc. In workflow research, CMI [1], OCoN [18], Crossflow [15,19], eFlow [14], define process service contracts for workflow process interconnection). To be more complete concerning process services, one may say that a process service can be seen as a software entity presenting process particularities and outcomes without totally revealing the process structure (*i.e.* its workflow implementation). A process service shows a functional abstraction of a process (or parts of a process) provided by an organisation. It specifies the amount of work that the organisation promises to carry out with a specific quality of service. It also specifies which parts of a workflow process it covers and how the requester could access to them. Proces service concept has been studied from several points of view: process service execution semantics abstraction [1], sub-workflow process service selection [19], dynamic process service activities configuration [14], process service control flow level abstraction [15], service methods and events wrapping [17], etc. Process service structure is to be seen as a co-operation pattern that relevantly supports dynamic workflow process interconnection and cooperation behaviours.

Compared to other approaches, *process service exchange approach* supports enterprise cooperation modelling in a very effective way. Actually, by it forces of abstracting enterprise workflow processes to be interconnected, process services are the *most adapted to build high level models for enterprise cooperation and generic models independent of workflow process particularities.* Moreover, process service exchange approach offers a high level paradigm which is very open to extensions dealing with other approaches basic paradigms (*e.g.* communication: message passing, data interoperability ; coordination: event synchronisation ; execution control: data access control, transaction management, etc.).

Hence, to build our dynamic enterprise workflow process model, we have chosen the process service exchange approach. For *process presentation problem*, we consider processes, beyond any process model, as services which are accessible object entities that possess object classification (*category*) and accept all object features (inheritance, overloading, etc.). Let call these specific services "process services". These process services possess application specific interface (*API*) and access rights to this API (*visibility contract*). A set of category specific typed values describe (*profiles*) of these process services. Our proposition for process

presentation is in the same vain of workflow object vision of the OMG [20] even if this later do not take into account workflow application specific properties. We propose innovative service discovery concepts and algorithms which can improve OMG Trading Service. As far as *dynamic process interconnection problem* is concerned, we merge the W*f*MC *nested sub-process model* [6] with process service out-sourcing based interconnection [1,14,15]. Our constraint was to interconnect workflow processes without changing neither their classical definition nor execution manner. So, we improved W*f*MC *nested sub-process model* which was created for build-time process interconnection with new dynamic paradigms (*discovery, negotiation, and wrapping*) that will enable processes to be composed at run-time. By the fact, our proposition improves classical "publish-find-bind" service oriented approach with symmetric aspects and negotiation. Our process service interconnection model will enable co-operating enterprises to structure, classify, and compare process services, to select dynamically a provided process service among those matching a required process service, and finally to keep possible their business processes co-operating through process service wrapping. This process service wrapping will realize the out-sourcing based interconnection of each workflow process interconnected couples. A process service may concern either long e-transactions (e.g. out-sourcing the development of pieces of software, subscription to full e-learning sessions, etc.), or short e-transactions (e.g. online book commands, enactment of administrative processes, data exchange rendezvous in a virtual enterprise, etc.).

To illustrate our approach, let us consider the following example within an e-learning context (figure 1). This example context considers five enterprises: KManager (a knowledge management and e-learning enterprise), WAgency (a web agency enterprise), CoolHost (a site hosting enterprise), MediAgency (a multimedia agency enterprise), and e-Store (an e-learning content collection enterprise). Each of these enterprises possess several workflow systems to manage their everyday business processes. Let KManager be a service requester enterprise. Let WAgency, CoolHost, MediAgency, and e-Store be four (among other) service provider enterprises. Let notice, that each enterprise can be both a requester and a provider of process services. On the one hand, KManager requires three types of services: a portal development service (portalDevSrv), a portal hosting service (portalHostSrv), and an e-learning content collection service (moduleCollectorSrv). Such requested services can be implemented by several provided services. On the other hand, WAgency proposes an Internet site development process service (webBoosterSrv), CoolHost proposes an Internet site hosting process service (hostEasySrv), and e-Store proposes an e-learning content construction process service (e-robotSrv).

To the indeterminism problem of interconnecting couples of provided and requested services, we proposed a symmetric publishing-discovery-negotiation service approach. In fact, after discovery and negotiation sessions of published process services, KManager chooses WAgency process service webBoosterSrv which matches portalDevSrv, CoolHost process service hostEasySrv which matches portalHostSrv, and e-Store process service e-robotSrv which matches moduleCollectorSrv. The interconnection is instance-based (i.e. each process service provider can interconnect instances of their process services to different requesters).

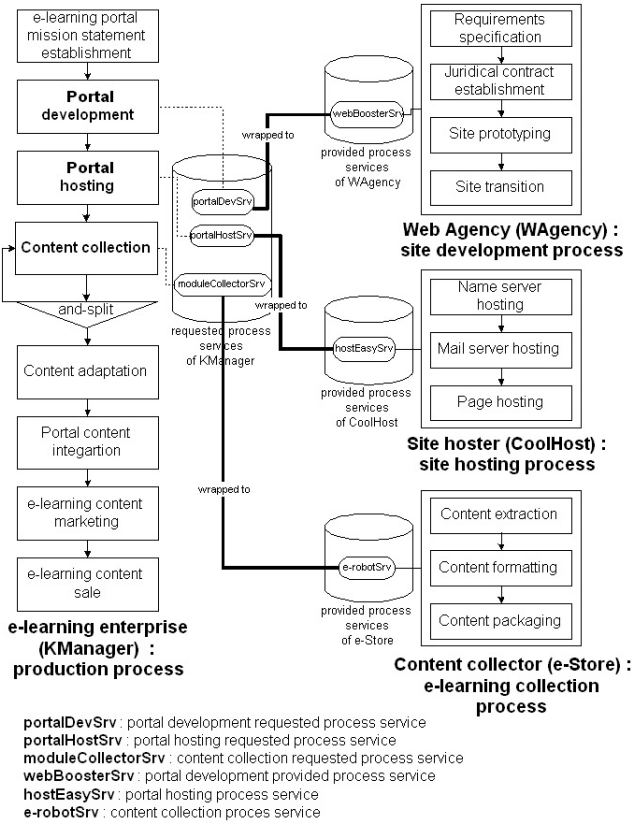


Fig. 1. An e-learning interconnection example

3 Our Process Service Interconnection Model

The modelling of process service interconnection is based on a metamodel describing our service oriented approach, on structures participating to our enterprise process service interconnection model and on facilities presenting the dynamics of our model and its operational aspects.

3.1 Process Service Approach: Meta Model

To tackle enterprise process interconnection problems, our approach considers three abstraction levels and thus is structured in three layers: workflow layer (implementation level), process layer (object abstraction level), and process service layer (presentation and access level).

Figure 2 shows our oriented process service approach presenting enterprise workflow processes, evolving inside monolithic and heterogeneous workflow management systems, as processes able to be interconnected through process services.

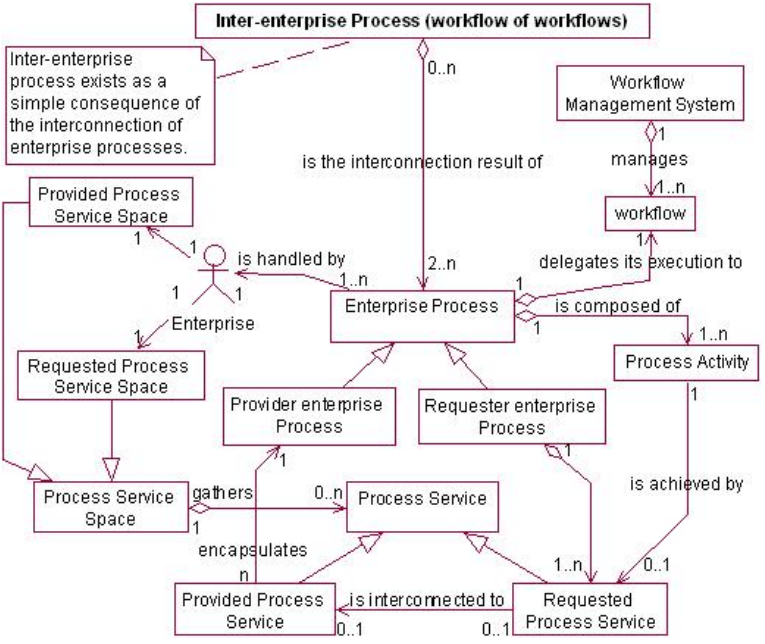


Fig. 2. Interconnection Approach: Meta Model

This interconnection yields an inter-enterprise process that represents "*a workflow of workflows*" whose management is distributed on all interconnected entreprises as follows:

- *processes:* Each enterprise possesses several processes. A process is composed of several process activities ;
- *workflows:* Each process delegates the execution of its activities to a workflow. The management of a workflow is specific to its WFMS engine (workflow management system) ;
- *process services:* An enterprise process activity can be achieved by the entreprise means or by an external service. We call a service achieving a process activity a process service. Each enterprise possesses a *requested process service space* and a *provided process service space*. A requested process service describes needs to accomplish a process activity. A provided process service encapsulates a process that represents an ability to achieve a process activity. Process interconnection is done through the wrapping of their requested and provided process services.

Figure 3 describes enterprise collaboration within our process service oriented approach. The operation <publish> represents the *publishing* (requesting and providing) of a process service, the operation <find> represents the *discovery* of a process service, while the operation <negotiate> represents a process service

parameters *negotiation*. Finally, the operation <bind> represents the dynamic *interconnection* between an abstract process service (a requested process service) and a concrete process service offer (a provided process service). Beside the fact that our approach supports negotiation facility (which is not ensured by classical service approaches), its strengthens these approaches with symmetric aspects of process service publishing, discovery, negotiation and interconnection.

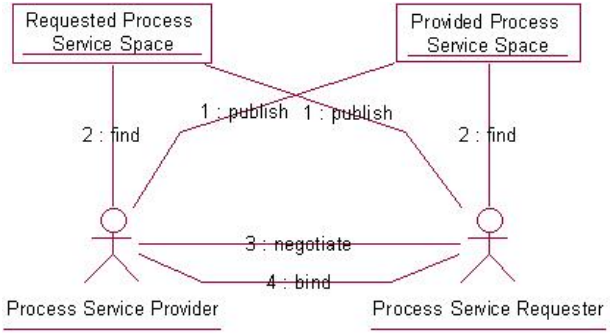


Fig. 3. Interconnection Approach: Collaborations

3.2 Process Service Interconnection Model: Structures

Process Structure. Our process interconnection model is initially based on the F. Leymann and D. Roller process definition model [12]. If this model can be applied very well for traditional workflows (within one enterprise), it does not consider explicitly process interconnection. Our objective is to enrich this model with new concepts and definitions in order to support some process interconnection aspects. Most of studied interconnection process models focus on process control flow and data flow definition without caring about two important process access points: process instance methods and process instance events.

The UML class diagram of figure 4 defines a process structure as follows:

- a process is defined by a process graph and a process interface ;
- a process graph describes the process control-flow structure [12]. It is the composition of nodes (process activities), edges (process activities transitions) and conditions (transition guards defined by business rules predicates);
- a process interface (or API) is the composition of methods (process instance reading and updating methods) and events (process event notification that are triggered off from a process instance during its execution). A process interface is specific to each process definition.

Figure 5 presents a process graph example and figure 6 a process interface example.

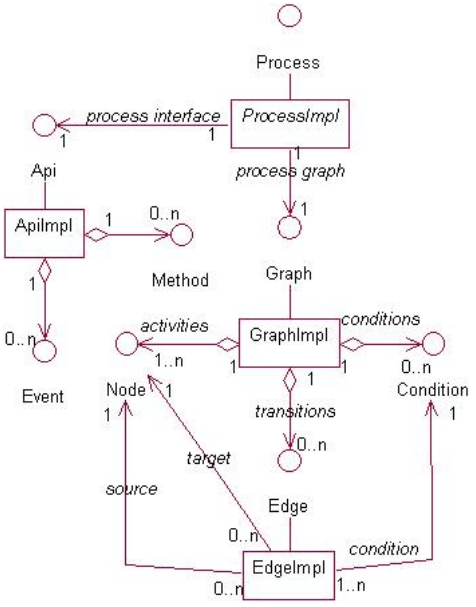


Fig. 4. Process Structure

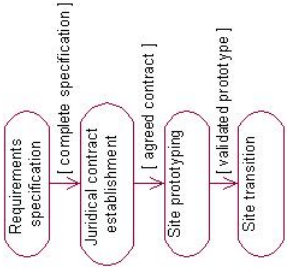


Fig. 5. WAgency Process Graph

Process Service Structure. The UML class diagram of the figure 7 defines the process service structure as follows:

- a process service is defined as a specific process wrapper that has a category, a profile and a visibility contract ;
- a process service category determines the process object type and its classification. Categories represent any enterprise agreed object ontologies ;
- a process service profile describes a relational structure defining a set of process named-typed-values attributes. Profile concept is in the same vein of OMG Trading Service properties ;

m_1	fetchProcessInstanceState(..)
m_2	fetchActivityInstanceState(..)
m_3	fetchWorkItem(..)
m_4	getProcessInstanceAttributeValue(..)
m_5	fetchWorkItemAttribute(..)
m_6	getWorkItem(..)
m_7	fetchActivityInstanceAttribute(..)
m_8	fetchActivityInstance(..)
m_9	getActivityInstance(..)
m_{10}	getWorkItemAttributeValue(..)
m_{11}	getActivityInstanceAttributeValue(..)
m_{12}	fetchProcessInstanceAttribute(..)

e_1	TerminatedProcessInstanceNotification
e_2	StartedProcessInstanceNotification
e_3	TerminatedActivityInstanceNotification
e_4	StartedActivityInstanceNotification
e_5	AvailableNewDataNotification

Fig. 6. WAgency process instance methods and events

– a process service visibility contract represents a subset of the wrapped process interface, "hiding" the complete process interface.

