Coordination explicite d'activités coopératives

François Charoy

Université Henri Poincaré – Nancy 1

Département de formation doctorale en informatique **École doctorale IAEM Lorraine**
UFR STMIA

Coordination explicite d'activités coopératives

MÉMOIRE

présentée et soutenue publiquement le 23 juin 2008

pour l'obtention de l'

Habilitation à diriger des recherches de l'université Henri Poincaré – Nancy 1

(spécialité informatique)

par

François Charoy

Composition du jury

Rapporteurs :	Christine Collet
	Michel Léonard
	Françoise Détienne
Examinateurs :	Olivier Festor
	Claude Godart

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

Mis en page avec la classe thloria.

Remerciements

La liste des personnes en qui je suis redevable est devenue trop importante pour que je puisse tous les citer ici.

Je commencerai cependant par remercier les membres du jury qui m'ont fait l'honneur de bien vouloir juger mon travail et d'assister à ma soutenance.

Le travail universitaire est une oeuvre collective. Celui-ci doit beaucoup à tous les collègues et amis, avec qui j'ai eu l'occasion de travailler, d'échanger, de me confronter parfois au cours des années passées à l'Université, que ce soit au LORIA, dans l'équipe Genielog puis ECOO, à l'IUT Nancy-Charlemagne au département Info-Com, ou depuis peu à l'ESIAL. Je tiens également à les remercier ici sincèrement.

Table des matières

Chapitre 1

Curriculum Vitae

- Date et lieu de naissance : le 20 décembre 1964 à Nancy
- Nationalité : française
- Situation de famille : Vie Maritale
- Adresse : 47, rue des tiercelins, 54000 Nancy
- Fonction : Maitre de conférences

1.1 Titres universitaires français

1.2 Expérience professionnelle (10 ans)

Septembre 2005 - Maitre de conférence à l'ESIAL, Université Henri Poincaré Nancy 1

Septembre 2002 - Septembre 2005 Maitre de conférence à l'IUT Nancy-Charlemagne, Université de Nancy 2

Septembre 2000 - Aout 2002 En délégation à l'INRIA dans le projet ECOO

Septembre 1993 - Aout 2005 Maitre de conférence à l'IUT Nancy-Charlemagne, Université de Nancy 2

1.3 Principaux résultats de recherche

Depuis plus de dix ans, je m'intéresse au problème de la coordination d'activités coopératives, ou comment faire pour aider des personnes distribuées dans le temps et dans l'espace à coopérer de façon efficace. L'objectif final est fournir un cadre à des individus et/ou à des systèmes leur permettant d'atteindre un objectif commun de façon coordonnée et sûre.

Ce problème a plusieurs dimensions.

Il a d'abord une dimension humaine et sociale. Il faut essayer de comprendre ce qu'est le travail en groupe, comment se fait la coordination des activités, comment cette coordination peut être assistée et quels en sont les enjeux. Cette dimension n'est pas au coeur de notre travail mais elle en donne les directions. En effet, la coordination du travail coopératif ne peut se faire sans prendre en compte les méthodes, les besoins et les aspirations des personnes qui participent à ce travail. Cela nécessite de prendre en compte et de réaliser des analyses d'usage d'outils

coopératifs et en particulier d'outils supports à la coordination. La difficulté ici réside dans le fait que le travail de groupe peut correspondre à des situations, des conditions et des contextes très différents.

Un groupe de deux ou trois personnes n'a pas les mêmes besoins ni les mêmes contraintes qu'un groupe de 15 ou qu'une communauté de plusieurs centaines ou milliers de personnes. La nature des activités a également un impact sur les besoins. Coordonner l'écriture d'un article est différents de la coordination du développement d'un logiciel, de celle d'une équipe médicale lors d'un traitement, de celle d'un ensemble d'organisation répondant à un appel d'offre ou de la rédaction d'une encyclopédie. D'autre conditions ont également un impact : l'histoire commune et la confiance entre les personnes collaborant, leur dispersion géographique, l'intérêt qu'elles ont a aboutir au résultat.

Il a ensuite une dimension qui s'attache aux modèles et aux concepts du travail collaboratif. Nous essayons de définir et de formaliser des modèles qui permettent de représenter puis de contrôler les processus mis en oeuvre dans une activité coopérative. Pour cela nous avons exploré deux approches. Une de ces approches est fondée sur la logique temporelle dont le but était d'exprimer des contraintes sur les états produits par les interactions. L'autre approche est fondée sur le modèle de procédé pour permettre de décrire les activités d'un procédé collaboratif et ses dépendances, tout en laissant suffisamment de flexibilité aux participants pour l'adapter au cours de son exécution.

Une dernière dimension s'attache à la sureté de l'exécution des procédés et des échanges de données pendant cette exécution. Notre but est de fournir des garanties sur l'exécution des procédés (atomicité, sérialisation, isolation, contrôle d'accès) tout en respectant les contraintes particulières du travail coopératif (flexibilité, adaptabilité, évolutivité). Pour cela, nous avons travaillé à la fois sur la flexibilité de l'exécution avec comme direction principale, la possibilité pour les utilisateurs de définir les contraintes d'exécution d'un procédé de façon localisée et non de façon globale (pour tout le procédé) et sur la flexibilité des modèles avec comme direction le fait que dans un procédé coopératif, les utilisateurs doivent être maitre de sa définition.

Ces considérations, qui ont bien sur évoluées dans le temps nous ont permis de conduire différents types de travaux et d'obtenir différents résultats théoriques et pratiques intégrant ces différentes dimensions.

1.3.1 Modèle hybride de gestion des interactions coopératives

La première approche que nous avons développée consistait à considérer l'évolution de l'état des objets dans un environnement coopératif. En partant du principe que la coopération qui nous intéressait était celle qui impliquait des manipulations sur des données partagées, nous avons proposé d'utiliser la logique temporelle pour contraindre les changements d'état de ces objets dans le temps. En nous appuyant sur un algorithme de transformation d'un ensemble de formules de logique en automate déterministe, il est possible d'autoriser ou non des changements d'états de l'ensemble des objets partagés. Ces contraintes dynamiques ont l'avantage de permettre de contrôler l'avancement d'un procédé coopératif sans définir formellement ce procédé. Avec cette méthode, il est possible de décrire l'état du système à atteindre, les états du système intermédiaire par lesquels il faut passer, sans faire d'hypothèse sur les outils à utiliser ou les activités à exécuter pour atteindre cette état. C'est le résultat principal de cette approche.

Elle a un avantage important : elle permet le contrôle sans imposer de processus. Elle peut ainsi éviter à des utilisateurs de terminer un projet coopératif (atteindre un état de satisfaction final) sans passer par des états intermédiaires (par exemple, il est facile d'exprimer le fait qu'un article doit toujours passer par l'état revisé avant de passer à l'état publié).

Cette approche a deux inconvénients majeurs. Le premier est la difficulté à exprimer toutes les contraintes que l'on voudrait voir apparaitre sous la forme d'états du système. Il faut souvent étendre le système pour ajouter des informations représentant des actions des utilisateurs. Le second concerne la difficulté d'expression des contraintes en logique temporelle ainsi que leur difficulté d'interprétation par un utilisateur. Si le nombre de contraintes augmente, il devient difficile de comprendre ce qu'il faut faire pour produire les états corrects successifs. Les outils fournis par la logique temporelle ne sont pas d'une grande aide dans ce cas.

Ces raisons nous ont amené à utiliser une nouvelle approche, permettant de décrire plus explicitement les activités à mener pour atteindre un objectif. Cette approche consiste à adapter les système de gestion de processus ou système de gestion de workflow à des activités coopératives.

Publications sur cette partie : [30, 32, 31]

1.3.2 Flexibilité de l'exécution des procédés coopératifs

Depuis une quinzaine d'années, de nombreux travaux ont été menés pour modéliser, automatiser et contrôler l'exécution de processus de services, par analogie aux plus classiques processus industriels. La première étape fut la formalisation des processus sous différentes formes pour ensuite pouvoir les contrôler et les exécuter. Les limites de cette approche sont apparues très rapidement en raison de la plasticité de la mise en oeuvre des processus de services. Entre la définition des processus, perçus par les analystes métiers et leur exécution dans un contexte réel, les différences, divergences et exceptions peuvent être nombreuses. C'est d'autant plus vrai que les processus sont complexes et concernent des activités créatives. Ces constatations ont eu deux effets différents. L'abandon par une partie de la communauté des travaux de recherche dans cette direction et la mise en oeuvre par une autre partie de cette communauté de modèles supportant une plus grande flexibilité dans la modélisation et l'évolution de la définition des processus. Nous avons pris une voie un peu différente en considérant la flexibilité au niveau de l'exécution plutôt qu'au niveau des modèles. Cette approche se fonde sur les hypothèses suivantes :

- Dans un contexte coopératif, il est difficile de modéliser l'ensemble du processus à l'avance. En outre, la définition de ce processus est susceptible d'évoluer au cours de son exécution.
- Même lorsqu'on est capable de modéliser le processus, il est nécessaire de pouvoir prendre, pour des raisons d'efficacité, des libertés avec son exécution.

A partir de ces hypothèses, nous avons proposé un modèle inspiré des langages à base de prototype, en renonçant au schéma traditionnel, définition du procédé/instance du procédé : une instance de procédé est définie au cours de son exécution. Elle peut ensuite être clonée pour créer de nouvelles instances par "analogie". Ces nouvelles instances peuvent à nouveau évoluer en fonction des besoins particuliers du nouveau procédé.

L'exécution du procédé devient elle-même flexible grâce au mécanisme d'anticipation qui permet de démarrer l'exécution d'une activité, même si toutes les conditions pour son exécution ne sont pas remplies. Le modèle garantit cependant qu'à un moment, ces conditions seront remplies avant de pouvoir terminer l'exécution de l'activité. Une trace de l'exécution à la fin du procédé donnera l'impression que tout s'est déroulé normalement même si certaines activités ont démarré en avance.

La combinaison d'un modèle dynamique, associé à un mécanisme d'exécution flexible nous est apparu comme une voie nécessaire à l'adoption des technologies de type workflow dans un contexte coopératif.

Une des critiques que l'on peut faire à cette approche est sans doute sa "trop grande" flexibilité. En effet, un processus "complétement" coopératif n'est pas la règle. En général, pour atteindre un objectif, un groupe va alterner des périodes où la gestion des activités est rela-

tivement libre et d'autres où le besoin de contrôle et d'assurance est plus important que ce soit au niveau du modèle ou de l'exécution. Certaines activités ne peuvent pas être anticipées (publier un article par exemple). Certaines parties d'un procédé ne doivent pas pouvoir être changées dynamiquement (les phases de validation ou les phases imposées par une certification par exemple). Pour permettre une réelle flexibilité dans la définition et l'exécution des procédés et pour permettre un passage fluide de phase coopératives à des phases plus contrôlées, nous proposons de séparer la définition du modèle de celle des contraintes d'exécution. Il doit être possible de spécifier différentes contraintes comportementales pour un procédé au cours de son exécution. Cette spécification doit en outre pouvoir être faite indépendamment du modèle de procédé lui même.

Publications sur cette partie : [21, 15, 17, 16, 19, 18, 20]

1.3.3 Sphères de comportements

Un procédé coopératif n'est pas une entité homogène dont toutes les activités doivent respecter les mêmes conditions d'exécution. Cette constatation est valable également pour des procédés plus classiques. Ainsi, en gardant le même modèle pour la définition d'un procédé, on peut imaginer des modes d'exécution différents pour toute ou partie des activités de ce procédé. Ces modes, ou ces contraintes d'exécution peuvent s'appliquer à l'ensemble du procédé, à une activité ou à un sous ensemble des activités du procédé. Cette constatation nous a amené à penser qu'il devait être possible de trouver une approche générale, adaptable à différents types de contraintes qui devait permettre de spécifier les modes et les contraintes indépendamment du modèle du procédé. C'est ce que nous avons appelé les sphères de comportement. Pratiquement, une sphère est un sous-ensemble des activités d'un procédé. En fonction du type de sphère, on pourra attacher des contraintes particulières à l'exécution des activités faisant partie de la sphère. Il faut noter ici que la contrainte s'applique à la sphère et non aux activités individuellement.

Nous avons ainsi identifié un certain nombre de propriétés qui peuvent s'appliquer à un sous ensemble des activités d'un procédé.

- les sphères d'atomicité, de compensation, d'isolation qui contrôle les aspects transactionnels de l'exécution d'un procédé.
- les sphères liées à l'organisation de l'exécution (distribution des roles, des affectations des taches, du temps d'exécution)
- les sphères liées aux règles d'exécution ou d'évolution.

Une partie du travail dans ce domaine a consisté à appliquer à des sous-ensembles d'activités de processus des règles appliquées habituellement à l'ensemble des processus. Nous nous sommes en particulier intéressé aux aspects transactionnels (atomicité, isolation, compensation).

Publications sur cette partie : [22, 24, 23, 25, 26]

1.3.4 Une idée des perspectives

Ces différents travaux nous ont permis de progresser dans la compréhension du problème et dans la proposition de solutions permettant une coordination adaptée au travail coopératif. Cette question reste cependant largement ouverte pour plusieurs raisons.

La première concerne la compréhension des modes de travail coopératif et des besoins de coordination qui ne peuvent être considérés de façon monolithique. En outre, l'évolution des moyens de communication (mobile, ubiquitaire) et des moyens de productions (coopératifs, massivement collaboratifs) modifie le contexte, les attentes et les usages. Pour progresser dans ce domaine, nous proposons d'étudier les besoins et les modes de coordination dans des conditions extrèmes

(crises, santé) ou ayant des contraintes particulières (e-gouvernement, e-justice). Dansle domaine de la gestion de crise, des plans et des procédures de gestion existent préliminairement à la crise mais ces plans et ces procédures peuvent être remises en cause en permanence en raison de l'évolution des évênements. En outre, ce type d'activité implique des besoins fort en terme de communication, de conscience de groupe et de partage d'information et mettent en oeuvre de façon très dynamique des organisations ou des compétences très différentes. D'autres dimensions auxquelles nous nous intéressons sont à prendre en compte comme la sureté de fonctionnement et la sécurité qui peuvent être également critique.

Ces différents domaines ont des contraintes en terme de flexibilité, de réactivité et d'adaptabilité qui peuvent varier de façon importantes dans le temps et l'espace. Les travaux récents de Jakob Bardram sur l'Activity Computing nous conforte dans l'idée que des études sont nécessaires pour réconcilier la vision formelle d'activité ou de tâches présente dans la gestion de procédés et les formes d'activités, d'actions ou d'opérations que nous menons chaque jour, consciemment ou inconsciemment pour atteindre nos objectifs personnels ou collectifs.

La seconde raison concerne l'intégration des procédés dans l'architecture globale des systèmes d'information d'entreprise. Les systèmes de gestions de procédés métiers vont être au coeur du fonctionnement des organisations et même des relations entre organisations. Ceci implique de nouvelles contraintes liées aux relations de confiance et aux problèmes de sécurité pour les organisations qui vont collaborer pour exécuter et rendre un service. Des exemples critiques sont déjà à l'oeuvre dans le domaine de l'e-justice par exemple au niveau européen ou des procédures débordent le cadre des juridiction nationales et nécessitent à la fois une collaboration entre administrations mais également la garantie pour chacun de protéger ses procédures et ses données. Flexibilité et sureté de fonctionnement, coopération et sécurité sont des notions qui sont souvent difficiles à concilier. Ces problèmes sont au coeur de l'intégration des procédés dans un urbanisme informationnel complexe et hétérogène.

La dernière raison est liée aux deux premières et concerne les conditions d'exécution des procédés et les moyens de leur contrôles. L'architecture des futurs systèmes de gestion de procédés devra prendre en compte les nouveaux modes de travail et d'organisation émergents et s'y adapter. Il est difficile d'envisager pour les raisons évoqués précédemment un système de gestion de la coordination global et centralisé pour contrôler et distribuer l'exécution d'activités qui concernent des organisations différentes et des personnes mobiles et pouvant utiliser des terminaux variés durant leur journée de travail. D'une part, un utilisateur devra pouvoir collationner les informations concernant son activité en provenance de plusieurs sources potentielles et d'autre part, le contrôle de l'exécution d'un procédé ou de ses parties (on peut parler ici aussi de service) ne pourra pas être centralisé. De la même façon qu'un travail important a été effectué dans le domaine des bases de données sur la réplication, la répartition et la distribution des informations, ce même travail doit être fait dans le domaine des procédés.

Les perspectives de mon travail sont donc triples :

- la compréhension des modes de travail coopératifs et les moyens de supporter leur coordination en particulier dans des contextes critiques.
- l'intégration des contraintes organisationelles et informationnelles pour coordonner des activités entre organisations.
- le support pour permettre un contrôle distribué de leur exécution garantissant à la fois la sureté de fonctionnement du système et la sécurité au niveau de l'accès au données et surtout aux activités.

1.4 Réalisations logicielles

Mon travail s'est appuyé sur plusieurs développements logiciels dont le but était à la fois de valider pratiquement les idées proposées et de les expérimenter. En effet, dans le domaine du travail coopératif, il est difficile de progresser sans tester dans des conditions réalistes et/ou expérimentales les idées proposées. Ces développements ont été de différentes natures. Certains sont de simples prototypes dont le but était de faire la preuve des concepts proposés (Motu, Corvette). D'autres ont l'objectif plus ambitieux d'être utilisés dans des contextes réels en vue d'expérimentation (Coopéra) et même de mise en production (Bonita).

1.4.1 Motu

Le prototype Motu permet à une équipe virtuelle de s'organiser et de travailler. Il se situe dans la lignée d'outils comme BSCW, TeamScope ou SourceForge avec une attention particulière sur la coordination d'une équipe virtuelle avec comme domaine d'application l'architecture et le batiment qui sont fortement demandeurs dans ce domaine.

Il intègre des services de partage d'objets, de communication, de gestion de tâche et d'awareness. Le partage d'objets permet la stockage de fichiers multiversionnés sécurisés par un système de contrôle de droits d'accès et un système de contrôle de la concurrence fondé sur des transactions coopératives.

Dans le prototype Motu, nous avons mis en oeuvre la partie gestion de procédés flexibles. Ce module nous a permis de faire la preuve de concept des propositions que nous avons faites sur ce type de workflow et de commencer le travail sur Bonita.

Publications : [8]

1.4.2 Corvette

Le logiciel Corvette a été développé dans le cadre d'une coopération provée avec Hitachi R&D à Kawasaki (cf. 1.5.1). Il s'agit d'un prototype de système de workflow coopératif obtenu en couplant le logiciel de workflow commercial WorkCoordinator de chez Hitachi au modèle de transactions coopératives de Motu.

Ce prototype a été développé en Java.

Ce système a permis de montrer qu'il était possible d'intégrer le protocol de transactions coopératives de Coo avec un moteur de workflow flexible.

Publications [2, 1]

1.4.3 Coopéra

La plateforme Coopera est une dérivation de la plateforme de travail coopératif ToxicFarm. Elle est le résultat de deux années de travail avec dans le cadre du projet RIAM Coopéra pour tenter d'adapter ToxicFarm à un public de jeunes élèves et promouvoir le travail coopératif inter-écoles. Cette plateforme a été développée en PHP/MySQL. Elle intègre les résultats de l'équipe Ecoo dans le domaine du partage d'objets et de la coordination. Le modèle de coordination implanté est un modèle de workflow simplifié, intégrant planification et procédé. Coopéra a servi de support à une expérimentation en situation (cf 1.5.2)

Publications : [5, 11, 27, 28, 4, 14, 13]

1.4.4 Bonita

Bonita est un moteur de workflow flexible adapté à la fois à des applications coopératives et à des applications plus traditionnelles. C'est le premier moteur de workflow a avoir été développé intégralement sur une base J2EE. Il permet à la fois une définition et une modification dynamique de procédés mais également l'utilisation de standard comme XPDL.

Le développement de Bonita se poursuit actuellement dans le cadre du consortium ObjectWeb. Bonita fait partie de l'offre de Bull pour le développement de services. Conformément à l'objectif que nous avions u début de son développement, Bonita a atteint un niveau de maturité qui permet de l'utiliser en production. Bonita nous a également servi de base pour expérimenter différents mécanismes comme par exemple la distribution de l'exécution des processus (Thèse CNAM de Joan Erhart) ou la multi-instantiation (DEA d'Adnene Guabtni). Bonita est actuellement étendu pour prendre en compte de façon plus systématique les aspects organisationnels et à travers eux les problèmes de gestion des droits et de sécurité dans des workflows coopératifs. Bonita a fait l'objet de présentation dans de nombreuses conférences professionnelles (JavaOne 2004 [29],JavaPolis, Linux Solutions,...) et d'articles sur des sites spécialisés à forte audience (theServerSide,InfoQ). Une initiative est en cours pour rapprocher les différents projets de développement de moteur de workflow libre et pour proposer une interface standard commune (avec JBoss jBPM et Sun).

Publications : [6]

1.5 Recherches contractuelles

Durant ces années, mes travaux se sont appuyés à divers degrés sur des projets de recherche nationaux et européens en relation avec l'industrie et la recherche, avec des objectifs de collaboration, de valorisation ou de transfert.

1.5.1 Corvette (2000-2001)

Ce travail s'est fait en collaboration avec le laboratoire Hitachi SDL à Kawasaki.

La conception et la réalisation de systèmes complexes nécessite de plus en plus la coordination et la collaboration d'individus distribués entre les divisions d'une entreprise (idée d'équipe virtuelle) ou de plusieurs entreprises (idée d'entreprise virtuelle). Les outils de workflow permettent une coordination du travail, mais pour des applications de type administratif ou de production. Ils montrent rapidement certaines limites lorsque appliqués à des applications créatives comme les applications de conception. L'objectif de cette collaboration a été de développer une nouvelle technologie, dans le contexte d'Internet et du Web, pour dépasser ces limites en s'appuyant sur la technologie workflow de Hitachi et les modèles de coordination développés dans ECOO.

Ce projet a aboutit à la réalisation d'un prototype et à la publication de plusieurs articles décrivant les conclusions de l'expérience [2, 1].

1.5.2 Coopéra

Le projet Coopera est un projet de recherche et développement mené par l'entreprise Jeriko, très présente dans le marché du Multimédia interactif, l'équipe ECOO, le projet Codisant de l'université de Nancy 2 et le laboratoire Gr@mmsci de l'Université de Bordeaux 3 et labélisé par le RIAM.

Ce projet avait pour but de mettre au point et d'expérimenter une plate-forme coopérative à destination des écoles et des collèges. Cette plate-forme devait permettre à des classes ou à des groupes d'élèves de coopérer à travers Internet pour mener à bien des projets de différents ordres. L'objectif pédagogique était d'apprendre aux enfants à travailler et à coopérer dans le monde de l'Internet. Dans ce projet, à forte composante d'analyse d'usages, nous avons utilisé la plate-forme ToxicFarm pour l'adapter à un nouveau public. Cette expérience avait pour but de nous permettre de valider les hypothèses faites au cours du développement de la plate-forme dans un contexte spécifique. Une pré-expérience a été conduite en mai et juin 2002. Cette étape initiale nous a permis de dégager les évolutions nécessaires à la réalisation de deux expériences de plus grande importance en 2002-2003 et 2003-2004. Une description détaillée des résultats de ce projet multidisciplinaire peuvent être trouvé dans le chapitre 7

Outre un prototype, ce projet a permis de mettre en place une collaboration interdisciplinaire fructueuse qui a débouchée sur de nombreuses publications [5, 11, 27, 28, 4, 14, 13].

1.5.3 Libresource

Le projet Libresource (RNTL) avait pour but de fournir un environnement de travail coopératif pour des équipes distribuées à la SourceForge, basé sur une architecture J2EE. Ce projet vient à la suite des projets Motu, ToxicFarm et Coopera. Une version de Libresource est disponible en Open Source. Une expérience d'intégration de Bonita et de Libresource a été effectuée dans le cadre d'une thèse d'ingénieur CNAM. Libresource est disponible en production mais l'intégration avec Bonita n'a pas été finalisée.

Les partenaires du projet étaient L'Université Henri Poincaré, l'INRIA, L'Université Paris 7 et la société Artenum.

`http://libresource.inria.fr`

1.5.4 Interop

Interop est un réseau d'excellence Européen dont l'objectif est de faire émerger une communauté autour de la problématique de l'interopérabilité des applications d'entreprise en Europe et de faire émerger des innovations et des technologies dans ce domaine. Le but d'Interop était en particulier de rassembler des chercheurs de domaines ayant une relation avec le problème de l'interopérabilité (Ontologies, modélisation d'entreprise, architecture et plateformes). L'intérêt d'un tel réseau est entre autre de nouer des relations scientifiques.

1.5.5 2ST

Nous avons aidé l'entreprise 2ST à faire évoluer leur système de gestion de contenu ION-CMS, appelé maintenant Kiwi CMS dans un environnement J2EE. Ce projet a été supporté par l'ANVAR Lorraine. KIWI-CMS est maintenant un produit Open Source distribué, bati sur un noyau Libresource.

1.6 Animation Scientifique

- Workshop co-chair IESA 2005 (First International Conference on Interoperability of Enterprise Software and Application)
- Comité de programme
  - BPM 2005 (Third International Conference on Business Process Management),

  - WBPMO 2007 (1st International Workshop on Business Process Management for Outsourcing),
  - WISE 2007 (The 8th International Conference on Web Information Systems Engineering),
  - ICEBE 2007 (IEEE International Conference on e-Business Engineering),
  - DEECS 2007 (The 3rd International Workshop on Data Engineering Issues in E-Commerce and Services)
- Organisation Chair de WISE 2007
- Relecteur pour de nombreuses conférences.

1.7 Enseignement

1.7.1 Systèmes et applications distribués

Mon implication dans des projets de développement, et l'encadrement d'ingénieurs m'a permis d'utiliser l'expérience acquise pour développer un cours sur le développement d'applications distribuées basé sur la technologie J2EE.

Le plan du cours est le suivant

- Introduction à la programmation d'applications distribuées
- L'environnement J2EE
- Les EJB (Session Beans, Entity Beans, Transformation objet/relationnel)
- Le contexte d'exécution des EJB
- Patrons de conception d'applications distribuées

Les TPs sont effectués sur différents serveurs d'applications en utilisant selon les cas différents environnements de développement (Jonas, Sun AS, Websphere, Jboss)

Ce cours permet aux étudiants d'acquérir les bases concernant les enjeux, les concepts et les technologies nécessaires à la conception, au développement et à la mise en oeuvre d'applications distribuées pour les entreprises. Ce cours évolue régulièrement pour tenir compte des avancées rapides dans ce domaine (JEE 5, Web Services, Micro-containers, injection de dépendances). Il est à destination d'étudiants de niveau Bac+5 et a été dispensé sous différentes formes aux étudiants de Master Pro IL, Réseau et SID ainsi qu'aux étudiants d'ESIAL troisième année. Les supports de cours et de TP sont disponibles sur la plateforme Moodle de L'université de Nancy.

1.7.2 Gestion de contenu et travail coopératif

Ce cours a pour objectif de faire connaitre aux étudiants les différents services disponibles pour permettre à des personnes de coopérer et de communiquer en utilisant Internet. Ce cours est à destination d'utilisateurs plutôt que de développeurs.

La première partie introduit les systèmes de gestion et de production de contenu (CMS, Blog, Wiki) et explique leurs enjeux et leur fonctionnalités.

La seconde partie introduit les systèmes permettant la gestion de communautés et la gestion de groupe de travail.

Les TP permettent aux étudiants de s'initier à leur déploiement et à leur utilisation et sont basés sur SPIP, Libresource, et d'autres outils disponible sur Internet.

Ce cours a été délivré à des étudiants de Licence Professionnelle en technologies de l'information et de la communication.

1.7.3 Développement pour le Web et composants

Ce cours pour objectif d'initier les étudiants à la problématique de la conception et du développement d'applications distribuées et de la programmation par composants. Ce cours introduit les notions de conteneur, d'injection de dépendance et d'inversion du contrôle, de "framework" et l'architecture MVC2. Il est a destination d'étudiants de niveau Bac+4. Le plan du cours est le suivant :

- Introduction à la notion de composant et de conteneur.
- Inversion du contrôle et injection de dépendance.
- Les composants Web : Servlet
- Le développement d'applications Web : JSP, Tag
- Les frameworks contrôleur : Struts, JSF

Ce cours évolue régulièrement pour prendre en compte à la fois les évolutions conceptuelles du domaine et les évolutions technologiques.

Ce cours a été délivré à des étudiants de niveau Bac+4 en informatique et à des étudiants de licence professionnelle.

1.7.4 Conception objet et patrons de conceptions

Ce cours a pour objectif de donner aux étudiants des méthodes et des outils pour concevoir un logiciel basé sur une approche objet. Ce cours est relativement classique aujourd'hui. Il introduit le langage UML pour la conception objet et il essaie de montrer quels sont les problèmes liés à la conception d'un logiciel basé sur un modèle objet en insistant sur les problèmes de cohésion et de couplage.

1.8 Responsabilités

J'ai eu l'occasion au cours de ces années de prendre des responsabilités pédagogiques et administratives.

1.8.1 Direction des études - Département Information-Communication

De 1993 à 2000, j'ai pris en charge la direction des études de l'option Publicité du département Information-Communication de l'IUT Charlemagne (60 étudiants). Le travail lié à cette charge consistait à effectuer la sélection des dossiers (600 chaque année) pour cette filière, réaliser les emplois du temps et faire évoluer le contenu pédagogique en fonction des nouveaux programmes pédagogiques nationaux. Durant cette période j'ai dirigé avec une collègue l'organisation à Nancy du "Challenge de la publicité", compétition amicale et sponsorisée entre les IUT ayant un option publicité en France. Cette compétition a toujours lieu a Nancy et réuni maintenant la plupart des formations en publicité de France.

En 2004, j'ai pris la responsabilité de la création d'une licence professionnelle de création publicitaire avec Jean-Grenier Godart. Cette licence a été ouverte à la rentrée 2005 avec un effectif de 28 étudiants.

1.8.2 Chef de département - Département Information-Communication

En 2004-2005, j'ai été nommé Chef du département Information de l'IUT Charlemagne. Le département Information-Communication compte environ 400 étudiants, répartis dans 4 options (communication, métiers du livre, documentation, publicité) et 4 années spéciales. Il comptait

en 2004 une licence pro en Communication et 2 nouvelles licences ont ouvertes en 2005 (Création publicitaire et Iconographie). J'ai quitté cette fonction suite à ma mutation à l'ESIAL.

1.8.3 Responsable de l'option Ingéniérie du Logiciel (IL) à l'ESIAL

Depuis la rentrée 2006, je suis responsable de l'option IL de l'ESIAL. Je participe également à la définition d'une nouvelle option Système et Applications Distribuées (SAD).

1.8.4 Membre élu du Conseil d'administration de l'Université de Nancy 2

J'ai été élu au conseil d'administration de l'Université de Nancy 2 en 2001. J'y suis resté jusqu'en 2005 date de ma mutation à l'Université Henri Poincaré. Pendant toute cette période, j'ai participé aux travaux du conseil et plus particulièrement à la commission du budget chargée de la mise en place de la LOLF.

1.8.5 Membre élu de la commission de spécialiste en 27ème section de l'Université de Nancy 2

J'ai fait partie pendant deux ans (2003-2005) de cette commission.

1.9 Encadrements

1.9.1 Co-encadrements de thèses

Durant ces années, j'ai participé avec Claude Godart à l'encadrement de plusieurs thèses.

Hala Skaf (1996)

Titre de la thèse : Une approche hybride pour gérer la cohèrence dans les environnements de développement coopératif

Daniela Grigori (2001)

Titre de la thèse : Eléments de flexibilité des systèmes de workflow pour la définition et l'exécution de procédés coopératifs

Adnene Guabtni (2006)

Titre de la thèse : Sphère de comportement pour la modélisation de procédés flexible

1.9.2 Encadrements de DEA

Adnene Guabtni (2003)

Titre du DEA : Multi-instanciation et itération dans un système de workflow coopératif

1.9.3 Ingénieurs et thèse CNAM

Miguel Valdes

Développement du moteur de workflow Bonita sur Jonas

Joan Erhahrt (2005)

Titre de la thèse CNAM : Intégration de Bonita à LibreSource

1.10 Publications

1.10.1 Publications dans des journaux

[3, 32, 28, 19, 1, 13, 20]

1.10.2 Publications dans des conférences internationales avec comité de lecture

[9, 30, 10, 33, 31, 21, 12, 8, 15, 17, 16, 2, 5, 27, 11, 18, 4, 14, 22, 24, 23, 6, 7, 25, 26]

Bibliographie

[1] Karim Baina, François Charoy, Claude Godart, S. El Hadri, Hala Skaf, S. Akifuji, T. Sakaguchi, Y. Seki, and M. Yoshioka. Corvette : a cooperative workflow for virtual teams coordination. *International Journal Networking and Virtual Organisations*, 2(3) :232–245, Dec 2004.

[2] Karim Baïna, François Charoy, Claude Godart, Daniela Grigory, Saad el Hadri, Hala Skaf, Shunsuke Akifuji, Toshiaki Sakaguchi, Yoko Seki, and Masaichiro Yoshioka. Corvette : A cooperative workflow development experiment. In *3rd IFIP Working Conference on Infrastructures for Virtual Enterprises - PRO-VE'2001, Sesimbra, Portugal* , May 2002.

[3] Gérôme Canals, Claude Godart, François Charoy, Pascal Molli, and Hala Skaf. Coo approach to support cooperation in software developments. *IEE Proceedings - Software*, 145(2-3) :79–84, Jun 1998.

[4] François Charoy, Claude Godart, Nicolas Gregori, Jean-Charles Hautecouverture, and Sébastien Jourdain. Coopera : An environment for teaching and learning internet cooperation. In *IADIS International Conference e-Society 2004, Avila, Espagne*, pages 323–330, Jul 2004.

[5] François Charoy, Claude Godart, Pascal Molli, Gerald Oster, Marc Patten, and Miguel Valdes. Services for virtual teams hosting : Toxicfarm introduction. In *Second International Workshop on Cooperative Internet Computing - CIC 2002, Hong Kong, China*, pages 105–112, Aug 2002.

[6] François Charoy, Adnene Guabtni, and Miguel Valdes Faura. A dynamic workflow management system for coordination of cooperative activities. In *Workshop on Dynamic Process Management - BPM 2006 International Workshops, BPD, BPI, ENEI, GPWW, DPM, semantics4ws, 04/09/2006*, volume 4103 of *in : LNCS, Business Process Management Workshops*, pages 205–216, Vienne/Autriche, 2006. Springer. Non.

[7] Khaled Gaaloul, François Charoy, and Claude Godart. Cooperative processes for scientific workflows. In *6th International Conference on Computational Science - ICCS 2006, 28/05/2006*, volume 3 of *in : Lecture Notes in Computer Science, International Conference on Computational Science*, pages 976–979, Reading/UK, 2006. Springer. Non.

[8] Claude Godart, Christophe Bouthier, Philippe Canalda, François Charoy, Pascal Molli, Olivier Perrin, Helene Saliou, Jean-Claude Bignon, Gilles Halin, and Olivier Malcurat. Asynchronous coordination of virtual teams in creative applications (co-design or co-engineering) : requirements and design criteria. In *Information Technologies for Virtual Enterprises*, Jan 2001.

[9] Claude Godart, Gérôme Canals, François Charoy, Pascal Molli, and Hala Skaf. Designing and implementing coo : Design process, architectural style, lessons learned. In *Proceedings International Conference on Software Engineering (ICSE18)*. IEEE Press, 1996.

[10] Claude Godart, Gérôme Canals, Francois Charoy, Pascal Molli, and Hala Skaf. Designing and implementing coo : Design process, architectural style, lessons learned. In *ICSE 18 (International Conference On Software Engineering), Berlin*, pages 342–352. IEEE Publishing Computer Society Press, mar 1996.

[11] Claude Godart, François Charoy, Marc Patten, Nicolas Grégori, Jean-Charles Hautecouverture, and Isabelle Faugeras. Coopéra : apprendre à coopérer et apprendre en coopérant. In *Conférence Internationale sur l'Enseignement Ouvert et en Ligne - ICOOL 2003, Ile Maurice*, Dec 2003.

[12] Claude Godart, François Charoy, Olivier Perrin, and Hala Skaf. Cooperative workflows to coordinate asynchronous cooperative applications in a simple way. In IEE Press, editor, *Parallel and Distributed Systems - PADS 2000*, pages 409–416. IEEE Computer Society, Jul 2000.

[13] Nicolas Grégori, Jean-Charles Hautecouverture, François Charoy, and Claude Godart. Combining ergonomics, culture and scenario for the design of a platform for cooperation. *Artificial Intelligence and Society*, 20(3) :384–402, 2006.

[14] Nicolas Grégori, Jean-Charles Hautecouverture, Claude Godart, and François Charoy. Ergonomie, culture, scénario : trois facteurs pour la conception d ?une plate-forme de coopération. In *Colloque ARCo'2004, Compiègne, France*, Dec 2004.

[15] Daniela Grigori, François Charoy, and Claude Godart. Anticipation to enhance flexibility of workflow execution. In H.C. Mayr, J. Lazansky, G. Quirchmayr, and P. Vogel, editors, *International Conference on Database and Expert Systems Applications - DEXA'2001, Munich, Germany*, volume 2113 of *Lecture Notes in Computer Science*, pages 264–273. Springer-Verlag, Sep 2001.

[16] Daniela Grigori, François Charoy, and Claude Godart. Flexible cooperative workflow management. In *13th International Conference on Control Systems and Computer Science - CSCS'2001, Bucarest, Roumanie*, Jun 2001.

[17] Daniela Grigori, François Charoy, and Claude Godart. Flexible data management and execution to support cooperative workflow : the coo approach. In *Proceedings of the Third International Symposium on Cooperative Database Systems for Advanced Applications - CODAS'2001, Beijing, China*, pages 139–146. IEEE, Apr 2001.

[18] Daniela Grigori, François Charoy, and Claude Godart. Coo-flow : a process model to support cooperative processes. In *Fifteenth International Conference on Software Engineering and Knowledge Engineering 2003 - SEKE'2003, San Francisco*, Jul 2003.

[19] Daniela Grigori, François Charoy, and Claude Godart. Coo-flow : a process technology to support cooperative processes. *International Journal of Software Engineering and Knowledge Engineering - IJSEKE Journal*, 14(1) :61–78, Feb 2004. World Scientific Publishing.

[20] Daniela Grigori, François Charoy, and Claude Godart. Enhancing the flexibility of workflow execution by activity anticipation. *International Journal of Business Process Integration and Management*, 1(3) :143–155, 2006.

[21] Daniela Grigori, Hala Skaf-Molli, and François Charoy. Adding flexibility in a cooperative workflow execution engine. In *8th International Conference on High Performance Computing and Networking Europe - HPCN Europe 2000, Amsterdam, Hollande*, May 2000.

[22] Adnene Guabtni and François Charoy. Multiple instantiation in a dynamic workflow environment. In Anne Persson and Janis Stirna, editors, *Advanced Information Systems Engineering, 16th International Conference, CAiSE 2004, Riga, Lavtia*, volume 3084 of *Lectures Notes in Computer Science*, pages 175–188. Springer, Jun 2004.

[23] Adnene Guabtni, François Charoy, and Claude Godart. Customizable isolation in transactional workflow. In *First International Conference on Interoperability of Enterprise Software and Applications - INTEROP-ESA'2005 First International Conference on Interoperability of Enterprise Software and Applications - INTEROP-ESA'2005*, Geneva/Switzerland, 02 2005. University of Geneva, Switzerland.

[24] Adnene Guabtni, François Charoy, and Claude Godart. Spheres of isolation : Adaptation of isolation levels to transactional workflow. In *Business Process Management*, pages 458–463, 2005.

[25] Adnene Guabtni, François Charoy, and Claude Godart. Concurrency management in transactional web services coordination. In *17th International Conference on Database and Expert Systems Applications - DEXA 2006, 04/09/2006*, volume 4080/2006 of *in : Lecture Notes in Computer Science, Database and Expert Systems Applications*, pages 592–601, Krakow, Poland, 2006. Gabriela Wagner, FAW, University of Linz, Austria, Springer Berlin / Heidelberg. Non.

[26] Adnene Guabtni, François Charoy, and Claude Godart. Using isolation spheres for cooperative processes correctness. In IEEE Beijing, editor, *The 10th International Conference on CSCW in Design (CSCWD 2006), 03/05/2006*, volume 1 of *Computer Supported Cooperative Work in Design 2006*, Nanjing, China, 2006. Southeast University. Non.

[27] Jean-Charles Hautecouverture, Nicolas Grégori, François Charoy, Claude Godart, Marc Patten, and Isabelle Faugeras. Coopera : Analyse de l'usage d'une plate-forme de coopération à destination d'enfants du primaire. In *Human Centered Processes - HCP'2003, Luxembourg*, May 2003.

[28] Jean-Charles Hautecouverture, Nicolas Grégori, François Charoy, Claude Godart, Marc Patten, and Isabelle Faugeras. Analyse d'usage d'une plate-forme de coopération : usage et développement logiciel. *In Cognito - Cahiers Romans de Sciences Cognitives*, 1(3) :45–77, 2004.

[29] Christophe Loridan and Miguel Valdes. Bonita :a java 2 platform, enterprise edition (j2ee) open source cooperative workflow system. In *Java One, 2004*, San Francisco, USA, 2004.

none. Non, This presentation describes the Bonita Workflow Management System developed by the ECOO team of LORIA. Colloque avec actes et comité de lecture. internationale.

[30] Hala Skaf, François Charoy, and Claude Godart. An hybrid approach to maintain consistency of cooperative software development activities. In *SEKE97 The Ninth International Conference on Software Engineering and Knowledge Engineering, Madrid*, jun 1997.

[31] Hala Skaf, François Charoy, and Claude Godart. Flexible integrity control of cooperative applications. In IEEE Computer Society, editor, *The Ninth International Workshop on Database and Expert Systems Applications - DEXA 98, Vienne, Autriche*, 1998.

[32] Hala Skaf, François Charoy, and Claude Godart. Maintaining shared workspaces consistency during software development. *International Journal of Software Engineering and Knowledge Engineering*, 9(5) :623–642, Oct 1999.

[33] Hala Skaf, Francois Charoy, and Claude Godart. Maintaining consistency of cooperative software development activities. In *Integrity in Database 6th International Workshop on Foundations of Models and Languages for Data and Objects, Dagstuhl*, sep 1996.

Chapitre 2

Introduction

La coopération entre personnes et entre organisations distribuées dans le temps et dans l'espace est devenue, avec la montée en puissance des réseaux et des applications, qui y sont liées une réalité. De grands projets, les plus visibles étant dans le domaine du développement de logiciels [9, 8], sont menés quotidiennement sans que leur participants aient besoin de se rencontrer. Chacun peut aujourd'hui produire des contenus ou des services en collaborant de façon plus ou moins formelle avec des personnes à distance. Les outils, les plate-formes sont nombreuses, adaptées à différents domaines, malheureusement sans doute trop peu interopérables. De plus, en général, elles ne fournissent que les outils habituels de communication, de partage et d'éditions collaboratives. Ces fonctionnalités peuvent être disponibles en ligne (Google docs, Wikipedia) ou intégrés aux outils des utilisateurs (Eclipse).

Pratiquement, nous sommes constamment confrontés dans notre travail quotidien à des problèmes de coopération pour écrire des articles, répondre à des appels d'offre, conduire des projets, rédiger des rapports d'activité. Dans chacun de ces cas, on se rend compte que la distance géographique, les habitudes de travail, la connaissance des partenaires ont un impact sur la coordination. On s'aperçoit rapidement que les outils qu'on utilise pour cette coordination (le courrier électronique, les échanges informels, les réunions) ne sont pas adaptés au suivi de l'avancement du travail. Il est également difficile d'y remédier simplement. C'est d'autant plus compliqué que les personnes impliquées appartiennent à des organisations différentes et/ou sont habitués à des outils, des méthodes hétérogènes.

Dans le même temps se développent d'autres modes de coopération entre organisations, basées sur des services publiés par ces organisations. La coordination et l'orchestration de services distribués pour fournir de nouveaux services à plus forte valeur ajoutée, plus flexibles et plus efficaces est un phénomène majeur du développement des technologies de l'information. Les questions liées à la production et à la consommation de services sont l'origine d'un courant en plein développement lié aux problèmes qui ne sont pas qu'informatique de cette ingéniérie des services.

Dans les deux cas, les questions qui se posent et auxquels nous allons nous intéresser sont liés à la coordination des activités entre des systèmes ou des personnes appartenant à des organisations différentes. Ces problèmes sont dus à plusieurs facteurs.

Le premier est celui de la flexibilité requise pour dans les processus d'orchestration qui gouverne la coordination des services. La distribution du contrôle des services fournis, les besoins de réactivité des organisations aux changements du contexte, la nature même des processus font qu'il est nécessaire de permettre dans de maitriser les changements qui peuvent intervenir dans leur définition ou lors de leur exécution.

Le second est celui des garanties que l'on veut assurer pour l'exécution des services. Permettre une grande flexibilité et une grande réactivité des organisations au changement n'empèche pas que certains critères, certaines qualités ou certaines règles doivent être garanties et/ou respectées lors de l'exécution des processus. Ainsi, dans certains cas, la sécurité, la sureté de fonctionnement doivent être garantis lors de l'exécution des processus pour donner aux clients des services des assurances sur le résultat attendu.

Le troisième est celui des répercutions sociales et opérationnelles de l'introduction de ces systèmes dans les organisations. Derrière la fourniture de certains services, il y a des activités humaines plus ou moins codifiées. La coordination de ces activités humaines ne se fait pas aussi naturellement que celles d'activités complétement automatisées. L'introduction de nouvelles technologies nécessite de prendre en compte l'histoire et la culture des hommes et des organisations pour garantir son succès. Cette question, même si elle déborde du cadre purement informatique, fait partie pleinement de nos préoccupations.

2.1 Historique

L'idée d'appliquer aux services les méthodes qui avaient permis l'augmentation de la productivité dans l'industrie manufacturière et en particulier son automatisation date des années 1970 [5]. Sous le terme générique d'"Office Automation" ou de "Computer Integrate Manufacturing", l'ambition des chercheurs à cette époque était de fournir les outils permettant de modéliser, d'automatiser et de contrôler les processus de services, le plus souvent dans les banques et les assurances. La grande utopie de cette période où les ordinateurs commençaient à investir les postes de travail était le bureau zéro papier[1]. Ces travaux ont permis l'apparition dans les années 80 des premiers systèmes connus sous les noms de Workflow Management System ou Système de gestion des flux de tâches. Ces systèmes ont évolué de façon relativement confidentielle jusque dans les années 90 avec la montée en puissance des réseaux qui ont permis de les envisager de façon beaucoup plus globale. Dès cette époque, où l'on considérait encore que l'homme et les organisations devaient s'adapter à la machine, les problèmes humains et organisationnels liés à l'automatisation et au contrôle des processus métiers sont devenus très prégnants et ont fait l'objet de différentes études et de débats [15, 12]. A cette époque sont apparues des formalisations des systèmes de workflow basés essentiellement sur les réseaux de Pétri [16], le Pi-calcul [14] ou l'Event calculus [3]. Les premiers se prêtent relativement bien à la représentation des flux de tâches. Les autres formalisations sont adaptées à différentes sortes de raisonnement et de contrôle sur la correction de ces représentations. Des travaux ont été mené en parallèle par les chercheurs issus des systèmes d'information et des bases de données et par ceux issus du travail coopératif. D'un coté, on utilisait des modèles formels de représentation des processus pour modéliser procédures métiers classiques. D'un autre coté, on tentait de comprendre les tenants et les aboutissants de la coordination d'activité pour proposer des modèles permettant de les représenter et d'assister les utilisateurs travaillant en groupe. La première communauté s'est structurée, a intégré les travaux sur les transactions longues et s'est intéressée naturellement à la question des procédés inter-organisationels. Une autre communauté, issue des systèmes distribués s'est emparée de problématique similaire mais dans une optique plus opérationnelle pour coordonner et intégrer des services s'exécutant sur des machines distribuées [6]. La communauté CSCW s'est elle un peu désintéressée du problème de la coordination d'activité coopérative. Après plusieurs années de polémiques et des propositions de modèles divers [11, 4, 7, 10] on peut

[1] on sait ce qu'il en est, même si des progrès notable sont en cours dans la virtualisation de procédures administratives.

regretter aujourd'hui que cette problématique ait disparue du champ de cette communauté. Il faut cependant noter des travaux qui prennent une direction un peu différente mais finalement assez proche comme l'Activity computing de Bardram [1] ou les travaux plus théorique de John Carroll et Mary Beth Rosson [2]. Notons également que si les communautés du Business Process Modeling et celle des Web Services se rencontrent et présentent des intersections notables, la communauté du CSCW n'est plus très concernée. Nous nous situons dans cette histoire à la limite entre les problématiques du travail coopératifs et celles des systèmes distribués et de la gestion des processus métiers. Notre ambition est toujours d'essayer de comprendre les questions liés à la coordination d'activités coopératives avec une approche plus technique que sociale. Nous allons cependant continuer en essayant décrire ce que nous entendons par coordonner des activités coopératives.

2.1.1 Coordination et coopération

La coopération est un principe relativement ancien, et même chargé idéologiquement. Dans leur article, Schmidt et Bannon [13] indiquent que le terme a été utilisé au 19ème siècle par les économistes pour désigner un travail impliquant plusieurs acteurs. Marx (1867) l'a défini comme "plusieurs individus travaillant ensemble de façon consciente dans le même processus de production ou dans des processus différents mais connectés". L'aspect le plus important dans la plupart des définitions qui ont été utilisées à cette époque concerne l'interdépendance dans le travail. A cela peut s'ajouter le fait que coopérer peut être en général considéré comme une nécessité. La dépendance fait qu'il est nécessaire de coopérer pour atteindre un objectif. Cette dépendance est positive (i.e. il ne s'agit pas de compétition pour une ressource).

Nous reprenons à notre compte cette définition. Dans ce contexte, les différents participants doivent non seulement s'occuper des tâches qu'ils ont à accomplir mais également de la coordination entre ces tâches.

2.1.2 Coordination, activités et procédés

Un projet coopératif ne peut s'exécuter correctement sans une forme coordination entre les membres du groupe concerné. Cette coordination peut avoir plusieurs dimensions mais elle se traduit généralement par une identification des dépendances entres les activités que chacun des participants doit mener. Ces dépendances peuvent être simplement temporelles (il faut faire une action avant une autre), liées à des ressources à produire (pour relire un document il faut qu'il ait été produit) ou liées à une règle spécifique d'organisation (une signature à obtenir par exemple). La division du travail est elle relative à des compétences, des disponibilités ou des autorisations.

Dans le cas d'un groupe de petite taille et co-présent, dont les membres se connaissent, ayant un histoire commune, pour un projet relativement court, cette coordination peut se faire de façon implicite en l'assurant par une communication verbale ou non, selon la connaissance implicite que chacun à des compétences des membres du groupe [2]. Une équipe de recherche devant réaliser un rapport d'activité peut atteindre son objectif sans utiliser de moyens très sophistiqués grâce à la connaissance des compétences, aux rôles connus et acceptés des membres de l'équipe et à la taille modeste du document à produire.

Dans le cas de projet plus long, ayant un plus grand nombre d'intervenants, éventuellement dispersés, cette coordination implicite est insuffisante. La mise en oeuvre de coordination explicite devient indispensable pour permettre le suivi des activités à réaliser, leur exécution, leur résultat. L'écriture du rapport d'activité d'une institution de taille importante va nécessiter une coordination beaucoup plus explicite, une répartition précise des rôles, des tâches à effectuer,

des délais à respecter, des procédures à suivre pour arriver à un résultat final satisfaisant. Ce problème peut également se poser pour un projet de taille plus petite (réponse à un appel d'offre) incluant des partenaires d'origines différentes, ayant peu l'habitude de travailler ensemble.

Pour ce genre de projets, il apparait nécessaire de proposer des outils permettant à la fois le support au travail coopératif et sa coordination. Ce support doit permettre au groupe de mettre en place simplement la description du travail à réaliser (les tâches ou les actions), leur ordonnancement, les délais d'exécution, les responsabilités et les autorisations. Il doit surtout permettre de suivre et de contrôler l'avancement du projet pour permettre à chacun d'avoir une vision claire sur ce qui a été fait, par qui, quand et ce qui reste à faire. Il doit également fournir des garanties sur les qualités de l'exécution. Le modèle obtenu doit être flexible pour pouvoir évoluer en fonction d'événements extérieurs, de retards ou d'autres problèmes imprévus comme des défaillances de partenaires.

2.2 Procédés coopératifs et procédés métiers

parmi les outils utilisés et aujourd'hui déployés pour assurer la coordination d'activités de service ou administrative se trouvent bien sur les système de gestion de workflow. Ces systèmes permettent de décrire explicitement les tâches à réaliser pour atteindre un but, leur dépendances et leurs enchainements, les données dont ils ont besoin et leur relation entre les tâches et l'organisation de l'entreprise. Les systèmes de gestion de workflow ont montré, avec quelques limites cependant leur utilité dans les entreprises. Ils permettent la modélisation et l'exécution de procédés métiers. Ils sont en revanche peu adaptés à la prise en charge des procédés coopératifs ou créatifs qui ont des caractéristiques que ces systèmes ne peuvent pas prendre en charge.

Les procédés coopératifs sont différents des procédés métiers par nature. Nous avons identifiés plusieurs différences entre un Système de gestion de Workflow qui doit supporter des procédés coopératifs d'un système de gestion de Workflow qui supporte des procédés métiers traditionnels.

La première différence se situe au niveau du ratio entre le nombre d'exécutions d'un processus et le nombre de définitions de processus. Il est petit pour les procédés coopératifs par rapport aux procédés métiers. En effet, un processus métier, de type bancaire, est défini de façon précise pour être exécuté un grand nombre de fois. Un processus coopératif, est lui en général partiellement défini, à partir de fragments de procédés, quasiment projet par projet. En poussant ce principe à l'extrême, chaque processus dans un environnement coopératif est unique. Cela signifie pratiquement que les participants au processus doivent être capable de concevoir ou au moins de modifier le processus.

La seconde différence concerne la structure plus simple des processus coopératifs. Les processus métiers qui sont peu souples sont en revanche relativement complexes. Lorsqu'un processus métier est conçu, le concepteur doit tenter de prévoir tous les cas possibles. Les processus coopératifs, sont en général plus simple. Ils consistent souvent en une suite d'activités s'exécutant en boucle ou multi instanciés. En outre, les exceptions doivent pouvoir être traités dynamiquement par les participants à ces procédés. Tous les cas possibles n'ont dont pas à être envisagés au départ. Nous considérons en fait que les processus coopératifs sont fait d'étapes successives qui amène à la production du résultat final du projet. Même si certaines des ces étapes peuvent être complexes, elles doivent rester compréhensibles par les participants au processus.

La troisième différence concerne le rythme d'évolution des processus coopératifs. Un processus métier classique est moins sujet aux changement que les processus coopératifs. La durée proportionnellement plus longue de ces derniers induit nécessairement un besoin plus grand d'évolution. Les changements dans l'environnement, dans les buts du procédé ont plus de chance

de se produire. Les procédés métiers ont eux une durée d'exécution plus courte et sont ainsi potentiellement plus stables.

La dernière différence concernent l'implication des participants aux processus coopératifs. Les processus coopératifs gouvernent des projets coopératifs où les participants sont concernés par un objectif commun. C'est en général moins le cas dans les processus métiers classiques où les utilisateurs sont concernés principalement par les tâches qu'ils ont à exécuter. Ainsi, les systèmes de gestion des procédé coopératifs doivent fournir aux utilisateurs une vue précise de ce qui doit être fait, de qui fait quoi et de ce qui reste à faire (même si ce qui reste à faire est toujours susceptible de changer). Nous considérons ainsi que contrairement à un processus métier, un processus coopératif est le résultat d'un consensus entre ses participants.

Bien que ces différences semblent relativement importantes, nous pensons qu'il est possible d'adapter les modèles de workflow et les systèmes de gestion de workflow pour permettre leur utilisation dans des contextes coopératifs. Nous pensons toujours que la notion d'activité est centrale même dans des processus gouvernés par des humains et que la description des activités et de leur dépendance est un moyen d'assister, de rendre plus efficace et de contrôler l'exécution des projets coopératifs.

2.3 Axes de recherche et Contributions

Comprendre comment peut s'effectuer la coordination d'activités coopératives et quel support il est possible fournir pour cette coordination est un travail complexe et faisant appel à de nombreux champs disciplinaires. Nous avons essayé de l'aborder en considérant les aspects qui nous semblaient important du point de vue de la modélisation et de l'exécution, en partant de notre culture informatique et en essayant de constamment prendre en compte les résultats issus des études et des travaux concernant les usages des technologies nouvelles. Trois axes principaux ont fait l'objets de travaux particuliers :

- Le contrôle de la coordination
- La flexibilité du modèle et de son exécution
- La mise en oeuvre de propriétés opérationnelles transversales.

2.3.1 Contrôle de la coordination

La première approche que nous avons étudiée se concentrait sur les données produites au cours d'un projet coopératif (chapitre 3). L'idée était de contrôler l'évolution de l'état de ces données vers un état de satisfaction final. En utilisant une forme de logique temporelle, il est possible de garantir qu'une donnée passe par différents états avant d'atteindre un état de satisfaction. Ces états correspondent aux transformations successives que doit subir un ensemble de données au cours du projet. Cette approche permet de mettre en oeuvre une forme de contrôle d'une coordination implicite. C'est sa limite. Elle ne permet pas aux utilisateurs de savoir comment arriver à l'état final.

2.3.2 Exécution des procédés

La seconde approche concerne l'adaptation des systèmes de gestion de procédés aux procédés coopératifs(chapitre 4). Les systèmes de gestion de procédés permettent de décrire de façon explicite des activités et leur dépendance temporelles. A l'origine, ils sont adaptés à des procédés métiers, relativement bien balisés et surtout répétables. Leur adaptation à des procédés coopératifs nécessite de rendre leur modèle adaptable et leur exécution flexible. Nous avons proposé

un modèle de procédés et un modèle d'exécution permettant d'atteindre cet objectif. Ce modèle a été mis en oeuvre dans un prototype, Bonita, qui a ensuite été transféré pour être maintenant industrialisé (chapitre 5).

2.3.3 Comportement des procédés

Un axe important de recherche que nous avons entrepris se trouve dans la suite logique des deux précédents (chapitre 6). La modélisation d'un procédé est une description du programme des actions à effectuer pour atteindre un objectif. Le script de ce procédé, son découpage en activités est dépendant du but à atteindre mais également et surtout de l'organisation des acteurs, de leur compétences supposées et des roles qu'ils peuvent prendre ou des autorisations qu'ils peuvent avoir. Ce découpage a un impact sur la granularité et l'enchainement des activités.

Il peut également être requis pour un procédé qu'il s'exécute en totalité ou en partie avec des qualités particulières (atomicité, isolation, compensation par exemple). L'attachement des qualités attendues de l'exécution doit avoir un impact minimal sur la définition du procédé lui-même. Pour cela, nous proposons une approche séparant définition du procédé et définition des qualités d'exécution en introduisant la notion générale de sphères de comportement. Une sphère de comportement regroupe un sous-ensemble d'activités d'un procédé et garanti une qualité d'exécution pour ce sous ensemble. Ainsi, il doit être possible de décider si un sous ensemble des activités d'un procédé doit être exécuté de façon isolé du reste des activités du système, ou si cette ensemble doit s'exécuter de façon atomique.

2.3.4 Etude de la coopération

Travailler sur les problèmes liés à la coopération ne peut se faire sans prendre en compte les utilisateurs et sans comprendre leur comportement (chapitre 7). L'acculturation de ceux-ci aux nouvelles méthodes de travail et aux protocoles de coopération en utilisant Internet ne se fait que progressivement. Nous avons donc contribué à une étude dans le cadre d'un projet multi-disciplinaires destiné à comprendre comment des élèves de plusieurs classes de CM2 se comportaient lors d'une collaboration à distance utilisant une plate-forme dédiée. Cette étude, utilisant essentiellement un approche ethnographique et constructiviste nous a permis de mettre en évidence les différentes dimensions en jeu dans un travail coopératif à savoir l'ergonomie bien sur, mais également la culture en terme de coopération et les scénarios proposés. Ces différentes dimensions ont un impact sur la façon dont les participants appréhendent la coopération et donc comprennent et utilise les outils mis à leur disposition. Une approche active leur permet de se construire progressivement une représentation des concepts mis en jeu, en particulier concernant le partage d'objets. C'est en fait plus le scénario et l'expérience du travail en groupe qui déterminent le succès du projet que l'ergonomie de l'outil.

2.4 Organisation du mémoire

Le mémoire suit la progression des contributions. Le chapitre qui suit est consacré à l'étude du contrôle de la coordination implicite pour des activités coopératives. Ce travail, par son résultat est à l'origine de la suite de nos travaux. Il nous a amené à nous intéresser plus précisément à la modélisation de la coordination des activités coopératives, en utilisant des modèles de procédés et aux moyens de rendre leur exécution et leur contrôle plus flexible pour les adapter au contexte du travail coopératif. Nous montrons ensuite comment le modèle que nous avons obtenu a pu être mis en oeuvre dans un système de gestion de workflow dynamique. C'est le chapitre sur

Bonita. Nous poursuivons par la présentation de nos derniers travaux sur les moyens que nous proposons pour séparer description des procédés et propriétés dynamiques de ces procédés à l'aide de ce que nous avons appelé les sphères de comportement. Avant de conclure et de décrire la façon dont nous envisageons la suite de ce travail, l'avant dernier chapitre décrit l'expérience que nous avons mené en collaboration avec une équipe de psycho-sociologues pour tenter de comprendre les mécanismes qui régissent les comportements coopératifs.

Bibliographie

[1] Jakob Bardram, Jonathan Bunde-Pedersen, and Mads Soegaard. Support for activity-based computing in a personal computing operating system. In *CHI '06 : Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 211–220, New York, NY, USA, 2006. ACM Press.

[2] John M. Carroll, Mary B. Rosson, Gregorio Convertino, and Craig H. Ganoe. Awareness and teamwork in computer-supported collaborations. *Interacting with Computers*, 18(1) :21–46, January 2006.

[3] Nihan K. Cicekli and Yakup Yildirim. Formalizing workflows using the event calculus. pages 222+. 2000.

[4] Paul Dourish, Jim Holmes, Allan Maclean, Pernille Marqvardsen, and Alex Zbyslaw. Freeflow : mediating between representation and action in workflow systems. In *CSCW '96 : Proceedings of the 1996 ACM conference on Computer supported cooperative work*, pages 190–198, New York, NY, USA, 1996. ACM Press.

[5] Clarence A. Ellis and Gary J. Nutt. Office information systems and computer science. *ACM Comput. Surv.*, 12(1) :27–60, March 1980.

[6] D. Georgakopoulos and L. Maciaszek. *Proceedings of 9th International Workshop on Research Issues in Data Engineering (Virtual Enterprise).* IEEE Press, 1999.

[7] Natalie S. Glance, Daniele S. Pagani, and Remo Pareschi. Generalized process structure grammars gpsg for flexible representations of work. In *CSCW '96 : Proceedings of the 1996 ACM conference on Computer supported cooperative work*, pages 180–189, New York, NY, USA, 1996. ACM Press.

[8] James D. Herbsleb, Audris Mockus, and Roy T. Fielding. Two case studies of open source software development : Apache and mozilla. *ACM Trans. Softw. Eng. Methodol.*, 11(3) :309–346, July 2002.

[9] James D. Herbsleb, Audris Mockus, Thomas A. Finholt, and Rebecca E. Grinter. An empirical study of global software development : Distance and speed. In *23rd International Conference on Software Engineering (ICSE'01)*, 2001.

[10] Raul Medina-Mora, Terry Winograd, Rodrigo Flores, and Fernando Flores. The action workflow approach to workflow management technology. In *CSCW '92 : Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, pages 281–288, New York, NY, USA, 1992. ACM Press.

[11] Wolfgang Prinz and Sabine Kolvenbach. Support for workflows in a ministerial environment. In *CSCW '96 : Proceedings of the 1996 ACM conference on Computer supported cooperative work*, pages 199–208, New York, NY, USA, 1996. ACM Press.

[12] K. Schmidt. *The critical role of workplace studies in CSCW.* Cambridge University Press, 1999.

[13] Kjeld Schmidt and Liam Bannon. Taking cscw seriously. *Computer Supported Cooperative Work (CSCW)*, V1(1) :7–40, March 1992.

[14] Howard Smith and Peter Fingar. Workflow is just a pi process, November 2003.

[15] Lucy A. Suchman. *Plans and Situated Actions : The Problem of Human-Machine Communication (Learning in Doing : Social, Cognitive & Computational Perspectives)*. Cambridge University Press, November 1987.

[16] W. M. P. van der Aalst, K. M. van Hee, and Houben. Modelling workflow management systems with high-level petri nets. In *Proceedings of the second Workshop on Computer-Supported Cooperative Work, Petri nets and related formalisms*, pages 31–50, 1994.

Chapitre 3

Une approche déclarative de la coordination

3.1 Introduction

Dans cette partie, nous décrirons l'approche qui est à l'origine de notre travail sur la modélisation et l'exécution des workflows coopératifs.

Avant de parler de coordination, nous nous intéressons au contrôle de l'exécution correcte d'un ensemble d'activités coopératives. L'idée à l'origine de ce travail était non pas de considérer explicitement les activités à exécuter mais de considérer les états par lesquels doit passer le système pour atteindre un objectif défini.

Un exemple classique qui illustre le type de problème auquel nous cherchons à apporter une réponse est l'écriture d'un document à plusieurs, associé à un procédé de type édition/révision/soumission. La soumission a lieu lorsque la revue ne produit plus de nouveau commentaire, i.e. lorsque le document final a atteint un état qu'on peut considérer comme stable. Ce procédé est de type interactif (ce sont des utilisateurs qui sont impliqués et qui exécutent les activités), itératif (plusieurs cycles sont nécessaires pour atteindre l'objectif) et de longue durée. En outre, il peut parfois s'avérer nécessaire de tricher avec ce procédé pour accélérer le transit de données des éditeurs vers les relecteurs et des commentaires dans l'autre sens. Ceci n'est pas sans danger et nécessite donc des protocoles particuliers.

L'idée ici est de permettre un accès sûr à des données partagées, tout en contrôlant la façon dont évolue l'état de ces objets, plutôt que de décrire explicitement les activités à exécuter pour atteindre cet objectif. L'approche proposée tente d'éviter de décrire explicitement le programme de la coopération à l'aide d'un modèle de description de procédés classique

Nous avons proposé pour cela une approche hybride qui essaie de prendre en compte la dimension transactionnelle pour le partage des données et la dimension sémantique qui permet de garantir le bon comportement des transactions. La philosophie générale de l'approche est de laisser les utilisateurs coopérer, à l'initiative de chacun, à condition qu'ils acceptent de payer le prix des resynchronisations.

3.2 Présentation générale de l'approche

Cette approche a été initialement développée dans le cadre de l'environnement COO pour aider à la coopération dans le développement de logiciels.

De manière générale, dans un environnement de développement, les objets du développement (spécification, code, ...) sont stockés dans un référentiel (une base de données, un système de gestion de configuration ou un système de gestion de fichiers) et les utilisateurs associés aux activités accèdent à ces objets pour les modifier. Les participants activités sont amenées naturellement à coopérer afin de réaliser leurs objectifs.

3.2.1 Principe de correction dans COO

L'approche de COO [14] consiste à séparer le contrôle de la cohérence du référentiel en deux dimensions : *la correction des interactions coopératives* et *la correction individuelle des activités coopératives* [12, 13].

La correction des interactions coopératives se charge de régler les problèmes de concurrence d'accès aux données partagées entre les différentes activités. Pour cela, nous avons réutilisé le travail de la thèse de Pascal Molli[10]

La correction individuelle des activités coopératives se charge de prévenir des erreurs sémantiques qui peuvent être introduites par les utilisateurs au sein du chaque activité individuellement. Ce contrôle s'intéresse aux valeurs des produits (le produit est-il testé, le code est-il compilé sans erreurs ?) ainsi qu'aux séquences des valeurs des produits (le produit a-t-il été révisé avant d'être livré au client ?). Cette dimension de correction est assurée par un mécanisme de vérification des contraintes.

La figure 3.1 montre :

1. l'ensemble de toutes les exécutions ;
2. l'ensemble des exécutions coopératives dont les interactions sont correctes ;
3. l'ensemble des exécutions coopératives dont les activités sont individuellement correctes ;
4. l'ensemble des exécutions coopératives dont les interactions sont correctes et les activités sont individuellement correctes (en gris foncé). C'est cet ensemble d'exécutions qui nous s'intéresse.

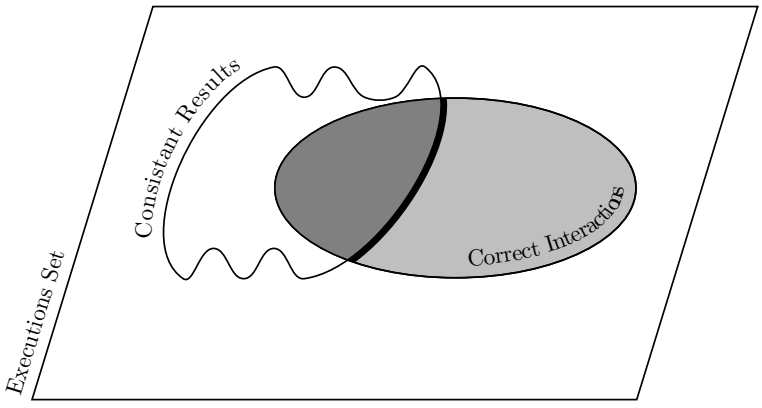


FIG. 3.1 – Ensembles d'exécutions

3.2.2 Correction des interactions coopératives

Pour régler les problèmes de synchronisation entre activités s'exécutant en parallèle, COO utilise une approche transactionnelle et chaque activité de développement se déroule au sein d'une transaction coopérative (COO-transaction) [10].

Les COO-transactions se distinguent des transactions ACID [5] classiques en relâchant la propriété d'isolation. Concrètement, deux transactions vont pouvoir coopérer en s'échangeant des données au cours de leurs exécutions au travers du référentiel. Les autres propriétés *atomicité*, *consistance* et *durabilité* sont conservées.

Le modèle des COO-transactions repose sur un critère de correction syntaxique : la COO-sérialisabilité [10]. Ce critère peut être vu comme une extension de la sérialisabilité pour supporter des exécutions coopératives telles que celles présentées. Il permet de garantir des propriétés générales sur les interactions. Ces propriétés sont : pas de lectures impropres ni de mises à jour perdues [11]. La correction des interactions coopératives est assurée par un protocole syntaxique (COO-protocole) basé sur la COO-sérialisabilité. Le COO-protocole contrôle l'échange des résultats entre les transactions coopératives et il s'appuie, intuitivement, sur les règles de synchronisation suivantes :

- si une transaction produit un résultat intermédiaire alors elle doit produire le résultat final correspondant ;
- si une transaction lit un résultat intermédiaire d'une autre transaction alors elle doit lire le résultat final correspondant ;
- si une transaction lit un résultat intermédiaire d'une autre transaction alors elle devient dépendante de cette transaction. Cette dépendance est levée si la transaction lit le résultat final correspondant ;
- en cas d'un cycle de dépendance entre les transactions, les transactions sont groupées pour former ce que nous appelons *un groupe de transactions coopératives*. Les transactions appartenant à ce groupe terminent leurs exécutions en même temps de manière indivisible.

Par exemple, l'exécution de la figure 3.2 est COO-sérialisable : A_2 a bien lu le résultat final de *graph* produite par la transaction A_1. Ceci implique que les interactions coopératives de A_1 et A_2 sont correctes.

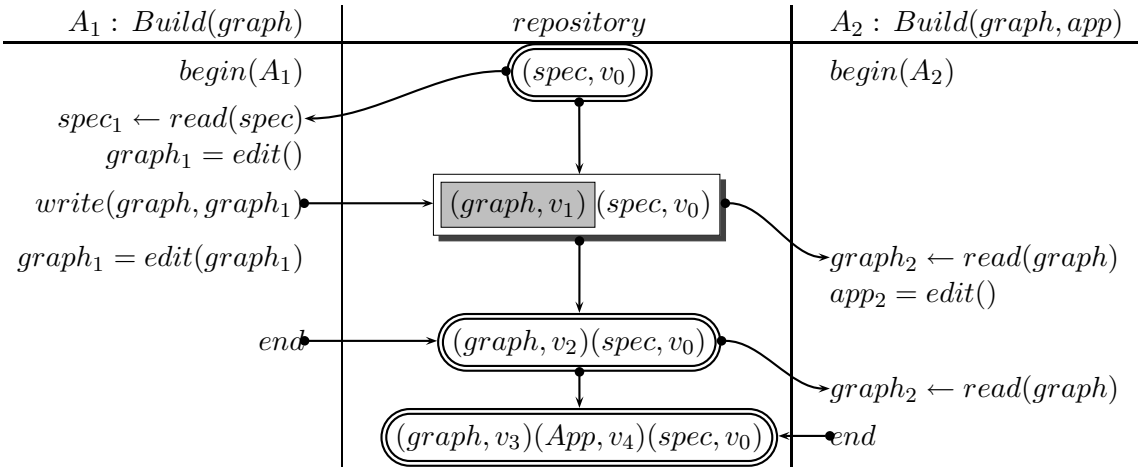


FIG. 3.2 – Exécutions coopératives

Supposons maintenant que la transaction A_2 n'ait pas lu le résultat final de *graph* produit

par A_1, ce qui donne l'exécution suivante :

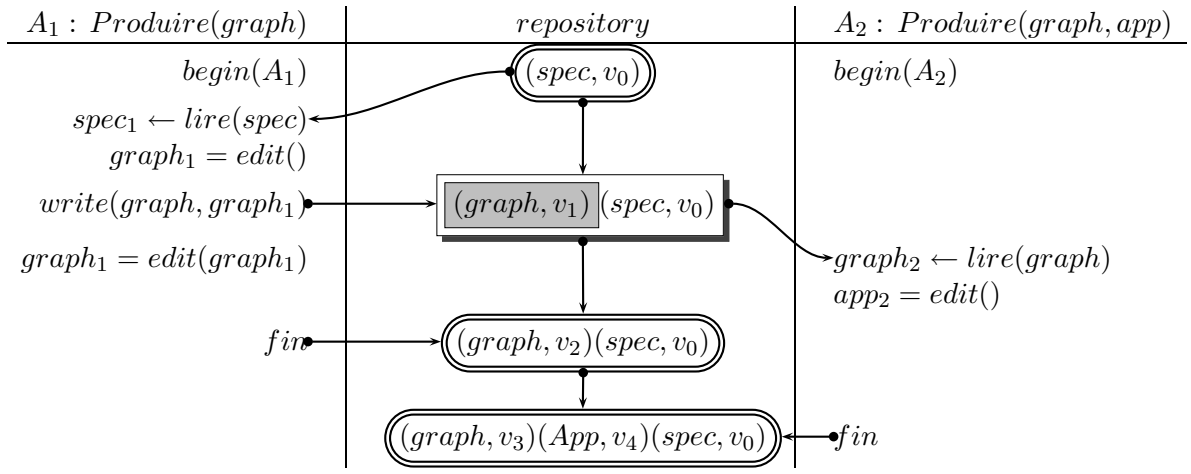


FIG. 3.3 – *Interaction coopérative incorrecte*

L'exécution de la figure 3.3 n'est pas COO-sérialisable : l'utilisateur de A_2 a lu un résultat intermédiaire de *graph* mais il n'a pas relu le résultat final correspondant. Il y a une lecture impropre sur l'objet *graph*. La transaction A_2 ne peut donc pas terminer son exécution. Dans ce cas là, l'opération de terminaison de la transaction est annulée et la transaction elle-même reste active. De cette façon, le protocole syntaxique de COO basé sur la COO-sérialisabilité n'oblige jamais les utilisateurs à annuler leur travail et garantit des propriétés générales sur les interactions [1, 11].