Figures 8 and 9 present a process service structure example.

Process Interconnection through Process Services. The W*f*MC has established a well known problem of *process interconnection by nested sub-process model*. The W*f*MC *nested sub-process model* expresses that an instance of an activity A_{i_j} belonging to a process P_i enacts remotely a known instance of an other process P_k and waits for its completion. Moreover, the W*f*MC has defined eight levels of process interoperability. The coexistence is the second lowest interoperability level (among these eight levels). Coexistent process interconnection are processes that do not possess any common interoperability standard. They meanwhile share, the same environment -machine or operating system or network- to be able to manage and achieve parts of the same process [6].

Process interconnection through process services aims to resolve the problem of *coexistent process interconnection through a dynamic variant of nested sub-process model*. That means that an instance of an activity A_{i_j} of a process P_i discovers dynamically a process P_k that suits its realisation profile, adapts it, wraps to it, instantiates it, enacts it dynamically, cooperates with it, and waits for its completion. Thus, a process service structure will be the glue keeping possible this dynamic coexistent process interconnection as shown in the UML class diagram of figure 10. Our complete approach is detailed in [21].

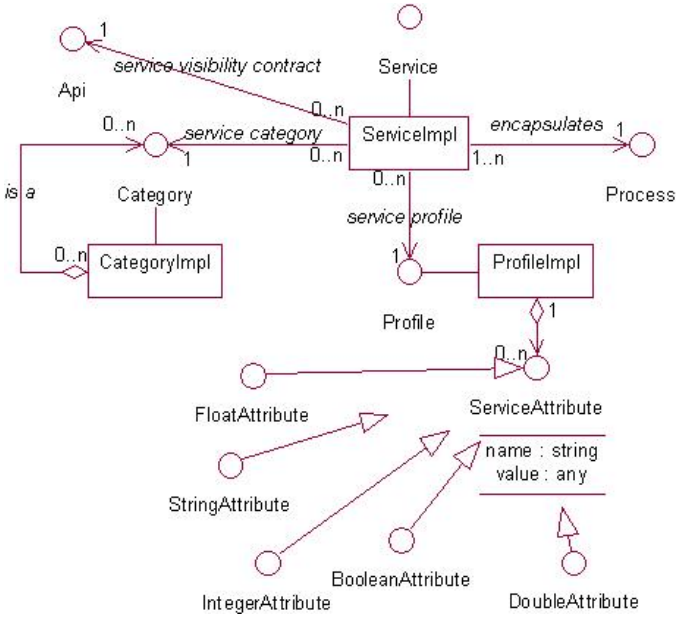


Fig. 7. Process Service Structure

m_1	fetchProcessInstanceState(..)
m_4	getProcessInstanceAttributeValue(..)
m_{11}	getActivityInstanceAttributeValue(..)
m_{12}	fetchProcessInstanceAttribute(..)

e_1	TerminatedProcessInstanceNotification
e_2	StartedProcessInstanceNotification
e_5	AvailableNewDataNotification

Fig. 8. WAgency.Provided.WebBoosterSrv methods visibility contract M_2 and events visibility contract EV_3

3.3 Process Service Interconnection Model: Dynamics

Process service structure is a wrapper that represents a functional and semantic abstraction of a process. It enables classification, indexing, comparison, and discovery of a certain type of process. This supposes that enterprises, within each business community, agreed about a common process service language (e.g. business key concept ontologies, business service taxonomies,...) to define and understand process services. Research and normalisation work are still emergent in this promising field. The dynamics of process service interconnection model will be presented through its publishing, discovering, negotiating and interconnecting process services facilities.

(*category* = e_learning_portal_development,
name = "WAgency.WebBoosterSrv",
profile = (duration=3 (month),
price=100 (Keuro),
dynamic_sites = true,
dbms = "MySQL",
XML_use = true,
Java_use = true,
JSP_use = true,
flash_use = false),
process = site_development_process)

Fig. 9. WAgency.Provided.WebBoosterSrv process service structure

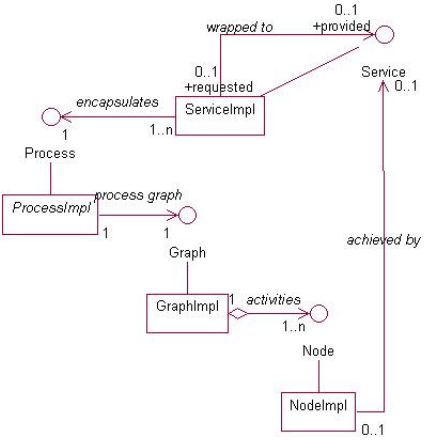


Fig. 10. Process Interconnection through Services

Process Service Spaces. The UML class diagram of figure 11 defines the process service spaces structure as follows:

- a process service space is a set composed of process services ;
- an enterprise possesses four types of process service spaces: (1) a private process service space (gathering all process services that the enterprise creates and keeps private to other enterprises before their publishing) ; (2) a requested process service space (public process service space gathering all process services that the enterprise requests (expressing the need of out-sourcing to an external enterprise). Each requested service knows its requester enterprise) ; (3) a provided process service space (public process service space gathering all process services the enterprise can achieve by her own means (expresses the capability to handle a requested process service of an external enterprise). Each provided service knows its provider) ; and (4) a wrapped process service space (public process service space gathering all process ser-

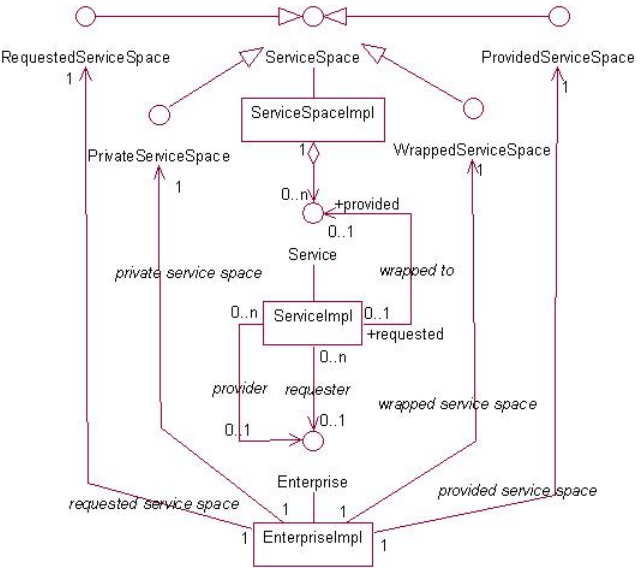


Fig. 11. Enterprise Process Service Spaces

vices that have been already wrapped -their requesting or providing need has been satisfied by an external process service-. Each wrapped service knows both its requester and its provider).

Process Service Publishing. Process service publishing deals with the communication of service description to other enterprises in order to find common agreement for enterprise process interconnections. Publishing of a process service is the result of the following steps:

1. The enterprise creates its process service within its private process space ;
2. the enterprise solicits its private process service space to publish (request of provide) its created process service ;
3. the private process service space constructs a clone of the process service and adds it to the requested process service space or to the provided process service space depending of the publishing type.

Actually, every subscribed enterprise possesses views on other enterprise requested and provided process service spaces. Every subscribed enterprise view is updated after each publishing event. These views enable enterprises to discover, to negotiate, and to be interconnected through other enterprise published process services. Figure 12 presents some process services among those defined and published by the five entreprises.

Process Service Discovery. Process service discovery deals with the application of algorithms that enable evaluation and comparison of process services

Enterprise	Process services (R = requested, P = provided)	
KManager	R	portalDevSrv (handles the developpement of a portal),
	R	portalHostSrv (handles a portal hosting),
	R	moduleCollectorSrv (collects an e-learning module).
e-Store	P	e-robotSrv (collects e-learning content)
MediAgency	P	mediArtSrv (achieves multimedia content)
WAgency	P	webBoosterSrv (achieves a web site)
CoolHost	P	hostEasySrv (handles the hosting of a web site)

Fig. 12. Some defined and published process services

in order to help process service requesters (or providers) to find the process services that match their requested (or provided) process services in the best way. Discovery of a process service (for instance a requested one) is the result of the following steps:

1. The enterprise creates and provides a process service we will name "provided" ;
2. let $n+1$ bet the current subscribed enterprise number,
   for ($set := \emptyset$, i := 1; i ≤ n; i++) loop
   a) the enterprise retrieves descriptions of process services published in the requested service space of the enterprise e_i :
   $services_i := \{rs_{i1}, rs_{i2}, \ldots\}$;
   b) $set := set \cup services_i$;
   end loop ;
3. the enterprise executes a neighbourhood algorithm on the process service provided among the set of possible requested process services *set*. This algorithm iterates on each element of *set* to find out the requested process services that suit the needs of the process service "provided". This neighbourhood algorithm is based on distances and matching measures that could be instantiated according to the application ;
4. The discovery finishes by building a set of requested process services $\{nrs_1, nrs_2, \ldots\}$ that are neighbour to the process service "provided".

Process service discovery is symmetric, it means that process service requesters can also discover provided process services that suit their needs. More details about matching and distance measures can be found in [22]. Actually, views on enterprise requested and provided process service spaces are organised using these matching measures. That enables enterprises to browse a restricted computed projection of the wide process service space.

Process Service Negotiation. Process service negotiation enables to decide dynamically and to adapt all interconnection parameters between processes to be interconnected (e.g. profile, visibility contract, process graph accessibility, etc.). Negotiation of a process service is the result of the following steps:

1. *solicitation phase:* the client contacts primarily the server and expresses its negotiation request (e.g. a new process service profile, a new process service visibility contract, etc.) ;
2. *effective negotiation phase:* both client and server exchange messages to build acceptable solutions set. An exchange protocol handles turn taking rules for expressing negotiation acts ;
3. *selection phase:* finally, the client or the server chooses a solution among those expressed during negotiation phase. The server takes its decisions according to the selected solution (e.g. allocating a solicited resource, tuning access rights for some service, etc.).

Actually, each subscribed enterprise is not only able to browse its views of other enterprise requested and provided process service spaces. It can also dynamically change these views by negotiating with process services publishers to customise their services for suiting its workflow process out-sourcing needs or offers. More details about generic negotiation component can be found in [23].

3.4 Process Service Interconnection Model: Protocol

Our process service interconnection protocol enables enterprise workflow processes to be interconnected and to cooperate through the following four steps:

1. *Workflow processes definition:* every workflow process is defined as a graph that manages process execution model within an enterprise chosen workflow management system.
2. *Workflow processes adaptation:* (a) *workflow process service provider adaptation:* each process service category is represented by an interface (ConcreteProcessInterface) that extends Process interface. A workflow process providing services of a certain category has to be associated to a program or a class that implements the category related interface. This adaptation has to be written in a language supported or inter-operable with the WFMS. (b) *workflow process service requester adaptation:* a workflow process requesting services has to express that its out-sourced activities (interface Node) are achieved by external process services. This adaptation has to program, in a language supported by the WFMS, the enactment of the requested process services and the cooperation protocol used to interact.
3. *Dynamic workflow processes interconnection by process services:* (a) *process service definition:* process services are created by requester and provider enterprises within their private process service spaces. Process services are defined by their name, textual description, category, profile, encapsulated process (if the process service is to be provided) and visibility contract that desires (if it is to be requested) or that permits (if it is to be provided). (b) *process service publishing:* process services are published by enterprises as provided or requested in their respective accessible requested or provided process service spaces. (c) *process service discovering:* process services publisher can look for process services suiting its process service definition needs.

(d) *process service negotiation:* process services publishers can negotiate with each others their process service requested and provided definitions (profile and visibility contract) to agree on common satisfying process service definition. (e) *process service wrapping:* process services wrapping deals with committing and dispatching, at run-time, agreed process service definitions (profile and visibility contract) on both process service views (requester and provider views).

4. *Workflow processes cooperation through process services:* workflow processes can cooperate through process services that adapt them. This cooperation between wrapped process services can vary from method invocation or event passing (according to agreed visibility contract), to data exchange or synchronisation on process execution states. Workflow processes cooperation through process services is to be considered as a generic paradigm that admits a wide panel of process cooperation modes.

4 Implementation and Experimentation

Our dynamic workflow process interconnection model has been implemented within our co-operative environment *DISCOBOLE* (DIStributed CO-operation and Business prOcess on LinE). *DISCOBOLE* integrates process and process service structures with their manipulation algorithms as innovative *CORBA application objects* within *process interconnection and cooperation facilities*. Through these facilities *DISCOBOLE* supplies mechanisms for process interconnection and cooperation applications. *DISCOBOLE* is implemented in Java on the CORBA broker architecture JacORB.

To experiment our enterprise workflow process interconnection model, we have deployed our e-learning enterprise context using *DISCOBOLE*. Each of the four enterprises KManager, e-Store, WAgency, and CoolHost uses a workflow management system to manage its business processes. *DISCOBOLE* is the environment that will enable them to interconnect dynamically these processes. In our experimentation, we selected and used three heterogeneous WFMS to model enterprise business processes: a lightweight component based WFMS *Breeze* [24], an object oriented PetriNets WFMS *Renew* [25], and a W*f*MC compliant WFMS *WorkCoordinator* (or WCO) [26]. These three WFMS are written in different languages and evolve in different environments. In order to keep possible the interconnection of their defined workflow processes, adaptation is achieved according to the WFMS supported language and environment (*Breeze* and *Renew* workflow processes adaptations have been programmed in Java on CORBA, while *WorkCoordinator* workflow processes adaptation has been programmed on a C++ CORBA bus). Thanks to this experimentation of our process service interconnection model, we effectively succeed to dynamically interconnect heterogeneous workflow processes. Moreover, another learned lesson from experimentation is that most of adaptation glue is redundant programming patterns that can be easily generated into WFMS supported languages to keep the enterprise focused on defining services and interconnecting processes through these services.