3.2.3 Correction individuelle des activités (idée)

Le protocole syntaxique de COO garantit la correction des interactions coopératives mais cet aspect n'est pas suffisant pour que l'utilisateur puisse travailler de manière sûre à l'abri des erreurs qui peuvent être provoquées par un mauvais déroulement des activités.

Reprenons l'exécution de la figure 3.2. Les interactions coopératives de cette exécution sont correctes au sens de COO : la transaction *Produire(graph,app)* a bien lu la valeur finale de *graph*. Mais nous n'avons aucune garantie sur la qualité de *graph* (*graph* est-il testé ?, *graph* est-il compilé sans erreurs ?) ni sur la façon dont il est fabriqué (*graph* a-t-il été compilé correctement avant d'être testé ?). C'est l'objectif de la correction individuelle d'apporter cette garantie.

3.3 Modélisation de la correction individuelle

Une façon d'assurer la correction individuelle des activités est de définir un ensemble de contraintes [2] dans le référentiel et de vérifier que les résultats des activités (transactions) respectent bien ces contraintes.

Les questions sont maintenant :

– quels types de contraintes choisir ?

– comment tester la validation des contraintes ?

[2] Les contraintes doivent exprimer la cohérence des objets contenus dans le référentiel et doivent empêcher l'occurrence d'un objet qui ne respecte pas ces contraintes.

Nous avons choisi un modèle de contraintes classique, mais nous verrons que sa mise en œuvre nécessite le développement de nouveaux algorithmes.

3.3.1 Quelles contraintes ?

Il est reconnu que la qualité d'un produit ne dépend pas seulement de ses valeurs mais aussi de son procédé de fabrication. Ainsi, les contraintes doivent garantir la qualité des produits logiciels et la qualité des procédés de fabrication. Elles garantissent des propriétés sur les valeurs des produits logiciels et des propriétés sur la façon dont ces valeurs sont produites.

Nous avons identifié deux types des contraintes : *les contraintes de sécurité* et *les contraintes de vivacité.*

Contraintes de sécurité

Elles décrivent les propriétés des objets qui doivent être valides dans chaque état du référentiel. Elles peuvent être vues comme des contraintes d'intégrité statiques dans le domaine des bases de données. Ces contraintes représentent, en quelques sorte, les critères de qualité. Par exemple : " le code de chaque module est compilé". Intuitivement, cette contrainte exprime le fait qu'à chaque instant le code de tous les modules doit être compilé. Cette contrainte sera vérifiée chaque fois que le code d'un module sera modifié.

Les *contraintes de sécurité* ne permettent pas de décrire des contraintes sur le procédé de fabrication *i.e.* sur les transitions entre les états et sur les séquences d'états. Par exemple, nous ne pouvons pas définir une contrainte de la forme "le produit n'est jamais livré avant d'être testé". Cependant, ce type de contraintes est nécessaire. Pour cela, nous avons introduit les *contraintes de vivacité.*

Contraintes de vivacité

Ces contraintes décrivent les propriétés qui doivent être valides sur une séquence d'états du référentiel. Elles peuvent être vues comme des contraintes dynamiques dans le domaine des bases de données [8]. Elles représentent, en quelques sortes, les critères de qualité des procédés de fabrication d'un produit. Par exemple :" le code d'un module n'est jamais produit avant que la conception soit approuvée". Cette contrainte signifie qu'il est nécessaire d'avoir un état du référentiel dans lequel la conception du module est approuvée avant d'avoir un état du référentiel dans lequel le code de ce module est développé.

Représentation des contraintes

Nous représentons les contraintes de vivacité et de sécurité sous la forme de formules de logique temporelle. Notre modèle s'appuie sur le travail de Lipeck et Zen [7]. Chaque contrainte est représentée par :

$$c_i(x_1 : t_1, \ldots, x_n : t_n) : \phi(x_1, \ldots, x_n)$$

où c_i est l'identificateur de la contrainte, $x_1, \ldots, x_n$ sont des variables libres, $t_1, \ldots t_n$ sont respectivement les types de $x_1, \ldots x_n$ (nos objets sont typés, chaque objet a un type, par exemple, *modA : Module* signifie que l'objet *modA* est de type Module).

ϕ est la formule à vérifier. C'est une formule de logique de premier ordre avec des opérateurs temporels (always, sometime, after, before ...). Par exemple, la contrainte de sécurité *la conception de chaque module est approuvée* peut-être définie par :

$c_1(concept : Conception)$: **always** [3] approuvée(concept)

La variable *concept* est de type *Conception*, *approuvée* est un prédicat qui vaut vrai si la conception est approuvée et faux dans le cas contraire. Cette contrainte signifie que tous les objets de type *Conception* sont approuvés.

3.4 Maintien des contraintes et transactions coopératives

Définir des contraintes sémantiques pour garantir des propriétés sur des valeurs ou sur une séquence de valeurs n'est pas nouveau. Ceci est souvent utilisé dans le domaine des bases de données [6]. Nous pouvons comparer les contraintes sémantiques que nous voulons intégrer dans notre environnement aux contraintes d'intégrité des bases de données. La question que nous posons maintenant est : est-ce que les mécanismes classiques de maintien des contraintes dans les bases de données sont transposables dans notre environnement ?

3.4.1 Inadéquation des mécanismes classiques

Principes de base des mécanismes

De façon générale, les mécanismes classiques de maintien des contraintes suivent les principes suivants [3, 9] :

- les contraintes sont relâchées pendant l'exécution de la transaction et elles sont testées à la fin. Ceci signifie que les états intermédiaires produits pendant l'exécution de la transaction ne sont pas soumis aux contraintes ;
- les contraintes susceptibles d'être mises en défaut par la transaction doivent être évaluées à sa terminaison. Nous nous basons sur l'hypothèse que la base se trouve dans un état cohérent avant la transaction. Cette hypothèse réduit considérablement le nombre des contraintes à évaluer ;
- si à sa terminaison la transaction ne respecte pas les contraintes, elle est annulée.

Ces principes reposent sur le fait que les transactions classiques sont isolées [5]. Une transaction ne peut pas observer les états intermédiaires de la base produits durant l'exécution d'une autre transaction. En se basant sur les principes précédents et sur la propriété d'isolation, nous pouvons dire que chaque transaction est une transformation cohérente de la base : elle prend la base dans un état cohérent dans lequel toutes les contraintes sont vérifiées et la rend dans un autre état cohérent.

Prenons par exemple une contrainte d'intégrité définie dans la base de données d'une entreprise : "le salaire d'un employé doit être supérieur au SMIC". Le mécanisme du maintien des contraintes testera cette contrainte chaque fois que le salaire sera modifié. Si la modification des salaires entraîne le non respect de la contrainte, la transaction sera annulée.

Limitations des mécanismes classiques

Malheureusement, ces mécanismes ne sont pas directement transposables dans un milieu coopératif :

1. les transactions coopératives ne sont pas isolées. Une transaction peut voir les résultats intermédiaires des autres transactions : comment évaluer les contraintes en présence des résultats intermédiaires ?

[3] Pour simplifier, les contraintes de sécurité sont représentées, par la suite, sans l'opérateur temporel **always**.

2. l'annulation systématique des transactions en cas de non respect des contraintes n'est pas acceptable dans notre contexte où les transactions sont de longue durée : plusieurs jours à plusieurs mois. Les contraintes ne doivent pas provoquer l'annulation involontaire des transactions.

Ainsi avons-nous besoin de définir un nouveau mécanisme pour maintenir les contraintes dans un milieu coopératif. Ce mécanisme doit être adapté à la nature des transactions coopératives : il ne doit forcer les transactions à annuler leur travail, ni bloquer la coopération mais au contraire la favoriser. Enfin, il ne doit pas entrer en conflit avec le protocole syntaxique (cf 3.2.2)

3.4.2 Principe de base de mécanismes de maintien des contraintes en milieu coopératif

Pour vérifier les contraintes, nous allons appliquer les principes suivants [16] :
- les contraintes susceptibles d'être mises en défaut par la transaction sont vérifiées à la fin de la transaction. Ces contraintes sont dites *contraintes critiques*. Une contrainte est critique par rapport à une transaction si elle peut être instanciée par des objets manipulés par la transaction. (On se base sur les types de variables dans l'interface des contraintes et les types d'objets manipulés par la transaction pour trouver ces contraintes).
- si, à sa terminaison, la transaction ne respecte pas ces contraintes critiques, seule l'opération de terminaison est annulée et l'utilisateur continue à travailler jusqu'à la satisfaction des contraintes. Ainsi, le mécanisme de maintien des contraintes ne force pas l'annulation des transactions mais les forcent à converger vers des résultats cohérents.

Les résultats intermédiaires, produits en cours d'exécution, échappent aux contraintes. En fait, imposer que les résultats intermédiaires respectent toutes les contraintes nie l'idée de résultat intermédiaire [4]. Les contraintes ne réduisent pas la flexibilité de l'environnement : même en présence de contraintes, l'utilisateur d'une transaction coopérative peut travailler à sa guise et surtout coopérer (échanger des résultats intermédiaires).

Impacts de la coopération

La visibilité des résultats intermédiaires au moment de l'évaluation des contraintes rend le degré de cohérence garanti par les transactions coopératives plus faible que celui assuré par les transactions traditionnelles. En fait, à la terminaison d'une transaction coopérative seules ses contraintes critiques sont vérifiées :
- sur la base des résultats finaux de la transaction ;
- mais il peut exister dans le nouvel état du référentiel, produit à la terminaison de la transaction en publiant ses résultats finaux, des résultats intermédiaires (pas forcement cohérents) venant d'autres transactions en cours.

Toutes les contraintes ne sont alors pas vérifiées dans ce nouvel état et le référentiel n'est cohérent par rapport à toutes les contraintes qu'à la fin de toutes les transactions.

Prenons l'exemple suivant. Supposons que nous ayons les transactions t_1 *: Modifier(conceptA)* et t_2 *: Modifier(codA)*. t_1 modifie la conception du module A, et t_2 modifie le code du module A. Soient les contraintes suivantes :
- c_1 : toutes les conceptions sont approuvées. Cette contrainte peut-être exprimée par : "$c_1(concept : Conception) : appr(concept)$", *appr* est un prédicat qui vaut vrai si la variable *concept* est approuvée et faux sinon. Seules les transactions qui manipulent les "conceptions" peuvent instancier la contrainte c_1 ;

– c_2 : tous les codes sont compilés sans erreurs. Cette contrainte peut-être exprimée par : "$c_2(cod : Code) : compilé(cod)$", *com* est un prédicat qui vaut vrai si la variable *cod* est compilée correctement et faux sinon. Seules les transactions qui manipulent les "codes" peuvent instancier la contrainte c_2.

Les contraintes c_1 et c_2 sont définies dans le référentiel. Considérons l'exécution de t_1 et t_2 de la figure 3.4.

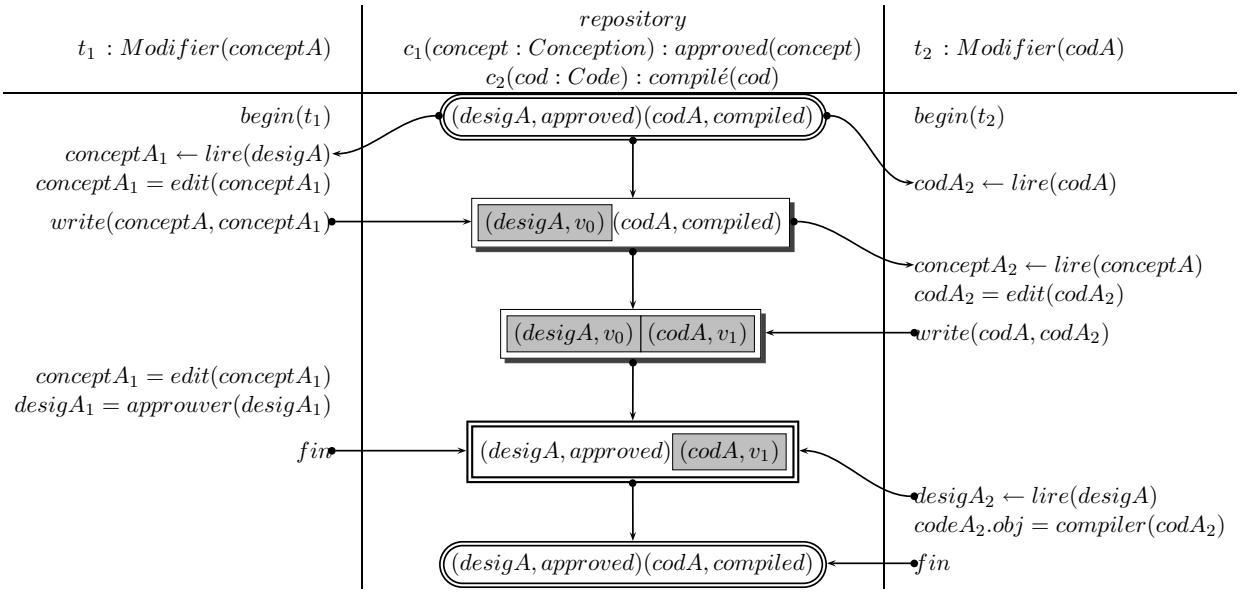


FIG. 3.4 – *Les états du référentiel produisent à la terminaison de t_1, t_2*

– t_1 modifie la conception de module A (*conceptA*), puis elle produit un résultat intermédiaire de *conceptA* ;
– t_2 lit cette valeur intermédiaire, elle modifie le code puis elle publie un résultat intermédiaire du code ;
– après quelques modifications de la conception, t_1 essaye de terminer son exécution. t_1 peut terminer son exécution : le protocole syntaxique (cf 3.2.2) est vérifié (t_1 n'a lu aucun résultat intermédiaire de t_2) et sa contrainte critique c_1 est vérifiée (le résultat final de t_1, la conception du module A (*conceptA*), est approuvée) ;
– l'état du référentiel produit à la terminaison de t_1 est cohérent par rapport à la contrainte c_1. Nous symbolisons un tel état par : $(desigA, approved)(codA, \neg com)$. Dans cet état, *conceptA* respecte bien la contrainte c_1, mais l'objet *codA* est un résultat intermédiaire et nous n'avons aucune garantie sur cet objet. Le fait d'avoir des résultats intermédiaires venant de la transaction t_2 dans l'état produit par t_1 n'est pas gênant car l'objet *codA* deviendra final (cohérent) à la terminaison de la transaction t_2 comme le montre la figure 3.4 ;
– à la fin de t_1, l'objet *conceptA* est cohérent et à la fin de la transaction t_2 l'objet *codA* est cohérent. Ainsi, à la fin de t_1 et t_2 tous les objets du référentiel sont cohérents, le référentiel est donc dans un état cohérent.

Bien que le degré de cohérence assuré par les transactions coopératives soit plus faible que celui des transactions traditionnelles, il est suffisant dans un milieu coopératif. Il permet de garantir à l'utilisateur qu'il est sur la bonne voie, et qu'il pourra atteindre dans le futur un état cohérent par rapport aux contraintes.

En d'autres termes, en milieu coopératif, seuls les objets modifiés par la transaction doivent être cohérents. Par contre, il faut garantir que tous les objets seront cohérents à la fin de toutes les transactions. Cela est assuré d'une part par le protocole syntaxique (cf 3.2.2) et d'autre part, par le mécanisme de maintien des contraintes qui garantit la cohérence des résultats finaux.

Dans la section qui suit, nous verrons une autre conséquence de la rupture de l'isolation des transactions : des résultats intermédiaires dans le référentiel peuvent empêcher l'évaluation immédiate des contraintes.

3.4.3 Stratégies d'évaluation des contraintes en présence de résultats intermédiaires

La visibilité des résultats intermédiaires introduit des difficultés supplémentaires dans la vérification des contraintes. Avant de tester la validation des contraintes, il est nécessaire de tester si elles sont évaluables *i.e.* de tester si tous les objets impliqués dans cette contrainte sont des résultats finaux (rappelons que les résultats intermédiaires ne sont pas soumis aux contraintes).

Prenons, par exemple, la contrainte c_1 décrivant le fait que la valeur de x est toujours inférieure à celle de y $(x < y)$(c_1 peut exprimer par exemple un critère de qualité sur x et y qui peuvent être le nombre de lignes des documents de spécification, de conception, de code ...).

Considérons que la valeur initiale de l'objet x est égale à 100, et la valeur initiale de y est égale à 150. Supposons que la transaction t_1 modifie l'objet x et la transaction t_2 modifie l'objet y, comme le montre la figure 3.5.

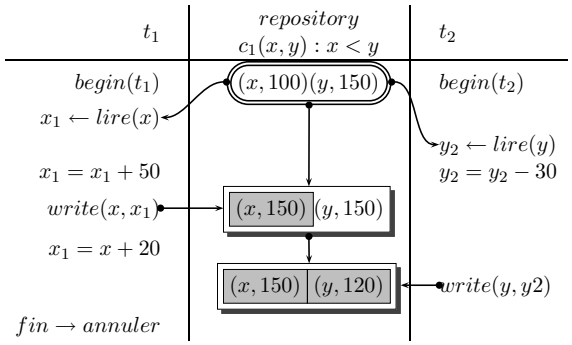


FIG. 3.5 – *La contrainte c_1 n'est pas évaluable à la terminaison de t_1*

- t_1 publie un résultat intermédiaire de x et t_2 publie un résultat intermédiaire de y.
- Après quelques modifications, t_1 décide de terminer son exécution. Il faut donc déterminer ses contraintes critiques, tester si elles sont évaluables et les valider.
- La contrainte c_1 est critique pour la transaction t_1, mais c_1 n'est pas évaluable car l'objet y impliqué dans c_1 est un résultat intermédiaire de t_2. Dans une telle situation la transaction t_1 ne peut pas valider c_1.

Pour résoudre ce type de problème, les trois stratégies suivantes peuvent être considérées [19] :

1. la transaction termine son exécution sans évaluer ses contraintes critiques ;
2. elle évalue ses contraintes critiques sur les dernières valeurs cohérentes disponibles des objets ;
3. elle retarde sa terminaison jusqu'à ce qu'elle puisse arriver à évaluer ses contraintes.

La première solution présente le risque qu'il ne soit jamais possible d'atteindre un résultat correct. La seconde stratégie fonde la correction sur des objets de "générations" différentes par rapport à la marche normal du procédé. La dernière solution peut entraîner dans certains cas un "interblocage" entre les transactions. En fait, arriver à un interblocage entre les transactions signifie que chaque transaction a besoin de l'autre. Une solution à ce problème d'interblocage est de faire coopérer les transactions [4] pour qu'elles puissent terminer leur exécution simultanément tout produisant un état correct du référentiel.

Contraintes et groupe de transactions

Dans le cas d'un cycle de dépendance entre les transactions coopératives, les transactions sont groupées par le protocole syntaxique et forment alors ce que nous appelons un *groupe de transactions coopératives* [10]. Un cycle de dépendance apparaît entre deux transactions lorsque la première transaction accède à un résultat intermédiaire de la seconde transaction et réciproquement. Les transactions appartenant à *un groupe de transactions coopératives* terminent leur exécution de manière indivisible et produisent un seul état du référentiel.

Vis-à-vis des contraintes, un *groupe de transactions coopératives* peut-être vu comme une seule transaction, comme si les transactions appartenant au groupe n'avaient jamais existé individuellement.

Le groupe de transactions termine son exécution lorsque la dernière transaction du groupe termine son exécution. Les contraintes critiques de groupe sont l'ensemble des contraintes critiques de toutes les transactions du groupe. Nous pouvons traiter la terminaison de la dernière transaction de groupe comme celle d'une transaction ordinaire : à sa terminaison elle doit produire un état "cohérent" du référentiel. Dans cet état toutes les contraintes critiques des transactions du groupe doivent être évaluables et valides [16].

3.4.4 Synthèse

Nous avons présenté l'impact de la rupture de l'isolation et du regroupement des transactions coopératives sur la vérification des contraintes. Dans un environnement coopératif, les contraintes ne sont pas toujours évaluables. Ainsi, à la terminaison d'une transaction coopérative, il est nécessaire de tester d'abord si les contraintes sont évaluables avant de tester leur validation.

L'algorithme du maintien des contraintes dans un environnement coopératif est défini de la façon suivante :

Définition 3.4.1

Si (Contraintes non-évaluables) **alors**
    Appliquer stratégie 2 ou 3
Sinon
    Si (contraintes non-vérifiées) **alors**
        Continuer à travailler
    Sinon
        Produire un nouvel état (Commit)
    fsi
fsi

[4] Bien sûr, c'est de la responsabilité individuelle de chaque utilisateur de coopérer avec les autres utilisateurs pour qu'il puisse finir son travail. L'environnement ne force pas l'utilisateur à coopérer.

Dans le cas où les contraintes sont évaluables, il faut tester leur validation ; si les contraintes ne sont pas valides alors c'est la responsabilité de l'utilisateur de continuer à travailler jusqu'à la satisfaction des contraintes.

3.5 Conclusion

L'approche proposée est une première étape vers la coordination d'activités coopératives. Elle permet de décrire les états par lesquels un système doit passer pour atteindre un objectif. Combiné à un système permettant la modification concurrente de données, l'approche fournit la garantie que si l'état final est atteint, toutes les contraintes (statiques et dynamiques) ont été respectées. La coopération au sens où nous l'entendons est possible. En revanche, en ce qui concerne la coordination, le problème n'est qu'en partie traité. En effet, les contraintes et le protocole transactionnel garantissent une exécution correcte de l'activité coopérative mais ils ne fournissent que peu de support aux protagonistes concernant une réelle coordination.

Cette approche a montré également un effet de bord intéressant. En effet, il est possible de trouver des cas ou la coopération entre activités permet d'atteindre un état cohérent. Il est également possible de trouver des cas ou le contrôle des contraintes entraine des échanges de données entre activités qui permettent d'éviter des itérations.

La faisabilité de cette proposition a été montrée par le développement d'un prototype intégrant les deux dimensions et par quelques expérimentations.

Cependant, le travail dans cette direction a été interrompu depuis quelques années. En effet, il est possible de décrire des situations complexes à l'aide de la logique temporelle mais plus le nombre d'expression augmente plus il devient difficile de comprendre ce qui se passe et quelles sont les opérations nécessaires pour passer d'un état à un autre. Une façon de faire consiste à visualiser le graphe utilisé par l'algorithme d'évaluation des contraintes mais celui ci devient également rapidement très compliqué. C'est pourquoi, nous nous sommes tournés vers des solutions adaptées des modèles de workflow. L'approche workflow n'est pas déclarative et oblige de décrire explicitement les opérations à effectuer pour atteindre un objectif, mais elle a l'avantage d'être plus simple à mettre en oeuvre et plus compréhensible par l'utilisateur final. C'est ce que nous allons voir dans la partie suivante.

3.6 Publications

[15, 19, 16, 16, 17, 2, 18]

Bibliographie

[1] G. Canals, P. Molli, and C. Godart. Concurrency control for cooperating software processes. In *Proceedings of the 1996 Workshop on Advanced Transaction Models and Architecture (ATMA'96)*, Goa, India, 1996.

[2] Gérôme Canals, Claude Godart, François Charoy, Pascal Molli, and Hala Skaf. Coo approach to support cooperation in software developments. *IEE Proceedings - Software*, 145(2-3) :79–84, Jun 1998.

[3] M. Gertz. *Diagnosis and Repair of Constraint Violations in Database Systems.* PhD thesis, Institut für Informatik Universität Hannover, 1996.

[4] C. Godart. COO : a Transaction Model to support COOperating software developers COOrdination. In *4th European Software Engineering Conference (ESEC4), Garmisch, LNCS 717*, 1993.

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

[6] P.W. Grefen and P.M. Apers. Integrity Control in Relational Database Systems – An Overview. *Data and Knowledge Engineering, 10(2)*, pages 187–223, 1993.

[7] U. W. Lipeck and H. Zhou. Monitoring Dynamic Integrity Constraints on Finite State sequences and Existence Intervals. In G.Saake J.Goers, A.Heuer, editor, *Proceedings of the 3rd International Workshop on Foundation of Models and Languages for Data and Objects*, pages 115–130, 1991.

[8] Udo W. Lipeck. Stepwise specification of dynamic database behaviour. In *SIGMOD '86 : Proceedings of the 1986 ACM SIGMOD international conference on Management of data*, pages 387–397, New York, NY, USA, 1986. ACM Press.

[9] H. Martin. *Contrôle de la cohérence dans les bases objets : Une approche par le comportement*. PhD thesis, Joseph Fourier-Grenoble 1, 1991.

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

[11] Pascal Molli, Manuel Munier, Gérôme Canals, François Charoy, and Claude Godart. Cooserializability : A correctness criterion for cooperative executions. Rapport de recherche, 1997.

[12] K. Ramamritham and P.K. Chrysanthis. In Search of Acceptability Criteria : Database Consistency Requirements and Transaction Correctness Properties. In Özsu, Dayal, and Valduriez, editors, *Distributed Object Management*. Morgan Kauffman, 1993.

[13] K. Ramamritham and P.K. Chrysanthis. A taxonomy of correctness criteria in database applications. *The VLDB Journal*, 5(5) :85–97, 1996.

[14] Hala Skaf. *Une approche hybride pour gérer la cohérence dans les environnements de développement coopératif*. Thèse en informatique, Université de Nancy I – Centre de Recherche en Informatique de Nancy, 1997.

[15] Hala Skaf, F. Charoy, and Claude Godart. Une approche hybride pour contrôler les activités de développement de logiciel. Rapport interne, Centre de Recherche en Informatique de Nancy, Vandoeuvre-lès-Nancy, 1996.

[16] Hala Skaf, François Charoy, and Claude Godart. An hybrid approach to maintain consistency of cooperative software development activities. In *SEKE97 The Ninth International Conference on Software Engineering and Knowledge Engineering, Madrid*, jun 1997.

[17] Hala Skaf, François Charoy, and Claude Godart. Flexible integrity control of cooperative applications. In IEEE Computer Society, editor, *The Ninth International Workshop on Database and Expert Systems Applications - DEXA 98, Vienne, Autriche*, 1998.

[18] Hala Skaf, François Charoy, and Claude Godart. Maintaining shared workspaces consistency during software development. *International Journal of Software Engineering and Knowledge Engineering*, 9(5) :623–642, Oct 1999.

[19] Hala Skaf, Francois Charoy, and Claude Godart. Maintaining consistency of cooperative software development activities. In *Integrity in Database 6th International Workshop on Foundations of Models and Languages for Data and Objects, Dagstuhl*, sep 1996.

Chapitre 4

Execution flexible de procédés coopératifs

4.1 Introduction

Les modèles de workflow sont depuis leur origine destinés à la modélisation et à l'automatisation des procédés administratifs, de production ou de services. Ces procédés sont utilisés pour coordonner des activités bien définies qui s'exécutent de manière isolée. Si on peut considérer aujourd'hui qu'ils sont de mieux en mieux adaptés à ce genre de tâche, leur limites sont rapidement atteintes lorsqu'il s'agit de les appliquer à d'autres classes d'applications et en particulier les activités coopératives.

C'est à ce type d'activités et aux moyens de les coordonner que nous nous sommes intéressés ici. Pour ce travail, nous avons fait plusieurs hypothèses :

- Les modèles de façon générale sont relativement simples à comprendre et sont finalement assez adaptés à la description d'enchainements d'activités, y compris des activités coopératives. Nous considérons donc qu'il n'est pas vraiment nécessaire d'agir au niveau des modèles.
- Il est possible en revanche de les rendre plus flexibles en changeant la façon de les interpréter et en particulier en relachant les conditions d'exécution d'une activité.

L'approche proposée ici consiste à permettre une représentation explicite du "programme d'un processus coopératif" à l'aide d'un formalisme simple et compréhensible par ses utilisateurs. Un processus coopératif sera représenté, comme dans les systèmes de workflow, par un ensemble d'activités (les tâches à effectuer pour atteindre l'objectif) et par des dépendances entre ces activités. C'est en réduisant les contraintes sur l'exécution du programme et sur son évolutivité que nous comptons pour permettre une approche plus flexible et donc adaptée au contexte auquel nous nous intéressons.

De plus, nous reprenons l'idée d'espace de travail partagé et donc de transaction coopérative pour contrôler les échanges de données entre les activités d'un processus. Ici encore, une activité correspondra à une transaction coopérative et sera contrôlée par le protocole transactionnel Coo.

4.2 Quels besoins pour l'ingéniérie coopérative

Les modèles de workflow traditionnels permettent de décrire des activités et leurs enchainements et les conditions de leur activation[16]. Ils font en général une hypothèse forte sur l'exécution des activités : une activité dans un système de gestion de workflow est de courte

durée et son exécution doit être atomique. Ainsi l'exécution d'un procédé est une succession d'exécutions d'activités respectant les propriétés transactionnelles classiques. Elles ne sont donc pas sujettes aux problèmes de concurrence traditionnelles qui sont ainsi réglés.

Dans le cadre d'activités coopératives, cette approche n'est bien sure pas adaptée. Une activité dans un processus coopérative peut être démarrée, interrompue, reprise.

4.2.1 Besoin d'anticipation

La plupart du temps, lors d'un processus coopératif, les activités se chevauchent. Certaines peuvent démarrer alors que les précédentes ne sont pas totalement terminées et n'ont fourni que des résultats partiels ou intermédiaires. Cette façon de travailler est souvent mise en oeuvre dans de nombreuses activités créatives en considérant que le travail fait à partir d'un résultat non définitif mais suffisamment avancé ne sera jamais complétement remis en cause. Ainsi, des activités peuvent commencer même si toutes les conditions pour leur exécutions ne sont pas complétement réunies :

- Les activités précédentes ne sont pas terminées (il est possible de travailler sur des brouillons)
- les conditions d'activation sont fausses ou ne peuvent pas être testées (une autorisation manque, un test ne passe pas)
- Il manque des données en entrée mais celles disponibles peuvent permettre de travailler

Permettre à des activités de démarrer dans de telles conditions est ce que nous avons appelé l'anticipation. La figure 4.1 montre une exécution sans et avec anticipation. Dans le second cas, l'activité d'édition fournit une brouillon qui permet à la revue de commencer plus tôt. Le processus a une chance de terminer plus tôt.

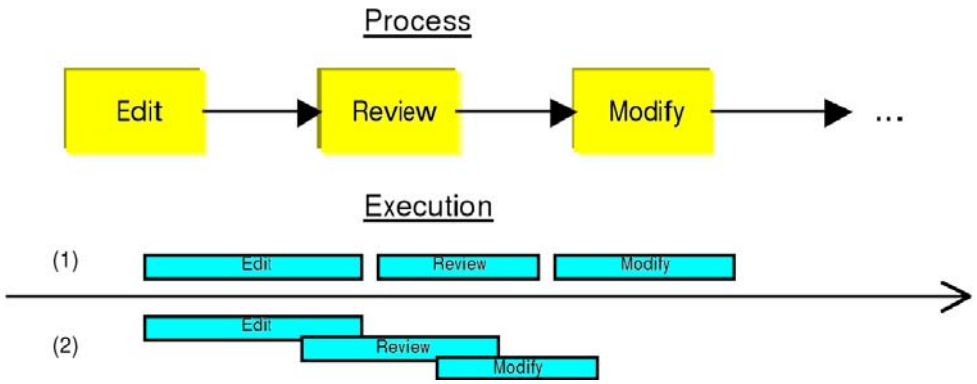


Fig. 4.1 – Exécution avec (1) ou sans (2) anticipation

En plus de la possibilité de démarrer plus tôt, l'anticipation peut être utilisée pour fournir des retours aux activités précédentes sur leurs résultats potentiels. Dans un processus coopératif, il est naturel de penser que l'utilisateur d'un résultat puisse fournir des commentaires ou des remarques au producteur de ce résultat si celui-ci n'a pas encore considéré sa tache comme terminée. C'est bien évidemment un effet non controlé de la faculté d'anticipation, mais c'est une façon de procéder relativement courante. Dans un cycle de développement classique, les développeurs travaillent à partir de spécifications abouties. S'ils ont la possibilité d'obtenir des versions préliminaires de la spécification, ils ont alors la possibilité de demander des modifications en dehors du cycle formel de validation et ainsi d'accélerer le processus. La figure 4.2 illustre ce phénomène.

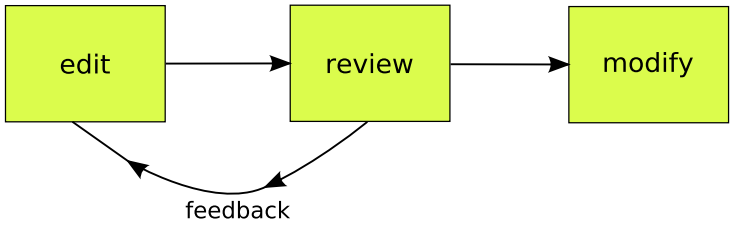


FIG. 4.2 – Feedback

Le modèle proposé ici est donc bati à partir d'un environnement qui garanti la cohérence des données dans un espace partagé et un modèle d'exécution flexible, incluant l'idée d'anticipation.

Ici aussi, nous avons utilisé le modèle des transactions coopératives de Coo pour garantir les échanges de résultats entre activités concurrentes. Les activités sont longues, elles sont complexes et ne peuvent donc être réduites à de simples transactions atomiques. Elles doivent pouvoir publier des résultats au cours de leur exécution. Il faut pouvoir contrôler ces échanges.

L'exécution en parallèle d'activités susceptibles d'interagir n'est pas réellement prise en compte dans les systèmes de gestion de workflow classiques. Nous avons proposé, plutôt que de définir un nouveau modèle, de permettre une interprétation plus flexible des "scripts" que celle des systèmes classiques. Nous considérons donc qu'il est possible de représenter des procédés coopératifs comme des procédés administratifs ou de production classique et qu'il suffit d'en rendre l'exécution plus flexible pour qu'ils deviennent utilisables dans de nouveaux contextes en augmentant le parallélisme et les interactions entre les activités.

4.3 Workflow Coopératifs : l'approche Coo

4.3.1 Introduction

L'approche que nous avons proposée est donc basée d'un part sur un modèle de transaction coopératives qui permet de relâcher l'isolation de l'exécution d'activités et d'autre part sur un modèle de procédé étendu pour permettre l'anticipation. Une activité n'est donc plus vue comme une boite noire qui prend un résultat en entrée et qui produit un résultat en sortie mais comme une boite "grise" qui peut produire des résultats intermédiaires au cours de son exécution et qui peut "interagir" avec les autres activités.

Nous ne reviendrons pas ici sur le modèle de transactions, qui a déjà été rapidement décrit dans la partie précédente.

4.3.2 Anticipation

Dans les systèmes de workflow traditionnels, une activité démarre uniquement si les précédentes sont terminées. Ce comportement est justifié par le fait que les activités sont courtes et atomiques. Relativement à l'exécution du processus, il est même possible de négliger leur durée, ce que font la plupart des modèles. Cette hypothèse n'est pas tenable pour des activités coopératives où la durée d'exécution des activités peut durer plusieurs jours et où il sera fréquent de pouvoir observer l'exécution simultanée de plusieurs activités. Nous considérons comme nous l'avons déjà dit que les exécutions d'activités successives peuvent se chevaucher.