5 Conclusion and Perspectives

Our paper is a contribution in enterprise workflow interconnection domain which is a hot reseach topic as far as the current B2B (Business to Business) boom is concerned. In spite of normalisation efforts, WFMS are still presenting monolithic and heterogeneous drawbacks. Thanks to processes and process services frameworks, our model bypasses these drawbacks by enabling enterprise workflow processes interconnection within a "*workflow of workflows*" in which several workflow management systems coexist. Our model has been developed to support a wide panel of workflow management systems and experimented to prove the realisability of dynamic enterprise workflow processes. After dealing with process interconnection problems, we are tackling problems of *composed process service enactment and execution control* [27].

Acknowledgements. We would like to thank W. Gaaloul, S. Baïna, and A. Larhlimi for their development participation within the *DISCOBOLE* project, and Dr. B. Benatallah from UNSW, Sydney, Australia for this paper's review.

References

1. D. Baker, D. Georgakopoulos, H. Schuster, A. Cassandra, and A. Cichocki. Providing Customized Process and Situation Awarness in the Collaboration Management Infrastructure. In *4th IFCIS Int. Conf. on Cooperative Information Systems (CoopIS'99)*, pages 79–91, Edinburgh, Scotland, September 2–4, 1999. IEEE Computer Society Press.
2. 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.
3. G. Alonso, D. Agrawal, and A. El Abbadi. Process Synchronisation in Workflow Management Systems. In *8th IEEE Symposium on Parallel and Distributed Processing (SPDS'97)*, New Orleans, Louisiana, October 1996.
4. C. A. Ellis. *Computer Supported Cooperative Work*, chapter Workflow Technology. John Wiley and Sons, 1999.
5. 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.
6. L. Fischer, editor. *The Workflow Handbook 2001.* Published in association with the Workflow Management Coalition (WfMC), October 2000.
7. W. M. P. van der Aalst. Interorganizational workflows: An approach based on message sequence charts and Petri nets, System Analysis and Modeling, 34(3):335–367, 1999.
8. P. Dewan and H-H. Shen. Flexible meta access-control for collaborative applications. In *Proceedings of ACM Conference on Computer-Supported Cooperative Work (CSCW'98)*, Primitives for Building Flexibile Groupware Systems, pages 247–256. ACM Press, 1998.
9. C. Godart, O. Perrin, and H. Skaf. COO: a workflow operator to improve cooperative modelling in virtual processes. In *9th IEEE International Workshop on Research Issues on Data Engineering: Information Technology For Virtual Enterprises (RIDE-VE'99)*, Sydney, Australia, March 23–24, 1999.

10. J. Puustjärvi. *Transactional Workflows*. PhD thesis, Department of Computer Science, University of Helsinki, Finland, 1999.
11. G. Alonso, C. Hagen, and A. Lazcano. Process in Electronic Commerce. In *ICDS workshop on Electronic Commerce and Web-Based Applications*, Austin, Texas, USA, June 1999.
12. F. Leymann and D. Roller. *Production Workflow, Concepts and Techniques*. Prentice-Hall, Inc., 2000.
13. G. Piccinelli. Distributed Workflow Management: The TEAM Model. In *3rd IFCIS Int. Conf. on Cooperative Information Systems (CoopIS'98)*, pages 292–299, New York City, New York, USA, August 20–22, 1998. IEEE-CS Press.
14. F. Casati, S. Ilnicki, L. J. Jin, and M. C. Shan. eFlow: an Open, Flexible, and Configurable Approach to Service Composition. In *2nd International Workshop on Advance Issues of E-Commerce and Web-Based Information Systems (WECWIS'00)*, pages 125–132, Milpitas, California, June 8–9, 2000.
15. 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.
16. L. Kutvonen. *Trading services in open distributed environments*. Th se en informatique, Department of Computer Science, University of Helsinki, Finland, 1998.
17. B. Benatallah, B. Medjahed, A. Boughettaya, A. Elmagarmid, and J. Beard. Composing and Maintaining Web-based Virtual Enterprises. In *1st Workshop on Technologies for E-Services, In Cooperation with VLDB'2000 (TES'00)*, Cairo, Egypt, September 14–15, 2000.
18. H. Giese and G. Wirtz. The OCoN Approach for Object-Oriented Distributed Software Systems Modeling. In *Software Engineering and Petri Nets, Workshop within the 21st International Conference on Application and Theory of Petri Nets, Aarhus, Denmark, June 26*, 2000.
19. J. Klingemann, J. Wasch, and K. Aberer. Deriving service models in cross organizational workflows. In *9th International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises (ITVE'99)*, Sydney, Australia, 1999. IEEE Computer Society Press.
20. OMG. *Workflow Management Facility Specification, V 1.2*. OMG (Object Management Group), ww.omg.org, April 2000.
21. Karim Baïna. *Un Modèle Orienté Services Procédés pour l'Interconnexion et la Coopération des Procédés d'Entreprises*. Ph.D thesis in Computer Science, Université Henri Poincaré (Nancy 1), May 16, 2003.
22. K. Baïna, K. Benali, and C. Godart. A process service model for dynamic enterprise process interconnection. In C. Batini, F. Giunchiglia, P. Giorgini, and M. Mecella, editors, *9th Int. Conf. on Cooperative Information Systems, In Cooperation with IFCIS (CoopIS'01)*, number 2172 in LNCS, pages 239–254, Trento, Italy, September 5–7, 2001. Springer-Verlag.
23. 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)*, number 1901 in LNCS, pages 224–235, Eilat, Israel, September 6–8, 2000. Springer-Verlag.
24. DSTC. *Breeze: workflow with ease*. DSTC (Distributed Systems Technology Centre), Australia, www.dstc.edu.au/Downloads/, February 15, 2002.
25. O. Kummer, F. Wienberg, and M. Duvigneau. *Renew – User Guide*. University of Hamburg, Department for Informatics, Theoretical Foundations Group, Distributed Systems Group, Germany, www.renew.de, July 3, 2001.

26. Hitachi. *WorkCoordinator Workflow System.* Hitachi Ltd., www.hitachi.co.jp/Prod/comp/soft1/wco/, 2002.
27. K. Baïna, S. Tata, and K. Benali. A model for process service interaction. In W. van der Aalst, A. ter Hofstede, and M. Weske, editors, *Conference on Business Process Management, on the Application of Formal Methods to "Process-Aware" Information Systems (BPM'03)*, number 2678 in LNCS, pages 261–275, Eindhoven, The Netherlands, June 26–27, 2003. Springer-Verlag.

Wil van der Aalst
Arthur ter Hofstede
Mathias Weske (Eds.)

LNCS 2678

Business Process Management

International Conference, BPM 2003
Eindhoven, The Netherlands, June 2003
Proceedings

Springer

A Model for Process Service Interaction

Karim Baïna[1], Samir Tata[2], and Khalid Benali[1]

[1] LORIA-INRIA-CNRS (UMR 7503)
Campus Scientifique B.P. 239 54506 Vanœuvre-lès-Nancy - FRANCE
{baina,benali}@loria.fr

[2] Institut National des Télécommunications
9, rue Charles Fourier 91011 Evry Cedex - FRANCE
Samir.Tata@int-evry.fr

Abstract. The design and the achievement of any consequent project imply the involvement of several people, teams and even enterprises. These enterprises interact and exchange data, more and more often, through Internet and the "Web". However, exchanging data is not enough for working together, it is also necessary to control and manage these exchanges occurring within business processes. Cooperation between enterprises means interconnecting and coordinating their business processes. If a wide spectrum of tools for work coordination exists, they have been unfortunately developed to only suit the internal needs of enterprises. Thus, existing work coordination systems are not adapted to inter-enterprise cooperation. This paper presents a promising approach for interconnecting processes based on service interaction. Its aim is to formally present a model for enterprise process interconnection and coordination through service interaction based on information sharing between process services, and process services coordination.

1 Introduction

Due to business process automation development, process interconnection and coordination has become an important matter. If a large number of business process management systems exist (e.g. workflows, shared agendas, project managers, to do lists), they have been mainly developed to suit internal needs of enterprises.

In workflow management systems (WFMS), existing interconnection and/or coordination solutions are mostly static and depend on specific business process definition languages and WFMS platforms and with regards to private data exchange formats, etc. To improve generic process interconnection support within existing WFMS, related interconnection models deal with awareness and dataflow formalization between inter-leaving processes (*e.g.* shared dataspace models [17], message passing mechanisms [3], event subscription and notification paradigms [11], remote object invocation [24,16], transfer protocol extension [4]), or with inter-leaving process control (*e.g.* transactional protocols [18,8,2]).

W.M.P. van der Aalst et al. (Eds.): BPM 2003, LNCS 2678, pp. 261–275, 2003.

Another way to connect and coordinate enterprise processes is based on the web. For instance, several architectures based on electronic service exchange between applications exist (e.g. IONA Orbix E2A [13], Sun ONE [19], Microsoft .NET [15]). However, current specification of Service Oriented Architectures (SOA), based on service definition language (WSDL), service discovery registries [22,12] and service invocation protocol (SOAP)[23], are still in the beginning of their progress. Actually, web services frameworks are not enough mature to support enterprise process interconnection. For instance, SOA lacks a lot of rich mechanisms like service matching, service negotiation and contracting, service sessions, service transactions, service licensing, service security, service communication and coordination, etc.

In our opinion, the more promising and the more generic approach for interconnecting processes should be service oriented. In other words, to develop a model for enterprise process interconnection we can use process service interaction. The service concept has been defined in many research fields: object oriented research [16], process modeling research [10,6], distributed system research, etc. In the field of workflow, a process service may be seen as a software entity which is able to present process particularities and outcomes without totally revealing its structure (i.e. its implantation in a WFMS or project management tool). A process service offers functional abstraction of a process (or a sub-process) supplied by an enterprise or an enterprise accepts to fulfill with a predefined quality of service. A process service specifies the work load that an enterprise accept to fulfill with a predefined quality of service. Process service concept has been studied from several points of view : abstract semantics of process service execution, sub-process service selection, activity coordination of dynamic process service, process service control flow abstraction, process service methods and events wrapping [5], and process service composition [9]. For us, a process service is seen as a pattern supporting cooperation and dynamic interconnection between enterprise processes.

Since we beleve that a service oriented approache is a promising approach and since the SOA approach lak a lot of rich mechanisms, we are developing a general service oriented approach with the following models :

- a negotiation model for computer supported cooperative work we have developed in [14] and we have applied to process services to improve SOA facilities and dynamics [1],
- a model for dynamic selection and interconnection of process service contracts we have developed in [1], and
- a model for process service interaction we present in this paper.

While our previous works dealed with atomic process servicesa [1], this work focuses on composition, cooperation and interaction of process services for enterprises process interconnection. Trying to go further limits of SOA, we present a generic process service interaction model for interconnection of enterprise processes. We will start, in section 2, by defining our process service approach which is at the heart of our process service interaction model. Then we will present the two dimensions on which our process service interaction model is based on :

information sharing between process services in section 3, and process services coordination in section 4. Section 5 will sum up our interaction model. And finally, section 6 will give some concluding remarks.

2 Process Interconnection through Process Service Interaction

Existing process management systems often consider process instance as a black box (process method calls, execution events, and intermediate data are not visible from outside). This approach does not suit process cooperation needs. One of the aim of our process service interaction model is to allow processes to cooperate through partial visibility of their resources (e.g. data sharing, group awareness) by permitting access to a set of their own methods and events [1].

Service exchange is a high level cooperation paradigm. It's based upon existing cooperation paradigms : *production* (dataspace sharing), *communication* (common message format, remote method invocation, notification messages, group awareness), and *coordination* (synchronization, coherence criteria checking). Therefore, service exchange offers a better expressiveness and comprehensiveness for cooperation situations. Our aim is to develop a service structure pattern supporting cross organizational processes cooperation.

A process service can represent either the description of a task that an enterprise may wish to out-source (e.g. because of time lack, cost reducing, or skill needs) or a specific task that an enterprise is known to be skilled to accomplish. This task may be managed by an automatic, a semi-automatic, or a manual process. It may be performable by a unique "provider" (atomic process service), or may need cooperation of several providers (composite process service) (e.g. because it needs several skills not necessary within the scope of a single provider). We distinguish thus between two types of process services : atomic process service and composite process service. The atomic process service represents the interface of a process, which its structure is hidden. The composite process service is constituted by a set of process services which may be themselves atomic or composite. A requester can compose the needed process services by using process services supplied by several providers. To do that, the requester needs a set tools to describe and manage interactions of the different provided process services. The interaction model we propose in this paper takes place in this context. It focus on process service coordination by data sharing and control flow management. We define our process service interaction model as 5 sets :

- a set of access contracts on shared data,
- a set of visible method execution contracts,
- a set of visible event reception contracts,
- a set of coordination contracts based on shared data states,
- a set of coordination contracts based on process service states.