L'anticipation est le moyen que nous proposons pour permettre ce comportement tout en conservant les avantages d'une coordination explicite de l'exécution des activités. Elle doit permettre le démarrage d'une activité plus tôt par rapport au flôt de contrôle décrit par le modèle

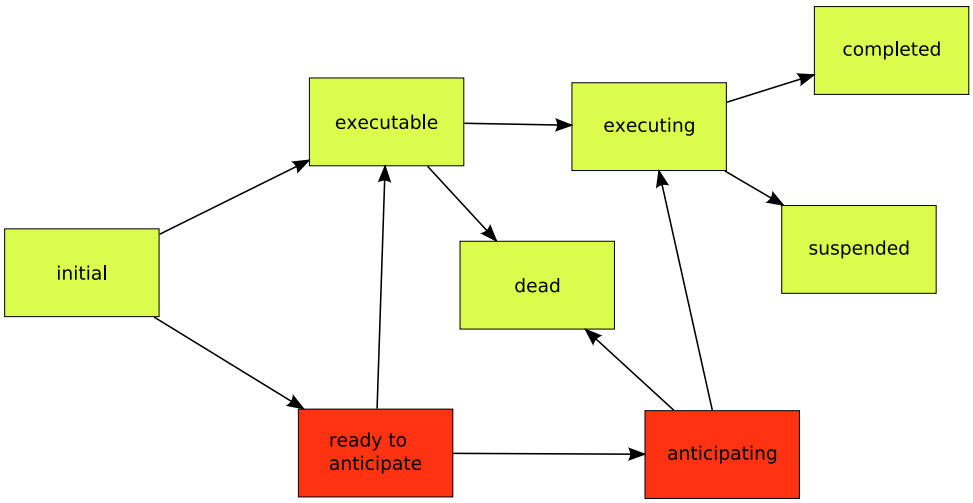


FIG. 4.3 – Les états d'une activité

de procédé. L'idée est que même si une activité démarre son exécution alors que toutes les conditions pour son démarrage ne sont pas remplies, à un moment, ces conditions le seront et elle passera alors dans un état d'exécution normal, comme s'il n'y avait pas eu d'anticipation. Ainsi, nous faisons l'hypothèse qu'ayant démarré son exécution plus tôt que prévue, elle se terminera plus tôt. L'anticipation permet une exécution plus flexible tout en préservant l'ordre de terminaison des activités.

Ce comportement est relativement classique, même dans des activités peu coopérative. Nous avons souvent l'occasion de la pratiquer en particulier dans les cas où le résultat d'une activité est prévisible. Ainsi, lorsqu'un chercheur doit partir en mission, il doit demander une autorisation d'absence et un ordre de mission avant de préparer son voyage. Le processus enchaine donc l'activité de demande d'ordre de mission, suivi par la préparation du voyage. En général, nous n'attendons pas l'ordre de mission pour faire les réservations d'avion et d'hotel, celle-ci pouvant prendre du temps à être signée, mais nous faisons l'hypothèse que la mission sera signée à partir du moment où un accord verbal, informel a été obtenu qui permet de démarrer le processus.

La possibilité d'anticiper tout en conservant un contrôle normal de l'exécution demande des modifications au modèle d'exécution d'un procédé. Nous avons donc étendu le modèle traditionnel pour prendre en compte l'anticipation. Deux nouveaux états ont été ajoutés : prêt à anticiper et anticipable. Le premier état indique que l'activité peut être démarrée prématurément, le second qu'elle a démarrée alors que toutes ses conditions normales de démarrage n'ont pas été vérifiées. La figure 4.3 montre le diagramme de transition des états d'une activité prenant en compte ces nouveaux états.

Le passage d'une activité de l'état initial à l'état prêt à anticiper peut se faire selon plusieurs stratégies :

1. Anticipation libre : un activité peut anticiper n'importe quand (l'état initial et l'état prêt à anticiper sont les mêmes). Dans ce cas, une activité peut terminer son exécution quand les précédentes ont terminées.
2. Anticipation dictée par le flot de contrôle - une activité peut démarrer quand ses prédécesseurs ont démarré (sont dans l'état anticipating ou executing). Dans ce cas on retrouve une dépendance entre le démarrage d'une activité et le démarrage de l'activité suivante. Le modèle d'exécution assure également que la terminaison d'une activité ne peut se faire

que si les activités précédentes ont terminées.

3. Anticipation dictée par le flot de données - Une activité peut démarrer seulement si toutes ses données en entrée sont disponibles, y compris sous une forme intermédiaire. activities).

4. Anticipation dictée par le flot de données et de contrôle - les deux précédentes conditions sont nécessaires pour démarrer une activité, toutes les activités précédentes ont démarré et les données en entrée de l'activité existent.

Ces différentes stratégies donnent plus ou moins de souplesse à l'exécution. Celles qui mettent en jeu les données sont plus compliquées à mettre en place parce qu'elles nécessitent de scruter en permanence les changements d'état du système. L'anticipation dictée par le flot de contrôle est un bon compromis qui fournit à la fois un certain contrôle et donne la flexibilité nécessaire.

La garantie principale fournie par le modèle d'exécution, c'est qu'à un moment de l'exécution du procédé, chaque activité passera par l'état executing, i.e. que toutes ses conditions d'activations ont été remplies. L'ajout de l'état d'anticipation, par rapport à un type de dépendance particulier entre les activités garanti qu'à la fin de l'exécution, si l'on retire les état correspondant à une anticipation, l'exécution du procédé suit le modèle classique.

Ainsi, le modèle est plus souple et plus sur que d'autre modèles qui propose des relations de dépendance différentes. Ce besoin a été exprimé dans plusieurs articles [10, 15]. Contrairement à ces approches, où les politiques de coordination sont directement décrites dans les modèles, l'anticipation permet d'interpréter les flot de contrôle de façon flexible à l'exécution.

4.3.3 Anticipation et transactions coopératives

Un processus coopératif peut être mis en oeuvre en encapsulant chaque activité d'un processus dans une transaction. La flexibilité de l'exécution permet aux utilisateurs d'adapter leur mode de travail à leur besoin tout en restant sous le contrôle du gestionnaire de procédés. Le système transactionnel garanti l'accès concurrent aux données.

La combinaison du modèle flexible d'exécution et d'un modèle de transaction qui rompt l'isolation permet d'obtenir de nouveaux comportement à l'exécution. En particulier, il devient possible à des activités qui se succèdent de partager des résultats et donc à une activité de commencer à travailler à partir de résultats intermédiaires. La parallélisme est donc augmenté par ce mode de fonctionnement. La figure 4.4 montre comment l'anticipation permet cette coopération entre activités qui se succèdent et comment le protocole transactionnel permet la coopération entre des activités concurrentes.

4.3.4 Contraintes sur la lecture des données

Les échanges de données entre activités génèrent de nouvelles dépendances entre ces activités. Il faut garantir que les dépendances générées par ces échanges ne provoquent pas de dépendances contradictoires avec l'ordre normal de terminaison des activités déterminé par le procédé. En particulier, il faut interdire à une activité de lire une donnée produite par une de ses activités suivantes. Cela imposerait un ordre contraire de terminaison à celui du procédé en créant un cycle de dépendance.

Ainsi, il faut étendre le protocole pour interdire la lecture par une activité d'une donnée produite par une autre activité qui contradirait l'ordre de dépendance.

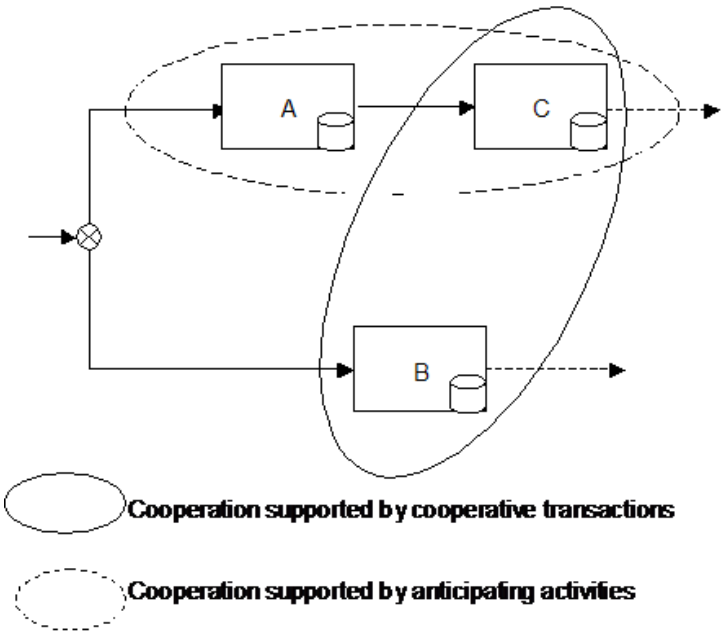


FIG. 4.4 – Deux possibilités de coopération

4.4 Mise en oeuvre

Le modèle de procédé décrit ici a été mis en oeuvre et évalué dans plusieurs contextes. Il a d'abord été intégré au prototype MOTU qui a servit pour valider un certain nombre d'idées de l'équipe ECOO dans le contexte de la conception et de la construction de batiment. La figure 4.5 est un exemple d'exécution montrant les liste de tâches de trois utilisateurs et les différents états correspondants.

Il a également été implanté dans le moteur de workflow Bonita qui est une initiative plus importante sur laquelle nous reviendrons dans un prochain chapitre (5).

4.5 Travaux liés

Les travaux sur la modélisation et le contrôle de l'exécution de procédés sont nombreux. Parmi ceux-ci, les recherches pour permettre la coordination d'activités coopératives restent importants et ont retrouvé récemment un nouvel intérêt.

Ces travaux ont pris naissance dans les années 70 avec la recherche sur les système de gestion documentaires (Office Information Systems). L'échec relatif de ces systèmes a interrompu ces recherches jusqu'à la montée en puissance d'Internet et à l'apparition d'une communauté de chercheurs s'intéressant au travail coopératif, qui s'est emparé des travaux faits parallélement sur la modélisation de procédés et le workflow dans les années 90.

Une polémique s'est fait jour à ce moment entre plusieurs tendances. Une de ces tendances soutenait qu'il n'était pas vraiment possible de modéliser ni de contrôler un processus coopératif en particulier pour des activités créatives. Une autre, née à la suite des travaux de Lucy Suchman [13] considérait plutôt qu'un processus n'était qu'une ressource pour l'action, et n'avait donc pas vocation à être totalement contrôlée. L'aspect important mis en avant par ces auteurs

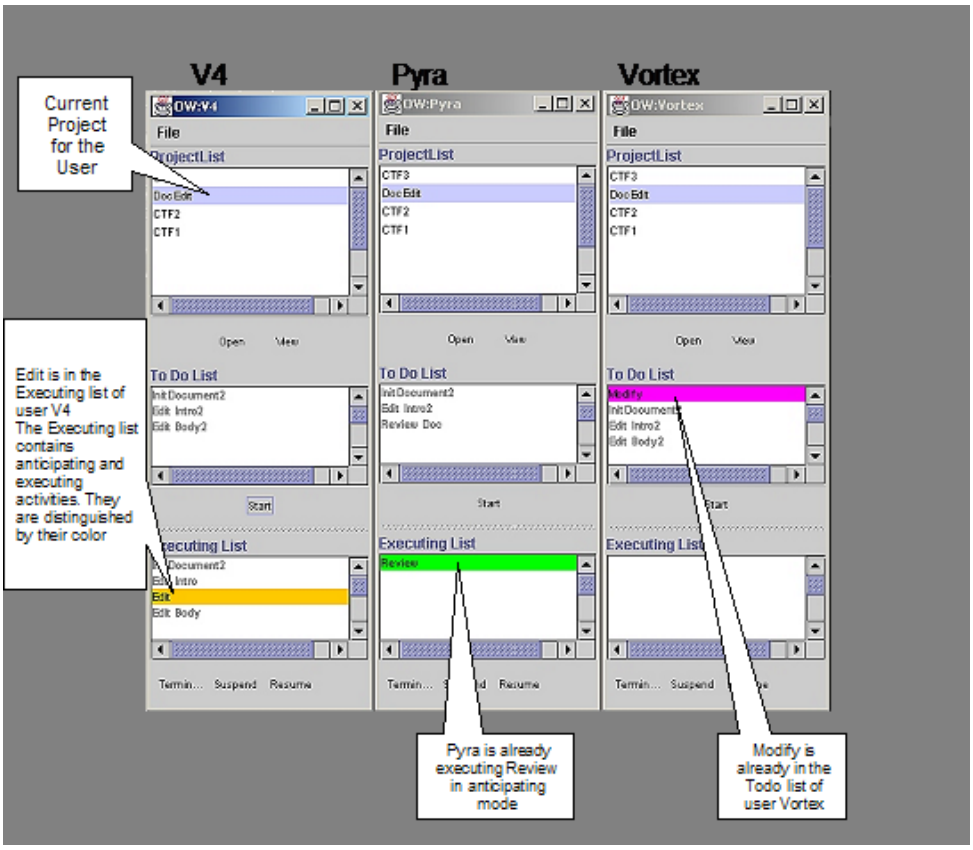


FIG. 4.5 – Trois listes de tâches

était qu'un système de workflow devait surtout fournir un moyen à ses utilisateurs de se situer dans le contexte du procédé qu'il exécutait. Une dernière tendance, issue plutôt du monde des systèmes d'informations et du domaine s'intéressant à la modélisation de procédés s'est tout de même évertuée à tenter de proposer des modèles, flexibles. La tendance dans ce groupe était de proposer des modèles capables d'évoluer dynamiquement, ou de supporter des cas d'exception, en considérant que tout ne pouvait pas être prévu au départ d'un procédé coopératif. ADEPTflex [11], Chautauqua [2], WASA [14] et WIDE [1] sont différents exemples de systèmes qui fournissent des primitives permettant de modifier un procédé pendant son exécution.

L'approche proposée ici est relativement différente de toutes celles qui ont été présentées. Nous considérons en fait qu'il n'est pas nécessaire de compliquer les modèles mais qu'il suffit en revanche de rendre l'exécution plus flexible pour la rendre plus conforme à la façon de travailler des utilisateurs. Cela n'empêche pas qu'il soit nécessaire de permettre de faire également évoluer le modèle dynamiquement. Nous reviendrons sur ce aspect dans la partie présentant l'implantation du système de gestion de workflow que nous avons initié.

Pendant quelques années, avec l'arrivée de services web et des problèmes liés aux procédés inter-organisationnels, les travaux sur les procédés flexibles, adaptés aux processus coopératifs ont été un peu moins importants. Mais on peut noter que cette thématique réapparait de façon prégnante avec l'apparition de différents workshops qui lui sont consacrés. La problématique qui se trouve posée reste toujours la même : celle de la relation entre un modèle de procédé et son exécution. Elle apparait également dans le domaine des services où la mise en pratique des procédés de services se trouve confrontée aux même difficultés liés à leur évolution et à leur dynamicité. Nous reviendrons sur ces aspects lors de la description des perspectives.

[9, 3, 5, 4, 7, 6, 12, 8]

Bibliographie

[1] Stefano Ceri, Paul W. P. J. Grefen, and Gabriel Sanchez. WIDE : A distributed architecture for workflow management. In *RIDE*, pages 0–, 1997.

[2] Clarence A. Ellis and Carlos Maltzahn. The chautauqua workflow system. In *HICSS (4)*, pages 427–, 1997.

[3] Daniela Grigori, François Charoy, and Claude Godart. Anticipation to enhance flexibility of workflow execution. In H.C. Mayr, J. Lazansky, G. Quirchmayr, and P. Vogel, editors, *International Conference on Database and Expert Systems Applications - DEXA'2001, Munich, Germany*, volume 2113 of *Lecture Notes in Computer Science*, pages 264–273. Springer-Verlag, Sep 2001.

[4] Daniela Grigori, François Charoy, and Claude Godart. Flexible cooperative workflow management. In *13th International Conference on Control Systems and Computer Science - CSCS'2001, Bucarest, Roumanie*, Jun 2001.

[5] Daniela Grigori, François Charoy, and Claude Godart. Flexible data management and execution to support cooperative workflow : the coo approach. In *Proceedings of the Third International Symposium on Cooperative Database Systems for Advanced Applications - CODAS'2001, Beijing, China*, pages 139–146. IEEE, Apr 2001.

[6] Daniela Grigori, François Charoy, and Claude Godart. Coo-flow : a process model to support cooperative processes. In *Fifteenth International Conference on Software Engineering and Knowledge Engineering 2003 - SEKE'2003, San Francisco*, Jul 2003.

[7] Daniela Grigori, François Charoy, and Claude Godart. Coo-flow : a process technology to support cooperative processes. *International Journal of Software Engineering and Knowledge Engineering - IJSEKE Journal*, 14(1) :61–78, Feb 2004. World Scientific Publishing.

[8] Daniela Grigori, François Charoy, and Claude Godart. Enhancing the flexibility of workflow execution by activity anticipation. *International Journal of Business Process Integration and Management*, 1(3) :143–155, 2006.

[9] Daniela Grigori, Hala Skaf-Molli, and François Charoy. Adding flexibility in a cooperative workflow execution engine. In *8th International Conference on High Performance Computing and Networking Europe - HPCN Europe 2000, Amsterdam, Hollande*, May 2000.

[10] Franck Leymann. *Production Workflow*. Prentice Hall, 1999.

[11] M. Reichert and P. Dadam. Adept flex : Supporting Dynamic Changes of Workflows Without Losing Control. *Journal of Intelligent Information Systems*, 10(2) :93–129, 1998.

[12] Hala Skaf, François Charoy, and Claude Godart. Maintaining shared workspaces consistency during software development. *International Journal of Software Engineering and Knowledge Engineering*, 9(5) :623–642, Oct 1999.

[13] Lucy A. Suchman. *Plans and Situated Actions : The Problem of Human-Machine Communication (Learning in Doing : Social, Cognitive & Computational Perspectives)*. Cambridge University Press, November 1987.

[14] Gottfried Vossen and Mathias Weske. The wasa2 object-oriented workflow management system. In Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh, editors, *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*, pages 587–589. ACM Press, 1999.

[15] Mathias Weske. Flexible modeling and execution of workflow activities. In *31st Hawaii International Conference on System Sciences, Software Technology Track (Vol VII)*, 1996.

[16] WFMC. The workflow reference model. Technical report, The Workflow Management Coalition, 1995.

Chapitre 5

Bonita

5.1 Introduction

Bonita est un moteur de workflow développé depuis 2003 sur une plateforme J2EE. L'objectif initial de son développement était de mettre en oeuvre et de valider les idées que nous avons développées sur les modèles de workflow pour la coordination d'activités coopératives, en particulier dans le cadre du projet Cocao.

5.2 Le cahier de charges de Bonita

Le besoin d'un meilleur support pour la coordination explicite des activités pour processus coopératif est souvent exprimé mais très difficile à mettre en oeuvre. La question centrale, la même qui s'est posée pour l'adoption des calendriers partagés et des autres outils de groupware en général, c'est celle du bénéfice que peut en tirer l'utilisateur final et pas seulement l'organisation qui le fait travailler. En général, ces outils sont vu comme des émanations de management. Ils demandent un effort supplémentaires aux employés pour le bénéfice principal de ce management (convoquer des réunions, suivre et contrôler le travail et l'occupation des employés). La réaction classique des agents dans ce cas est de peu, pas ou mal utiliser les outils. Ce comportement a été identifié également dans le cas de l'automatisation de processus de service. Nous pensons que l'acceptation des outils de coordination explicite dans le cadre de projets coopératifs implique que les utilisateurs aient une idée claire des bénéfices qu'ils peuvent en tirer. Il est facile de constater que même si le besoin se fait sentir d'un meilleur support pour la coordination dans les équipes distribuées, même s'il existe une volonté de mettre en oeuvre des procédures claires au début d'un projet, l'utilisation de ces procédures et des outils pour les contrôler sont vite oubliés lorsque le projet commence et que la pression pour obtenir des résultats augmente. Si les utilisateur perdent l'impression que suivre le processus est utile pour eux, ils arrêtent de l'utiliser ou ne l'utilisent pas de façon correcte et régulière[1, 4]. C'est en tenant compte de ces résultats et de ces constatations que nous avons défini quelques principes à prendre en compte pour la mise en oeuvre d'un système de gestion de workflow pour des environnement coopératifs.

La conception du processus Dans un environnement coopératif, le plan et le processus à suivre est en général le résultat d'un consensus entre les membres de l'équipe du proejt. Des décisions sont prises et des actions à réaliser sont distribuées entre les membres de l'équipe. La plupart du temps, ces actions sont des petits processus qui doivent être raffinés et connectés entre eux pour assurer la connection globale du groupe.

Par exemple, dans le cas d'un développement de logiciel, le planning du développement est décidé puis raffiné. La création du processus complet à partir de rien serait la plupart du temps un travail long et risqué. En général, lorsqu'un processus coopératif est démarré, même en utilisant une coordination implicite, les participants se réfèrent à des processus existants qu'ils ont déjà exécutés précédemment. Ils réutilisent en partie leur expérience pour le nouveau projet. Cette étape est d'autant plus simple que les participants une histoire commune de coopération. Un système de gestion des processus coopératif doit permettre de reproduire ce comportement en permettant la réutilisation de fragments de processus qui ont déjà été exécutés.

Un système de gestion de processus coopératif doit permettre la réutilisation d'instances de processus ou de fragments qui doivent pouvoir être intégrés facilement dans le nouveau processus (par exemple un fragment de processus pour la revue d'articles ou pour la livraison de logiciel dans un projet de développement).

Controle du processus par les utilisateurs Les utilisateurs participant à un processus coopératif doivent être capable de suivre et de modifier ce processus. Les processus métiers classiques sont en général contraints par les managers et ne peuvent pas être modifiés par les utilisateurs finaux. Ce n'est pas acceptable et en tous cas pas acceptés dans les processus coopératifs. Nous pensons qu'un processus coopératif est en fait le résultat de son exécution. Le processus est lui même un produit du processus coopératif. Chaque participant doit avoir la possibilité d'ajouter, de supprimer et de changer les activités et leurs enchainements.

Automatisation des activités L'intérêt de l'utilisation d'un système de workflow n'est pas toujours clair pour les utilisateurs. De nombreuses expériences ont montré que si les utilisateurs n'y voient pas d'intérêt, ils essaieront de ne pas les utiliser par tous les moyens (voir [6] et les nombreux travaux qui y font référence). Un système de gestion des processus coopératifs doit fournir une assistance et permettre l'automatisation des tâches les plus répétitives et ne pas être simplement un outil de contrôle pour les gestionnaires. Il doit également être suffisamment intégré à l'environnement pour aider les utilisateurs à retrouvers les objets sur lesquels ils travaillent, les partagers, suivre leurs modifications. Nous faisons l'hypothèse que si ces conditions sont remplies, il y a plus de chance pour que les utilisateurs contribuent à l'évolution du processus.

Le modèle de Bonita a été développé en essayant de tenir compte de ces considérations et de ces hypothèses dans le but de pouvoir les expérimenter.

5.3 Le modèle de Bonita

Les différents points que nous avons décrits ci-dessus nous ont amenés à tenter de définir un modèle simple, flexible et dynamique de modélisation et d'exécution de processus. Nous nous sommes en particulier inspirés des langages à prototypes [7]. Ces langages ne font pas la différence entre classes ou modèles et instances. Une nouvelle instance n'est pas créée à partir d'une définition mais en clonant une instance existante puis en la modifiant éventuellement. Dans Bonita, une instance de procédé peut être créée directement à partir de l'API de gestion des processus. Elle peut ensuite être modifiée dynamiquement. Un nouveau processus peut être créé à partir de zéro ou en clonant un processus existant. La définition du processus peut se faire activité par activité ou en important des processus entiers. Sur la figure 5.1, la partie gauche montre l'état d'un processus pour un utilisateur et sur la droite l'état du processus. Cette fenêtre est également un éditeur qui permet de changer le processus au cours de son exécution.

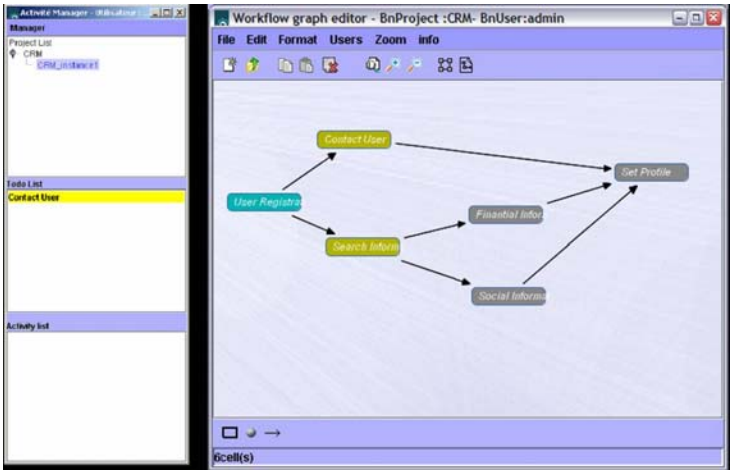


FIG. 5.1 – L'interface de définition et d'exécution

Un processus est composé d'un ensemble d'activités qui doivent être exécutées pour atteindre un objectif. Les activités ont des dépendances entre elles sur lesquelles nous reviendrons. Un processus a également des participants qui peuvent prendre différents rôles lors de l'exécution du processus. Les roles décrivent les relations entre les activités et les utilisateurs qui peuvent les exécuter.

Les contraintes sur la définition d'un processus sont minimales pour faciliter sa définition par les utilisateurs finaux.

Un processus est créé par un utilisateur qui en devient le "propriétaire". A cet instant, le processus est démarré. Le propriétaire peut ajouter des activités et des dépendances entre elles. Dès qu'une activité est ajoutée, elle peut être démarrée, sauf si on lui ajoute une dépendance avec une autre activité. Le processus est terminé lorsque toutes les activités sont terminées ou lorsque le processus est explicitement terminé ou annulé par son créateur.

Les activités peuvent avoir différentes états : initial, executable, executing, anticipable, anticipating, cancelled, aborted, terminated. Une activité peut être exécutée dès qu'elle a été créée. Elle est alors dans l'état "executable".

L'exécution flexible des procédés est possible grace à l'anticipation que nous avons décrite au chapitre précédent. Le modèle est là complétement mis en oeuvre.

De nouvelles activités et de nouvelles dépendances peuvent être créées pendant l'exécution du processus. Les seules contraintes concernent l'état des activités. Il n'est pas possible de changer les activités et les dépendances concernant les activités en cours ou terminées.

Le moteur de workflow est capable de calculer les listes de tâches à faire et d'activités en cours d'exécution et de notifier les utilisateurs de chaque changement qui le concerne.

5.3.1 La construction des processus

La construction d'un processus activité par activité n'est certainement pas une solution viable pour un processus coopératif. En outre, il est difficile de réutiliser un processus coopératif d'un projet à l'autre. Le temps de conception du processus ne peut être compensé par un nombre d'exécutions important.

Le processus doit donc pouvoir être créé simplement en se basant sur des expériences précédentes. Par exemple, l'écriture coopérative d'un document est un processus exécuté fréquemment. Si un projet contient une étape de ce type, il doit être possible de retrouver différents

fragments de processus qui fournissent une solution à mettre en oeuvre (planification, édition, revue, publication ou planification, production, édition, intégration, publication).

L'exemple suivant montre différentes étapes d'un projet coopératif :

Fig. 5.2 – Le processus initial

Sur la figure 5.2, un petit processus d'édition est utilisé pour démarrer la production d'un document. Plusieurs activités sont créées. Le processus est démarré. Un peu plus tard, des étapes de validation/soumission sont ajoutées au processus (figure 5.3).

A partir de ce résultat, un nouveau document doit être produit. Le processus d'édition est importé à nouveau et connecté à l'activité de validation (figure 5.4).

L'importation et le clonage de processus sont les deux principaux mécanismes que nous proposont pour permettre ce type de comportement. Cette approche est inspirée de loin des langages à base de prototype. N'importe quelle instance de processus, en cours d'exécution ou terminée peut être utilisée pour instantier un nouveau processus. Dans ce cas, les activités du processus source sont remises à leur état initial et toutes les propriétés du processus original sont importées pour constituer le nouveau processus, à l'exception des utilisateurs. Un processus peut être construit en composant et en important différents processus et en créant des dépendances entre leurs activités. Durant cette phase, le processus peut être suspendu. S'il ne l'est pas, l'état des activités est mis à jour immédiatement pour refléter leur nouvel état.

5.3.2 Flot de données

La gestion des données dans Bonita est réalisée de façon simple. A un procédé sont associées deux types de données. Les données des activités et les données des processus. Les données associées aux activités servent également de paramètres en entrée et en sortie de ces activités. Les données en sortie sont propagées en suivant les liens de dépendances entre les activités. Cette approche est relativement simple mais elle permet de gérer la plupart des cas. En outre elle évite les problèmes liés à l'évolution dynamique du procédé.

De plus, la plupart des projets coopératifs sont fondés sur l'utilisation d'espaces de travail partagé. Les donnés d'un procédé sont locales à celui-ci et ne peuvent être efficacement utilisés pour cela. Les espaces partagés sont en général mis en oeuvre à l'aide de répertoires partagés ou de systèmes de gestion de versions. Dans ce cas, chaque activité peut être associée à une copie privé de cet espace partagé. Le démarrage de l'activité correspond à la création d'une copie privée et la terminaison de l'activité correspond à la publication des résultats finaux dans

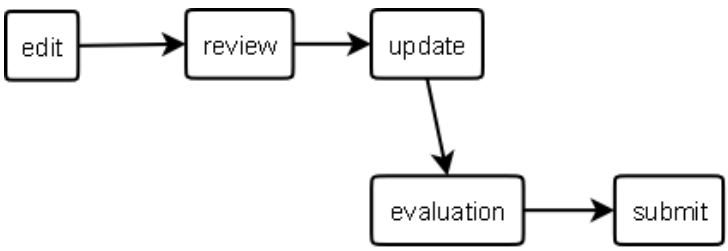


Fig. 5.3 – Le processus est complété avec des activités de validation

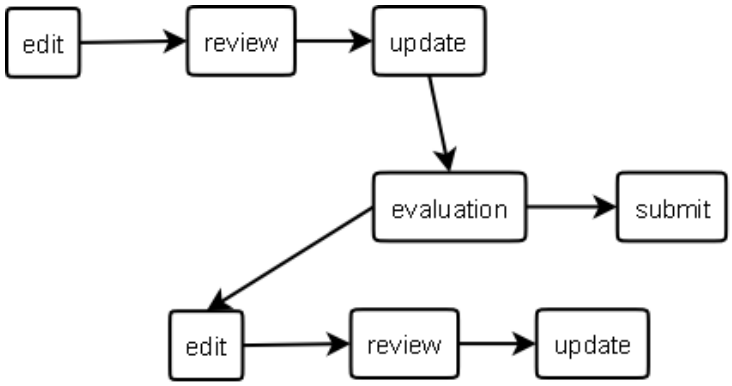


FIG. 5.4 – Le processus d'édition est importé à nouveau pour être réexécuté

l'espace partagé. Les conflits et les problèmes de concurrences sont gérés par l'espace partagés et dépendent du protocole qu'il met en oeuvre. Nous avons eu l'occasion de tester et de valider cette approche dans différents projets (Corvette, Cocao et Coopéra entre autres).

5.3.3 Correction des procédés

De nombreux travaux ont été mené pour décrire et vérifier des propriétés liés à la correction de la définition des procédés. Ces propriétés sont en général basées sur celles vérifiables pour des réseaux de Pétri[8] ou de l'"Event Calculus"[3].

Dans le contexte de Bonita, nous avons pris le parti de restreindre au minimum les contraintes sur le procédé pour en simplifier la définition par les utilisateurs et leur donner le maximum de flexibilité. Seuls les cycles sont détectés et interdits sauf lors de l'utilisation d'itérateurs. Ainsi, un processus construit dynamiquement est considéré comme toujours valide. Les activités sont exécutables dès qu'elles remplissent les conditions pour leur "anticipation". Un procédés est considéré comme terminé lorsque toutes ses activités sont dans l'état "dead" (non accessible) ou terminées. Ainsi, nous considérons que la flexibilité est plus importante que la consistance dans un contexte coopératif. Un procédé donné ne sera pas, comme c'est le cas pour les procédés métiers, exécuté un grand nombre de fois. Il reste toujours possible (ce n'est pas fait actuellement) de mettre en oeuvre des algorithmes de vérifications pour détecter les anomalies sur un procédé, sans toutefois les interdire.

5.3.4 L'automatisation des activités

L'acceptation par les utilisateurs du contrôle par un procédé dépend du bénéfice que ces utilisateurs peuvent espérer en obtenir pour eux mêmes. L'automatisation de parties du procédé est un de ces bénéfices potentiels. Bien que de nombreuses activités d'un projet coopératif soient gérées par les utilisateurs, beaucoup d'entre elles peuvent être automatisées. Les tests, la génération de la documentation, la construction des espaces de travail, la livraison des rapports peuvent être implantés par des services fournis dans l'environnement d'exécution du procédé.

Bonita permet ainsi d'attacher des scripts (appelés hooks) aux changements d'état des activités. Plusieurs hooks peuvent être attachés au même changement d'état. Par exemple, lorsqu'un utilisateur a fini son activité, il est possible automatiquement de contrôler la validité de son résultat (les tests unitaires passent), de publier ce qu'il a fait dans l'espace partagé et d'exécuter des tests d'intégration dans cet espace partagé.

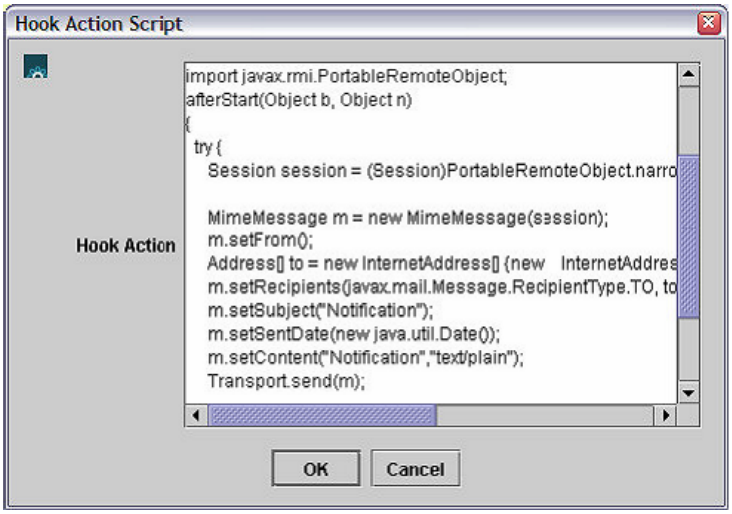


FIG. 5.5 – L'implantation d'un Hook

Certains types d'activités peuvent même être définis comme complétement automatiques. Dès qu'elles deviennent exécutable, tous leurs scripts sont exécutés.

L'échec de l'exécution d'un hook peut servir à annuler un changement d'état. Les hooks peuvent ainsi servir à valider le démarrage d'une activité (pre-start hook) ou à valider sa terminaison (pre-terminate hook). Si la vérification faite par le hook échoue, l'activité n'est pas démarrée ou terminée selon le cas. Les changements d'état des activités sont atomiques et incluent l'exécution des hooks précédant ce changement d'état. Nous avons ainsi récemment connecté un moteur de règle à Bonita pour mettre en place des contraintes de sécurité dynamique de type "séparation de devoirs".

Bien sur ce mécanisme peut rendre la définition d'un processus plus compliquée. Elle demande en particulier un effort de programmation. Dans une version récente de Bonita, une librairie de hook prédéfinie pour des actions génériques a été construite, ce qui permet de les utiliser simplement et de les attacher à n'importe quelle activité. Cette librairie peut être augmentée en fonction des besoins. En outre, les hooks peuvent être également écrits en utilisant un langage de scripts. Les hooks nous servent également pour étendre simplement le moteur, pour l'associer par exemple avec un moteur de règles pour vérifier des contraintes sur l'état du procédé.

Certains hooks peuvent être utilisé pour modifier dynamiquement le processus courant. Nous envisageons de les utiliser pour générer automatiquement des activités de compensation lorsqu'un activité échoue ou est annulée.

5.3.5 La gestion de l'organisation

La gestion des roles et les relations avec l'organisation ont été mis en oeuvre relativement simplement dans la version initiale de Bonita. Des procédures et des plugins peuvent être écrits pour connecter le moteur de workflow aux outils de l'organisation (Annuaire LDAP par exemple) et pour affecter automatiquement des personnes aux procédés en cours d'exécution en fonction de procédures et de règles spécifiques à une orgranisation.

5.3.6 Conscience du procédé

Chaque événement (modification, changement d'état) d'un procédé produit un événement qui est publié dans une queue de message. Les clients du système peuvent s'enregistrer sur cette queue pour être notifiés de ces événements. Ils peuvent par exemple choisir d'être informés de la terminaison de chaque activité des procédés dont ils font partie. Ils seront alors notifiés par messagerie instantannée ou par courrier électronique. C'est une forme assez primaire de conscience du procédé mais nous considérons que la progression d'un procédé coopératif n'est pas très rapide. Le nombre d'événements générés sera donc relativement faible. En revanche, ils permettront de rappeler à un utilisateur, pris éventuellement par d'autres taches qu'il participe à un projet et qu'il doit éventuellement s'intéresser aux choses qu'il peut avoir à y faire.

5.4 L'implantation

Bonita est actuellement disponible comme un logiciel libre (bonita.objectweb.org). Il est supporté par le consortium ObjectWeb et Bull. Il est cependant plus utilisé pour ses fonctionnalités traditionnelles de système de gestion de workflow que pour les extensions aux procédés coopératifs même si celles-ci sont souvent considérées comme un bon argument pour favoriser son adoption.

Bonita est implanté sur une plateforme J2EE. La persistence des procédés est gérée par le mécanisme de transformation Objet/Relationnel des EJBs. Plusieurs interfaces sont fournies aux clients pour manipuler les procédés. Les plus importantes sont l'interface de définition de procédés et l'interface pour les utilisateurs désirant interagir avec les procédés existants.