To illustrate our approach, figure 1 shows an example of process service interconnection within an e-learning context. Let *KManager* (an e-learning enterprise)

be a service requester enterprise. Let *WAgency* (a web agency enterprise), *CoolHost* (a site hosting enterprise), and *e-Store* (an e-learning content collection enterprise) be three (among others) service provider enterprises. These enterprises cooperate by providing (and requesting) process services to be achieved within their cooperation. Process services represent either their skills ans requirements within their cooperation partnership. The enterprise *KManager* is the main actor of this cooperation and its aim is to produce e-learning sessions. To do so, it uses process services provided by its partners. Figure 1 shows how our model can be applied to interconnect business processes by outsourcing and composing process services.

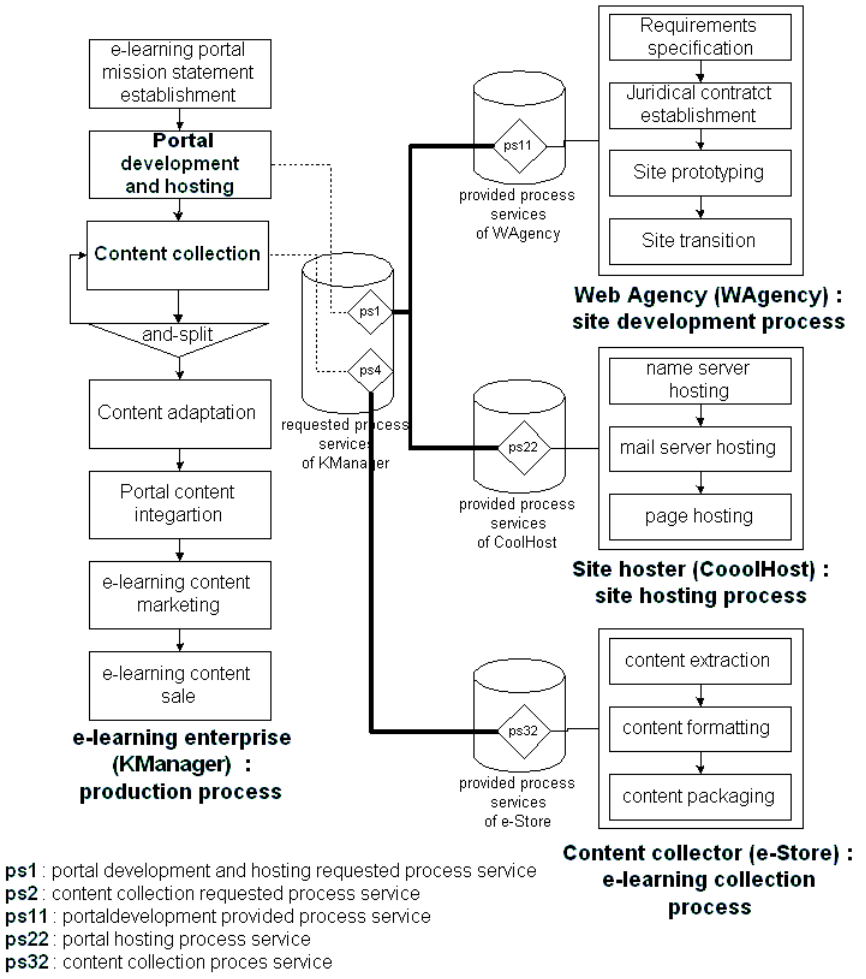


Fig. 1. An interconnection example of an e-learning enterprise process services

Within this example, we consider the following set of process services PS, set of data D, set of access rights AR, set of methods M, and set of events E:

- $PS = PS_{prv} \cup PS_{req}$: the union of provided and requested process services where
  - $PS_{prv} = \{ps_{11}$ (portal development process service), ps_{22} (portal hosting process service), ps_{32} (content collection process service)$\}$,
  - $PS_{req} = \{ps_1 := (ps_{11}, ps_{22})$ (composite service), $ps_4 := ps_{32}$ (atomic process service)$\}$;
- $D = \{d_1$: portal development judicial contract, d_2: portal hosting judicial contract, d_3: graphical charter of the portal development, d_4: portal prototype...$\}$;
- $AR =$ {read, write};
- $M = M_{ps_{11}} \cup M_{ps_{22}} \cup M_{ps_{32}}$ where
  - $M_{ps_{11}} =$ {getPortalDevelopmentAllocatedHumanResources(),...},
  - $M_{ps_{22}} =$ {getPortalHostingStatus(),...},
  - $M_{ps_{32}} =$ {getProcessInstanceAchievedModuleDuration(),...}
- $E = E_{ps_{11}} \cup E_{ps_{22}} \cup E_{ps_{32}}$ where
  - $E_{ps_{11}} =$ {ASK_FOR_MORE_SPECIFICATION,...},
  - $E_{ps_{22}} =$ {BEGIN_PORTAL_HOSTING...},
  - $E_{ps_{32}} =$ {END_MODULE_COLLECTION...}.

According to the approach we have presented above, we aim to interconnect processes through process service interaction. We describe interaction through two dimensions: data sharing and process service coordination. We present, in the following, these features that we will integrate into our model for process service interaction.

3 Process Service Information Sharing

The behavior of process service interactions depends on the nature of operations they perform on the shared information. Process services can share common data and/or access to private data using methods published by their process service owners. These accesses have to be controlled so that every process service provider plays its real role in the composition. For this reason, we define access contracts for both shared and private data.

We define an access contract as the association of a service, an access right, and an object. We denote an access contract by *Access*.
$Access(ps, o, ar) \stackrel{\text{def}}{=}$ the process service ps has the access right ar on the object o modelling some. We express access contracts for shared data in the example presented in section 2 as:

- $Access(ps_1, d_1, write)$ (the composite service for portal hosting and development can write the portal development contract),
- $Access(ps_{22}, d_3, read)$ (the provided service for portal hosting can read graphical charter of the portal development).

In a composite process service, each process service provides a set of access points to its resources for other cooperating process services. These points include methods remote process services can execute in order to access to private data. The access to these methods is controlled by execution rights:

$Exec(ps_i, M^{'}) \stackrel{\text{def}}{=} \forall m_{ps_j} \in M^{'}$, the process service ps_i can execute the method m of the process service ps_j $(i \neq j)$.

In the example presented in section 2 methods of the provided services ps_{11} and ps_{22}, the required process services ps_1 and ps_4 are $Exec(ps_1, M^{'}_{ps_{11}} \cup M^{'}_{ps_{22}})$ and $Exec(ps_4, M^{'}_{ps_{32}})$ where

- $M^{'}_{ps_{11}}$ = {getPortalDevelopmentAllocatedHumanResources(), getPortalDevelopmentStatus(), checkinDocument(), checkoutDocument()} (NB: $M^{'}_{ps_{11}} \subset M_{ps_{11}}$),
- $M^{'}_{ps_{22}}$ = {getPortalHostingStatus(), checkinDocument(), checkoutDocument()} (NB: $M^{'}_{ps_{22}} \subset M_{ps_{22}}$),
- $M^{'}_{ps_{32}}$ = {getProcessInstanceAchievedModuleDuration(), checkinDocument(), checkoutDocument()} (NB: $M^{'}_{ps_{32}} \subset M_{ps_{32}}$).

In addition to method visibility, it is necessary to provide to cooperating process services means to coordinate their actions in an informal way and to work in an autonomous way while being aware of what occurs in the composite process service. For this reason, each process service has to provide events allowing other services to know the state of its execution and its private data. Since each process service is interested in a specific information, it is beneficial to control the visibility of process service events:

$Recept(ps_i, E') \stackrel{\text{def}}{=} \forall e_{s_j} \in E'$, the process service ps_i can receive the visible event e of the process service ps_j $(i \neq j)$.

Concerning event visibility, the process services ps_1 and ps_4 can receive from the process services ps_{11}, ps_{22} and ps_{32} the following sets of events : $Recept(ps_1, E^{'}_{ps_{11}} \cup E^{'}_{ps_{22}})$ and $Recept(ps_4, E^{'}_{ps_{32}})$ where

- $E^{'}_{ps_{11}}$ = {ASK_FOR_MORE_SPECIFICATION,ENDPORTAL_DEVELOPMENT} (NB: $E^{'}_{ps_{11}} \subset E_{ps_{11}}$),
- $E^{'}_{ps_{22}}$ = {BEGIN_PORTAL_HOSTING,ASKFORMORE_SPECIFICATION} (NB: $E^{'}_{ps_{22}} \subset E_{ps_{22}}$),
- $E^{'}_{ps_{32}}$ = {END_MODULE_COLLECTION, PRODUCED_NEW_MODULE_VERSION} (NB: $E^{'}_{ps_{32}} \subset E_{ps_{32}}$).

4 Process Service Coordination

We define process service coordination as a constrained interaction (data sharing, event reception, and method invocation). This coordination can be based on their states or on the states of their shared data.

4.1 Coordination Based on Process Service States

We define a coordination contract based on process service states as a set of coordination rules. These rules denote an association of two process services and a type of coordination. Each coordination type defines a constraint on the states of both process services. In order to give some coordination rule examples, we introduce the concepts of process service states, state sequences, and process service view sequence. Each process service has a state which reflects its situation or circumstances that describe it at a given time. We call a state sequence of a process service ps, that we denote S^{ps}, the ordered complete set of states of ps. We can define S^{ps} as $S^{ps} = \{s_0^{ps}, s_1^{ps}, \dots\}$, where s_0^{ps} is the initial state of ps and s_1^{ps}, s_2^{ps},... some of its successive states. A state of a process service ps_1 is observed by a process service ps_2 if this latter has executed a method of ps_1 that returns a value reflecting a state of ps_1, or if ps_2 receives an event from ps_1 showing its state.

For a process service ps_1, we call a view sequence of process service ps_2, noted $S_{ps_2}^{ps_1}$, the subset of S^{ps_1} (states of ps_1) observed by ps_2. Considering a process service with a begin and an end, the set of methods it can execute and the set of events it can receive are finite. Thus, the set $S_{ps_2}^{ps_1}$ is finite. Let $S_{ps_2}^{ps_1} = \{s_{ps_2,0}^{ps_1}, s_{ps_2,1}^{ps_1}, \dots, s_{ps_2,n-1}^{ps_1}\}$ be the state sequence of the process service ps_1 observed by ps_2, where n is the number of ps_1 states observed by ps_2. $s_{ps_2,0}^{ps_1}$ is the initial state of ps_1 observed by ps_2 after its beginning and $s_{ps_2,n-1}^{ps_1}$ is the last state of ps_1 observed by ps_2 before its end. Note that there may be some states of ps_1 not observed by ps_2 and which can be produced between two states of ps_1 observed by ps_2. We call view sequence of a process service ps all its states and all the states of other process services that it can observe. Let $V^{ps} = S^{ps} \cup_{ps_1 \in PS} S_{ps}^{ps_1}$ be the view sequence of the process service ps. This set represents the states of the process service that ps is aware about.

Let $SCoor(ps_1, ps_2, coortype)$ be a coordination rule based on process service states, where ps_1 and ps_2 are two process services from the process service set PS and *coortype* be a coordination type belonging to the state based coordination type set $SCoorT$. Coordination types we present are inspired from several works. We mention, among others, those achieved in the transactional field [7], and those accomplished in workflow interconnection field [24]. In order to introduce these coordination types, we use the following notations. Let ps be a composite process service, ps_i and ps_j two of its process service components. Let S^{ps_i} and S^{ps_j} be their respective state sequences. Let S (s state sequence) be the union of S^{ps_i} and S^{ps_j}. We supply S by a partial order relation we denote $\rightarrow$ that handles the causality of S elements.

Considering a process service as a transaction with a begin (enact), an end (commit, abort), and a life cycle, we can express dependencies on process service states using the transactional approach presented in [7]. Process service state significant events include Enacted, Preempted, Committed, Aborted,....

Here are some types of state based process service coordination for describing:

- serial dependency of two process services (*ECD*: Enact-On-Commit Dependency):

$$SCoor(ps_2, ps_1, ECD) \stackrel{\text{def}}{=} \mathsf{Enacted}_{ps_2} \in S \Rightarrow \mathsf{Committed}_{ps_1} \rightarrow \mathsf{Enacted}_{ps_2}$$

– commit dependency of two process services (*CD*: Commit Dependency):

$$SCoor(ps_2,\ ps_1,\ CD) \stackrel{\text{def}}{=}$$
$$\mathsf{Committed}_{ps_2} \in S \Rightarrow (\mathsf{Committed}_{ps_1} \in S \Rightarrow$$
$$(\mathsf{Committed}_{ps_1} \rightarrow \mathsf{Committed}_{ps_2}))$$

Interface 4 of the W*f*MC (Workflow Management Coalition) [24] supports workflow processes interconnection by proposing paradigms for workflow interoperability. Based on the approach of the W*f*MC paradigms, we define two new process service state based coordination types describing:

– asynchronous externalisation of a sub process service (*CMD*: Chained Model Dependency):

$$SCoor(ps_2,\ ps_1,\ CMD) \stackrel{\text{def}}{=}$$
$$\mathsf{Enacted}_{ps_2} \in S \Rightarrow \mathsf{Enacted}_{\ ps_1} \rightarrow \mathsf{Enacted}_{ps_2}$$

– synchronous externalisation of a sub process service (*NMD*: Nested sub-process Model Dependency)

$$SCoor(ps_2, ps_1, NMD) \stackrel{\text{def}}{=}$$
$$(\mathsf{Enacted}_{ps_2} \in S \Rightarrow \mathsf{Enacted}_{ps_1} \rightarrow \mathsf{Enacted}_{ps_2}) \wedge$$
$$(\mathsf{Committed}_{ps_1} \in S \Rightarrow \mathsf{Committed}_{ps_2} \rightarrow \mathsf{Committed}_{ps_1})$$

These types of process service state based coordination have been given as simple examples. We have developed other coordination types that we do not present in this paper[1].

The following state based coordination contracts are expressed for the process service of the section 2 :
$SCoor(ps_{11},\ ps_1,\ NMD)$: the process service ps_{11} (portal development provided process service) is a nested sub-process of the process service ps_1 (portal development and hosting requested process service) and $SCoor(ps_{22},\ ps_1,\ NMD)$: the process service ps_{22} (portal hosting provided process service) is a nested sub-process of the process service ps_1.

4.2 Coordination Based on Data States

We define a coordination contract based on data states as the association of two process services, an object (modeling the data), and a type of coordination ([20], [21]). To define formally coordination contracts, we use the notion of object state

[1] An example of coordination types we have developed is Parallel Synchronized Model Dependency: ps_1 and ps_2 are parallel synchronized if they agree on a synchronization point within which they exchange data then they continue their executions normally.

sequence and process service view sequence that we define in the following before presenting coordination contracts based on data states.

We call the state sequence of an object o, denoted S^o, the ordered set of the states of o. We can define S^o as $S^o = \{s_0^o,\ s_1^o, \ldots\}$, where s_0^o is the initial state of the object and $s_1^o,\ s_2^o,\ \ldots$ are the successive states of o. An object can change its state when a process service performs an operation that changes the value of one of its attributes. A process service can observe a state of the object if it performs an operation which returns a value reflecting the state of this object. Note that no assumptions are made on the granularity of a state change. Depending on the context and of the application being specified, this may vary from one single character change in a textual document to a CAD [2] object replacement. We call the view sequence of a process service ps on an object o, noted S_{ps}^o, the ordered subset of S^o (the state sequence of o) restricted to the states observed by ps.

Given an object, a coordination contract binds two process services to express the fact that there is a constraint on their view sequences. The coordination is a *process service coordination* if it expresses constraints on the sequence of states observed by one or the other of the both process services. It is a *state coordination*, if it expresses constraints on only one state observed by one or the other of the both process services.

For a given object o, the process service ps_1 is coordinated to the process service ps_2 on o when $S_{ps_1}^o$ is included in $s_{ps_2}^o$. We note this coordination type *SvCoor* (Process Service Coordination).

$$DCoor(ps_1, ps_2, o, SvCoor) \stackrel{\text{def}}{=} S_{ps_1}^o \subset S_{ps_2}^o.$$

ps_1 and ps_2 are state coordinated if there exists a state $s_{ps_1,i}^o$ observed by ps_1 and a state $s_{ps_2,j}^o$ observed by ps_2 that are identical. We note this coordination type *StateCoor* (State Coordination).

$$DCoor(ps_1,\ ps_2,\ o,\ StateCoor) \stackrel{\text{def}}{=}$$
$$\exists\ i \in \{0..n-1\},\ \exists j \in \{0..m-1\}\ s_{ps_1,i}^o = s_{ps_2,j}^o$$

where n (respectively m) denotes the number of states observed by ps_1 (respectively ps_2).

This type of coordination can be used to specify the more general case of two process services coordinated on an arbitrary state. However, there is a lot of situations in which such coordination occurs at the beginning or at the end of one or both of the coordinated process services.

For this reason, we provide specialized coordination types corresponding to the coordination of :

- a process service initial state and another process service state (Initial state Coordination):

[2] Computer Aided Design.

$$DCoor(ps_1, ps_2, o, ICoor) \stackrel{\text{def}}{=} s^o_{ps_1,0} \in S^o_{ps_2}$$

– the initial states of two process services (Initial States Coordination):

$$DCoor(ps_1, ps_2, o, IsCoor) \stackrel{\text{def}}{=} s^o_{ps_1,0} = s^o_{ps_2,0}$$

– a process service final state and another process service state (Final state Coordination):

$$DCoor(ps_1, ps_2, o, FCoor) \stackrel{\text{def}}{=} s^o_{ps_1,n-1} \in S^o_{ps_2},$$
where n is the number of the states of o observed by ps_1

– the final states of two process services (Final States Coordination):

$$DCoor(ps_1, ps_2, o, FsCoor) \stackrel{\text{def}}{=} s^o_{ps_1,n-1} = s^o_{ps_2,m-1}$$
where n (respectively m) is the number of the states of o observed by ps_1 (respectively ps_2)

– a process service final state with another process service initial state (Serial Coordination):

$$DCoor(ps_1, ps_2, o, SrlCoor) \stackrel{\text{def}}{=} s^o_{ps_1,n-1} = s^o_{ps_2,0}$$

– two state sequences of two process services (Total Coordination):

$$DCoor(ps_1, ps_2, o, TotCoor) \stackrel{\text{def}}{=} n = m \wedge \forall\ i\ \in\ \{0..n-1\}, s^o_{ps_1,i} = s^o_{ps_2,i}$$

There are some dependencies between the different coordination types that lead to implicit coordination types when we define some other stronger coordination type. For instance we have the following dependencies:

– if the process service ps_1 is coordinated to the process service ps_2, then the ps_1 initial state is coordinated to another ps_2 state:

$$DCoor(ps_1, ps_2, o, SvCoor) \Rightarrow DCoor(ps_1, ps_2, o, ICoor)$$

– if the initial states of ps_1 and ps_2 are coordinated, then the ps_1 initial state is coordinated to another ps_2 state and the ps_2 initial state is coordinated to another ps_1 state:

$$DCoor(ps_1, ps_2, o, IsCoor) \Rightarrow$$
$$DCoor(ps_1, ps_2, o, ICoor) \wedge DCoor(ps_2, ps_1, o, ICoor)$$

– if the process service ps_1 is coordinated to the process service ps_2, then the ps_1 final state is coordinated to another ps_2 state:

$$DCoor(ps_1, ps_2, o, SvCoor) \Rightarrow DCoor(ps_1, ps_2, o, FCoor)$$

– if the final states of ps_1 and ps_2 are coordinated, then the ps_1 final state is coordinated to another ps_2 state and the ps_2 final state is coordinated to another ps_1 state:

$$DCoor(ps_1, ps_2, o, FsCoor) \Rightarrow DCoor(ps_1, ps_2, o, FCoor) \wedge DCoor(ps_2, ps_1, o, FCoor)$$

– we quote, in addition, here that the total coordination is a strong coordination type that implicit implies other ones. If ps_1 and ps_2 are totally coordinated, then
  - ps_1 is coordinated to ps_2 and ps_2 is coordinated to ps_1:

$$DCoor(ps_1, ps_2, o, TotCoor) \Rightarrow DCoor(ps_1, ps_2, o, SvCoor) \wedge DCoor(ps_2, ps_1, o, SvCoor)$$

  - the final states of ps_1 and ps_2 are coordinated:

$$DCoor(ps_1, ps_2, o, TotCoor) \Rightarrow DCoor(ps_1, ps_2, o, FsCoor)$$

  - the ps_1 final state is coordinated to another ps_2 state and the ps_2 final state is coordinated to another ps_1 state:

$$DCoor(ps_1, ps_2, o, TotCoor) \Rightarrow DCoor(ps_1, ps_2, o, FCoor) \wedge DCoor(ps_2, ps_1, o, FCoor)$$

  - the initial states of ps_1 and ps_2 are coordinated:

$$DCoor(ps_1, ps_2, o, TotCoor) \Rightarrow DCoor(ps_1, ps_2, o, IsCoor)$$

  - the ps_1 initial state is coordinated to another ps_2 state and the ps_2 initial state is coordinated to another ps_1 state:

$$DCoor(ps_1, ps_2, o, TotCoor) \Rightarrow DCoor(ps_1, ps_2, o, ICoor) \wedge DCoor(ps_2, ps_1, o, ICoor)$$

In the example presented in section 2 the coordination contracts $DCoor(ps_1, ps_{11}, d_1, TotCoor)$ based on the data states of the process services shows that the required process service for portal development and hosting ps_1, and the provided process service for portal development ps_{11} are totally coordinated on the portal development contract d_1.

5 Process Service Interaction Model

In this section, we propose an interaction model which supports information sharing and coordination between cooperating process services. Thus, this model can describe needs of roles and coordination management for cooperating process services.

Formally, process service interaction model is represented by the tuple (PS, D, CL) where PS is the set of process services, D is the set of shared documents, and CL (Contract List) is a set of cooperation contracts. Each contract binds a set of process services participating to a cooperation by:

- DAL (Data Access List), a set of dynamic access rights on shared documents;
- MAL (Method Access List), a set of dynamic execution rights on methods made accessible by process services;
- EAL (Event Access List), a set of dynamic reception rights on events produced by process services;
- DCL (Data Coordination List), a set of dynamic coordination rules based on process service shared information state;
- SCL (Service Coordination List), a set of dynamic coordination rules based on process service states.

By dynamic, we mean negotiation based customisation [14].

Here is an example of the tuple (PS, D, CL):

$PS = PS_{frn} \cup PS_{req}$: the union of the following provided and requested process services $PS_{frn} = \{ps_{11}$: portal development process service, ps_{22}: portal hosting process service, ps_{32}: content collection process service$\}$ and $PS_{req} = \{ps_1 := (ps_{11}, ps_{22})$ (composed process service), $ps_4 := ps_{32}$ (atomic process service)$\}$

$D = \{ d_1$: portal development judicial contract, d_2: portal hosting judicial contract, d_3: portal development graphical charter, d_4: portal prototype, d_5 : e-learning module requirements,... $\}$

and CL:

$DAL = \{Access(ps_1, d_1, read),\ Access(ps_1, d_1, write),\ Access(ps_{22},\ d_3,\ read),\ ...\}$,
$MAL = \{Exec(ps_1, M^{'}_{ps_{11}} \cup M^{'}_{ps_{22}}),\ Exec(ps_4, M^{'}_{ps_{32}}),\ ...\}$,
$EAL = \{Recept(ps_1,\ E^{'}_{ps_{11}} \cup E^{'}_{ps_{22}}),\ Recept(ps_4,\ E^{'}_{ps_{32}}),\ ...\}$,
$DCL = \{DCoor(ps_1,\ ps_{11},\ d_1,\ TotCoor),\ DCoor(ps_{11},\ ps_{22},\ d_3,\ IsCoor),\ ...\}$,
$SCL = \{SCoor(ps_{11},\ ps_1,\ NMD),\ SCoor(ps_{22},\ ps_1,\ NMD), ...\}$.

with

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

Cooperation rules define explicitly allowed, refused, and compulsory operations. Information sharing rules, we may define inside a cooperation contract,

assign (negative, or positive) rights on shared documents, on accessible events and/or methods. Each positive access right enables its possessing process service to execute a set of operations on shared or private documents, or to receive events reflecting states of data of other process service. Negative information sharing rules prohibit to a process service to execute a set of operations or the reception of a set of events. To respect certain coordination rules inside a cooperation contract, the participating process services have to observe the shared information states. Actually, if it exists a coordination rule of $DCoor(ps_1, ps_2, d, coortype)$ type, then ps have to observe at least a state of d when $coortype$ belongs to $\{IsCoor, FsCoor, TotCoor, SrlCoor\}$.

For a process service ps that participate to a cooperation contracts, we can define consequently a set of operations that ps can execute, a set of operations that ps cannot execute, and a set of operations that ps ought to execute. Process service interaction model defines cooperation contracts between process services. These contracts model rules for process interconnection. However, cooperation consistency has to be maintained in order to avoid contradiction between agreed contracts. A process service ps can respect a cooperation contract if every compulsory operations are accessible, and every compulsory operations are not prohibited. A cooperation contract is thus coherent if it can be respected by every cooperating process services.

6 Conclusion

Our paper is a contribution in the field of process interconnection. Our process service interaction model allows communication and coordination between process services aiming to interconnect processes, and, therefore, to support inter-enterprises cooperation. Actually, our process service interaction model encompasses information sharing between process services, and process services coordination. Therefore our model allows management of roles and coordination of services through access rights on shared data, execution rights on process services methods, reception rights on process services events, and coordination rights based on data states or on process service states.

Although implementation is not in the intended scope of this paper, we can mention that we are currently developing an implementation of our process service interaction model within *DISCOBOLE* (DIStributed COoperation and Business prOcess on LinE), our cooperative platform for process service interconnection. *DISCOBOLE* is developed in Java on a CORBA bus. Our implementation relies on a process service structure which may be seen as the development of a design pattern for cooperation. Actually, a process service may be related to proxy and adapter patterns. While the proxy pattern limits accesses to process resources (data access rights $Access(ps, d, dr)$, method visibility $Exec(ps,\ M)$, visibility of process service events $Recept(ps,\ E)$), the adapter pattern provides a new interface to the adapted process service (ability to exchange data, coordination ability, ability to control interaction consistency).

Our next step will be fully explore the cooperation pattern paradigm we have used to implement our interaction service model. In fact, our model description and our implementation of it with patterns present two orthogonal views of process service interaction. We plan to cross fertilize and validate formally these two orthogonal views to produce an integrated and complete view.