La version 3 actuellement disponible fonctionne sur Jonas et JBoss. Elle fournit une interface Web et un éditeur intégré dans Eclipse, ce qui facilite la définition et l'implantation des procédés. En outre, une interface d'importation de description de procédés décrits en XPDL a été ajoutée pour répondre aux besoins des utilisateurs de Bonita. La figure 5.6 donne une idée de l'architecture de Bonita.

Bonita a été également intégré avec la plateforme de gestion de contenu eXo pour automatiser la production et la publication (figure 5.7.

5.5 Conclusion

Grace aux efforts de Bull et d'ObjectWeb, Bonita a atteint une maturité qui lui permet maintenant d'être déployé dans des applications réelles. Il a cependant fallu remettre en cause certains des choix initiaux pour permettre son industrialisation. La flexibilité du modèle était obtenu au détriment des performances pour certaines applications. L'instantiation par clonage n'est pas très efficace lorsque les processus à instantier sont nombreux (ce qui n'était pas une hypothèse à l'origine). Le système a donc été étendu pour permettre de faire la différence entre modèle et instance de manière à permettre l'optimisation de la création de procédés. La possibilité de faire évoluer simplement les instances reste cependant un argument important pour l'adoption de Bonita dans des environnements où la définition complète du procédé est difficile. C'est le cas dans des applications de e-gouvernement pour lesquelles Bonita est utilisé. Le mécanisme d'awareness est également désactivé par défaut, toujour pour des raisons de performance. Ce critère est majeur pour le déploiement industriel d'un système de Workflow. Il ne l'était pas pour nous lors de la conception initiale. En outre, nous avions misé sur une technologie pas assez mature (J2EE 1.4) qui ne permet pas le passage à l'échelle aussi simplement que prévu.

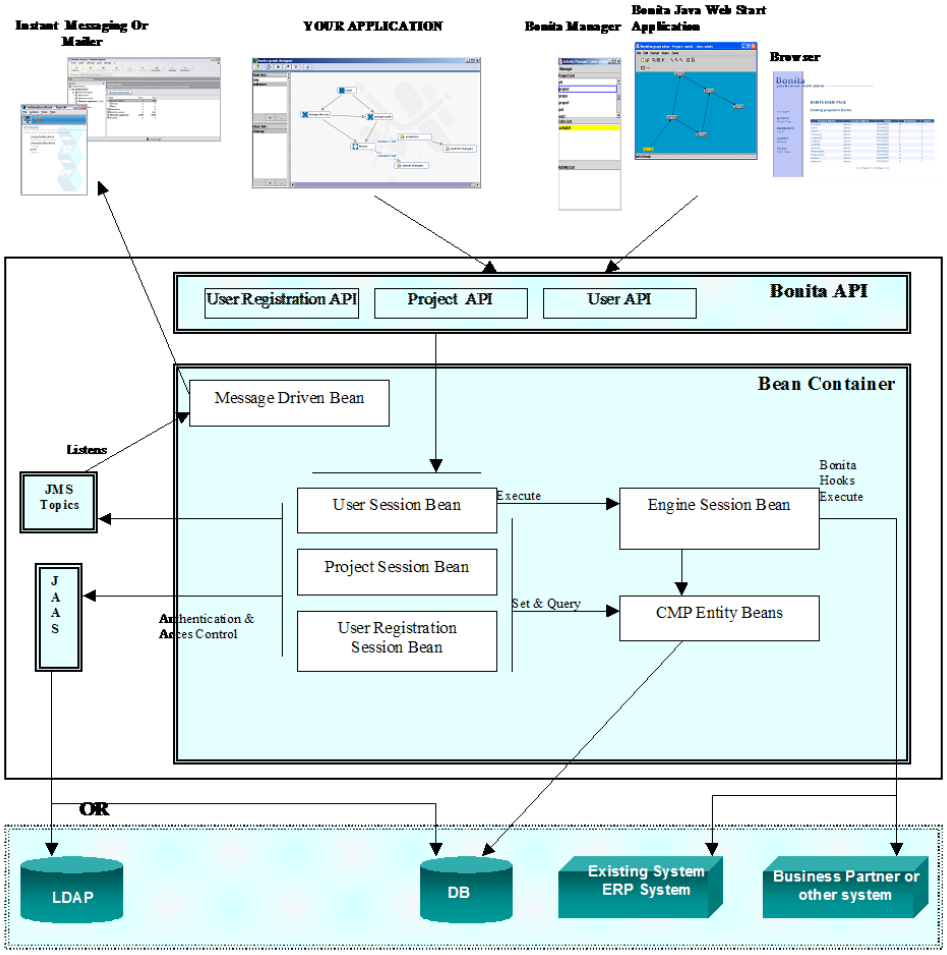


FIG. 5.6 – Architecture of Bonita

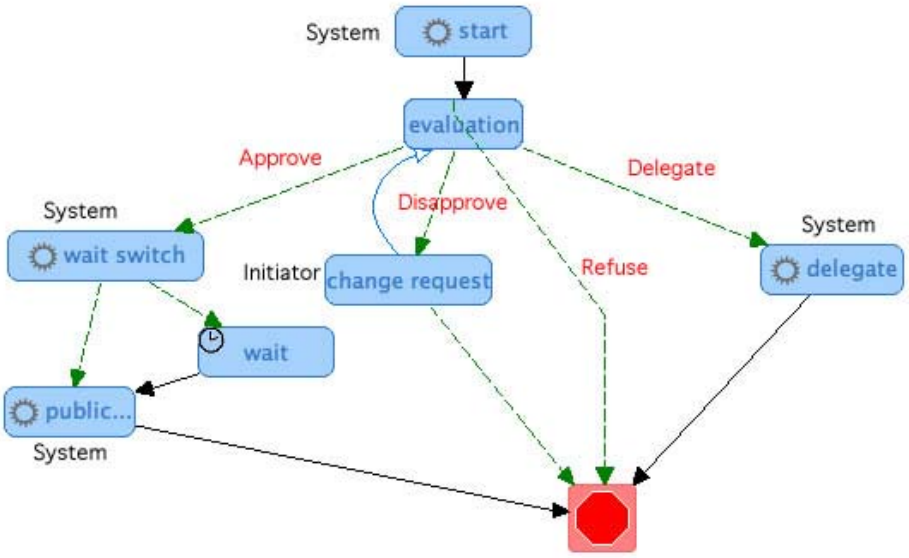


FIG. 5.7 – Intégration de Bonita et d'Exo

Bonita va maintenant passer à une autre étape grâce à une collaboration avec JBoss pour tenter de proposer une nouvelle version d'un moteur de workflow générique (Process Virtual Machine) sur lequel il sera possible de mettre en oeuvre différents modèles spécifiques de Workflow (XPDL, BPEL). D'autres extensions et travaux sont en cours sont en cours. En particulier, une expérience est actuellement menée pour connecter Bonita à un moteur de règles pour exprimer de façon simple des règles de sécurité sur les processus (Séparation des devoirs par exemple).

5.6 Publications

[1, 5, 2]

Bibliographie

[1] François Charoy, Claude Godart, Nicolas Gregori, Jean-Charles Hautecouverture, and Sébastien Jourdain. Coopera : An environment for teaching and learning internet cooperation. In *IADIS International Conference e-Society 2004, Avila, Espagne*, pages 323–330, Jul 2004.

[2] François Charoy, Adnene Guabtni, and Miguel Valdes Faura. A dynamic workflow management system for coordination of cooperative activities. In *Workshop on Dynamic Process Management - BPM 2006 International Workshops, BPD, BPI, ENEI, GPWW, DPM, semantics4ws, 04/09/2006*, volume 4103 of *in : LNCS, Business Process Management Workshops*, pages 205–216, Vienne/Autriche, 2006. Springer. Non.

[3] Nihan K. Cicekli and Yakup Yildirim. Formalizing workflows using the event calculus. pages 222+. 2000.

[4] Thomas Herrmann and Marcel Hoffmann. The metamorphoses of workflow projects in their early stages. *Computer Supported Cooperative Work (CSCW)*, 14(5) :399 – 432, October 2005.

[5] Christophe Loridan and Miguel Valdes. Bonita :a java 2 platform, enterprise edition (j2ee) open source cooperative workflow system. In *Java One, 2004*, San Francisco, USA, 2004. none. Non, This presentation describes the Bonita Workflow Management System developed by the ECOO team of LORIA. Colloque avec actes et comité de lecture. internationale.

[6] Lucy A. Suchman. *Plans and Situated Actions : The Problem of Human-Machine Communication (Learning in Doing : Social, Cognitive & Computational Perspectives)*. Cambridge University Press, November 1987.

[7] David Ungar, Craig Chambers, Bay-Wei Chang, and Urs Holzle. Organizing programs without classes. *Lisp and Symbolic Computation*, 4(3), 1991.

[8] W.M.P van der Aalst. The application of petri nets for workflow management. *The Journal of Circuits, Systems and Computers*, 8(1) :21–66, 1998.

Chapitre 6

Sphères de comportement

6.1 Introduction

La modélisation, l'exécution de procédés et les systèmes de gestion de workflow sont en production depuis plusieurs années avec des succès divers. Les propositions d'évolutions sont de plus en plus sophistiquées et comme nous l'avons décrit précédemment cherchent à augmenter la flexibilité de la définition et de l'exécution des procédés. Les travaux que nous avons menés et l'étude des travaux des autres chercheurs du domaine nous ont amené à penser qu'il n'était pas possible de proposer une solution assurant à la fois la flexibilité requise par les utilisateurs et le contrôle nécessaire à la gestion des ressources d'une entreprise. Un procédé, qui implante un service fournit par une organisation est une suite d'actions ou d'activités qui s'exécutent avec des contraintes différentes selon les moments de l'exécution du service. Par exemple, certaines parties peuvent être peu contraintes comme les phases de conception ou de définition et d'autres peuvent l'être plus lorsqu'il s'agit de valider un produit avant son expédition. De même, il est possible de mettre un peu de flexibilité dans la procédure de préparation d'un voyage mais il y aura forcément plus de contrôles lors de la phase de remboursement des frais (il n'est pas question de mettre en paiement si tous les papiers ne sont pas disponibles). La définition d'un procédé doit pouvoir prendre en compte les différents niveaux de contraintes et les différentes conditions d'exécution qu'on veut pouvoir associer à son exécution. En revanche nous pensons que cette flexibilité ne doit pas se faire au détriment de la simplicité d'expression des processus. Pour cela, nous proposons de séparer la définition des procédés des propriétés attachées à leur exécution.

6.2 Etat de l'art

De nombreuses tentatives ont été faites pour tenter de définir de façon plus précise le comportement des processus en étendant leur langage de définition par de nouveaux opérateurs ou par de nouvelles propriétés. Pour chaque comportement désiré, il est possible de proposer une utilisation particulière des opérateurs des modèles de workflow pour obtenir ce comportement[1]. Cette approche a pour principal inconvénient d'adapter la description des procédés aux qualités que l'on veut obtenir pour son exécution au prix de la modification de la façon de travailler décrite à l'origine.

D'autres travaux se sont intéressés à des alternatives pour introduire de la flexibilité dans les systèmes de gestion de workflow tout en préservant leur lisibilité. Ces approches ont en commun de tenter de séparer les propriétés attendues de l'exécution des procédés de leur définition. C'est

ce que nous avons tenté de faire en proposant de généraliser cette approche en séparant la définition des procédés de la description de leur propriétés comportementales.

6.2.1 Principe des sphères de comportement

Les sphères de comportement permettent de séparer la définition d'un procédé de la spécification de son comportement. La plupart des approches actuelles obligent à associer les propriétés d'exécutions à l'ensemble du procédé ou aux activités une par une, alors quelles ne sont parfois nécessaires que pour un sous-ensemble de ses activités. Par exemple l'atomicité peut n'être nécessaire que pour une partie critique du processus et pas pour l'ensemble du processus. La compensation elle peut ne pas avoir de sens activité par activité mais doit pouvoir être associée à un groupe d'activités. Nous nous sommes proposés de définir précisément ce que peuvent être les sphères de comportement et ce qu'elles permettent.

6.2.2 L'idée : les sphères de contrôle

L'idée d'associer des propriétés dynamiques à un ensemble d'élements est inspirée des spheres de contrôle dans le traitement des données [10]. Les sphères de contrôle ont été introduites pour définir des "frontières" autour d'ensemble d'activités dans les systèmes de traitement de données. Elles étaients utilisées pour faire du recouvrement, de l'audit, de la validation et du remplacement de procédures dans des systèmes distribués de traitement de données.

Nous avons proposé d'appliquer ce principe aux procédés métiers pour fournir un cadre flexible permettant de définir et de gérer le comportement d'un processus métier en cours d'exécution. Mais tout d'abord, nous allons définir rapidement ce qu'est une sphère de comportement de notre point de vue.

Une sphère de comportement peut être simplement décrite comme un sous-ensemble d'activités d'un processus à laquelle nous associons un propriété qui doit être garantie pour le groupe d'activités désignées.

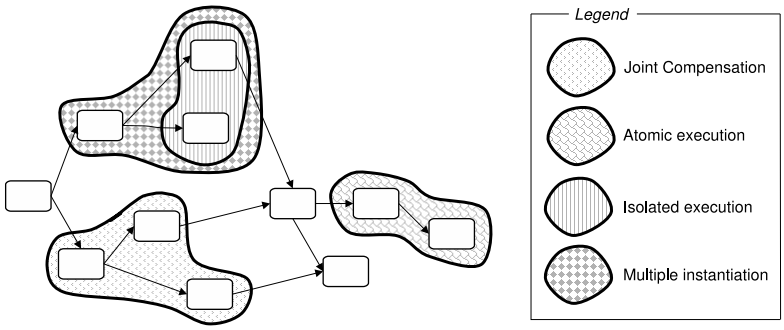


Fig. 6.1 – Exemple de sphères de comportement.

La figure 6.1 montre un procédé sur lequel sont définies un certain nombre de sphères. C'est l'idée principale de l'approche. Le procédé est défini par un analyste en tenant compte des besoins et des méthodes mise en oeuvre dans une organisation. Ensuite les propriétés que l'on veut garantir à l'exécution sont "plaquées" sous forme de sphères sur cette définition. Cette séparation permettra même d'associer différents contextes d'exécution à une même spécification de procédé. Trois principes doivent être respectés par rapport à l'application de la propriété pour que l'on puisse parler de sphère.

Premier principe : La séparation des préoccupations entre la définition du processus et la spécification des propriétés comportementales.

Second principe : Le comportement attaché à la sphère est appliqué aux groupe d'activités comme un tout. Le comportement n'est pas attaché aux activités individuellement.

Troisième principe : L'utilisation des sphères de comportement doit apporter plus de flexibilité au concepteur du procédé, aux administrateurs et aux utilisateurs.

Le premier principe implique que la spécification du comportement ne doit pas avoir d'impact à la définition du processus et que le processus ne doit pas être définit en fonction des propriétés du processus, ce qui peut être parfois le cas dans une approche classique. Par exemple, l'atomicité peut être assurée pour l'ensemble d'un processus si celui-ci respecte les contraintes des transaction flexibles. Cela implique cependant de respecter des contraintes particulières sur la définition du procédé qui peuvent ne pas être respectueuses de la façon de travailler ou des contraintes de l'organisation.

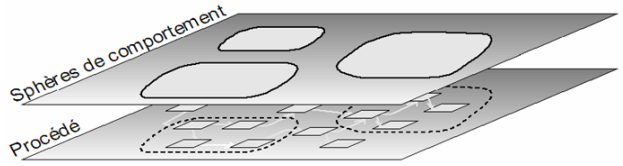


FIG. 6.2 – Séparation des préoccupations

Le second principe assure que l'application d'une propriété à un groupe d'activités a un résultat observé différent de son application à chacune des activités séparément. Ainsi, il est différent de dire que l'exécution d'un groupe d'activité est atomique et que chacune des activités du groupe est atomique. En revanche, dire qu'un groupe d'activité est affecté aux personnes pouvant prendre un rôle donné est équivalent à l'affectation de chacune des activités à ce même rôle.

Le troisième principe impose que l'utilisation de sphères rende la définition des propriétés attachées à l'exécution des procédés plus souple et plus flexible que d'autres méthodes comme l'ajout d'opérateurs au modèle ou l'utilisation d'autres techniques. Il doit rendre la définition des procédés plus flexible pour le concepteur et éventuellement pour les utilisateurs.

Chaque propriété que l'on voudra mettre en oeuvre en utilisant l'approche des sphères devra satisfaire à ces principes. Le premier et le second sont relativement facile à vérifier. Le troisième peut se révéler un peu plus "subjectif".

Dans le domaine de la gestion de procédés, il est déjà possible de considérer la plupart des propriétés appliquées à un processus de façon global comme pouvant également être mise en oeuvre sur un sous-ensemble du procédé. C'est le cas par exemple pour l'atomicité qui a déjà fait l'objet d'études dans ce sens [12, 2, 15]. Cette proposition est une réponse à certains détracteurs des procédés transactionnels qui pensent que dans la plupart des cas, la mise en oeuvre de transactions dans les procédés est contre-productive. C'est souvent vrai, mais il existe des cas où l'on veut assurer cette propriété pour une partie de l'exécution. L'approche de Derks propose d'identifier les parties de procédés qui doivent s'exécuter de façon atomique.

Les sphères de compensation de Leymann [11] relèvent d'une autre logique. Dans les modèles de workflow transactionnels classiques, l'annulation d'une activité pour revenir à l'état initial

n'est pas toujours possible. Une activité de compensation est une activité qui permet, sans forcément annuler les effets d'une activité, de revenir à un état acceptable pour le processus. Par exemple, si un billet d'avion a été acheté et que le voyage est annulé, le remboursement du billet d'avion n'est pas forcément intégral. On ne revient pas à l'état initial mais à un état acceptable. Une sphère de compensation permet de décrire la façon de compenser l'action d'un ensemble d'activités. La compensation peut elle même être réalisée par l'exécution d'un autre ensemble d'activités sans qu'il y ait une correspondance entre les activités compensées et les activités de compensation.

Ces deux approches respectent les principes que nous avons associés à la notion de sphère. Elles fournissent en particulier une plus grande flexibilité dans la définition des propriétés qu'un processus doit respecter lors de son exécution.

D'autres propositions se rapprochent de l'idée de sphère. Les poches de flexibilité proposées dans [13] en font partie. Une poche de flexibilité correspond à un ensemble d'activités pour lesquelles les dépendances ne sont pas complétement définies et sont laissées à la charge des utilisateurs exécutant le processus, sous réserve de respecter certaine contraintes. Par exemple, dans une poche de flexibilité, on peut trouver trois activités qui doivent être exécutées en série, sans imposer l'ordre dans lequel elle doivent être exécutées. Les poches de flexibilité ne sont pas exactement des sphères parce que leur définition impose une modification (la possibilité d'avoir un processus non complétement défini) et l'extension (les contraintes) du modèle. Il n'y a pas dans ce cas séparation des préoccupations. Il serait cependant intéressant d'étudier l'approche pour voir s'il ne serait pas possible de séparer la définition des poches de flexibilité de la définition du processus lui-même.

6.3 Contributions

Les sphères de comportement sont utilisées principalement pour exprimer des propriétés transactionnelles pour des procédés métiers. C'est relativement normal dans la mesure ou celles-ci sont critiques pour assurer une exécution correcte des procédés. De notre coté, nous nous sommes intéressés à deux types propriétés qui pouvaient bénéficier de l'approche. Les sphères de multi-instantiation et les sphéres d'isolation. Les premières correspondent à une propriété opérationnelle et les secondes à une propriété transactionnelle peut étudiée dans les systèmes de workflow.

6.3.1 Sphères d'instantiation multiple

Etre capable d'exécuter des tâches un nombre indéterminé de fois, en série ou en parallèle est un des besoins classique lors de la définition de procédés. Certains systèmes de gestion de workflow fournissent cette possibilité en utilisant des opérateurs spéciaux d'instantiation multiples. Le terme itération est utilisé dans ce cas pour définir des exécutions multiples en série. Il se trouve cependant que les solutions proposées imposent de nombreuses contraintes pour les concepteurs de workflow et limitent la flexibilité de la modélisation. L'utilisation d'opérateurs spécifiques compliquent et encombrent le modèle de procédé et le rende plus difficilement lisible pour l'utilisateur final.

Les patrons d'instantiation multiple ont été analysés précisément dans [1],[16] et [3]. Les solutions pour les mettre en oeuvre permettent en général de définir ces patrons pour une activité mais pas pour un groupe d'activités. Certaines solutions proposent des combinaisons d'opérateurs de bases, en fixant à l'avance un nombre maximum d'exécution des activités (voir

figure 6.3). Un des patrons (le 15 ou il n'y a pas de connaissance a priori du nombre d'instances) n'a tout simplement pas de solution pour sa mise en oeuvre.

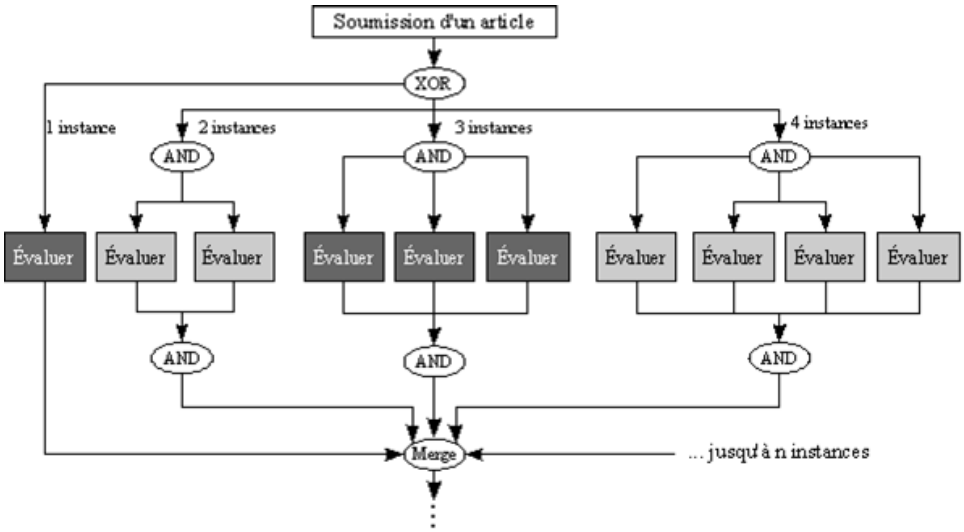


FIG. 6.3 – Instanciation multiple bornée

Le principe des sphères d'instantiation multiple est de considérer séparément la conception du procédé et le nombre de fois qu'un groupe d'activités doit être exécuté. Dans l'exemple simplifié de la figure 6.4, le procédé est défini comme une suite d'étapes sans se préoccuper du nombre de modules qui devront être développés. Les étapes concernant les modules seront exécutées en parallèle autant de fois qu'il y a de modules en fonction de la spécification faite par le concepteur de la sphère d'instantiation multiple.

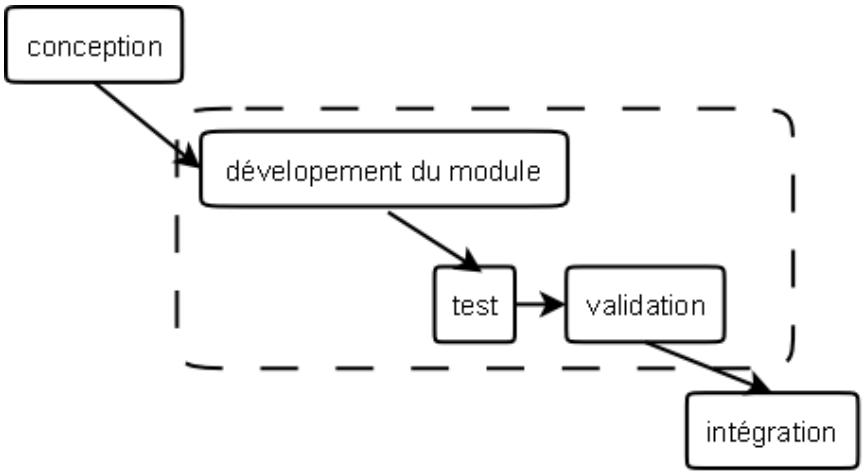


FIG. 6.4 – Instanciation multiple d'un ensemble d'activités

Deux types de sphères d'instantiation multiple ont été identifiés : itérative et parallèle. Chacune a ses propres propriétés ou contraintes

Definition. (Sphère d'instatiation multiple) *Une sphère d'instatiation multiple ω est un*

tuple ($A_\omega, TYPE_\omega, PARAM_\omega$) représentant un groupe d'activités A_ω qui doit être exécuté plusieurs fois en série ou en parallèle. $TYPE_\omega \in \{P,S\}$ représente le type d'instantiation. $PARAM_\omega$ détermine le nombre d'exécution. Ce nombre peut être un constante ou une variable du procédé, ou encore dans le cas d'une exécution en série une condition évaluée chaque fois que la sphère doit être exécutée à nouveau.

Dans le cas d'instantiations multiples en parallèle, le nombre d'instances à créer est évalué dès que la première activité de la sphère est prête à être démarrée. Dans le cas d'instantiation multiple en série, une condition est évaluée lorsque toutes les activités de la sphères ont été exécutées pour décider s'il y a lieu de la réinstancier. Certaines activités peuvent être définies comme des "break activity", forçant l'évaluation de la condition et l'éventuelle réinstantiation avant la fin de la sphère en cours. C'est un peu l'équivalent de l'anticipation pour les sphères.

Les activités composant une sphère peuvent être choisies arbitrairement. Il existe cependant quelques contraintes qui ont été décrites précisément dans [5] sur le structure et les dépendances entre les activités des sphères et du procédé. La plus notable est qu'il ne doit pas y avoir de flot de contrôle partant d'une activité de la sphère vers une activité hors de la sphère puis revenant vers une autre activité de la sphère. Ce type de "cycle" provoquerait un "deadlock" qui ne permettrait pas la terminaison correcte de l'ensemble des activités de la sphère. En revanche, il est possible d'avoir des imbrications de sphères d'instantiation voire même des intersections entre sphères même si dans ce dernier cas, il peut devenir difficile d'anticiper le résultat final de l'instantiation.

6.3.2 Les sphères d'isolation

De nombreux travaux se sont intéressés au problème de l'atomicité des procédés métiers. Des modèles basés sur les modèles de transactions avancées ont été proposés. On en trouve des adaptations dans le domaine des Web Services, utilisant en particulier le principe des activités de compensation pour annuler partiellement les effets d'une activité déjà exécutée et revenir à un état initial satisfaisant en cas d'échec d'un procédé. En revanche, la plupart de ces modèles fonctionnent pratiquement comme si le procédé s'exécutait tout seul. Le problème de l'isolation est beaucoup moins bien pris en compte. Le travail le plus abouti dans ce domaine est certainement la proposition de [14] qui propose un modèle unifié pour l'atomicité et l'isolation dans les procédés transactionnels. Le reproche principal fait à ces modèles est qu'ils imposent une structure et des contraintes d'exécution relativement forte pour résoudre des problèmes relativement rares. Nous avons proposé une approche simplifiée de contrôle de l'isolation pour des parties critiques des procédés. La contrainte ne porte plus sur l'ensemble du procédé pour lequel elle n'est pas forcément indispensable mais pour une partie pour laquelle elle peut être critique. Par exemple, si un procédé contient une séquence de validation puis de publication, il n'est pas forcément acceptable que la donnée à publier soit modifiée après sa validation. Cette simple séquence d'un procédé de production de document doit donc être protégée contre les exécutions concurrentes. Une sphère d'isolation pourra ainsi avoir un double rôle. D'une part contrôler les accès aux données manipulées par les activités d'une sphère d'isolation et celles manipulées par les activités en dehors de la sphère. Elle pourra servir également à contrôler les échanges entre les activités d'une sphère. Ces deux dimensions seront appelées par la suite cohérence pour les interactions entre la sphère et l'extérieur de la sphère et cohésion pour les interactions à l'intérieur de la sphère.

La propriété de cohésion est basée sur la notion de vue commune des données. Cela signifie que toutes les activités de la sphère ont la même vue sur les données auxquelles elles accèdent. Pour augmenter la flexibilité, différents niveaux de flexibilité, inspirés de ceux des bases de

données peuvent être appliqués à la cohésion.
- *Read Uncommitted* Ce niveau permet d'utiliser des données non commitées pendant leur exécution.
- *Read Committed* Ce niveau permet aux activités d'une sphère d'utiliser uniquement des données commitées durant son exécution.
- *Repeatable Read* Ce niveau assure que les données utilisées par une activité d'une sphère ne seront pas modifiées au cours de son exécution.
- *Serialisable* Ce niveau assure une exécution sérialisable des activités de la sphère. Cette contrainte est la plus forte. Il n'y aura aucune interaction au niveau des données pour les activités de la sphère.

La cohérence d'une sphère détermine la façon dont ses activités contrôlent l'accès aux données par des activités en dehors de la sphère (le reste du monde). Le but de la cohérence est d'assurer une exécution transactionnelle du groupe d'activités constituant la sphère. En particulier, elle permet d'assurer que les données modifiées à l'intérieur de la sphère ne seront visibles que lorsque toutes les activités de la sphère auront fini leur exécution. De même qu'il y a différents niveaux de cohésion, il y a également différents niveaux de cohérence pour permettre une plus grande flexibilité dans la définition du comportement de la sphère. Ces niveaux sont les suivants :

- *Cooperative coherence* : Toutes les valeurs produites par les activités de la sphères sont visibles par les activités en dehors de la sphère.
- *Activity coherence* : Seules les valeurs validées modifiées par les activités de la sphère sont visible en dehors de la sphère.
- *Sphere coherence* : Les valeurs produitent par les activités de la sphère ne sont visible que lorsque toutes les activités de la sphère ont terminées leur exécution.

Une sphère d'isolation sera définie à partir d'un ensemble d'activités d'un processus, d'un niveau de cohérence et d'un niveau de cohésion, controlant ainsi les interactions entre les activités de la sphère et avec l'extérieur. Dans le cas le plus fort, on pourra parler d'extra-sphere serialisabilité. La sphère est sérialisée avec les autres activités mais les activités de la sphère elle même ne le sont pas (figure 6.5). A l'inverse on pourra parler d'intra-sphere serialisabilité lorsque l'on voudra sérialiser les activités de la sphère entre elles simplement (figure 6.6). La combinaison des deux nous ramène à une sérialisabilité classique.

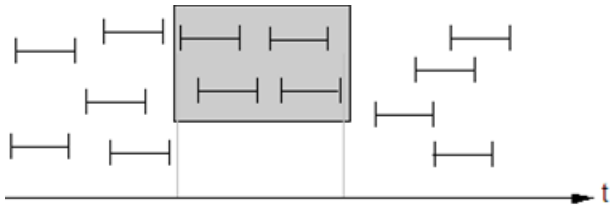


Fig. 6.5 – Extra Sphère Sérialisabilité

Les sphères d'isolation respectent bien les principes que nous avons énoncés concernant les sphères de comportement. Isoler un groupe d'activités est différent d'isoler chacune de ses activités. Le problème de l'isolation est bien un problème orthogonal à la définition du procédé et la flexibilité de l'expression des propriétés est accrue par cette approche.

Le choix des niveaux de cohérence et de cohésion permet de choisir entre douze combinaisons possibles pour définir la stratégie d'échanges de données à l'intérieur de la sphère et avec l'extérieur (voir figure 6.8).

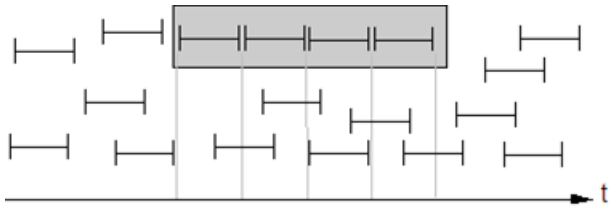


FIG. 6.6 – Intra Sphère Sérialisabilité

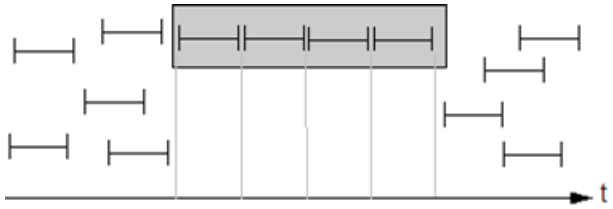


FIG. 6.7 – Sphère Sérialisabilité

Cooperation phenomena →		Disrupted Cooperation	Dirty Read Cooperation	Fuzzy Read Cooperation	Phantom Read Cooperation	External Dirty Read	External Misleading Read
No Isolation Sphere		yes	yes	yes	yes	yes	yes
Cohesion level	**Coherence level**						
Read Uncommited	Cooperative	no	yes	yes	yes	yes	yes
Read Commited	Cooperative	no	no	yes	yes	yes	yes
Repeatable Read	Cooperative	no	no	no	yes	yes	yes
Serializable	Cooperative	no	no	no	no	yes	yes
Read Uncommited	Activity	no	yes	yes	yes	no	yes
Read Commited	Activity	no	no	yes	yes	no	yes
Repeatable Read	Activity	no	no	no	yes	no	yes
Serializable	Activity	no	no	no	no	no	yes
Read Uncommited	Sphere	no	yes	yes	yes	no	no
Read Commited	Sphere	no	no	yes	yes	no	no
Repeatable Read	Sphere	no	no	no	yes	no	no
Serializable	Sphere	no	no	no	no	no	no

FIG. 6.8 – Les niveaux et ce qu'ils permettent

Les sphères peuvent être imbriquées mais contrairement aux sphères de multi-instantiation, l'intersection entre sphères n'a pas de sens. Seule l'imbrication est permise. Les règles de portée des niveaux de cohésion et de cohérence sont relativement simples. Pour une sphère, on considère que le niveau de cohérence qui s'applique est celui de la sphère immédiatement parente. En ce qui concerne la cohésion, on considère les sous-sphères comme une activité de la sphère. Le niveau de cohésion est alors celui de la sphère.

Les différents critères que nous avons définis dans cette partie permettent de garantir différents niveaux de cohésion et de cohérence pour les accès aux données par une activité d'un procédé, en fonction de la sphère à laquelle elle appartient et des propriétés qui y sont attachées. Il faut maintenant pouvoir vérifier que ces critères sont bien respectés à l'exécution.

6.3.3 Le protocole CCDP

Les sphères d'isolations permettent de préciser la visibilité des données entre différents groupes d'activités. Il faut pouvoir contrôler dynamiquement les accès des activités à ces données. Pour cela, nous avons proposé un protocole pessimiste, basé sur des verrous relatifs aux données et aux sphères qui les posent.

Lorsqu'une activité, appartenant à une sphère ou pas tente d'accéder à une donnée sous le contrôle du protocole elle va essayer de mettre un verrou relatif à sa sphère sur la donnée. Ce verrou ne pourra être posé que s'il est compatible avec les verrous déjà posés par les autres activités sur cette donnée. Nous avons proposé un algorithme décrit dans [4] pour vérifier ce protocole.

6.3.4 La mise en oeuvre

Les sphères d'isolation ont été mise en oeuvre dans le cadre d'une plate-forme WebServices. Un éditeur de diagramme BPMN a été étendu pour permettre la définition graphique des sphères et leur spécification. Cet éditeur produit un code XML qui est ensuite transformé dans le code BPEL équivalent, étendu avec les notations correspondantes aux sphères (cf. figure 6.9).

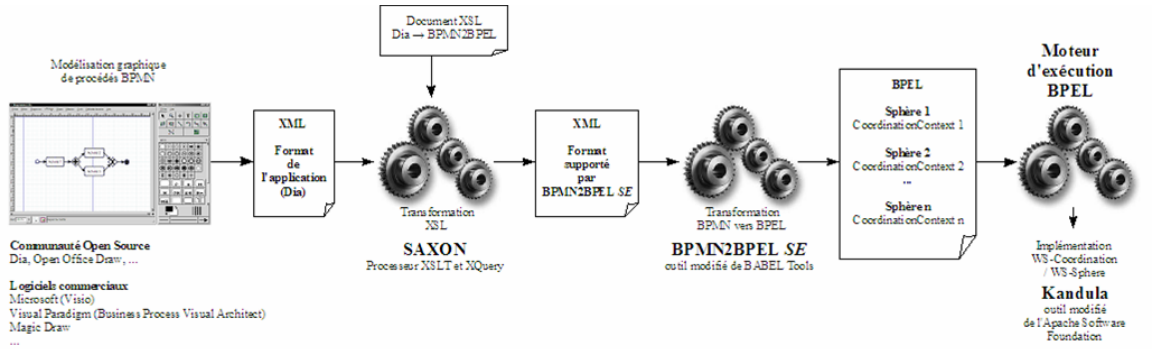


FIG. 6.9 – L'implantation

Le protocole CCDP a lui été mis en oeuvre sous la forme d'une spécification héritant de WS-Coordination. La figure 6.10 montre comment le service des sphères s'intègre dans l'ensemble des services de coordination, incluant les transaction atomiques et les business activity. La connection du moteur BPEL, étendu pour prendre en compte les sphères au service de coordination permet de mettre en oeuvre simplement les sphères d'isolation dans un contexte de WebServices.

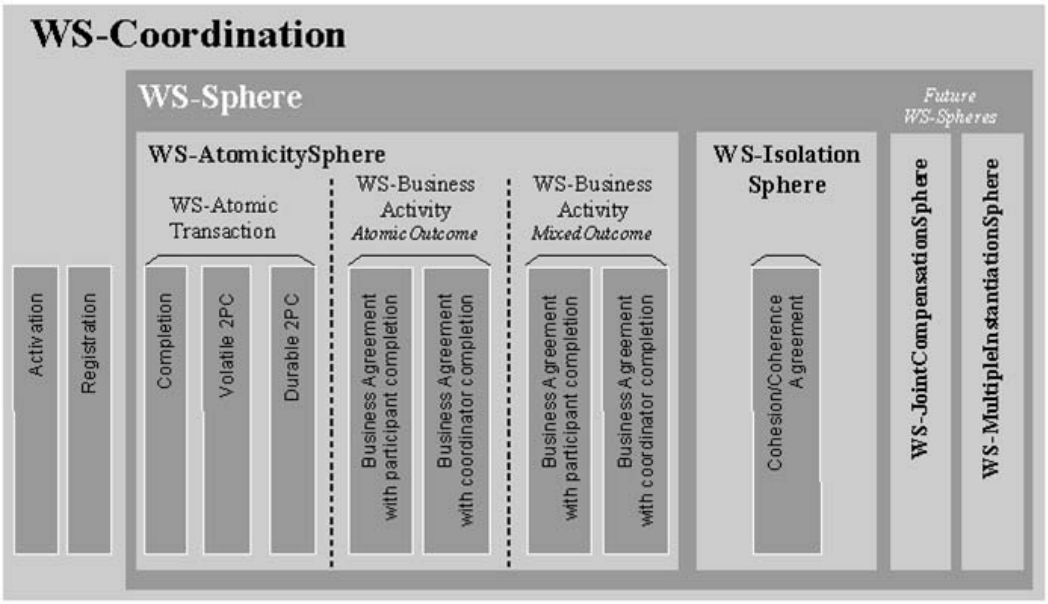


FIG. 6.10 – WS-Sphere

6.4 Conclusion sur les sphères de comportement

Les besoins de flexibilité concernant la définition des procédés et leur mise en oeuvre opérationnel sont importants. Ils sont d'autant plus importants que dans le cadre d'un procédé inter organisationnel, les contraintes que l'on peut contrôler et mettre en oeuvre dépendent non seulement de la forme du procédé mais de sa distribution entre les organisations. Il est donc important de proposer une approche qui permette de garantir le respect des contraintes pour des sous-parties de procédé, et surtout en évitant de remettre en cause leur définition. L'approche basée sur la notion de sphère qui est différente de l'idée de "scope" dans BPEL permet de faire un pas dans cette direction. Nous pensons que cette approche peut être utilisée pour d'autres dimensions, opérationnelles ou organisationnelles de la définition de procédés. Nous pensons en particulier à la définition des contraintes de sécurité sur des processus inter-organisationnels où les contraintes peuvent être différentes selon les organisations et donc selon les parties de procédés concernées. La distribution de l'exécution devrait également faire l'objet d'une étude. Un autre problème plus prospectif, concerne la définition des sphères elles mêmes. Pour l'instant, les sphères sont définies de façon explicite. Il pourrait être intéressant dans certains cas de les définir par intention, surtout dans la cas de procédés complexes, ou susceptible d'évoluer.

[5, 7, 6, 8, 9]

Bibliographie

[1] W. M. P. Van Der Aalst, A. H. M. Ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1) :5–51, July 2003.

[2] Wijnand Derks, Juliane Dehnert, Paul Grefen, and Willem Jonker. Customized atomicity specification for transactional workflow. In *Proceedings of the Third International Sym-*

posium on Cooperative Database Systems for Advanced Applications (CODAS'01), pages 140–147. IEEE Computer Society, 2001.

[3] M. Dumas and A. ter Hofstede. Uml activity diagrams as a workflow specification language. *In Proc. of the International Conference on the Unified Modeling Language (UML). Toronto, Canada, October 2001. Springer Verlag.*, 2001.

[4] Adnene Guabtni. *Behavioural Spheres for Flexible Business Process Modelling and Execution.* PhD thesis, Université Henri Poincaré Nancy 1, 2007.

[5] Adnene Guabtni and François Charoy. Multiple instantiation in a dynamic workflow environment. In Anne Persson and Janis Stirna, editors, *Advanced Information Systems Engineering, 16th International Conference, CAiSE 2004, Riga, Lavtia*, volume 3084 of *Lectures Notes in Computer Science*, pages 175–188. Springer, Jun 2004.

[6] Adnene Guabtni, François Charoy, and Claude Godart. Customizable isolation in transactional workflow. In *First International Conference on Interoperability of Enterprise Software and Applications - INTEROP-ESA'2005 First International Conference on Interoperability of Enterprise Software and Applications - INTEROP-ESA'2005*, Geneva/Switzerland, 02 2005. University of Geneva, Switzerland.

[7] Adnene Guabtni, François Charoy, and Claude Godart. Spheres of isolation : Adaptation of isolation levels to transactional workflow. In *Business Process Management*, pages 458–463, 2005.

[8] Adnene Guabtni, François Charoy, and Claude Godart. Concurrency management in transactional web services coordination. In *17th International Conference on Database and Expert Systems Applications - DEXA 2006, 04/09/2006*, volume 4080/2006 of *in : Lecture Notes in Computer Science, Database and Expert Systems Applications*, pages 592–601, Krakow, Poland, 2006. Gabriela Wagner, FAW, University of Linz, Austria, Springer Berlin / Heidelberg. Non.

[9] Adnene Guabtni, François Charoy, and Claude Godart. Using isolation spheres for cooperative processes correctness. In IEEE Beijing, editor, *The 10th International Conference on CSCW in Design (CSCWD 2006), 03/05/2006*, volume 1 of *Computer Supported Cooperative Work in Design 2006*, Nanjing, China, 2006. Southeast University. Non.

[10] Charles T. Davies Jr. Data processing spheres of control. *IBM Systems Journal 17(2) : 179-198*, 1978.

[11] Frank Leymann. Supporting business transactions via partial backward recovery in workflow management systems. In *BTW*, pages 51–70, 1995.

[12] Frank Leymann and Dieter Roller. *Production Workflow*, chapter Chapter 7 : Workflows and Transactions. Ed. Prentice Hall, Inc., Upper Saddle River, New Jersey, second edition edition, 2000.

[13] Shazia Wasim Sadiq, Maria E. Orlowska, and Wasim Sadiq. Specification and validation of process constraints for flexible workflows. *Inf. Syst.*, 30(5) :349–378, 2005.

[14] Heiko Schuldt, Gustavo Alonso, Catriel Beeri, and Hans-J&#246 ;rg Schek. Atomicity and isolation for transactional processes. *ACM Trans. Database Syst.*, 27(1) :63–116, 2002.

[15] Willem-Jan van den Heuvel and Sergei Artyshchev. Developing a three-dimensional transaction model for supporting atomicity spheres. *International Workshop on Web Services Research, Standardization, and Deployment*, 2002.

[16] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Advanced workflow patterns. In O. Etzion en P. Scheuermann, editor, *7th International Conference on Cooperative Information Systems (CoopIS 2000)*, volume 1901 of *Lecture Notes in Computer Science*, pages 18–29. Springer-Verlag, Berlin, 2000.

Chapitre 7

Coordination et coopération

L'objectif que nous nous sommes fixé nécessite de comprendre comment les personnes participant à un groupe de travail coopèrent et quels peuvent être leurs besoins. Lors du développement des différentes plate-formes dans l'équipe ECOO, nous avons fait de nombreuses hypothèses sur ces besoins, concernant aussi bien la façon de partager des objets que la coordination et les impacts de la distance et du temps sur la coopération [3]. L'utilisation de ces plate-formes pour des expériences informelles nous ont permis d'en évaluer empiriquement le bien fondé. L'opportunité de développer un projet multidisciplinaire pour étudier ces aspects était une occasion importante pour valider plus formellement ces hypothèses et les confronter dans des conditions particulières. Dans le cadre d'un projet labellisé RIAM, Coopéra, nous avons eu l'occasion de travailler à l'évaluation d'une plate-forme de coopération dans des écoles primaires.

Le projet Coopéra avait trois objectifs. Le premier était de produire une plateforme coopérative adaptée aux enfants dans le cadre de travaux pédagogiques. La méthodologie utilisée pour construire cette plate-forme nous amène à notre second objectif qui était suivre pour cette construction un processus participatif incluant des informaticiens, des psychologues, des spécialistes de la pédagogie et bien sur, les enseignants et les élèves des classes participants à l'expérience. Dans notre cas, il s'agissait de classes de CM2 de quatre écoles différentes. Le dernier objectif, et pas le moindre était de comprendre comment la coopération se produisait et l'impact de l'interface du système sur la façon dont les enfants coopéraient et socialisaient à travers elle.

Au cours de ce projet nous nous sommes principalement intéressé à la dynamique de la coopération, i.e. aux facteurs qui favorisent la coopération et ceux qui la restreignent. La méthodologie utilisée pour comprendre comment se produit la coopération et comment les enfants entre en relation est basée sur l'ethnomethodologie et la théorie des actes de langage. Ce choix, guidé par les psychologues de l'équipe était motivé par le fait que ces méthodes permettent d'avoir une compréhension très fine des interactions des utilisateurs. Cette méthode est aussi plus facile à mettre en oeuvre dans une classe que des questionnaires et des interviews. Elle permet en autorisant une intervention directe des expérimentateurs une acculturation des enfants à la coopération médiatisée par des ordinateurs. Cette expérience, si elle ne nous a pas permis de valider totalement l'approche proposée par la plate-forme a permis tout de même de montrer l'importance égale de plusieurs dimensions dans la dynamique de la coopération : la culture des enfants concernant la coopération, le scénario proposé et l'ergonomie du système.

7.1 La méthode et les participants

La méthode suivie, qui nous paraissait appropriée pour étudier l'utilisation de la plate-forme dans des conditions réelles, était basée sur le principe de l'action située. Elle consistait à observer les acteurs dans des situations de travail naturelles. L'approche choisie était donc ethnographique [9] et la théorie servant de base à l'analyse est la théorie de l'activité [22]. Cette théorie est particulièrement adaptée dans un cadre pédagogique pour décrire les actions comme des chaines d'opérations effectuées par des individus sans y penser. L'objectif des observateurs est de clarifier cette chaine d'opérations et de montrer comment elle se transforme en action. Cette méthode est aujourd'hui promue par de nombreux chercheurs dans le domaine du travail coopératif pour comprendre comment les utilisateurs adoptent certains dispositifs techniques pour les aider dans leur travail. L'autre alternative aurait été d'adopter une approche expérimentale mais le contexte du travail coopératif asynchrone se prête mal aux dispositifs contrôlés en raison des nombreuses interactions sociales induites et de la durée nécessaire à l'exécution du travail en groupe. En outre notre but n'était pas la comparaison d'un nombre important de groupes dans des conditions expérimentales variées. De façon générale, l'approche expérimentale, si elle se prête bien à des études sur de dispositifs de travail coopératif synchrone est peu adaptée à des projets coopératifs de longue durée [19]

Pour finir, dans le contexte de cette expérience, nous avions à faire à deux types d'utilisateurs : les élèves et les enseignants. Les élèves étaient les acteurs directes de l'étude mais les enseignants avaient également un rôle important dans la préparation des travaux et dans la conception de la plate-forme. L'approche choisie de ce point de vue était une approche inspirée de la conception participative [6].

7.1.1 La constitution des situations naturelles

L'activité humaine est complexe. Elle transforme son environnement en même temps que l'environnement la transforme dans un couplage action/environnement [16, 21]. Comme nous l'avons déjà dit, mettre en place un système expérimental pour comprendre des situations aussi complexes est illusoire. Il est même nécessaire de mettre en place des situations dans lesquelles des pratiques naturelles peuvent avoir lieu. Vygotsky [22] propose une méthode pour atteindre ce but. Cette méthode a été suivie en constituant dans chaque classe des paires d'élèves qui travaillaient dans leur propre classe sur un projet conçu par leurs enseignants. C'est une pratique classique de l'utilisation des ordinateurs dans les classes. L'avantage était de pouvoir analyser non seulement les interactions des éleves avec la machine mais également la conversation qui se produisait au cours des séances. Celles-ci étaient filmées pour conserver des traces linguistiques et matérielles. De plus nous étions présents lors des séances, comme observateurs des activités.

7.1.2 L'analyse des interactions conversationnelles

Les interactions enregistrées concernaient principalement le langage, les manipulations des objets (souris, clavier) et les événements à l'écran. L'analyse de la partie linguistique s'est appuyée sur la théorie des actes de langage [17, 20]. Cette approche n'est pas équivalente à l'analyse conversationnelle de l'éthnométhodologie mais elle permet de rendre compte des dimensions cognitives et sociales des conversations. Elle permet de rendre compte en même temps des signifiés construits par le sujet et des relations sociales qui apparaissent pendant l'activité. Néanmoins, l'étude porte plus qu'uniquement sur le langage produit. Nous nous sommes également intéressés aux objets manipulés parce qu'ils sont les moyens de la coordination des actions et de la

production de la connaissance partagée.

7.1.3 Le rôle des acteurs/utilisateurs dans la conception

Nous avons considéré dès le départ que le fait de se placer dans un contexte naturel de travail pour mener l'étude nécessitait d'impliquer le plus tôt possible les utilisateurs dans le processus de conception. Nous ne les avons pas considéré comme des testeurs ni des évaluateurs mais comme des prescripteurs. C'est pourquoi les objectifs des situations étaient de comprendre les usages dans un contexte de coopération nécessitant des synchronisations complexes : opérationnelles (concernant la plate-forme), cognitives (concernant la connaissance produite) ou sociales (concernant la conscience de groupe). Cet objectif était important en raison du peu d'habitude de ce type de travail des enfants et des enseignants, même s'ils étaient volontaires pour travailler dans ce sens.

7.1.4 Ergonomie, scénario et culture

Le développement de la plateforme de coopération à la destination d'élèves doit prendre en compte des besoins particuliers en terme d'ergonomie. La plateforme doit bien sure être efficace et utilisable. Elle doit également permettre la mise oeuvre des scénarios pédagogiques. C'est le scénario qui doit permettre de situer l'action et de construire des situations d'observation naturelles. Les scénarios doivent impliquer une forme de coopération. Ils doivent être motivant pour les élèves et ils doivent impliquer les enseignants dans le projet. La construction des situations de travail est particulièrement sensible de ce point de vue. La construction de la plateforme doit aussi tenir compte des relations sociales entre les individus. Elle s'inscrit donc dans un environnement culturel complexe.

Les analyses produites doivent rendre possible la caractérisation de la plateforme selon ces trois dimensions essentielles : l'ergonomie, les scénarios et la culture.

7.1.5 Le protocole expérimental

Nous avons distingué deux types de participants, les élèves et les enseignants. Les élèves sont les utilisateurs directs. Ils travaillent en coopération entre les différentes écoles. Les enseignants sont plutôt des administrateurs de la plate-forme. Ce sont eux qui définissent les scénarios pédagogiques. Nous avons travaillé avec des écoles primaires de Nancy et de la banlieue (CM2). L'accès aux ordinateurs avait lieu dans des salles dédiées et les élèves travaillent à deux par ordinateur pour des sessions de durée limitée. Trois projets ont été menés sur trois versions successives de la plate-forme.

Le premier projet, "Poèmes" consistait pour des binômes de chaque école à coopérer pour produire, illustrer et mettre en forme des poèmes. La coopération a donné lieu à la production d'un recueil. Le second projet, "Operette" avait pour but de construire un site web sur l'opéra de Nancy. L'objectif pour les élèves était d'apprendre à partager des compétences et des connaissances, puis d'articuler des idées pour arriver à un résultat commun. Le dernier projet consistait en la production d'un journal en ligne à partir de thèmes choisis par les élèves.

7.2 La plateforme de coopération

La plate-forme de coopération était basée sur un prototype développée dans l'équipe Ecoo en PHP/MySQL, la ToxicFarm, orientée à l'origine vers le développement de logiciel. Cette plate-

forme était étendue avec des mécanismes spécifiques pour améliorer la conscience de groupe.

7.2.1 Le modèle de coopération

Le modèle de coopération implanté dans la plateforme originale est relativement classique. Il est basé sur le protocole Copy/Modify/Merge (figure 7.1). Sur la plate-forme est stocké l'espace partagé des projets. Un participant à un projet crée d'abord un espace privé pour pouvoir modifier les fichiers de cet espace partagé. Ensuite, il synchronise cet espace sur son ordinateur pour pouvoir effectivement modifier les fichiers de l'espace privé, puis publier ses travaux dans l'espace partagé. Les conflits doivent être résolus avant la publications des objets modifiés dans l'espace partagé. L'idée principale du protocole est d'essayer de prévenir les conflits en indiquant aux utilisateurs les objets modifiés par leurs partenaires avant leur publication, pour éviter les modifications concurrentes (figure 7.2).

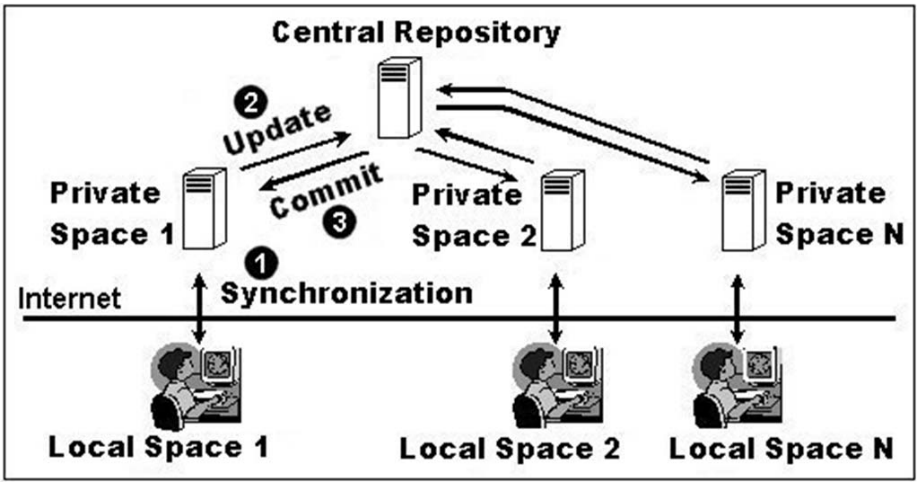


FIG. 7.1 – Le modèle de partage de fichiers

7.3 L'analyse d'usage

Nous ne détaillerons pas ici l'analyse d'usage. Un exemple complet de cette analyse a été publié [13]. Cette analyse a été produite en identifiant des séquences de travail significative et en procédant à une micro analyse de ces séquences. Elles ont permis de travailler sur les problèmes d'ergonomie mais également sur les problèmes culturels. Nous avons ainsi pu noter que les élèves étaient plus concentré sur l'utilisation de la plateforme (respecter des consignes, atteindre leur objectif) que sur la coopération avec les autres. Nous avons pu vérifier que le facteur le plus important pour engager cette coopération de façon efficace, indépendamment de l'ergonomie et du cadre culturel, est le scénario qui doit réellement engager à la coopération.

L'ergonomie de l'interface est un aspect fondamental. Un des objectifs de celle mise en oeuvre était de permettre aux élèves de comprendre la dynamique de la coopération (le travail sur des documents communs). Lors des deux premières expérimentations, les élèves avaient du mal à comprendre les icones de l'interface et les situations auxquelles ils étaient confrontés. La troisième version de l'interface a été largement améliorée pour montrer de façon explicite le modèle de partage de fichier, réduisant ainsi l'effort cognitif à faire pour le comprendre.

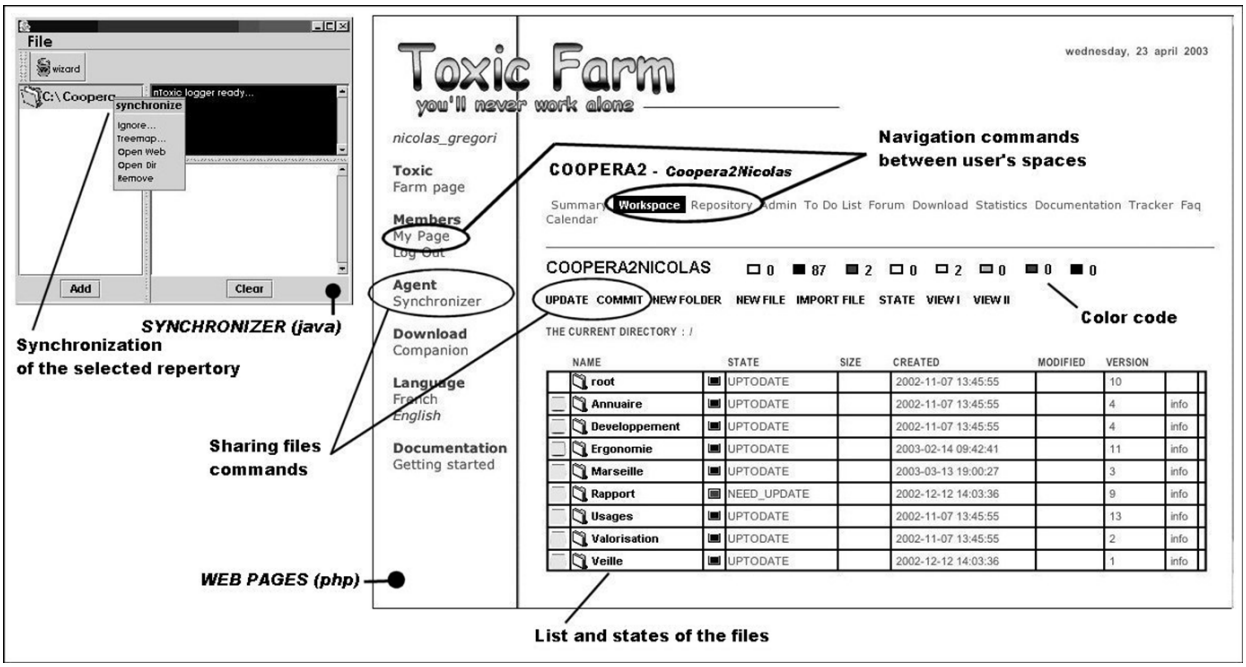


FIG. 7.2 – La plate-forme ToxicFarm

7.3.1 Les aspects historico-culturels

La dynamique de la coopération ne peut être séparée de l'émergence de la conscience de groupe [7, 8]. Cette conscience est le résultat de la capacité des acteurs à penser à la place, au rôle et aux activités de tous les membres du projet. Plus précisément, la conscience de groupe est caractérisée par la capacité des acteurs à discerner les intentions et les attentes des autres envers eux-mêmes et par des comportements intentionels envers les autres. Mettre des enfants dans une dynamique relationnelle nécessite de savoir traduire les propriétés du système en signes. Cette traduction s'exprime lorsque les enfants visualisent ces signes. Les enfants doivent non seulement comprendre ces signes, mais ils doivent également les relier à leur sens dans la dynamique de la coopération. Lorsque les utilisateurs convertissent ces signes en sens, il ne font pas qu'interragir avec le système mais avec d'autres utilisateurs à travers le système.

Cette traduction était un problème pour Coopera. Nous avons pu l'observer durant la première expérience. Cela nous a amené à mettre en place des animations pour permettre l'appropriation par les enfants de la relation à eux-mêmes et aux autres et au modèle de partage de fichiers qui supporte ce modèle relationnel. Ces animations ont eu lieu dans les classes en présence de tous les enfants. Leur but était de reproduire le fonctionnement du système avec des objets réels. Les enfants pouvaient les manipuler pendant l'animation. Ce choix, inspiré de Vitgosky, était motivé par le fait que le concept d'acquisition dépend de la façon dont l'expérience est vécue. Cette expérience ne doit pas seulement être cognitive. Elle doit aussi être physique et émotionnelle.

7.3.2 Les scénarios

Le scénario est une partie importante dans l'utilisation de l'outil. Le but à atteindre était d'avoir un impact sur la motivation et la satisfaction des utilisateurs quand ils utilisent le système.

C'est particulièrement important pour son appropriation.

Plusieurs modèles de scénario (figure 7.3) ont été définis avec différents niveaux de coopération. Avec le premier modèle, individuel, chaque acteur doit faire son travail lui même. Le second modèle, inter-individuel/fichiers partagés est le premier niveau de coopération. Les utilisateurs sont groupés en équipe. Ils partagent le même fichier mais il n'y a pas de dépendance dans leur production. Le troisième modèle est inter-individuel/inter-article. C'est le second niveau de coopération. Les individus partagent le même fichier et il y a des dépendances dans leur production. Le dernier modèle est inter-individuel/intra-fichier. C'est le troisième niveau de coopération. Il est mis en oeuvre lorsque les utilisateurs sont groupés en équipes entre les classes. Ils partagent le même fichier et produisent le même document. C'est le plus haut niveau de coopération.

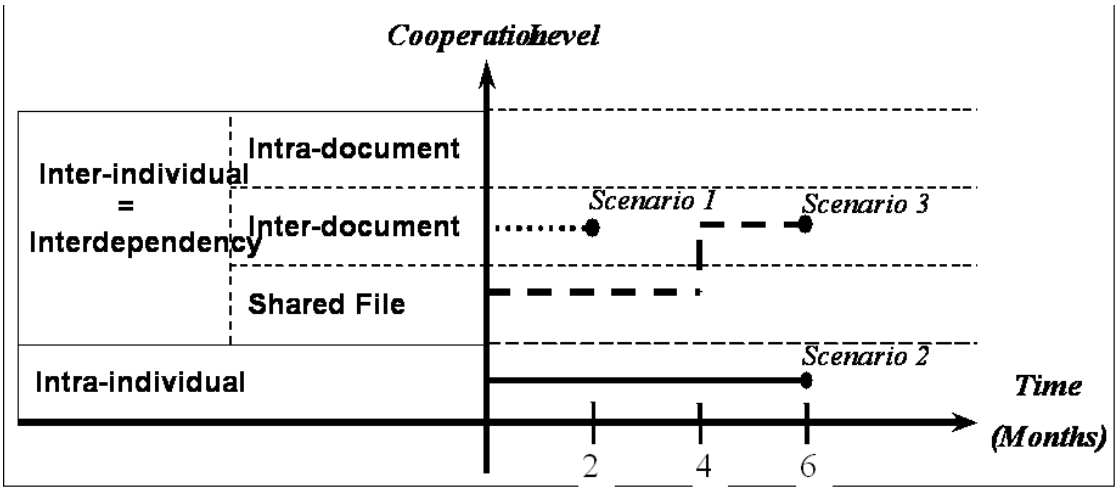


Fig. 7.3 – Les scénarios

Le scénario du troisième projet est apparu comme étant le plus approprié pour la coopération entre des élèves de CM2. Ils ont assez de temps pour s'habituer au système et faire leur travail. Ils sont intéressés par l'objectif et sont motivés pour coopérer. Certaines contraintes liées au fonctionnement des classes doivent cependant être prises en compte. La première concerne la fréquence d'utilisation des ordinateurs par les élèves. Ils n'avaient qu'une séance par semaine dédiée à ce projet. La seconde contrainte est que les élèves d'une même classe doivent tous utiliser le système en même temps. Les interdépendances entre les productions ne peuvent être mises en place que dans un second temps après que toutes les classes aient effectuées au moins une séance sur le projet.

7.3.3 Relations entre les facteurs d'utilisabilité

les trois facteurs que nous avons étudiés, l'ergonomie, la cadre culturel et les scénarios d'apprentissage sont très intriqués. Ils contribuent tous à l'utilisabilité de la plateforme.

Par exemple, les propriétés ergonomiques de Coopéra 1 (figure 7.4) n'étaient pas satisfaisantes et les enfants ont eu des difficultés à interpréter les situations qu'ils rencontraient. Néamoins, la plupart d'entre eux ont effectué leur tâche sans trop de difficulté.

Il y a plusieurs explications à cela. Tout d'abord, il faut se souvenir que l'appropriation du modèle de partage des fichiers par les enfants à eu lieu lors d'une animation. Le scénario de l'animation n'impliquait pas beaucoup d'interactions coopératives. En fait, la situation qu'il

rencontrait était toujours la même et ils n'avaient qu'à répéter le même procédé. Les difficultés sont venues du fait qu'ils essayaient juste de se souvenir et de répéter ce procédé sans être aidé par la plateforme.

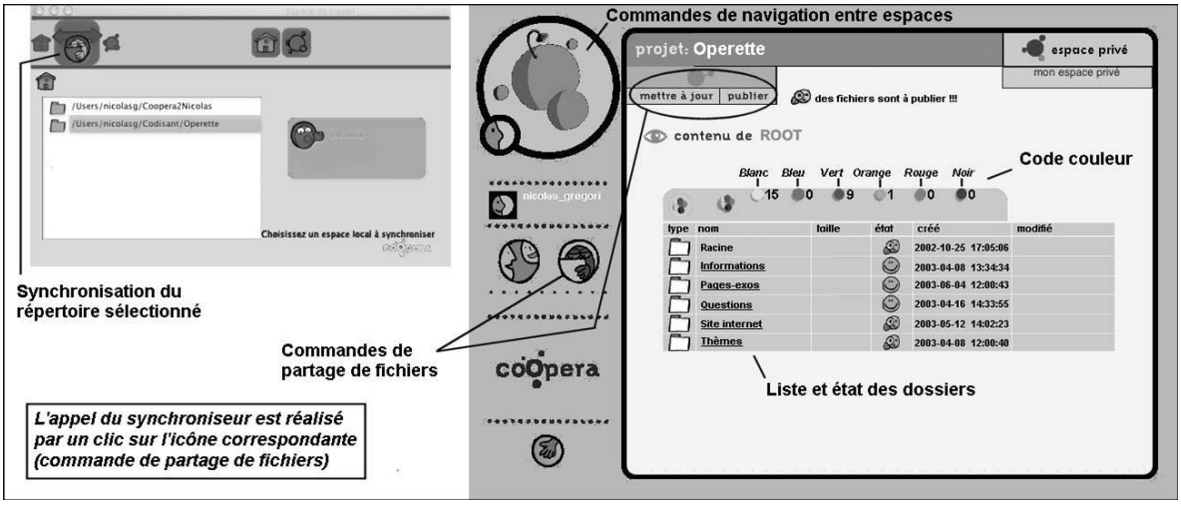


FIG. 7.4 – Coopera 1

Des leçons ont été tirées des problèmes rencontrés avec cette interface pour mettre en place une interface rendant beaucoup plus explicite le modèle de partage de fichiers (figure 7.5).

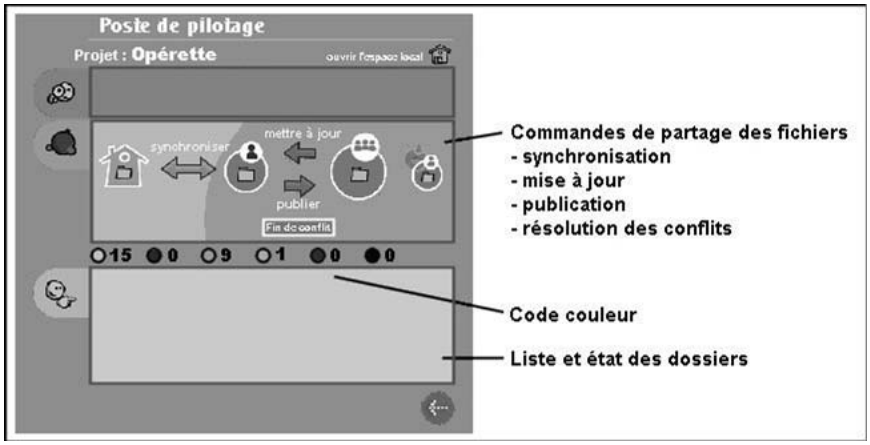


FIG. 7.5 – La version finale de Coopéra

7.4 Discussion

La difficulté principale rencontrée par les enfants était liée à l'exécution de la dynamique de coopération. Ils ne comprenaient pas spontanément les intentions des autres. De plus, ils ne généraient pas d'activités intentionnelles en direction des autres. Cette difficulté provient principalement des facteurs historico-culturels. Le schéma de coopération mis en oeuvre dans la plate-forme est en rupture avec leur pratique du travail en groupe. Il n'est pas commun

lors d'un exercice à l'école de travailler à plusieurs sur le même fichier et d'autoriser les autres à modifier une production personnelle. Même après les animations dans les classes, l'appropriation du modèle relationnel a nécessité plusieurs séquence de travail sur le système. En fait, il a fallu qu'ils travaillent d'abord sur les fonctions fondamentales (la synchronisation, la publication et les mises à jour) de la plateforme avant qu'émerge un comportement intentionnel puis que la conscience de groupe apparaisse.

A ce niveau, l'ergonomie est importante. Il est fondamental que les enfants puissent utiliser ces fonctions avec le minimum de charge cognitive. En outre, le rôle de l'enseignant est fondamental pour générer la conscience de groupe. Il doit expliquer aux enfants les comportements intentionnels qu'ils doivent percevoir et adopter. Il doit leur permettre d'intérioriser ces intentions. D'ailleurs, l'intérêt de la plate-forme est de forcer les enfants à être moins centrés sur eux mêmes. Il doivent également produire une pensée réflexive sur leur place et leurs actions à l'intérieur du projet coopératif. Ainsi, la conscience de groupe, dans ce contexte particulier, n'est pas une condition sine qua non pour utiliser la plateforme. C'est surtout un but à atteindre à travers son utilisation. De ce point de vue, les animations avaient deux objectifs : initier les enfants aux fonctions nécessaires pour mettre en oeuvre la dynamique coopérative et leur expliquer le mode relationnel proposé par le système.

A la fin nous avions atteint l'état suivant. Les élèves savaient utiliser la plateforme. Ils savaient exécuter les commandes. Ils étaient capable de participer aux activités du groupe. En revanche, ils avaient toujours des difficultés à relier leurs actions au modèle de partage de fichiers.

La méthodologie utilisée nous a permis de bien identifier ces problèmes. Elle était réellement centrée sur l'activité des enfants en situation. Ils étaient encouragés à parler et à exprimer leurs sentiments. Ils avaient pu expérimenter physiquement les concepts de la plateforme durant des animations préliminaires. Ceci nous a permis d'affirmer que pour analyser une situation coopérative, se concentrer uniquement sur le point de vue ergonomique est insuffisant. En faisant cela, nous aurions pu avoir l'impression que tout allait bien. En fait, l'ergonomie, la culture et le scenario sont complémentaires. Ils doivent être considérés simultanément pour étudier des activités coopératives.

7.5 La coordination

La dynamique de coopération étudiée dans ce travail s'est concentrée principalement sur le problème du partage de fichiers. Les problèmes de coordination étaient essentiellement réglés par des réunions entre l'équipe et les enseignants ou par courrier électronique. La vitesse d'avancement relativement lente du projet et la proximité géographique des écoles permettaient ce type d'interactions. Nous avons cependant réfléchit dans le cadre de Coopéra 2 à un outil permettant de supporter la coordination entre les écoles et de contrôler et d'assister à l'avancement du projet. Un modèle de workflow simplifié et adapté a été mis oeuvre dans le système. Ce modèle est une tentative pour intégrer planification et coordination. La planification est un moyen pour décomposer le projet en grandes étapes avec leurs objectifs et leur délai (figure 7.6).

A l'intérieur de chaque étape sont ensuite définies des activités et leurs dépendances. Les activités sont ensuites distribuées aux élèves (figure 7.7). Une activité est exécutée autant de fois qu'elle a d'acteurs. Pour simplifier l'utilisation du système, à chaque activité devait être connecté un espace privé destiné à recueillir les données produites lors de l'activité puis à les publier à la fin de l'activité. Cette connection est équivalente à celle qui a été mise en place dans le projet Corvette [1]. Les activités sont également un espace de communication permettant de commenter leur résultat et les difficultés rencontrées pour les mener à bien 7.7.