References

1. Baïna, K., Benali, K. and Godart, C.: A process service model for dynamic enterprise process interconnection. In *9th Int. Conf. on Cooperative Information Systems, In Cooperation with VLDB 2001*, volume 2172 of *LNCS*, pages 239–254, Trento, Italy, September 5–7, 2001. Springer-Verlag.
2. K. Baïna, F. Charoy, C. Godart, D. Grigori, S. el Hadri, H. Skaf, S. Akifuji, T. Sakaguchi, Y. Seki, and M. Yoshioka. CORVETTE: A Cooperative Workflow Development Experiment. In L. M. Camarinha-Matos, editor, *3rd IFIP Working Conference on Collaborative Business Ecosystems and Virtual Enterprises (PRO-VE'02)*, pages 169–180, Sesimbra, Portugal, May 1–3, 2002. Kluwer Academic Publishers.
3. 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)*, pages 336–347, Edinburgh, Scotland, September 2–4, 1999. IEEE Computer Society Press.
4. 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.
5. B. Benatallah, B. Medjahed, A. Boughettaya, A. Elmagarmid, and J. Beard. Composing and maintaining web-based virtual enterprises. In *1st Workshop on Technologies for E-Services, In cooperation with VLDB 2000 (TES'00)*, Cairo, Egypt, September 14–15, 2000.
6. F. Casati, S. Ilnicki, L. J. Jin, and M. C. Shan. eflow: an open, flexible, and configurable approach to service composition. In *2nd International Workshop on Advance Issues of E-Commerce and Web-Based Information Systems (WECWIS'00)*, pages 125–132, Milpitas, California, June 8–9, 2000.
7. P-K. Chrysanthis and K. Ramamritham. ACTA: The SAGA continues. In *Database Transaction Models for Advanced Applications*, pages 349–397. 1992.
8. K. Dutta, D. VanderMeer, A. Datta, and K. Ramamritham. User Action Recovery in Internet SAGAs (iSAGAs). In F. Casati, D. Georgakopoulos, and M-C. Shan, editors, *2nd Workshop on Technologies for E-Services, In Cooperation with VLDB'2001 (TES'01)*, number 2193, pages 132–146, Rome, Italy, September 14–15, 2001. Springer-Verlag.
9. M.-C. Fauvet, M. Dumas, B. Benatallah, and H. Paik. Peer-to-peer traced execution of composite services. In F. Casati, D. Georgakopoulos, and M-C. Shan, editors, *2nd Workshop on Technologies for E-Services, In cooperation with VLDB 2001 (TES'01)*, volume 2193 of *LNCS*, pages 103–117, Rome, Italy, September 14–15, 2001. Springer-Verlag.
10. 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*, 24(6), 1999.
11. 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.

12. IBM. *IBM open sources technology for accessing UDDI Business Registry.* www-3.ibm.com/services/uddi/announce012501.html, January 2001.
13. IONA. *Orbix E2A, Web Services Integration Platfom.* IONA, www.iona.com, 2002.
14. 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.
15. Microsoft. *Microsoft .NET.* www.microsoft.com/net/, 2001.
16. OMG. Workflow Management Facility Convenience Document combining dtc/99-07-05 dtc/2000-02-03 (WF RTF 1.3 Report). *OMG, ww.omg.org*, February 2000.
17. 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.
18. 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.
19. Sun. Sun Open Net Environment (Sun ONE). *Sun microsystems, www.sun.com/sunone/*, 2002.
20. S. 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.
21. S. Tata. Policies for Cooperative Virtual Teams. In *Proceedings of the 5th International Conference on Coordination Models and Languages* COORDINATION 2002, York, UK, April 8–11, 2002.
22. UDDI.Org. UDDI V. 2.0 API Specification. *www.uddi.org*, June 8, 2001.
23. W3C. Simple Object Access Protocol (SOAP) V. 1.1. *W3C (World Wide Web Consortium), www.w3.org/TR/SOAP/*, 2000.
24. WFMC. Workflow Standard – Interoperability, Abstract Specification, WFMC-TC-1012, V. 1.0. *Workflow Management Coalition, www.wfmc.org*, October 20, 1996.

Information Interaction Intelligence

Actes des 2e Assises nationales du GdR i3

Rédacteur
Jacques Le Maitre

Nancy
décembre 2002

CÉPADUÈS-ÉDITIONS

Collaboration / Coopération

Khalid Benali[1], Grégory Bourguin[2], Bertrand David[3], Alain Derycke[4], Christine Ferraris[5]

[1]LORIA, Campus Scientifique, Nancy
benali@loria.fr

[2]Laboratoire d'Informatique du Littoral
BP 719, 62228 Calais Cedex
bourguin@lil.univ-littoral.fr

[3]Laboratoire ICTT, Ecole Centrale de Lyon, 69134 Ecully
Bertrand.David@ec-lyon.fr

[4]Laboratoire TRIGONE, Université des Sciences et Technologies de Lille, 59665 Villeneuve d'Ascq cedex,
alain.derycke@univ-lille1.fr

[5]Equipe SysCom, Université de Savoie,
Campus Scientifique, 73376 Le Bourget du Lac cedex,
Christine.Ferraris@univ-savoie.fr

Résumé. Le travail coopératif assisté par ordinateur (TCAO) se développe de façon considérable ces derniers temps. Sous cet acronyme, sont rassemblées des activités très différentes dont nous proposons une classification. Nous montrons également des aspects importants tant technologiques qu'organisationnels et humains s'y attachant. Nous terminons par les orientations de recherche qui émergent actuellement.

1 INTRODUCTION

L'objectif du TCAO (Travail Coopératif Assisté par Ordinateur), surtout connu sous le sigle anglo-saxon de CSCW (Computer-Supported Collaborative Work), est de permettre à un collectif d'acteurs de travailler

ensemble via une infrastructure informatique. Différentes formes et classifications du TCAO ont été dégagées. Parmi celles-ci, celle fondée sur la matrice espace-temps proposée par Ellis [15], garnie par Rodden et complétée par Dix et Grudin [24] caractérise assez bien des situations typiques du travail coopératif. La collaboration est exprimée par rapport au temps (asynchrone ou synchrone) et à la distance (local ou distant). D'autres caractéristiques souvent utilisées concernent la granularité informationnelle de manipulation et la granularité temporelle de mise à jour d'actions utilisateurs, la conscience des actions des autres (awareness), le partage de vues, …. Globalement, ces caractéristiques permettent de qualifier assez précisément chaque contexte coopératif.

2 CARACTERISATION ET EVOLUTION

2.1 Besoins

Les besoins identifiés pour les différentes applications coopératives dépendent des paramètres listés ci-dessus. Il semble toutefois important d'introduire des nuances car distinguer seulement interaction asynchrone et interaction synchrone ne permet pas de tenir compte des besoins qui sont à la fois qualitatifs et quantitatifs. En effet, entre les interactions fortement couplées (de type « electronic meeting ») et les interactions lâchement couplées (de type édition collaborative) les délais souhaitables d'action et de perception ne sont pas les mêmes. Il en est de même pour la notion de vue, vue partagée (WYSIWIS strict ou relâché), le besoin d'observabilité ou au contraire de « privacy ». Sans vouloir exprimer toutes ces nuances, il semble souhaitable de rappeler le trèfle d'Ellis [16] qui caractérise l'activité coopérative par trois espaces permettant respectivement de communiquer, de co-produire, de se coordonner. Toutes les applications ne couvrent pas de la même façon ces trois espaces; certains outils peuvent se situer à l'intersection de deux voire de trois espaces. Dans tous les cas, il semble souhaitable tant en analyse qu'en mise en oeuvre de se positionner par rapport à ceux-ci [11].

2.2 Classification

Lorsqu'il est question de classification, la plupart des travaux existants procèdent à une distinction selon le type de tâche auquel les collecticiels considérés apportent un support. Il est ainsi habituel de distinguer entre les Group Decision Support Systems, les « recommender systems », les

éditeurs collaboratifs et les supports à la co-conception. Une place spécifique est faite aux infrastructures ainsi qu'aux systèmes utilisés dans un contexte éducatif (e-learning au sens large du terme). Cette classification semble naturelle et correspond de plus à des développements et travaux de recherche entrepris dans ces différents cadres depuis plusieurs années.

Il nous a cependant semblé opportun de considérer un autre critère de classification. Il s'agit du degré de liberté laissé aux usagers engagés dans une activité coopérative pour intervenir sur leur environnement, critère que nous croiserons avec le type de tâche considérée (spécifique ou pas) ainsi que l'explicitation ou pas du modèle de la tâche. Nous pourrons ainsi classer les collecticiels présentés selon un axe allant du plus spécifique et du plus restreint en termes de fonctionnalités au plus générique et au plus ouvert pour les usagers.

2.2.1 Les outils élémentaires

A la base, nous trouvons les outils qui peuvent répondre à des besoins primitifs de communication entre acteurs : chat, forum, tableau blanc partagé, mail. Ils peuvent également servir à la coordination et à l'organisation, à travers la communication, et dans une moindre mesure d'espace de production (cas du tableau blanc – mais il s'agit là d'un espace non structuré et bien souvent volatile). Ils sont en général considérés comme des briques élémentaires qui vont entrer dans la composition de collecticiels plus élaborés.

2.2.2 Les collecticiels centrés « tâches »

Au niveau intermédiaire, nous plaçons les collecticiels construits autour d'une tâche spécifique (édition collaborative, mutualisation documentaire, co-conception, …). La tâche devient le sujet d'étude des concepteurs de collecticiels : ils cherchent à l'expliciter, en tout ou partie, étape après étape, dans le but de fournir un guide organisationnel ou un support logistique au groupe, dans un souci de coordination des actions des uns et des autres. Le focus retenu ici est celui du « process » et il est fait peu de cas des individus qui composent le groupe. Les fonctionnalités offertes dans les collecticiels concourent à fournir une aide dans ce sens.

Celles-ci varient grandement en fonction de l'explicitation ou non du modèle de la tâche. L'outil RtW (« Reading through Writing ») du projet européen Nimis [45] repose sur un modèle de tâche implicite dans lequel les ressources sont partagées entre les acteurs (des enfants). La tâche

consiste à construire phonétiquement des mots correspondant à des images. Elle n'est réalisable que par la mise en commun des ressources (les phonèmes). La collaboration va alors émerger de l'impossibilité qu'a chaque enfant de la mener à bien seul. L'outil leur fournit de fait un espace commun de construction du mot, dans lequel chacun peut déposer un phonème. Le projet « Memo-net » [32] s'appuie lui aussi sur un modèle de tâche implicite. Des acteurs sont invités à résoudre un problème d'analyse ou de synthèse en s'appuyant sur des concepts issus de ce modèle; ils disposent pour cela d'un forum structuré (étiqueté par ces concepts) qui leur permettra d'avoir une trace de la résolution du problème traité. Ces deux outils, parce qu'ils n'explicitent pas le modèle sur lequel ils sont construits, ne permettent aucune intervention des acteurs sur le processus et l'outil lui-même. Les acteurs se contentent d'utiliser les fonctionnalités et composants logiciels mis à leurs disposition pour résoudre la tâche.

L'explicitation du modèle, même partielle, fournit un premier moyen d'intervenir sur la tâche, et donc l'activité du groupe au delà. Ainsi, dans l'éditeur coopératif Byzance [8], il est supposé que les acteurs vont travailler sur des parties disjointes de documents, en écriture pour les modifier ou en lecture uniquement. La tâche de rédaction est alors vue comme l'agrégation des productions des uns et des autres, après fragmentation d'un texte initial. Les éléments issus de ce modèle de tâche sont des concepts tels que les rôles, les droits ou les fragments. Les acteurs disposent de fonctionnalités permettant de les manipuler. Ils acquièrent ainsi le moyen de commencer à influer sur leur activité.

Lorsque nous parlons d'explicitation dans ce qui précède, cela relève davantage d'une « intuition » de ce que doit être la tâche que d'une véritable explicitation. Considérons des systèmes servant de support à des tâches complexes tels que BSCW [3] dans le cadre de la mutualisation de ressources documentaires ou [10] pour le développement de logiciels en équipe. Leurs concepteurs ont avant tout songé à fournir des fonctionnalités pratiques à destination des acteurs (par exemple annotation de documents, recommandations pour le premier; découpage en sous-projets, « todo list » pour le second) plutôt qu'une description rigoureuse de la tâche. Celle-ci serait en effet très difficile voire impossible à obtenir; tout au plus peut-on en donner les grandes lignes. Il semble ici préférable de fournir les moyens d'agir sur les systèmes (ex : création de groupes et de rôles) et de les adapter en fonction des besoins.

Seuls les systèmes intégrant la technologie du workflow conduisent à une explicitation complète du modèle de la tâche. Le workflow fournit en effet un formalisme rigoureux pour le décrire et des mécanismes pour

l'automatiser. En ce sens, il s'agit d'une aide importante au travail de groupe pour la coordination des actions. A cet avantage, nous pouvons cependant opposer la difficulté d'explicitation du modèle, le manque de support au sein des groupes à la négociation de ces modèles (celle-ci est bien souvent inexistante) ainsi que la rigidité qui en découle. L'explicitation du modèle devrait cependant permettre sa modification en situation et c'est actuellement l'objectif des travaux menés dans le domaine du workflow adaptatif (voir ci-après).

2.2.3 Les collecticiels centrés « modèle »

Nous rassemblons ici l'ensemble des collecticiels qui reposent sur un modèle non plus d'une tâche spécifique à réaliser mais de l'activité conjointe en général. Ces modèles sont inspirés de travaux en sciences humaines. Ils fournissent des concepts permettant de décrire ce qu'est une activité de groupe et par là même de la construire. Mis à la disposition des usagers, ces concepts vont leur permettre de structurer leur groupe, leur espace de travail et de décrire les conditions de leur engagement dans l'activité. Il s'agit ici d'expliciter un niveau « méta » et les fonctionnalités qui vont avec pour permettre non seulement une construction conjointe de l'activité du groupe mais aussi son évolution dans le temps, au cours de l'activité elle-même.

Parmi les systèmes relevant de cette catégorie, nous trouvons des « méta-collecticiels » tels que Worlds [20] [46] fondé sur la théorie des « locaux », Prospero [13] qui s'appuie sur l'ethnométhodologie ou encore DARE [5] qui lui a recours à la théorie de l'Activité. Dans la même veine, les travaux réalisés à l'Université de Savoie proposent un modèle original (le modèle de participation) qui s'inspire à la fois de la théorie des locaux, de l'ethnométhodologie, de la linguistique, de la pragmatique et de la sémantique différentielle. Ce modèle est à la base des réalisations effectuées dans le cadre du projet de « cartable électronique » [35] qui, si elles visent un cadre applicatif particulier (celui de l'Education), voire pour certaines une tâche spécifique (le dessin entre enfants) [18], n'en fournissent pas moins un niveau « méta » et des composants et fonctionnalités associés à ce niveau, réutilisables dans d'autres applications.

3 ASPECTS IMPORTANTS

3.1 Fonctionnalités et services

Les éléments technologiques nécessaires pour supporter le travail coopératif sous toutes ses formes sont nombreux. Nous pouvons lister les suivants : distribution et synchronisation des objets partagés, accès concurrents à ces objets, gestion des droits d'accès, nommage, réplication, etc. Ne pouvant pas traiter tous ces aspects ici nous nous limiterons à la manipulation des objets par les acteurs coopérant et le contrôle de leur cohérence. En effet, les collecticiels reposent sur les mécanismes développés dans les systèmes distribués ou les gestionnaires de configurations: verrouillage des objets, passage de jeton (ou « prise de tour »), détection des dépendances (conflits résolus par les utilisateurs).

Afin de contrôler les mises à jour concurrentes effectuées par les différentes activités d'un système coopératif, il est possible d'utiliser un outil de gestion de configurations (RCS, ClearCase, Continuus, Adèle,…). Son rôle est d'assurer le stockage des données partagées, appelées ressources, tout en gardant une trace de leur évolution (généralement sous la forme de leurs versions successives) et en contrôlant les accès concurrents effectués par les activités. Par exemple, si deux activités modifient en parallèle une même ressource, elles vont chacune développer, à partir d'une version initiale de cette ressource, ce que l'on appelle une branche de versions. De cette façon, chacune des activités travaille sur sa propre copie de la ressource, sans être perturbée par les modifications effectuées par les autres. Lorsqu'elles ont terminé leur travail, une activité (éventuellement différente) est chargée de fusionner les branches afin de ne produire qu'une seule nouvelle version, sans que les modifications d'une des activités n'écrasent celles des autres. Le rôle du gestionnaire de configurations est alors de mémoriser le fait que cette nouvelle version est dérivée des précédentes.

La plupart des gestionnaires de configurations reposent sur une architecture client/serveur (référentiel centralisé, éventuellement répliqué et/ou partitionné sur plusieurs serveurs). Ils sont essentiellement concernés par les problèmes de concurrence d'accès à un référentiel : gestion des versions et des configurations de ressources partagées. Si l'on se place dans un cadre distribué dans lequel les activités réclament une certaine autonomie, on peut tirer profit d'une approche transactionnelle avec des modèles transactionnels étendus, des transactions de longues durées qui accèdent à une multitude de ressources distribuées. Le problème se « réduit » alors à ceux d'affaiblissement de l'atomicité et d'isolation, parmi les propriétés ACID (Atomicité, Cohérence, Isolation et Durabilité), pour les transactions longues, tout en assurant la cohérence et la fiabilité de l'exécution des tâches concurrentes

et en offrant des mécanismes de reprise (ou de recouvrement) après une panne, ainsi que des mécanismes de compensation permettant d'annuler logiquement les effets d'une activité.

Plusieurs travaux ont choisi l'approche transactionnelle en se basant sur des échanges via un système de gestion de bases de données ou sur un moniteur transactionnel. Parmi ces travaux, nous pouvons mentionner: [7], le projet ECOO (Environments pour la COOperation) [2][22], le projet DOM (Distributed Object Management) [21], le projet WISE (Workflow based Internet Services) [1][41], le projet WIDE (Workflow on Intelligent Distributed database Environment)[23], le projet COW (Cooperative and Open Workflow) [47], le projet TRANSCOOP (Transaction Mangement Support for Cooperative Applications) [39], etc.

Même si les techniques de base ne sont pas nouvelles, la gestion des données manipulées dans un collecticiel reste primordiale et la façon dont cette gestion sera faite influera grandement sur la perception des utilisateurs quant à la collaboration en cours.

3.2 Awareness

Lorsque des personnes travaillent en « co-présence », c'est-à-dire en même temps et dans le même lieu, elles échangent un certain nombre d'informations, certaines implicites et d'autres explicites. Ces informations créent chez chaque participant une conscience de groupe, qui lui permet :

- de comprendre et mesurer l'activité et la dynamique du groupe ;
- de situer sa propre action au sein du groupe, et ainsi de coordonner ses propres activités avec celles des autres.

Cette notion de conscience de groupe (awareness) est originellement définie par Dourish et Belloti [14] comme étant « la compréhension des activités des autres, qui permet de donner un contexte à sa propre activité ». Dans une équipe virtuelle, la disparition de la co-présence entraîne la disparition de certaines informations, particulièrement les informations implicites, qui entraîne à son tour la disparition de la conscience de groupe. Le but, dans le contexte des applications de collaboration/coopération, est donc de reconstruire la conscience de groupe grâce à des modèles et par le biais de mécanismes adaptés. Modéliser cette conscience de groupe nécessite :

- l'identification des informations utilisées en co-présence pour construire la conscience de groupe ;
- une synthèse claire et utilisable des nombreuses informations que l'on peut obtenir des systèmes informatiques ;
- une représentation à l'utilisateur d'une information pertinente mais non obstructive.

On distingue généralement trois formes de conscience de groupe : la « group awareness » a pour but de rendre tangible le travail en groupe et d'être conscient de l'état et des activités des partenaires; la « Workspace awareness » (conscience de l'espace de travail commun) se situe à un plus grand niveau de détail et permet aux utilisateurs travaillant sur les mêmes documents d'être au fait des modifications apportées par les partenaires situés dans le « même » espace de travail; enfin, la « process awareness » (conscience du procédé commun) permet aux participants de situer leurs actions dans le contexte plus large d'un projet « global » et coordonné [6].

Parmi les différents modèles proposés pour résoudre le problème de la conscience de groupe dans les outils de collaboration/coopération nous pouvons citer les modèles basés sur la publication/souscription et les modèles spatiaux. Le modèle publication/souscription peut grossièrement être décrit par : les producteurs d'information publient des évènements et des messages vers le système et les consommateurs d'information souscrivent des abonnements pour certaines catégories d'évènements déterminées soit par leur sujet, soit par leur contenu. Le modèle spatial, quant à lui, est fondé sur une métaphore spatiale et définit la conscience de groupe en terme de positions, proximité et distance à l'intérieur d'un espace virtuel commun.

Parmi les infrastructures permettant l'implantation de la conscience de groupe dans les systèmes coopératifs, nous pouvons citer Elvin [17] (développé par DSTC), Gryphon [25] (IBM) et Nessie [38] (GMD - Fraunhofer). Gryphon et Elvin utilisent la publication / souscription sur des données différentes : sujets et contenus pour le premier, contenus pour le second. Elvin vise par ailleurs au passage à l'échelle. Il fonctionne avec de très grands nombres de messages et d'utilisateurs et a servi au développement de collecticiels comme TickerTape et Orbit. Nessie, quant à lui, permet la notification synchrone ou asynchrone d'évènements qui sont présentés de différentes façons à l'utilisateur (popup, animation, sons,…).

3.3 Workflow

La coordination sociale s'effectuant sous la responsabilité exclusive des acteurs ayant montré ses limites, il s'agit d'utiliser l'ordinateur pour cette activité ou au moins partager cette activité entre l'ordinateur et les acteurs. Le Workflow constitue une réponse technologique à la gestion des processus. Malheureusement, s'il est facilement utilisable dans les processus très rigides, la souplesse et la flexibilité ne sont pas les points forts des outils actuellement disponibles. Dans la coordination de nombreux processus coopératifs, la rigidité n'est pas acceptable. Il est en effet nécessaire de permettre à la fois des interventions opportunistes au sein d'un processus globalement structuré et de respecter une certaine rigueur dans les différents sous-processus. Les travaux sur les Workflows adaptables tentent de proposer des approches permettant au Workflow de pouvoir être utilisé dans la coordination au sein des collecticiels synchrones [40].

4 VERS DES LOGICIELS DE TCAO EVOLUTIFS

4.1 Malléabilité et composants

Depuis déjà plusieurs années, le domaine de recherche sur le TCAO a identifié la malléabilité [37] des systèmes comme un besoin important pour toute application ou plate-forme [30]. Une première approche architecturale de la malléabilité assez répandue est celle liée aux composants et aux architectures cadres (frameworks) [3][26][27]. Le composant est l'unité de base réutilisable, évolutive et compréhensible supportant la malléabilité alors que l'architecture cadre fournit l'ossature de l'application. Ainsi, cette démarche offre un moyen simple d'évolution de l'application par la manipulation de sa structure au niveau d'abstraction des composants.

Néanmoins, si cette approche apparaît comme une solution intéressante, elle est loin de résoudre tous les problèmes liés à la mise en œuvre de la malléabilité. Il reste encore à déterminer les mécanismes d'évolution de ces derniers. Ces différents mécanismes ont été étudiés par Anders Morch [36] qui les définit en termes de paramétrage, d'intégration et d'extension. Le paramétrage offre une malléabilité limitée par un nombre de choix prédéterminés, comme par exemple choisir un composant parmi une liste figée de N. L'intégration consiste à greffer un

nouveau composant dans l'architecture de l'application. L'extension revient à modifier le code du composant lui-même. Le paramétrage, l'intégration et l'extension offrent de plus en plus de malléabilité mais demandent à son utilisateur de plus en plus grandes compétences en informatique. Ce point explique en partie pourquoi la plupart des systèmes de TCAO actuels orientent généralement leur malléabilité vers des utilisateurs développeurs ou experts du système plutôt que vers les utilisateurs finaux qui, paradoxalement et d'après les recherches en sciences humaines, sont ceux qui en ont le plus besoin [31].

4.2 Autres approches de la malléabilité

L'ensemble de ces problèmes et solutions ont été approfondis et pris en compte dans la création de l'architecture du système DARE développé par G. Bourguin [5]. Ce système est fondé sur la Théorie de l'Activité (TA) et veut offrir une malléabilité accessible aux utilisateurs finaux dans la démarche nommée Co-évolution [4] (voir ci-après). DARE est défini comme un collecticiel réflexif (au sens de Maes [33]) et repose sur une approche de type Implémentation Ouverte (IO) de Kickzales [28].

C'est ce même souci de donner la main aux utilisateurs finaux qui a conduit C. Martel à proposer un espace dédié à l'activité de configuration, reconfiguration, définition et modification du système [34]. En manipulant les concepts du modèle qui structure cet espace (acteurs, rôles, scénarios notamment), les utilisateurs ont la possibilité de décrire leur mode de participation à l'activité ainsi que les conditions du travail en groupe. Une architecture fondée sur ce modèle a été proposée et une implantation dans deux applications a permis de valider en partie le modèle [19][35].

Une autre approche de malléabilité, appelée flexibilité [44] a été mise en place dans la proposition de l'architecture AMF [42] pour les systèmes interactifs et dans sa version AMF-C [43] pour les systèmes coopératifs. Le modèle AMF-C se caractérise par l'approche multi-facettes qui conduit à identifier et mettre sous forme de pattern tout comportement stéréotypé. La possibilité de définir de nouvelles facettes chaque fois qu'une nouvelle fonctionnalité se stabilise, constitue un atout fondamental d'AMF-C. Les liens entre facettes qui expriment le « contrôle » sont présentés graphiquement à l'utilisateur, qui peut par leur manipulation choisir dynamiquement par exemple une présentation alternative, un autre comportement applicatif ou rendre la propagation des modifications plus ou moins globale (à aucun, à un certain nombre ou à la totalité des acteurs de la coopération).

4.3 La co-évolution

Notre vision est que le système coopératif produit doit pouvoir s'adapter aux besoins émergeants, à la transformation de besoins ayant servis à l'analyse initiale, ainsi qu'aux évolutions des contextes d'usages et des technologies utilisées. Ces systèmes peuvent être vus à double titre comme une intermédiation (un passage plus qu'une séparation) entre les concepteurs et les usagers:

- intermédiation entre les besoins, compétences, désirs… des usagers et les contraintes, technologiques ou autres, les compétences et les désirs des concepteurs ;

- intermédiation entre les deux parties pour en assurer la maintenance et l'évolution dans la durée. Cela implique, nous le justifierons après, que le système coopératif doit contenir sa propre représentation (un méta-modèle) et être capable de réflexivité (introspection et intercession).

Nous pensons, utilisant pour cela l'expérience acquise au sein de la communauté du TCAO, que cette transformation continue a une dimension collective et historique. C'est au sein de la communauté de pratique, au sens de la sociologie des usages, que les artefacts informatiques proposés doivent être adaptés afin de refléter la cristallisation des compétences et des attitudes de ses membres.