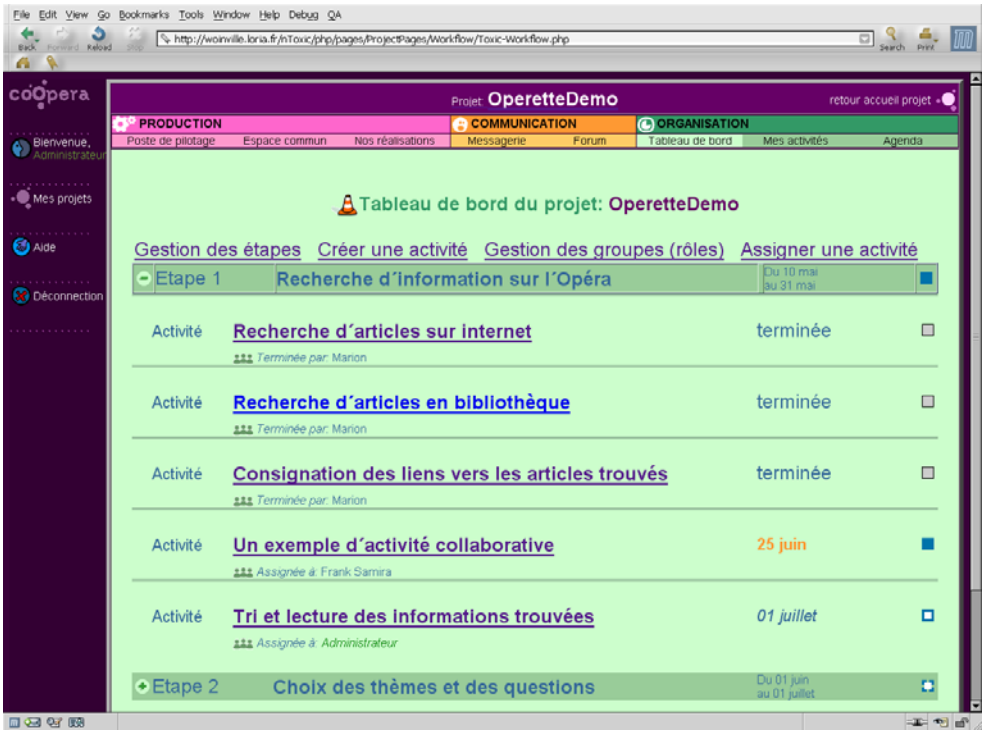


FIG. 7.6 – L'interface Workflow de Coopéra

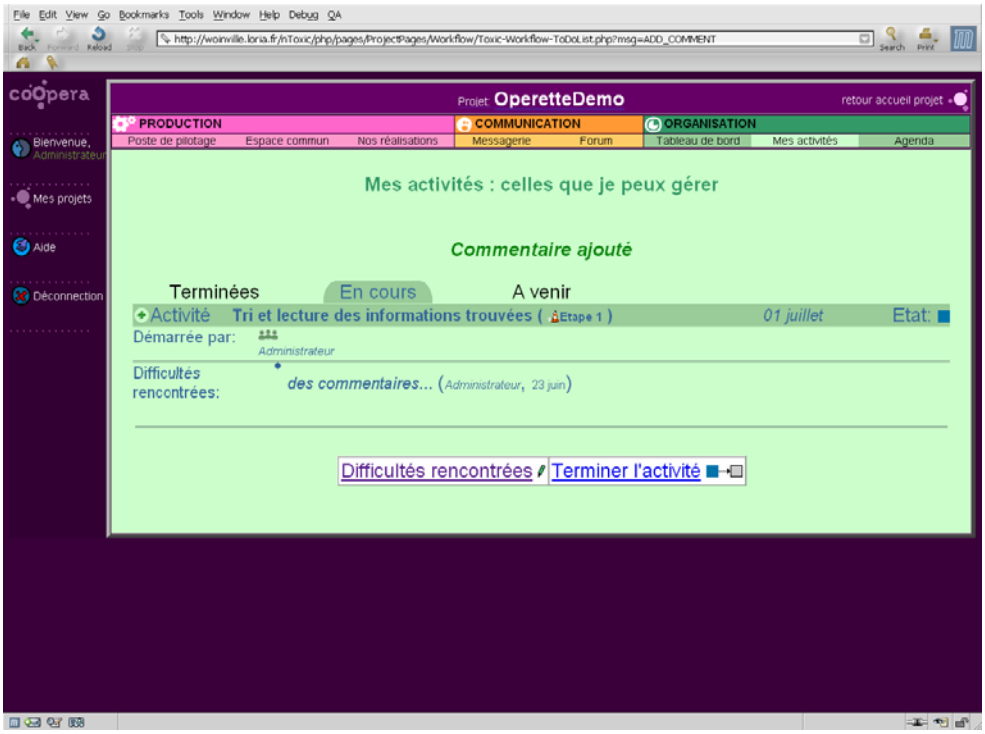


FIG. 7.7 – Une activité dans coopéra

Cette fonctionnalité a été peu étudiée au cours des expériences avec Coopéra. Si le besoin d'améliorer la coordination entre les classes s'est fait ressentir, en particulier dans l'optique d'une utilisation plus autonome de Coopéra par les écoles, la mise en place du système de gestion des activités a été réalisée trop tardivement pour en tirer des enseignements concrets. L'idée de combiner plan et procédé nécessiterait d'être étudiée plus en profondeur.

7.6 Conclusion

L'étude de la dynamique de la coopération, des moyens et des supports pour la faciliter, en particulier dans un contexte asynchrone est particulièrement difficile. Monter une expérience et arriver à mettre des utilisateurs en situation réelle de travail, les observer et analyser les données de l'observation prend énormément de temps. Il n'est pas étonnant que la plupart des travaux que l'on puisse trouver sur ce thème concernent soit des systèmes permettant de faciliter le travail collaboratif synchrone, soit des analyses de données a posteriori produites par des projets coopératifs comme le développement de logiciel libre [18] ou encore de simples interviews. Nous pensons cependant qu'une bonne compréhension des processus en jeu, nécessaires pour améliorer les outils et leur acceptation par les utilisateurs est nécessaire. La méthodologie mise en oeuvre dans cette expérience, bien que couteuse en temps, est une source importante d'enseignements. D'autres collaborations pluridisciplinaires devront être mises en place dans le futur pour affiner l'approche et ses résultats. Ce sera en particulier le cas dans le cadre du projet Icrisis de simulation de gestion de crises qui doit démarrer en septembre 2007.

Bibliographie

[1] Karim Baina, François Charoy, Claude Godart, S. El Hadri, Hala Skaf, S. Akifuji, T. Sakaguchi, Y. Seki, and M. Yoshioka. Corvette : a cooperative workflow for virtual teams coordination. *International Journal Networking and Virtual Organisations*, 2(3) :232–245, Dec 2004.

[2] Karim Baïna, François Charoy, Claude Godart, Daniela Grigory, Saad el Hadri, Hala Skaf, Shunsuke Akifuji, Toshiaki Sakaguchi, Yoko Seki, and Masaichiro Yoshioka. Corvette : A cooperative workflow development experiment. In *3rd IFIP Working Conference on Infrastructures for Virtual Enterprises - PRO-VE'2001, Sesimbra, Portugal* , May 2002.

[3] J. Bardram. Designing for the dynamics of cooperative work activities. In *CSCW'98*, 1998.

[4] François Charoy, Claude Godart, Nicolas Gregori, Jean-Charles Hautecouverture, and Sébastien Jourdain. Coopera : An environment for teaching and learning internet cooperation. In *IADIS International Conference e-Society 2004, Avila, Espagne*, pages 323–330, Jul 2004.

[5] François Charoy, Claude Godart, Pascal Molli, Gerald Oster, Marc Patten, and Miguel Valdes. Services for virtual teams hosting : Toxicfarm introduction. In *Second International Workshop on Cooperative Internet Computing - CIC 2002, Hong Kong, China*, pages 105–112, Aug 2002.

[6] Andy Crabtree, Michael Twidale, Jon O'Brien, and David M. Nichols. Talking in the library : Implications for the design of digital libraries. In *ACM DL*, pages 221–228, 1997.

[7] Paul Dourish. The parting of the ways : Divergence, data management and collaborative work. In *ECSCW*, pages 213–, 1995.

[8] Paul Dourish and Victoria Bellotti. Awareness and coordination in shared workspaces. In *CSCW*, pages 107–114, 1992.

[9] H. Garfinkel. *Studies in Ethnomethodology.* Prentice Hall, 1967.

[10] Claude Godart, Christophe Bouthier, Philippe Canalda, François Charoy, Pascal Molli, Olivier Perrin, Helene Saliou, Jean-Claude Bignon, Gilles Halin, and Olivier Malcurat. Asynchronous coordination of virtual teams in creative applications (co-design or co-engineering) : requirements and design criteria. In *Information Technologies for Virtual Enterprises*, Jan 2001.

[11] Claude Godart, François Charoy, Marc Patten, Nicolas Grégori, Jean-Charles Hautecouverture, and Isabelle Faugeras. Coopéra : apprendre à coopérer et apprendre en coopérant. In *Conférence Internationale sur l'Enseignement Ouvert et en Ligne - ICOOL 2003, Ile Maurice*, Dec 2003.

[12] Claude Godart, François Charoy, Olivier Perrin, and Hala Skaf. Cooperative workflows to coordinate asynchronous cooperative applications in a simple way. In IEE Press, editor, *Parallel and Distributed Systems - PADS 2000*, pages 409–416. IEEE Computer Society, Jul 2000.

[13] Nicolas Grégori, Jean-Charles Hautecouverture, François Charoy, and Claude Godart. Combining ergonomics, culture and scenario for the design of a platform for cooperation. *Artificial Intelligence and Society*, 20(3) :384–402, 2006.

[14] Jean-Charles Hautecouverture, Nicolas Grégori, François Charoy, Claude Godart, Marc Patten, and Isabelle Faugeras. Coopera : Analyse de l'usage d'une plate-forme de coopération à destination d'enfants du primaire. In *Human Centered Processes - HCP'2003, Luxembourg*, May 2003.

[15] Jean-Charles Hautecouverture, Nicolas Grégori, François Charoy, Claude Godart, Marc Patten, and Isabelle Faugeras. Analyse d'usage d'une plate-forme de coopération : usage et développement logiciel. *In Cognito - Cahiers Romans de Sciences Cognitives*, 1(3) :45–77, 2004.

[16] Edwin Hutchins. *Cognition in the Wild (Bradford Books).* The MIT Press, September 1996.

[17] Stephen C. Levinson. *Pragmatics (Cambridge Textbooks in Linguistics).* Cambridge University Press, June 1983.

[18] Audris Mockus, Roy T. Fielding, and James D. Herbsleb. Two case studies of open source software development : Apache and mozilla. *ACM Trans. Softw. Eng. Methodol.*, 11(3) :309–346, July 2002.

[19] Andrew F. Monk, Bonnie A. Nardi, Nigel Gilbert, Marilyn M. Mantei, and John McCarthy. Mixing oil and water ? : Ethnography versus experimental psychology in the study of computer-mediated communication. In *INTERCHI*, pages 3–6, 1993.

[20] John R. Searle and Daniel Vanderveken. *Foundations of Illocutionary Logic.* Cambridge University Press, July 1985.

[21] Lucy A. Suchman. *Plans and Situated Actions : The Problem of Human-Machine Communication (Learning in Doing : Social, Cognitive & Computational Perspectives)*. Cambridge University Press, November 1987.

[22] L. S. Vygotsky. *Mind in Society : The Development of Higher Psychological Processes*. Harvard University Press, November 1980.

Chapitre 8

Perspectives

Le Web est considéré aujourd'hui comme un système global d'information et de communication. Il devient également une plate-forme globale de services. Parmi les plus intéressants, on peut citer les services fournit par Amazon comme l'Amazon S3 pour stocker des données, l'Amazon E-Commerce Service qui fournir des services pour mettre en place une place un site de E-Commerce. D'autres services sont disponibles pour stocker et organiser des photos, vendre ou acheter sur des places de marché, accéder à des informations géographiques, produire des documents. A partir de ces services, il est possible de construire de de nouvelles applications et de nouveaux services.

Ainsi, il est possible aujourd'hui, en théorie, d'abstraire une entreprise à l'ensemble des services qu'elle peut rendre à ses clients. Un site de transporteur aérien peut être vu comme un ensemble de services de consultations d'horaires et de réservation de billets d'avion. Un site de vente aux enchères peut être vu comme un ensemble de services de recherche, de mise en vente, de suivi des enchères et d'achats. Un site de banque peut lui fournir des services de consultation de comptes bancaires, mais aussi et surtout des services de transfert sécurisé d'argent d'un compte à un autre. Une entreprise peut alors être fondée sur l'idée d'acheter des lots de billets à une compagnie aérienne puis les mettre aux enchères sur un site de vente en utilisant les services de la banque pour assurer le transfert d'argent du site de vente aux enchères vers son propre compte. Cette entreprise va mettre en place un nouveau service, décrit par un procédé d'achat, de mise en vente de billet puis de récupération du paiement et éventuellement de traitement des différents problèmes qui peuvent se poser au cours de son exécution. Le procédé métier de cet entreprise sera construit à travers des conversations avec les différents procédés indiqués. L'entreprise va orchestrer son service en gérant les conversations avec les services qu'elle utilise sans connaitre forcément les procédés mis en oeuvre ceux ci. Elle n'en connaitra que la choréographie qui décrit la conversation qu'un client doit avoir avec ces services. Cette connection entre procédés de différentes organisations est une tâche complexe et difficile à maitriser. Chaque procédé doit être mis en place et validé individuellement. Il sera confronté au cours de son existence à de nombreux problèmes liés en particulier à l'évolution des services composés. Chaque évolution des services utilisés pourra remettre en cause l'intégralité du modèle métier de l'entreprise. Celle-ci devra à la fois être capable de réagir au changement dans son environnement et être également capable d'évoluer pour fournir de nouveaux services à ses clients, intégrer de nouvelles compagnies aériennes dans son plan, proposer des billets sur différentes places de marché.

Cependant, ces évolutions amènent également leur lots récurrents d'anciens problèmes, réactualisés, ou de nouveaux problèmes liés aux usages et aux contraintes que l'on veut imposer sur ces usages.

Parallèlement, ces dernières années et ces derniers mois on vu apparaitre un ensemble très important d'applications disponibles en ligne et proposant des capacités de collaboration en ligne. Google Docs, la suite d'outils de Zoho[5], des outils de brainstorming en ligne (MindMeister, uBubbl) fournissent tous des fonctions permettant à un groupe d'utilisateur de collaborer pour produire un document en ligne. De plus, ces applications exposent des interfaces qui permettent différentes formes d'intégration dans des aggrégateurs comme Netvibes, PageFlakes ou iGoogle.

La question n'est donc plus aujourd'hui de fournir la plate-forme répondant à tous les besoins mais de permettre à un utilisateur ou à un groupe d'utilisateurs de composer l'environnement dont ils ont besoin pour atteindre leur objectif. La question de l'orchestration de l'utilisation de ces services reste cependant entière. Il ne suffit pas d'agréger des services pour coordonner une équipe. Le problème qui existait dans l'approche monolythique reste entier. Il est même compliqué par la décentralisation de l'exécution de ces services.

Ce domaine des services est à la frontière de l'informatique, des sciences de l'information et des sciences sociales et de l'organisation. Il offre de nombreuses perspectives et de nombreux domaines possibles d'applications, ouvrant des champs de recherche théorique, pratique et liés aux applications particulièrement intéressants. Ce sont certains de ces champs que nous nous proposons de prospecter dans les années à venir avec comme idée directrice le support aux activités coopératives.

Les besoins liés à la coordination d'activités coopératives sont grandissants. Des domaines considérés comme prioritaires au niveau européen font apparaitre des problèmes spécifiques et nouveaux qui peuvent trouver des réponses dans une recherche à la fois théorique et appliquée, prenant en compte la dimension socio-technique des verrous qui sont posés.

L'enjeu de la mise en oeuvre de nouvelles pratiques de gouvernance grâce et par les nouvelles technologies de l'information fait partie de ces priorités. Sous le terme de eGouvernement se regroupe un ensemble de problèmes qui s'approchent fortement des questions qui nous préoccupent : renforcer les coopérations entre différents services gouvernementaux et entre différents échelons administratifs. Dans ce contexte, de nombreux projets font appels à des compétences de différents services et nécessitent des productions, des avis et des décisions de nombreuses personnes. Suivre l'avancement de ces projets d'un bout à l'autre, de leur conception à leur évaluation nécessite d'être capable de coordonner des personnes mais également des procédés. De plus, la durée potentiellement longue de ces projets implique forcément de prendre en compte de nombreuses évolutions à tous les niveaux. En outre, il est également nécessaire de considérer les problèmes culturels, les différences de méthode, les questions légales pour mener à bien cette coordination.

8.1 Confiance et Sécurité dans les procédés inter-organisationnels

Une dimension particulièrement importante à prendre en compte est celui de la sécurité et de la confiance lors de la collaboration entre différentes organisations. Même si la volonté de coopérer existe entre plusieurs organisations, cette coopération n'aura lieu que s'il existe une certaine confiance entre eux et surtout si le système fournit des garanties sur la façon dont le procédé va s'exécuter. C'est le problème sur lequel nous avons commencé à travailler en collaboration avec une équipe de SAP Research.

Un procédé inter organisationnel n'est pas simplement un procédé qui fait appel à des services de différentes organisations. C'est un procédé dont l'exécution peut être distribuée à travers ces organisations. Cette distribution est possible à condition que ces organisations soient d'accord

[5] www.zoho.com

pour coopérer. Elles ont en général des raisons pour le faire. Ces raisons peuvent être politiques, juridiques ou de simple responsabilité. Cette coopération doit se faire en général en respectant des contraintes qui ne sont pas simplement organisationnelles. Chaque partenaire de la coopération tient à conserver ses prérogatives et en particulier l'exécution des parties de procédés qui le concerne. Ces contraintes peuvent être également d'ordre juridique ou réglementaire.

Il ne s'agit pas non plus de coordonner à l'aide d'un procédé global différents sous procédés. Un des cas qui nous intéresse concerne la gestion des mandats d'arrêt internationaux. Un procédé a été décidé au niveau européen qui permet à l'Espagne de demander à la France d'arrêter et d'extrader un ressortissant anglais sur son territoire. Un tel cas de figure demande une collaboration entre les juridications espagnole, française et anglaise. Au cours de cette collaboration, différentes contraintes doivent être satisfaites. Chaque juridiction doit avoir le contrôle sur la partie du procédé qu'elle exécute. Elle doit pouvoir partager certaines données de façon parfaitement contrôlée. Elle doit également pouvoir suivre l'exécution de la procédure dans son ensemble. Nous nous trouvons là face à la gageure de pouvoir concilier les différentes contradictions liés à des questions de confiance, de confidentialité, de sécurité et de traceabilité de l'exécution, tout en respectant un procédé agréé à l'avance. Ce problème est proche de celui de la décentralisation du contrôle dans l'orchestration de services web. La différence principale est que la décomposition du procédé doit pouvoir se faire de façon explicite en fonction des organisations et pas simplement de l'implantation des services.

Permettre d'assurer la distribution de cette exécution, tout en fournissant des garanties de sécurité, de confidentialité, et en permettant aux différents acteurs de suivre et de tracer l'exécution du procédé est un challenge important qui peut avoir également des applications dans d'autres domaines (la santé pour les relations entre les services de soins et l'administration des soins, la gestion de crise pour les services préfectoraux, militaires et civils, et même le eLearning entre les enseignants et les étudiants). Pour atteindre ce but, nous pensons pouvoir poursuivre nos travaux sur la séparation des préoccupations dans la définition et l'exécution des procédés en essayant d'y intégrer le problème de la sécurité et de la distribution contrôlée de l'exécution. Celle-ci pourra s'appuyer sur des travaux existants [16, 1, 2]. Concernant la sécurité, nous avons déjà commencé à travailler sur les questions de la séparation des devoirs et le problème de la délégation et de la révocation des droits dans un environnement distribué et hétérogène dans le cadre d'une thèse en collaboration avec SAP Research.

Les questions qui se posent forment ainsi un problème de recherche à plusieurs dimensions dont les réponses se trouvent dans la continuation des travaux que nous avons menés et dans la nécessité de collaborer avec d'autres sur des problématiques spécifiques.

Comment distribuer l'exécution d'un procédé ? Comment faire migrer des parties de procédés d'une organisation à une autre ? Comment partager, transmettre et sécuriser les données entres plusieurs organisations ? Comment aligner la gestion de la sécurité de différentes organisations ? Et surtout, comment faire tout cela de façon indépendante de la définition du procédé.

Toutes ces questions sont importantes pour la mise en place de solutions de procédés interorganisationnels dans les domaines auxquels nous pensons nous intéresser. Elles font l'objet d'études distinctes en particulier concernant la sécurité et la distribution mais il n'existe pas de travaux prenant en compte la coordination entre organisations, les problèmes de sécurité et de confiance entre ces organisations et la distribution de l'exécution du procédé sur différents sites. Ces questions sont bien sure liées à la problématique plus général que nous considérons qui est celle de la gestion de procédés de longues ou très longue durée.

8.2 Les procédés de longues durées

La plupart des questions que nous posions concernant les modèles de procédés dans un contexte coopératif se posent toujours. Aucun modèle n'a prouvé sa supériorité alors qu'il s'agit de prendre en compte des procédés complexes, évolutifs, contrôlés par ses utilisateurs et de durée potentiellement très longue. Des avancées notables ont été fait dans le domaine de systèmes orientés services avec les services Web et les langages d'orchestration comme BPEL ou les langages de choréographie comme WS-CDL [9]. Un des problèmes de ces langages est qu'ils sont très statiques. Un procédé BPEL est défini puis instancié. A l'instantiation il est lié à différents partenaires qui vont lui fournir les services dont il a besoin. Le problème c'est qu'une fois les services liés au procédé en cours d'exécution, il est pratiquement impossible d'en changer. De nombreux travaux s'intéressent en au problème de la composition dynamique de services, c'est à dire la possibilité de composer de façon sure des services au cours de l'exécution du procédé. En revanche, la question de la flexibilité et de la gestion des cas exceptionnels, même si elle est reconnue comme un dimension importante, plus importante sans doute que celle des transactions, pour l'adoption des systèmes de gestion de workflow ne semble pas résolue. Les travaux sur l'évolution des modèles ont atteint une certaine maturité théorique [12]. Il n'empêche qu'un workshop récent sur le sujet concluait que le problème réside toujours dans la séparation entre les modèles et les instances [15].

Les mécanismes qui gèrent l'orchestration de services par des procédés de type BPEL font au départ l'hypothèse d'une liaison statique des services appelés au moment de son instantiation. De nombreux travaux faisant appel à toutes les techniques classiques possibles proposent des solutions pour permettre sous certaines conditions la découverte et la composition dynamique de services à l'exécution. Les problèmes qui se posent dans le domaine de la composition de services sont d'abord la découverte de services répondant au besoin du procédé en cours d'exécution puis de la vérification de l'adéquation du protocole du service avec la chorégraphie mise en oeuvre et de la compatibilité entre les politiques mises en oeuvre par le service et les politiques acceptées par le procédé. On ne peut pas dire aujourd'hui qu'une solution ait émergée qui soit parfaitement opérationnelle. En outre, ces approches font l'hypothèse de procédés relativement stables au cours de leur exécution.

Dans le cas de procédés de très longue durée, éventuellement coopératifs, le problème de l'adaptation aux changements de l'environnement se pose de façon plus importante. L'évolution des instances de procédés (ou évolution ad-hoc) causée par des évènements spécifiques générés durant l'exécution du procédé (problèmes de délais, diminution des ressources, nouvelles ressources), par des changements dans les partenariat, ont un impact sur des parties du procédés. Les moteurs BPEL assurant l'exécution des processus nécessite des évolutions incluant :

- la possibilité de modifier le processus et de tracer l'histoire des changements. La gestion du changement est un aspect important dans la gestion de procédés de très longue durée. Nous pouvons nous inspirer pour cela des travaux de Dadam [12] sur les différentes possibilités d'évolution des processus métiers dans les systèmes de gestion de Workflow mais ces travaux ne prennent pas en compte les problèmes de distribution de l'exécution pour la migration des procédés, ni les aspects non fonctionnels de l'évolution (transactions, sécurité).
- le changement de partenaire lors de l'exécution d'un service est aussi une question qu'il faut traiter. Les travaux actuels proposent des solutions partielles pour la sélection dynamique de service [7, 13] mais ne traitent pas le problème du changement de partenaire après cette sélection en cas de défaillance du partenaire. Dans ce cas, il faut prévoir des techniques de retour en arrière partielle (partial rollback) pour refaire la sélection d'un

nouveau partenaire et reprendre la conversation entièrement ou partiellement avec ce nouveau partenaire. Dans ce cas, les politiques, le contrat et éventuellement le procédé doivent être changé pour prendre en compte les contraintes de ce nouveau partenariat.
– les modifications des propriétés d'exécution non fonctionnelles que nous avons évoqués plus haut font aussi partie du problème. Les pistes se situent dans ce domaine du coté de la programmation orientée aspects appliquée à des procédés BPEL [8, 11].

Notons que ce que nous proposons pour les procédés de très longues durées dont la cas d'utilisation que nous considérons se trouvent dans le domaine du BTP et du eGouvernement peut certainement s'appliquer également dans le domaine de la gestion de crises ou de la santé. En effet, dans ces deux derniers domaines, les procédés peuvent être très long, ou très courts mais avec un nombre d'événements importants, rapprochés et nécessitant des changements qui ne peuvent pas être tous prévus. Ceci nous amène à la dernière question qui nous intéresse, celle des utilisateurs dans un monde de services.

8.3 L'utilisateur au coeur des procédés

Les travaux des psychologues dans la lignée de Vygotsky [14] nous apprennent à considérer les processus humains sous l'angle de l'activité comme outil de transformation de soi et de l'environnement. Il est difficile, sans entrer dans les détails, lorsqu'on travaille sur la modélisation de la coordination d'activités coopératives de ne pas être interpellé par le cadre formel fournit par cette méthode d'analyse pour comprendre comment fonctionne les interactions entre les membres d'un même groupe pour atteindre un objectif donné. En outre, même si cette interprétation est un peu simpliste dans notre contexte, le rôle donné à l'histoire et à la culture dans l'approche et dans l'analyse fondée sur cette théorie peut être particulièrement intéressant pour analyser, comprendre et plus tard modéliser les comportements collectifs. En effet comme nous avons pu le deviner et parfois le constater de façon relativement informelle lors des expériences concernant Coopéra et de façon plus classique dans les processus d'apprentissage, les activités permettant aux enfants de publier leur résultats se transformaient rapidement en actions qu'ils arrivaient à faire de façon presque mécanique. C'est un mécanisme bien connu en apprentissage que celui de la conversion d'une activité vue au départ comme une suite d'actions bien décomposée qui se transforme elle même progressivement en une action.

Cette question mérite d'être étudié de façon plus précise pour les procédés coopératifs. Comment l'histoire et la culture commune d'un groupe permet de simplifier un processus coopératif dans le temps pour en optimiser la définition et s'assurer que les utilisateurs continuent à l'utiliser ? Pour cela, nous avons une source principale d'inspiration dans les travaux de Jakob Bardram sur l'Activity Computing [5]. Dans ses travaux les plus récents, Bardram propose de considérer une activité dans son ensemble et d'aider l'utilisateur (le personnel soignant un malade en l'occurrence) d'interrompre et de reprendre une activité dans différents contextes de façon instantanée. Dans l'approche qu'il a entrepris de mettre en oeuvre, la notion de coordination est peu ou pas prise en compte contrairement à ses travaux plus anciens [4, 3]. En mettant au centre des préoccupations l'utilisateur, ses activités et les procédés auxquels il participe et qu'il est capable de composer, il est possible de progresser à la fois dans la compréhension de la dynamique de la coordination d'un groupe de projet et dans celle de la composition de procédés personnels. Les travaux de John Carroll [6] sont également intéressants dans cette perspective parce qu'ils proposent de considérer le contexte et différents éléments de conscience de groupe pour permettre aux utilisateurs de se coordonner. Nous proposons en nous inspirant des systèmes d'orchestration et de workflow de considérer les personnes comme des services connectés à un

Personal Service Bus. Dans son travail quotidien, un individu est confronté à un ensemble de flux électroniques entrant (des évènements) sous forme de messages instantanés, des flux RSS, d'email, d'appels téléphoniques, de SMS. Il participe à des procédés interorganisationnels informels. Il génère également son propre flux de messages sortant dans un réseau social informel et hétérogène.

Prenons l'exemple d'un utilisateur participant à la réponse à un appel d'offre avec d'autres partenaires. La production du document de réponse est le résultat d'un procédé informel et souvent mal contrôlé. L'exécution de ce procédé peut également démarrer différents procédés internes (la signature d'un engagement par exemple). Dans un tel procédé, la simple coordination prend souvent un temps important. Le suivi de cette coordination est également difficile à faire, entrainant des retards, des travaux non pris en compte ou fait en double.

Nous proposons à moyen terme de travailler sur la composition de procédés traditionnels et des procédés personnels et sur un bus d'évènement agrégeant et dirigeant les évènements vers les tâches et les actions d'un individu (voir d'un groupe d'individus). Ce travail va commencer prochainement avec l'étude des procédés et des échanges mis en oeuvre dans des cas de simulation de crise et sur les outils permettant de contrôler et de suivre l'évolution des crises et des procédés de traitement de ces crises dans le cadre du projet iCrisis

8.4 Conclusion

Le domaine des services et de leur coordination est un domaine en pleine expansion que ce soit en recherche ou dans l'industrie. Certains commencent même à parler de "service science" à la frontière entre les sciences de l'information, de la gestion et de l'ingénierie. La question de la coordination des activités coopératives est tout aussi féconde même si depuis quelques années les travaux se concentrent plutôt sur ses aspects synchrones. Cependant, l'utilisation de nouveaux cadres formel d'études basés sur la théorie de l'activité permettent de pallier aux limites des approches purement cogniticiennes dans ce domaine [10]. Nous nous situons comme nous l'avons montré à la frontière entre ces deux domaines. Nous considérons ainsi les aspects "techniques" de la modélisation et de l'exécution des processus coopératifs en proposant des modèles et en les validant par des implantations. De ce point de vue, nous tentons de proposer une approche assurant une modélisation et une exécution flexible des procédés tout en étant capable de fournir des garanties qui nous parraissent indispensables concernant les aspects non fonctionnels comme les transactions ou plus récemment la sécurité. Nous considérons également les aspects humains de la coordination en essayant de comprendre comment l'individu ou le groupe peut profiter des supports à la coordination proposés pour passer plus de temps à travailler réellement sur le problème à résoudre et moins sur les aspects liés aux problèmes générés par la distribution dans l'espace et le temps des équipes et par l'hétérogénéité des cultures et des organisations. Pour finir, nous essayons de nous appuyer pour cela sur des exemples réels et des domaines où ces questions sont importantes comme le eGouvernement, la eSanté ou la gestion de crises qui fournissent des cas complexes et particulièrement intéressant d'interactions entre les organisations et les individus.

Bibliographie

[1] *Decentralized Orchestration of CompositeWeb Services*, 2006.

[2] *Towards a Reliable Distributed Web Service Execution Engine*, 2006.

[3] J. Bardram. Designing for the dynamics of cooperative work activities. In *CSCW'98*, 1998.

[4] Jakob Bardram. Plans as situated action : An activity theory approach to workflow systems. In *ECSCW*, pages 17–, 1997.

[5] Jakob Bardram, Jonathan Bunde-Pedersen, and Mads Soegaard. Support for activity-based computing in a personal computing operating system. In *CHI '06 : Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 211–220, New York, NY, USA, 2006. ACM Press.

[6] John M. Carroll, Mary B. Rosson, Gregorio Convertino, and Craig H. Ganoe. Awareness and teamwork in computer-supported collaborations. *Interacting with Computers*, 18(1) :21–46, January 2006.

[7] Fabio Casati and Ming-Chien Shan. Dynamic and adaptive composition of e-services. *Information Systems*, 26(3) :143–163, May 2001.

[8] Carine Courbis and Anthony Finkelstein. Weaving aspects into web service orchestrations. In *ICWS '05 : Proceedings of the IEEE International Conference on Web Services (ICWS'05)*, pages 219–226, Washington, DC, USA, 2005. IEEE Computer Society.

[9] Thomas Erl. *Service-Oriented Architecture : Concepts, Technology, and Design*. Prentice Hall PTR, August 2005.

[10] Victor Kaptelinin, Bonnie Nardi, Susanne B&#248 ;dker, John Carroll, Jim Hollan, Edwin Hutchins, and Terry Winograd. Post-cognitivist hci : second-wave theories. In *CHI '03 : CHI '03 extended abstracts on Human factors in computing systems*, pages 692–693. ACM Press, 2003.

[11] Woralak Kongdenfha, Régis Saint-Paul, Boualem Benatallah, and Fabio Casati. An aspect-oriented framework for service adaptation. pages 15–26. 2006.

[12] Stefanie Rinderle, Manfred Reichert, and Peter Dadam. Correctness criteria for dynamic changes in workflow systems–a survey. *Data & Knowledge Engineering*, 50(1) :9–34, July 2004.

[13] Paolo Traverso and Marco Pistore. Automated composition of semantic web services into executable processes. pages 380–394. 2004.

[14] L. S. Vygotsky. *Mind in Society : The Development of Higher Psychological Processes*. Harvard University Press, November 1980.

[15] Barbara Weber, Manfred Reichert, and Jan Mendling. 1stieee workshop on flexibility in process-aware information systems summary, June 2006.

[16] Ustun Yildiz and Claude Godart. Towards decentralized service orchestrations. In *SAC '07 : Proceedings of the 2007 ACM symposium on Applied computing*, pages 1662–1666, New York, NY, USA, 2007. ACM Press.

Bibliographie

[1] *Decentralized Orchestration of CompositeWeb Services*, 2006.

[2] *Towards a Reliable Distributed Web Service Execution Engine*, 2006.

[3] W. M. P. Van Der Aalst, A. H. M. Ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1) :5–51, July 2003.

[4] Karim Baina, François Charoy, Claude Godart, S. El Hadri, Hala Skaf, S. Akifuji, T. Sakaguchi, Y. Seki, and M. Yoshioka. Corvette : a cooperative workflow for virtual teams coordination. *International Journal Networking and Virtual Organisations*, 2(3) :232–245, Dec 2004.

[5] Karim Baïna, François Charoy, Claude Godart, Daniela Grigory, Saad el Hadri, Hala Skaf, Shunsuke Akifuji, Toshiaki Sakaguchi, Yoko Seki, and Masaichiro Yoshioka. Corvette : A cooperative workflow development experiment. In *3rd IFIP Working Conference on Infrastructures for Virtual Enterprises - PRO-VE'2001, Sesimbra, Portugal* , May 2002.

[6] J. Bardram. Designing for the dynamics of cooperative work activities. In *CSCW'98*, 1998.

[7] Jakob Bardram. Plans as situated action : An activity theory approach to workflow systems. In *ECSCW*, pages 17–, 1997.

[8] Jakob Bardram, Jonathan Bunde-Pedersen, and Mads Soegaard. Support for activity-based computing in a personal computing operating system. In *CHI '06 : Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 211–220, New York, NY, USA, 2006. ACM Press.

[9] G. Canals, P. Molli, and C. Godart. Concurrency control for cooperating software processes. In *Proceedings of the 1996 Workshop on Advanced Transaction Models and Architecture (ATMA'96)*, Goa, India, 1996.

[10] Gérôme Canals, Claude Godart, François Charoy, Pascal Molli, and Hala Skaf. Coo approach to support cooperation in software developments. *IEE Proceedings - Software*, 145(2-3) :79–84, Jun 1998.

[11] John M. Carroll, Mary B. Rosson, Gregorio Convertino, and Craig H. Ganoe. Awareness and teamwork in computer-supported collaborations. *Interacting with Computers*, 18(1) :21–46, January 2006.

[12] Fabio Casati and Ming-Chien Shan. Dynamic and adaptive composition of e-services. *Information Systems*, 26(3) :143–163, May 2001.

[13] Stefano Ceri, Paul W. P. J. Grefen, and Gabriel Sanchez. WIDE : A distributed architecture for workflow management. In *RIDE*, pages 0–, 1997.

[14] François Charoy, Claude Godart, Nicolas Gregori, Jean-Charles Hautecouverture, and Sébastien Jourdain. Coopera : An environment for teaching and learning internet cooperation. In *IADIS International Conference e-Society 2004, Avila, Espagne*, pages 323–330, Jul 2004.

[15] François Charoy, Claude Godart, Pascal Molli, Gerald Oster, Marc Patten, and Miguel Valdes. Services for virtual teams hosting : Toxicfarm introduction. In *Second International Workshop on Cooperative Internet Computing - CIC 2002, Hong Kong, China*, pages 105–112, Aug 2002.

[16] François Charoy, Adnene Guabtni, and Miguel Valdes Faura. A dynamic workflow management system for coordination of cooperative activities. In *Workshop on Dynamic Process Management - BPM 2006 International Workshops, BPD, BPI, ENEI, GPWW, DPM, semantics4ws, 04/09/2006*, volume 4103 of *in : LNCS, Business Process Management Workshops*, pages 205–216, Vienne/Autriche, 2006. Springer. Non.

[17] Nihan K. Cicekli and Yakup Yildirim. Formalizing workflows using the event calculus. pages 222+. 2000.

[18] Carine Courbis and Anthony Finkelstein. Weaving aspects into web service orchestrations. In *ICWS '05 : Proceedings of the IEEE International Conference on Web Services (ICWS'05)*, pages 219–226, Washington, DC, USA, 2005. IEEE Computer Society.

[19] Andy Crabtree, Michael Twidale, Jon O'Brien, and David M. Nichols. Talking in the library : Implications for the design of digital libraries. In *ACM DL*, pages 221–228, 1997.

[20] Wijnand Derks, Juliane Dehnert, Paul Grefen, and Willem Jonker. Customized atomicity specification for transactional workflow. In *Proceedings of the Third International Symposium on Cooperative Database Systems for Advanced Applications (CODAS'01)*, pages 140–147. IEEE Computer Society, 2001.

[21] Paul Dourish. The parting of the ways : Divergence, data management and collaborative work. In *ECSCW*, pages 213–, 1995.

[22] Paul Dourish and Victoria Bellotti. Awareness and coordination in shared workspaces. In *CSCW*, pages 107–114, 1992.

[23] Paul Dourish, Jim Holmes, Allan Maclean, Pernille Marqvardsen, and Alex Zbyslaw. Freeflow : mediating between representation and action in workflow systems. In *CSCW '96 : Proceedings of the 1996 ACM conference on Computer supported cooperative work*, pages 190–198, New York, NY, USA, 1996. ACM Press.

[24] M. Dumas and A. ter Hofstede. Uml activity diagrams as a workflow specification language. *In Proc. of the International Conference on the Unified Modeling Language (UML). Toronto, Canada, October 2001. Springer Verlag.*, 2001.

[25] Clarence A. Ellis and Carlos Maltzahn. The chautauqua workflow system. In *HICSS (4)*, pages 427–, 1997.

[26] Clarence A. Ellis and Gary J. Nutt. Office information systems and computer science. *ACM Comput. Surv.*, 12(1) :27–60, March 1980.