Le terme co-évolution a été choisi pour traduire le fait que les systèmes coopératifs doivent être continûment évolutifs, sans l'être toutefois de façon autonome ou auto-adaptative, car ils doivent rendre compte, de manière consciente, des évolutions des besoins, des attitudes et des compétences des usagers, individuellement ou collectivement. Mais il n'y a pas que le système qui doive évoluer, l'usager doit également transformer, adapter, ses pratiques et ses méthodes de travail, c'est-à-dire son rôle, pour satisfaire les besoins évolutifs de l'organisation [48]. C'est une vue plus large que la boucle « tâche/artefact » proposée par Carroll et Rosson [9] pour essayer de traduire la relation dialectique qui existait entre la spécification des besoins au travers de la tâche et l'artefact lui-même résultant du processus de conception. La co-évolution traduit tant les ajustements continus, négociés et socialement situés, des pratiques de la part des individus, que les ajustements apportés au comportement du système interactif considéré. Ceci peut-être synthétisé par le schéma donné dans la figure ci-après, où il apparaît que l'environnement global fixe le contexte de la tâche [29].

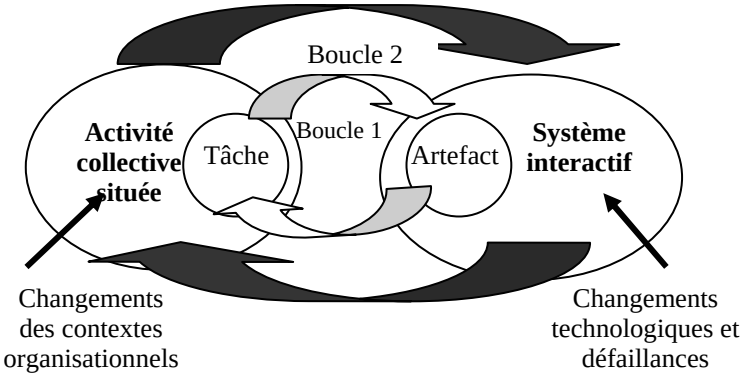


Illustration de la co-évolution

5 TENDANCES ACTUELLES

Après les développements d'applications spécifiques, puis de systèmes collaboratifs, une démarche plus industrielle se met en place en proposant des plates-formes intégrant des services et composants dédiés au travail de groupe. De nouvelles applications peuvent ainsi être plus rapidement développées en utilisant ces briques de base.

Les collecticiels actuellement développés tendent vers la co-évolution. Il s'agit de réaliser des plates-formes de travail de groupe, dans lesquelles les acteurs définissent eux-mêmes les modalités de l'organisation du groupe, son fonctionnement, les caractéristiques de l'espace partagé ainsi que les outils qu'ils souhaitent utiliser. L'espace de travail et ses lois sont ainsi configurés par les usagers eux-mêmes et ils évoluent de manière dynamique, au cours de l'activité elle-même. Cet idéal est encore loin cependant d'être atteint, la reconfiguration dynamique nécessitant bien souvent des compétences en informatique ou une connaissance des modèles théoriques sous-jacents que nous ne sommes pas en droit d'attendre des usagers. Elle bute de plus sur des verrous technologiques.

Par ailleurs, l'évolution actuelle du TCAO vers une plus grande disponibilité des acteurs passe entre autres par la prise en compte de leur mobilité et de l'utilisation de dispositifs légers. Cette approche est

qualifiée de TCAO capillaire par [12]. Le TCAO capillaire répond à un fort besoin émis par des utilisateurs qui ne travaillent plus seulement dans des bureaux devant des postes fixes. En plus de l'accès aux informations, de l'interaction et de la collaboration par la virtualisation de l'espace et du temps, il ajoute la mobilité des personnes, leur plus grande disponibilité et la contextualisation de leurs actions et informations grâce à la prise en compte du contexte précis, de leur localisation et de la location des objets sur lesquels portent le travail.

6 CONCLUSION

Dans cet article nous avons explicité l'évolution du travail coopératif. Nous en avons affiné la définition et décrit les principales caractéristiques. Nous avons pu constater le besoin d'allier flexibilité, robustesse et généricité pour répondre de façon satisfaisante aux nouvelles exigences liées entre autres à la prise en compte du contexte logique et/ou géographique obtenu par les capacités nouvelles des dispositifs nomades. Nous avons mis en évidence le besoin d'une nouvelle approche du cycle de vie des systèmes coopératifs qui prennent en compte la gestion, par les usagers en partie, des changements à opérer sur le système lui-même pour satisfaire les évolutions des tâches des usagers.

7 REFERENCES

[1] Alonso G., Hagen C., Lazcano A. Process in Electronic Commerce, ICDS workshop on Electronic Commerce and Web-Based Applications, Austin, Texas, USA, June 1999.

[2] Baïna K., Benali K., Godart C. A process service model for dynamic enterprise process interconnection, Proceedings of 6th International Conference on Cooperative Information Systems (CoopIS'2001), in cooperation with VLDB'2001, Trento, Italy, 2001, LNCS 2172.

[3] Bentley, R., Appelt W., Busbach U., Hinrichs E., Kerr D., Sikkel K., Trevor J., Woetzel G. Basic support for cooperative work on the World Wide Web.International Journal of Human-Computer Studies Vol. 46, pp. 827-846, 1997.

[4] Bourguin G., Derycke A., Tarby J.C. Beyond the Interface: Co-evolution Inside Interactive Systems – A proposal Founded on Activity Theory, Springer Verlag, People and Computer vol. 15 – Interaction without

Frontiers, proceedings of Human Computer Interaction 2001 (HCI'2001), Blandford, Vanderdonckt, Gray (eds), pp. 297-310

[5] Bourguin, G. Un support informatique à l'activité coopérative fondé sur la Théorie de l'Activité : le projet DARE, Ph. D. Thesis, Informatique, n° 2753, Université des Sciences et Technologies de Lille, France, 210 p., 2000

[6] Bouthier C., Canals G. Le contexte comme base de la conscience de groupe, CITE, Troyes, France, Novembre 2001.

[7] Breitbart, A. Deacon, H-J. Schek, A. Sheth, G. Weikum, Merging Application-centric and Data-centric Approaches to Support Transaction-oriented Multi-system Workflows, SIGMOD Record, pp 23-30, Vol 22, N° 3, 1993

[8] Byzance http://www.inrialpes.fr/opera.html

[9] Carroll J., Rosson, M. Getting around the Task-Artefact Framework: How to Make Claims and Design by scenario, ACM Transaction On office Information Systems, vol. 10 (2), 181-212. 1992.

[10] Charoy F., Godart C., Molli P., Oster G., Patten M., Valdes M. Services For Virtual Teams Hosting. ToxicFarm Introduction. Second International Workshop on Cooperative Internet Computing (CIC), August 18-19, 2002, Hong Kong, Kluwer Academics.

[11] David B. IHM pour les collecticiels. In Réseaux et Systèmes Répartis. Hermès, Paris, vol. 13, novembre 2001, pp. 169-206. ISBN 2-7462-0303-0.

[12] David B., Vaisman G., Saikali K. Evolution du Travail Coopératif Assisté par Ordinateur : Vers le TCAO « capillaire », CITE'2001, Coopération, Innovation et Technologie. 29 et 30 Novembre 2001. Université de Technologie de Troyes. Novembre 2001.

[13] Dourish P. Using Metalevel Techniques in Flexible Toolkit for CSCW Applications, ACM Transaction on Computer-Human Interaction, vol. 5, n°2, pp. 109-155. 1998.

[14] Dourish P., Bellotti V. Awareness and Coordination in Shared Workspaces, Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW-92), Toronto, Ontario, 1992, ACM Press, p. 107-114.

[15] Ellis C.A., Gibbs S.J., Rein G.L. Groupware: some issues and experiences, Communication of ACM, 34(1), January 1991, pp. 39-58.

[16] Ellis C., Wainer J., A conceptual model of Groupware, In Proc. CSCW'94, ACM Press, p. 79-88., 1994.

[17] Elvin – http://elvin.dstc.edu.au.

[18] Ferraris C., Martel C. Regulation in groupware: the example of a collaborative drawing tool for young children. Proceedings of CRIWG'2000, 6th international workshop on groupware, Madeira, Portugal, pp. 119-127. 2000.

[19] Ferraris C., Brunier P, Martel C**.** Constructing collaborative pedagogical situations in classrooms: a scenario and role based approach. Computer Support for Collaborative Learning 2002 (CSCL 2002), Boulder, Colorado, USA, 7-11 January 2002.

[20] Fitzpatrick G., Tolone W. J., and Kaplan S. M. Work, Locales and Distributed Social Worlds. Proceedings of the ECSCW'95, Stockholm, September, pp. 1-16. 1995.

[21] Georgakopoulos D., M. F. Hornick, F. Manola, M. L. Brodie, S. Heiler, F. Nayeri, B. Hurwitz, " An Extended Transaction Environment for Workflows in Distributed Object Computing", IEEE Data Engineering Bulletin, Vol.16, N° 2, June 1993

[22] Godart C., Perrin O., Skaf H., COO : a workflow operator to improve cooperative modelling in virtual processes", 9th International Workshop on Research Issues on Data Engineering : Information Technology For Virtual Enterprises (RIDE-VE'99), March 1999, Sydney, Australia

[23] Grefen P. Advanced Architectures for Transactional Workflows or Advanced Transactions in Workflow Architectures, International Process Technology Workshop (IPTW'99), 1999.

[24] Grudin J. CSCW: History and Focus. In Computer, 29(6), IEEE Computer Society, June 1992, pp. 27-35.

[25] Gryphon – http://www.research.ibm.com/gryphon

[26] Hoogstoel F. Une approche organisationnelle du Travail Coopératif Assisté par Ordinateur. Application au projet Co-Learn, Th. de Doctorat en Informatique, Université des Sciences et Technologies de Lille, 1995, n° 1487

[27] Hummes J., Merialdo B. Design of extensible component-based groupware, soumis pour être publié par Kluwer dans le Journal of CSCW, 1999, http://www.eurocom.fr/~hummes/docs/JCSCW/JCSCW.html

[28] Kiczales G., Bobrow D.G., Des Rivieres J. The Art of the Metaobject Protocol, MIT Press, August 1991, 335 p

[29] Kirsh, D. Distributed Cognition, Coordination and Environment Design. In Proceedings Of the European Cognitives Sciences Society (htttp://icl-server.ucsd.edu/~kirsh/articles/italy/published.html) 1999.

[30] Koch M. Teege G., Support for tailoring CSCW systems: adaptation by composition. Seventh Euromicro Workshop on Parallel an Distributed Processing PPD'99, 1999, pp 146-153.

[31] Kuutti K. The concept of activity as a basic unit of analysis for CSCW research, Proceeding of the second ECSCW'91 conference, Kluwers Academics Publishers, 1991, pp 249-264.

[32] Lewkowicz M., Zacklad M. A structured groupware for a collective decision-making aide.European Journal of Operational Research 136, 2002, pp.333-339.

[33] Maes P. Computational Reflection, Ph.D. Thesis, V.U.B, Brussels, 1987.

[34] Martel C., Ferraris C., De la régulation dans les collecticiels**,** Actes de IHM99, Montpellier 1999.

[35] Martel C., Vignollet L Educational Web Portal based on personalized and collaborative services. Proceedings of ICALT (International Conference on Advanced Learning Technologies), Madison, USA, August 6-8. 2001, http://www.cartable-electronique.fr,

[36] Morch A. Three levels of end-user tailoring: customization, integration, and extension, dans [37], 1995, pp 41-51

[37] Morch A. Method and Tools for Tailoring of Object-oriented Applications: An Evolving Artifacts Approach, part 1, Dr. Scient. Thesis Research Report 241, University of OSLO, Department of Informatics, 1997

[38] Nessie - http://orgwis.gmd.de/projects/nessie

[39] Puustjärvi J. Transactional Workflows, Department of Computer Science, University of Helsinki, Finland, 1999

[40] Saikali K., David B. Vers l'usage du Workflow pour la coordination dans les collecticiels. In Actes de IHM-HCI 2001, Lille, France, septembre 2001. ISBN 1-85233-515-7

[41] Schuldt H., Schek H.J., Alonso G. Transactional Coordination Agents for Composite Systems, Proceedings of the International Database Engineering and Applications Symposium, (IDEAS'99), Montreal, Canada, August 1999

[42] Tarpin-Bernard F., David B. AMF : un modèle d'architecture multi-agents multi-facettes. Techniques et Sciences Informatiques. Hermès. Paris. Vol. 18. No. 5. p. 555-586. Mai 1999.

[43] Tarpin-Bernard F., David B.T., Primet P. Frameworks and patterns for synchronous groupware: AMF-C approach, In Proceedings of EHCI'98, IFIP Working Conference on Engineering for HCI, Greece, Sept. 1998, 13 p.

[44] Tarpin-Bernard F. La flexibilité dans les collecticiels, Ecole thématique Documents & évolution du GDR I3, Tome 2, Cépaduès, 2000.

[45] Tewissen F., Lingnau A., Hoppe U., Mannhaupt G., Nischk D. Collaborative Writing in a Computer-integrated Classroom for Early Learning. Proceedings of Euro-Cscl 2001, Maastricht, Netherlands, Mars 22-24, 2001

[46] Tolone W. J. Introspect: a Meta-level Specification Framework for Dynamic, Evolvable Collaboration Support. Ph.D. Thesis. University of Illinois at Urbana-Champaign, 1996.

[47] Vantroys V., Peter Y. A WMF-Based workflow for e-learning, European Research Seminar on Advances in Distributed Systems, Advanced School and Workshop (ERSADS'01), Bologna, Italy, May 2001

[48] Vicente, K. J. HCI in the Global Knowledge-Based Economy: Designing to Support Worker Adaptation, ACM Transactions on Computer-Human Interaction, vol. 7, n°2, 263-280, 2000.