[27] Thomas Erl. *Service-Oriented Architecture : Concepts, Technology, and Design*. Prentice Hall PTR, August 2005.

[28] Khaled Gaaloul, François Charoy, and Claude Godart. Cooperative processes for scientific workflows. In *6th International Conference on Computational Science - ICCS 2006,*

28/05/2006, volume 3 of *in : Lecture Notes in Computer Science, International Conference on Computational Science*, pages 976–979, Reading/UK, 2006. Springer. Non.

[29] H. Garfinkel. *Studies in Ethnomethodology.* Prentice Hall, 1967.

[30] D. Georgakopoulos and L. Maciaszek. *Proceedings of 9th International Workshop on Research Issues in Data Engineering (Virtual Enterprise).* IEEE Press, 1999.

[31] M. Gertz. *Diagnosis and Repair of Constraint Violations in Database Systems.* PhD thesis, Institut für Informatik Universität Hannover, 1996.

[32] Natalie S. Glance, Daniele S. Pagani, and Remo Pareschi. Generalized process structure grammars gpsg for flexible representations of work. In *CSCW '96 : Proceedings of the 1996 ACM conference on Computer supported cooperative work*, pages 180–189, New York, NY, USA, 1996. ACM Press.

[33] C. Godart. COO : a Transaction Model to support COOperating software developers COOrdination. In *4th European Software Engineering Conference (ESEC4), Garmisch, LNCS 717*, 1993.

[34] Claude Godart, Christophe Bouthier, Philippe Canalda, François Charoy, Pascal Molli, Olivier Perrin, Helene Saliou, Jean-Claude Bignon, Gilles Halin, and Olivier Malcurat. Asynchronous coordination of virtual teams in creative applications (co-design or co-engineering) : requirements and design criteria. In *Information Technologies for Virtual Enterprises*, Jan 2001.

[35] Claude Godart, Gérôme Canals, François Charoy, Pascal Molli, and Hala Skaf. Designing and implementing coo : Design process, architectural style, lessons learned. In *Proceedings International Conference on Software Engineering (ICSE18).* IEEE Press, 1996.

[36] Claude Godart, Gérôme Canals, Francois Charoy, Pascal Molli, and Hala Skaf. Designing and implementing coo : Design process, architectural style, lessons learned. In *ICSE 18 (International Conference On Software Engineering), Berlin*, pages 342–352. IEEE Publishing Computer Society Press, mar 1996.

[37] Claude Godart, François Charoy, Marc Patten, Nicolas Grégori, Jean-Charles Hautecouverture, and Isabelle Faugeras. Coopéra : apprendre à coopérer et apprendre en coopérant. In *Conférence Internationale sur l'Enseignement Ouvert et en Ligne - ICOOL 2003, Ile Maurice*, Dec 2003.

[38] Claude Godart, François Charoy, Olivier Perrin, and Hala Skaf. Cooperative workflows to coordinate asynchronous cooperative applications in a simple way. In IEE Press, editor, *Parallel and Distributed Systems - PADS 2000*, pages 409–416. IEEE Computer Society, Jul 2000.

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

[40] P.W. Grefen and P.M. Apers. Integrity Control in Relational Database Systems – An Overview. *Data and Knowledge Engineering, 10(2)*, pages 187–223, 1993.

[41] Nicolas Grégori, Jean-Charles Hautecouverture, François Charoy, and Claude Godart. Combining ergonomics, culture and scenario for the design of a platform for cooperation. *Artificial Intelligence and Society*, 20(3) :384–402, 2006.

[42] Nicolas Grégori, Jean-Charles Hautecouverture, Claude Godart, and François Charoy. Ergonomie, culture, scénario : trois facteurs pour la conception d ?une plate-forme de coopération. In *Colloque ARCo'2004, Compiègne, France*, Dec 2004.

[43] Daniela Grigori, François Charoy, and Claude Godart. Anticipation to enhance flexibility of workflow execution. In H.C. Mayr, J. Lazansky, G. Quirchmayr, and P. Vogel, editors, *International Conference on Database and Expert Systems Applications - DEXA'2001, Munich, Germany*, volume 2113 of *Lecture Notes in Computer Science*, pages 264–273. Springer-Verlag, Sep 2001.

[44] Daniela Grigori, François Charoy, and Claude Godart. Flexible cooperative workflow management. In *13th International Conference on Control Systems and Computer Science - CSCS'2001, Bucarest, Roumanie*, Jun 2001.

[45] Daniela Grigori, François Charoy, and Claude Godart. Flexible data management and execution to support cooperative workflow : the coo approach. In *Proceedings of the Third International Symposium on Cooperative Database Systems for Advanced Applications - CODAS'2001, Beijing, China*, pages 139–146. IEEE, Apr 2001.

[46] Daniela Grigori, François Charoy, and Claude Godart. Coo-flow : a process model to support cooperative processes. In *Fifteenth International Conference on Software Engineering and Knowledge Engineering 2003 - SEKE'2003, San Francisco*, Jul 2003.

[47] Daniela Grigori, François Charoy, and Claude Godart. Coo-flow : a process technology to support cooperative processes. *International Journal of Software Engineering and Knowledge Engineering - IJSEKE Journal*, 14(1) :61–78, Feb 2004. World Scientific Publishing.

[48] Daniela Grigori, François Charoy, and Claude Godart. Enhancing the flexibility of workflow execution by activity anticipation. *International Journal of Business Process Integration and Management*, 1(3) :143–155, 2006.

[49] Daniela Grigori, Hala Skaf-Molli, and François Charoy. Adding flexibility in a cooperative workflow execution engine. In *8th International Conference on High Performance Computing and Networking Europe - HPCN Europe 2000, Amsterdam, Hollande*, May 2000.

[50] Adnene Guabtni. *Behavioural Spheres for Flexible Business Process Modelling and Execution*. PhD thesis, Université Henri Poincaré Nancy 1, 2007.

[51] Adnene Guabtni and François Charoy. Multiple instantiation in a dynamic workflow environment. In Anne Persson and Janis Stirna, editors, *Advanced Information Systems Engineering, 16th International Conference, CAiSE 2004, Riga, Lavtia*, volume 3084 of *Lectures Notes in Computer Science*, pages 175–188. Springer, Jun 2004.

[52] Adnene Guabtni, François Charoy, and Claude Godart. Customizable isolation in transactional workflow. In *First International Conference on Interoperability of Enterprise Software and Applications - INTEROP-ESA'2005 First International Conference on Interoperability of Enterprise Software and Applications - INTEROP-ESA'2005*, Geneva/Switzerland, 02 2005. University of Geneva, Switzerland.

[53] Adnene Guabtni, François Charoy, and Claude Godart. Spheres of isolation : Adaptation of isolation levels to transactional workflow. In *Business Process Management*, pages 458–463, 2005.

[54] Adnene Guabtni, François Charoy, and Claude Godart. Concurrency management in transactional web services coordination. In *17th International Conference on Database and Expert Systems Applications - DEXA 2006, 04/09/2006*, volume 4080/2006 of *in : Lecture Notes in Computer Science, Database and Expert Systems Applications*, pages 592–601, Krakow, Poland, 2006. Gabriela Wagner, FAW, University of Linz, Austria, Springer Berlin / Heidelberg. Non.

[55] Adnene Guabtni, François Charoy, and Claude Godart. Using isolation spheres for cooperative processes correctness. In IEEE Beijing, editor, *The 10th International Conference on CSCW in Design (CSCWD 2006), 03/05/2006*, volume 1 of *Computer Supported Cooperative Work in Design 2006*, Nanjing, China, 2006. Southeast University. Non.

[56] Jean-Charles Hautecouverture, Nicolas Grégori, François Charoy, Claude Godart, Marc Patten, and Isabelle Faugeras. Coopera : Analyse de l'usage d'une plate-forme de coopération à destination d'enfants du primaire. In *Human Centered Processes - HCP'2003, Luxembourg*, May 2003.

[57] Jean-Charles Hautecouverture, Nicolas Grégori, François Charoy, Claude Godart, Marc Patten, and Isabelle Faugeras. Analyse d'usage d'une plate-forme de coopération : usage et développement logiciel. *In Cognito - Cahiers Romans de Sciences Cognitives*, 1(3) :45–77, 2004.

[58] James D. Herbsleb, Audris Mockus, and Roy T. Fielding. Two case studies of open source software development : Apache and mozilla. *ACM Trans. Softw. Eng. Methodol.*, 11(3) :309–346, July 2002.

[59] James D. Herbsleb, Audris Mockus, Thomas A. Finholt, and Rebecca E. Grinter. An empirical study of global software development : Distance and speed. In *23rd International Conference on Software Engineering (ICSE'01)*, 2001.

[60] Thomas Herrmann and Marcel Hoffmann. The metamorphoses of workflow projects in their early stages. *Computer Supported Cooperative Work (CSCW)*, 14(5) :399 – 432, October 2005.

[61] Edwin Hutchins. *Cognition in the Wild (Bradford Books)*. The MIT Press, September 1996.

[62] Charles T. Davies Jr. Data processing spheres of control. *IBM Systems Journal 17(2) : 179-198*, 1978.

[63] Victor Kaptelinin, Bonnie Nardi, Susanne B&#248 ;dker, John Carroll, Jim Hollan, Edwin Hutchins, and Terry Winograd. Post-cognitivist hci : second-wave theories. In *CHI '03 : CHI '03 extended abstracts on Human factors in computing systems*, pages 692–693. ACM Press, 2003.

[64] Woralak Kongdenfha, Régis Saint-Paul, Boualem Benatallah, and Fabio Casati. An aspect-oriented framework for service adaptation. pages 15–26. 2006.

[65] Stephen C. Levinson. *Pragmatics (Cambridge Textbooks in Linguistics)*. Cambridge University Press, June 1983.

[66] Franck Leymann. *Production Workflow*. Prentice Hall, 1999.

[67] Frank Leymann. Supporting business transactions via partial backward recovery in workflow management systems. In *BTW*, pages 51–70, 1995.

[68] Frank Leymann and Dieter Roller. *Production Workflow*, chapter Chapter 7 : Workflows and Transactions. Ed. Prentice Hall, Inc., Upper Saddle River, New Jersey, second edition edition, 2000.

[69] U. W. Lipeck and H. Zhou. Monitoring Dynamic Integrity Constraints on Finite State sequences and Existence Intervals. In G.Saake J.Goers, A.Heuer, editor, *Proceedings of the 3rd International Workshop on Foundation of Models and Languages for Data and Objects*, pages 115–130, 1991.

[70] Udo W. Lipeck. Stepwise specification of dynamic database behaviour. In *SIGMOD '86 : Proceedings of the 1986 ACM SIGMOD international conference on Management of data*, pages 387–397, New York, NY, USA, 1986. ACM Press.

[71] Christophe Loridan and Miguel Valdes. Bonita :a java 2 platform, enterprise edition (j2ee) open source cooperative workflow system. In *Java One, 2004*, San Francisco, USA, 2004. none. Non, This presentation describes the Bonita Workflow Management System developed by the ECOO team of LORIA. Colloque avec actes et comité de lecture. internationale.

[72] H. Martin. *Contrôle de la cohérence dans les bases objets : Une approche par le comportement*. PhD thesis, Joseph Fourier-Grenoble 1, 1991.

[73] Raul Medina-Mora, Terry Winograd, Rodrigo Flores, and Fernando Flores. The action workflow approach to workflow management technology. In *CSCW '92 : Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, pages 281–288, New York, NY, USA, 1992. ACM Press.

[74] Audris Mockus, Roy T. Fielding, and James D. Herbsleb. Two case studies of open source software development : Apache and mozilla. *ACM Trans. Softw. Eng. Methodol.*, 11(3) :309–346, July 2002.

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

[76] Pascal Molli, Manuel Munier, Gérôme Canals, François Charoy, and Claude Godart. Cooserializability : A correctness criterion for cooperative executions. Rapport de recherche, 1997.

[77] Andrew F. Monk, Bonnie A. Nardi, Nigel Gilbert, Marilyn M. Mantei, and John McCarthy. Mixing oil and water ? : Ethnography versus experimental psychology in the study of computer-mediated communication. In *INTERCHI*, pages 3–6, 1993.

[78] Wolfgang Prinz and Sabine Kolvenbach. Support for workflows in a ministerial environment. In *CSCW '96 : Proceedings of the 1996 ACM conference on Computer supported cooperative work*, pages 199–208, New York, NY, USA, 1996. ACM Press.

[79] K. Ramamritham and P.K. Chrysanthis. In Search of Acceptability Criteria : Database Consistency Requirements and Transaction Correctness Properties. In Özsu, Dayal, and Valduriez, editors, *Distributed Object Management*. Morgan Kauffman, 1993.

[80] K. Ramamritham and P.K. Chrysanthis. A taxonomy of correctness criteria in database applications. *The VLDB Journal*, 5(5) :85–97, 1996.

[81] M. Reichert and P. Dadam. Adept flex : Supporting Dynamic Changes of Workflows Without Losing Control. *Journal of Intelligent Information Systems*, 10(2) :93–129, 1998.

[82] Stefanie Rinderle, Manfred Reichert, and Peter Dadam. Correctness criteria for dynamic changes in workflow systems–a survey. *Data & Knowledge Engineering*, 50(1) :9–34, July 2004.

[83] Shazia Wasim Sadiq, Maria E. Orlowska, and Wasim Sadiq. Specification and validation of process constraints for flexible workflows. *Inf. Syst.*, 30(5) :349–378, 2005.

[84] K. Schmidt. *The critical role of workplace studies in CSCW*. Cambridge University Press, 1999.

[85] Heiko Schuldt, Gustavo Alonso, Catriel Beeri, and Hans-J&#246 ;rg Schek. Atomicity and isolation for transactional processes. *ACM Trans. Database Syst.*, 27(1) :63–116, 2002.

[86] John R. Searle and Daniel Vanderveken. *Foundations of Illocutionary Logic*. Cambridge University Press, July 1985.

[87] Hala Skaf. *Une approche hybride pour gérer la cohérence dans les environnements de développement coopératif*. Thèse en informatique, Université de Nancy I – Centre de Recherche en Informatique de Nancy, 1997.

[88] Hala Skaf, F. Charoy, and Claude Godart. Une approche hybride pour contrôler les activités de développement de logiciel. Rapport interne, Centre de Recherche en Informatique de Nancy, Vandoeuvre-lès-Nancy, 1996.

[89] Hala Skaf, François Charoy, and Claude Godart. An hybrid approach to maintain consistency of cooperative software development activities. In *SEKE97 The Ninth International Conference on Software Engineering and Knowledge Engineering, Madrid*, jun 1997.

[90] Hala Skaf, François Charoy, and Claude Godart. Flexible integrity control of cooperative applications. In IEEE Computer Society, editor, *The Ninth International Workshop on Database and Expert Systems Applications - DEXA 98, Vienne, Autriche*, 1998.

[91] Hala Skaf, François Charoy, and Claude Godart. Maintaining shared workspaces consistency during software development. *International Journal of Software Engineering and Knowledge Engineering*, 9(5) :623–642, Oct 1999.

[92] Hala Skaf, Francois Charoy, and Claude Godart. Maintaining consistency of cooperative software development activities. In *Integrity in Database 6th International Workshop on Foundations of Models and Languages for Data and Objects, Dagstuhl*, sep 1996.

[93] Howard Smith and Peter Fingar. Workflow is just a pi process, November 2003.

[94] Lucy A. Suchman. *Plans and Situated Actions : The Problem of Human-Machine Communication (Learning in Doing : Social, Cognitive & Computational Perspectives)*. Cambridge University Press, November 1987.

[95] Paolo Traverso and Marco Pistore. Automated composition of semantic web services into executable processes. pages 380–394. 2004.

[96] David Ungar, Craig Chambers, Bay-Wei Chang, and Urs Holzle. Organizing programs without classes. *Lisp and Symbolic Computation*, 4(3), 1991.

[97] Willem-Jan van den Heuvel and Sergei Artyshchev. Developing a three-dimensional transaction model for supporting atomicity spheres. *International Workshop on Web Services Research, Standardization, and Deployment*, 2002.

[98] W. M. P. van der Aalst, K. M. van Hee, and Houben. Modelling workflow management systems with high-level petri nets. In *Proceedings of the second Workshop on Computer-Supported Cooperative Work, Petri nets and related formalisms*, pages 31–50, 1994.

[99] W.M.P van der Aalst. The application of petri nets for workflow management. *The Journal of Circuits, Systems and Computers*, 8(1) :21–66, 1998.

[100] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Advanced workflow patterns. In O. Etzion en P. Scheuermann, editor, *7th International Conference on Cooperative Information Systems (CoopIS 2000)*, volume 1901 of *Lecture Notes in Computer Science*, pages 18–29. Springer-Verlag, Berlin, 2000.

[101] Gottfried Vossen and Mathias Weske. The wasa2 object-oriented workflow management system. In Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh, editors, *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*, pages 587–589. ACM Press, 1999.

[102] L. S. Vygotsky. *Mind in Society : The Development of Higher Psychological Processes*. Harvard University Press, November 1980.

[103] Barbara Weber, Manfred Reichert, and Jan Mendling. 1stieee workshop on flexibility in process-aware information systems summary, June 2006.

[104] Mathias Weske. Flexible modeling and execution of workflow activities. In *31st Hawaii International Conference on System Sciences, Software Technology Track (Vol VII)*, 1996.

[105] WFMC. The workflow reference model. Technical report, The Workflow Management Coalition, 1995.

[106] Ustun Yildiz and Claude Godart. Towards decentralized service orchestrations. In *SAC '07 : Proceedings of the 2007 ACM symposium on Applied computing*, pages 1662–1666, New York, NY, USA, 2007. ACM Press.

Résumé

Le résumé.

Abstract

in english

Designing and Implementing *COO*: Design Process, Architectural Style, Lessons Learned

C. Godart*, G. Canals**, F. Charoy**, P.Molli* and H. Skaf*
CRIN-CNRS, BP 239,
F-54506 Vandoeuvre les Nancy cedex, France
e-mail: {godart,canals,charoy,molli,skaf}@loria.fr
* Université Henri Poincaré,ESSTIN ** Université de Nancy II

Abstract

This paper reports on the design and implementation of a Software Development Framework named COO. Its design process is firstly detailed and justified. Then, the paper emphasizes its layered and subject-oriented architecture. Particularly, it is shown how this architectural style leads to a very flexible and powerful way of defining, integrating and combining services in a Software Development Environment.

1 Introduction

The *COO* project has two orthogonal but interactive dimensions. One is about cooperation support in the software process, the other is about Software Development Environments (SDE) architectures.

In fact our work is originally related to cooperation support in Software Development Environments, or in how to allow inter-dependent sub-processes to progress in parallel both safely and correctly. It is when designing and implementing the *COO* framework with this objective in mind that a programming style progressively emerged and became a second dimension of the project.

This paper reports on this experience with the objective: (a) to depict a posteriori our design process, (b) to characterize our software architectural style and (c) to extract reusable design guidelines.

Section 2 provides a succinct description of our view of cooperation and cooperation support. Section 3 describes our design process and shows how we progressively derived the *COO* conceptual architecture. Section 4 describes how we use subject-oriented programming to implement *COO* and also introduces the *L*ayer-*S*ubject oriented Programming software architectural style. The paper concludes in section 5 with a discussion of lessons learned. It also offers practical guidance on design issues and indication of future direction.

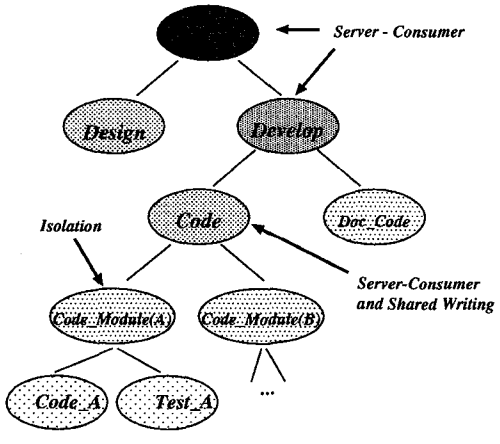


Figure 1: Heterogeneous cooperation policy

2 Cooperation in Software Development Environments

2.1 Cooperation support requirements

Cooperation paradigms We stressed three main paradigms in order to characterize cooperation and these are: *server-consumer*, *shared writing* and *server-mediator-consumer*.

The *server-consumer* paradigm corresponds to the case in which a process, hereinafter called the *consumer*, reads a result of another process, the *server*. There is cooperation when both the server and the consumer executes simultaneously.

The *shared writing* paradigm corresponds to the case in which two processes modify the same object at the same time. Shared writing can be seen as an extension of the *server-consumer* paradigm in which the *consumer* also modifies the object produced by the server.

The *server-mediator-consumer* paradigm corresponds to the case in which a third process, the *mediator*, is requested to connect the *consumer* with the

0270-5257/96 $5.00 © 1996 IEEE
Proceedings of ICSE-18

server, which can be unknown to the *consumer* and/or which can be inactive. This is typical of a *fix a bug* scenario, in which the process which discovers the bug requests a *change management* process to search for a process to fix this bug [16].

In addition, these paradigms can be combined. Two processes which are implied in a *server-consumer* relationship from the point of view of an object can also simultaneously write another object.

Heterogeneous cooperation policies Inside a project, each group must be allowed to define its proper cooperation policy. The project cooperation policy must integrate these local policies. This is illustrated in figure 1. The process *Project* breaks down into several subprocesses and different cooperation policies are used by the different sub-processes. *project* synchronizes *Design* and *Develop* following the *Server-consumer* paradigm. This is also the case for *Develop* which synchronizes *Doc_code* and *Code*. At the lower level, *Code* allows *shared writing* to developers who work on common shared objects (e.g a common *Makefile*). At the opposite, *Code_Module(A)* forbids any interactions between *Code_A* and *Test_A*: *Code_A* and *Test_A* execute in a serial way.

2.2 Cooperation support approaches

Four main approaches to cooperation support are currently considered in the state of the art. Firstly, a cooperation policy can be based solely on the responsibility of human agents, like in some configuration management systems (e.g CVS [6]). Secondly, it can be based on some *predefined synchronization strategies*, defined before the process starts. These strategies can be hard-coded like in traditional database transaction managers, or programmed like in Marvel [4] and Merlin [19]. Thirdly, it can also be *based on the knowledge* included in the software process model without any distinction of a specific knowledge related to cooperation support. This makes the hypothesis that all the interaction cases are forecast. Adele [5] and Spade [3] appears in this category. Finally, it can be a mixture of these approaches, and that is probably the more realistic approach.

2.3 The *COO* approach

The *COO*[1] project is a contribution to the continuing development of **Process Centered IPSE**[2]s, i.e. environments which have not a predefined behavior, but which can be parametrized by a software process model. *COO* is focused on cooperation support between software developers. That is the first dimension of the project. The second dimension is the study of SDE design and implementation and it is the main focus of this paper.

A *COO* process breaks down into sub-processes. At the leaves of the tree are atomic processes, called activities, which modify objects and execute atomically. Other processes, called tasks, solely synchronize their sub-processes but delegate object modifications to their enclosed activities.

Each process executes in a workspace. Thus we obtain a tree of workspaces which is isomorph to the process hierarchy. To operate an object, a process must transfer it to its proper workspace and it is the sole to be authorized to work in its workspace. Different processes can operate the same object, or different parts of the same object, in different workspaces. This is especially the case when they cooperate.

Cooperation policies In this context, a cooperation policy is distributed over the tree of processes. Each process node has its proper (sub-) cooperation policy which synchronizes its sub-processes. A local cooperation policy is mainly a set of rules which restricts the transfers from a process workspace to its sub-process workspaces, and mutually, from its children workspaces to its proper workspace.

3 Design Process

We explore now some design issues.

3.1 General Requirements

We started the design with three main objectives in mind:

1. not to develop *the* system, but a kernel from which different SDE can be generated,
2. to support at least the requirements of section 2.1,
3. to provide for *safe* cooperation, i.e. correctness of execution is a key issue. It is to support this requirement that we decided to based cooperation control on a safe predefined synchronization strategy in the continuation of database transaction models.

[1] *COO* stands for cooperation and coordination in the software process

[2] IPSE: Integrated Programming Support Environment

3.2 Design approach

We based our design on the two following principles:

1. There is a continuation between traditional concurrency control protocols and *predefined synchronization strategies* required in SDE,
2. There is a continuation between *predefined synchronization strategies* and *knowledge based synchronization*: predefined strategies make some hypothesis on the semantic of software engineering applications.

In other words, the principles of the processing of a traditional transaction can be generalized to these of cooperative processes. In addition, a SDE should provide support for both predefined synchronization strategies and knowledge based synchronization.

3.3 Deducing a conceptual architecture

As a consequence of design principle 1, we observed the processing of a traditional transaction with the objective to, in a first time, extract basic service layers, and in a second time, to generalize them to support synchronization strategies as needed in software processes (section 3.3.1). From these layers, we deduced the conceptual architecture introduced in section 3.3.2. Different specific architectures can be instantiated from this conceptual architecture. Our *COO* system is one of them (section 2-B). See [18], chapter 4 and chapter 5 for a more highly developed analysis of some european process-centered environments in the frame of this conceptual architecture.

3.3.1 Processing of a traditional transaction

Before a transaction is authorized to modify the object base, it goes through several layers which act as filters. We can distinguish between five layers:

1. *Control of the transaction call with regards to the interface definition.* This includes at least syntactical controls, report on abort or success ...

   This layer is not really affected by the characteristics of advanced transactions related to software processes. We call it *Interface* in our conceptual architecture.

2. *Control of the transaction call with regards to the process definition.* A transaction cannot execute if it is not in accordance with the process schedule.

   In traditional applications, e.g. banking, transactions are temporally constrained. For example, it is not accepted that one can withdraw some money from an account which does not have a positive balance.

   This is yet true for software processes. Any advanced protocol makes some hypothesis about the order in which (sub)-processes are issued. The difference is that control need not be done a priori, but can be done during or at the end of the processes. Typically, in order to assert consistency between code and design, a traditional transaction protocol would prevent a coder to begin before the designer has finished. At the opposite, a strategy to synchronize a coder and a designer in a real software process must allow the coder to start its job with an incomplete design but must prevent him to terminate its job without having taken into account all the modifications of the design.

   We call this layer *Knowledge* in our conceptual architecture.

3. *Control of the transaction call with regards to concurrency between transactions.*

   Several concurrency control protocols exist. Most of them assert the serializability of executions and support ACID transaction[3]. This step makes a hypothesis about the previous one: each transaction is an individual and correct piece of code, i.e. code which executes correctly if isolated.

   Even if things are more complicated in software development applications, especially due to their interactive and long term nature, different advanced concurrency control mechanisms have been developed. As with traditional classical protocols, their synchronization rules are established before the transaction starts its execution. We call this layer *Predefined strategies* in our conceptual architecture.

4. *Modification of copies*

   In most cases, a traditional transaction does not directly modify the objects of the object base. It operates on transient copies of objects, loaded in its own private space. Modifications are done locally in this private space.

[3] ACID means Atomicity, Consistency, Isolation and Durability. The general idea is that a process which executes as an ACID transaction has no interaction with other processes and that its effects on the object can be seen by other processes only when it terminates. The most popular protocol for ACID transactions is the two phases locking protocol (2PL).

This is largely more sensitive in our applications because of two things: first, the long duration of our processes imposes to manage persistent cooperative versions [11] of objects; second the number of points of view requests different representation of the same object in different workspaces. We call this layer *Workspace* in our conceptual architecture.

5. *Effective modification of the object base.*

   Finally, the modifications of copies are reflected into the object base; the transaction effect becomes effective either for its enclosing transaction in the case of a nested transaction [17], or in the object base in the case of a flat transaction or a root transaction.

   Software processes are generally nested and open, and a transaction can make a result visible, sometimes before it terminates, to any of its sibling transactions.

   Finally, our architecture rests on a basic layer called *Repository*, the objective of which is especially to store software artifacts in their different versions.

3.3.2 A layered conceptual architecture

This section gives some details of the layers we just pointed out.

Interface As classically, the interface must support control. It can also directly support cooperation, as an example by displaying notifications in a convivial way (see flags in Merlin [19]).

Knowledge This layer exploits the knowledge in the software process model in order to support the correctness of the enacted processes, either in combination with the *Predefined Strategies* layer if it exists or not. It can also be enriched to provide cooperation guidance to the users. In the event that it does not exits a *Predefined Strategies* layer [2], this layer is directly connected to the lower layers.

Predefined Strategies This layer is responsible for the description and the enforcement of predefined synchronization strategies. In addition, it must allow for the combination of different strategies and support recovery in case of failure. The *cooperative transaction model* of *COO*, the *Semantic Concurrency Control* of Marvel and the *Programmable Concurrency Control Mechanism* of Merlin belong to this category.

Workspaces A workspace is the set of objects and tools used by a user (or a group of users) to perform a (sub)-task. It manages the view (type and representation) that the user has on the object base. It allows a user to operate on objects in isolation when requested. At the same time and in combination with a synchronization strategy, it allows a user to coordinate its work with other interacting users. We consider that it is also a foundation to support cooperation. This idea comes especially from the *Software Configuration Community* and we find it, among others, in *Adèle* [5], *COO* [13] and *Epos* [10].

Repository The bottom layer is the *repository* layer which mainly provides for object and schema management services. The repository can be distributed. It must allow view definition and should support view integration. It must also provide an ACID transaction model (there is always a level of abstraction where transactions execute in a serializable way). It can provide a triggering mechanism.

3.4 The *COO* architecture

This section describes the *COO* architecture which instantiates the above conceptual architecture. It simply assumes that a *COO* process breaks down into sub-processes with atomic processes, called activities, at the leaves. We prefer to describe it with a bottom-up approach, starting with the repository layer, for a better comprehension.

Repository The *COO* repository, called P-RooT [8, 9] is an object oriented database. It implements an object orientation of the PCTE interfaces: its data model is based on an ERA data model extended with encapsulation, inheritance, dynamic binding and composite objects. Data types are grouped in schemas. The view of a process is defined by the union of a set of schemas it has defined as its working schema (we come back on the union of schema in section 4.2). It provides also an ACID transaction protocol which directly supports the isolation of atomic processes and a triggering mechanism based on a simple event-action model.

Workspaces There is one workspace per process. A workspace, except the root workspace, has a parent and can have several children. The workspace layer implements a base/sub-base architecture. It is mainly based on a cooperative version manager. To operate on an object, a process must transfer this object from one of its chain of ancestor workspace into its proper workspace (a *check_out* activity). This activity creates a new persistent version of the object. Thus several copies of the same logical object can exist in different workspaces. Versioning is transparent to processes and a process issues a request to a logical object. Mutually, one process can transfer an object

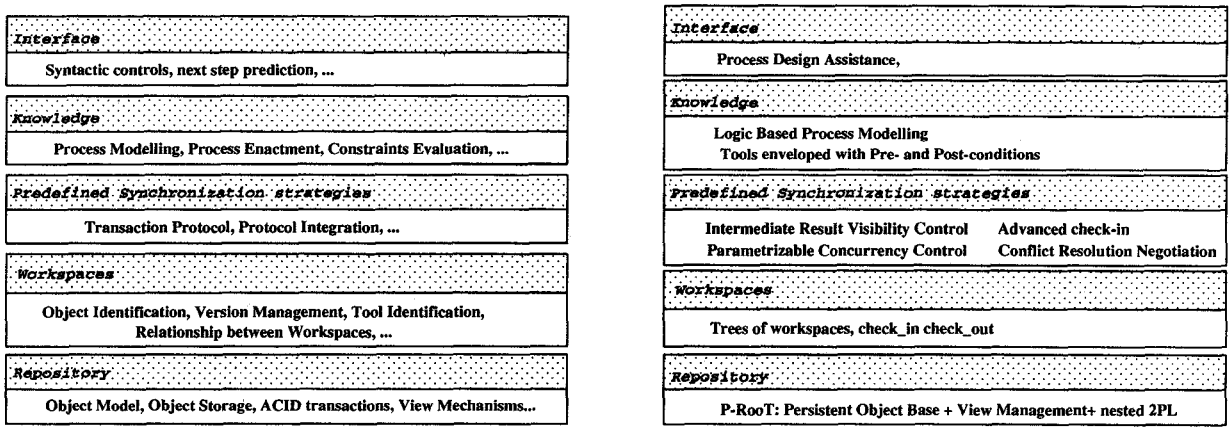


Figure 2: A. General Service Layers–B. COO Service Layers

from its workspace to its parent workspace (a *check_in* activity). A last transfer activity is defined to *update* the value of an object in a workspace with the corresponding value in the parent workspace. This activity creates a new version of the object by merging the two values of the object. *check_out*, *check_in* and *update* are atomic activities.

Predefined strategies *COO* implements a cooperative transaction model based on the correctness criteria described in [1]. It distinguishes between three levels of consistency: globally stable objects, locally stable objects (also called *intermediate values* or *results*) and non stable objects [13]. A process can make an intermediate result visible by explicitly *upward_commiting* it in its parent workspace. To maintain consistency with regards to intermediate results, we can impose that when a process A has read an intermediate value of a process B, it cannot *terminate* without *updating* this value with the corresponding final value. When the dependencies between interacting processes create a cycle, the deadlock is solved by a kind of two phase commit protocol. This directly supports the *server-consumer* and *share-writing* paradigms. *check_in* is now reserved for transfer of final results when a process terminates. Nevertheless, *advanced ckeck_ins* of a final result, i.e check_in before commit, are allowed to support the *server-mediator-consumer* paradigm. The behavior of the *COO* protocol is close to the nested 2PL protocol [17] relaxed with the ability to upward commit intermediate results. This rests either on the responsibility of human agents or can be programmed depending on the processes in conflict. A *COO* protocol can be heterogeneous in that a sub-process can run its proper protocol. This is achieved by, on the one hand, the ability to define a "lock" model[4] and, on the other hand the ability to attach a locking model to a workspace (a locking model overrides transfer operations with locking operations). We will study this aspect more thoroughly in section 4.3.

Knowledge The *predefined strategies* layer asserts consistency on the basis that processes are consistent with regards to their definition and to a mode of execution. In other words, it makes the hypothesis that, independently of parallelism of execution, processes are correct, i.e. they do not start or terminate their execution when it is not appropriate. In [14], we demonstrate how (atomic) processes can be decorated with pre- and post-conditions to assert this *pre-parallelism* correctness.

Interface We have not deeply investigated this aspect in *COO*. Nevertheless, our internal process organization can support process notifications when requested. In addition, we are developing an environment to support process model design.

4 Implementing *COO*

This section describes the technology used to implement *COO*. It is mainly based on *Subject-Oriented Programming* [15] which connected with a layered design approach produces an extremely effective software architectural style.

4.1 The *LS* software architectural style

Layering (*L*) is a natural way to get an abstract view of a complex system, but it is not precise enough

[4] when a process want to operate an object, it must set a lock on this object. Depending on the current lock on this object, it can either be authorized to operate this object or be asked to perform another operation, including wait and abort in the more simple cases

to express its architectural style. Layers can be connected in different ways which lead to different styles. We use *Subject-Oriented Programming* (*S*) to glue our layers together. The idea is that the concepts at a particular layer must not be understood as an implementation of the concepts of the upper layer, but that the different layers implement different points of view of the same concepts. In fact, in the *LS* style, layers are superimposed to produce the global view of the system. An analogy can be done with superimposition of color in photography (blue plus yellow mix to give green) in opposition to immiscible oil and water layers (the layer of oil rest on the layer of water with a very thin interface). Typically, in *COO*, a software artifact is the superimposition of the *repository* view, the *workspace* view, the *predefined strategies* view, the *knowledge* view and the *interface* view.

4.2 Subject-Oriented Programming in *P-RooT*

P-RooT is the repository layer of the *COO* system. Its central part is an object oriented database[5], the characteristic of which is to provide active support for view integration by means of a schema integration mechanism. More precisely, the view a tool (or a service) has on the object base is described by means of an Entity-Relationship like schema extended with a procedural attachment. To work with , processes must integrate the schemas of all the tools and services they wish to use. We call such an aggregate a *Working Context*. *Working contexts* form the basis for building environments in *COO*, the behavior of which depends on the aggregated schemas. Thus, the way this integration of schemas into working contexts is done is a key point in understanding how *COO* environments are built.

Schemas and working contexts Schemas describe objects structure and behavior. A particular and very important point is that the *same type* can have *multiple* and *different definitions* in different schemas. Thus, the observed structure and behavior of a given object may vary depending on the schemas used by the process which observes it. This is illustrated in figure 3, where the same object type, say `C-source` is defined twice: in `schema 1` with the `compile` and `white-edit` operations attached to it, and in `schema 2` with the `grey-edit` and `beautify` operations. Thus, depending on the schema in use, a `C-source` object can be `white-edited` and `compiled` or `grey-edited` and `beautified`.

A *working context* is now defined as an ordered list of schemas. Properties (structure and behavior) of an object type in a working context are built by doing the union of its properties in the different schemas. this union is computed by a dynamic binding procedure. It is important to note that thanks to the notion of a working context, the behavior of an object depends on its type but also on the context in which it is used. In addition, dynamic binding allows this context to dynamically evolves.

Names binding and Horizontal overriding As said above, the union of properties of an object type is dynamically computed. This is quite easy to do except in the case where properties in different schemas have the same name. Naming conflicts are solved on the basis of the order in which schemas appear in the working context: for a given name, the selected definition is the first found while traversing the schema list. This way of binding names to definitions, when applied to methods, leads to the notion of *horizontal overriding*.

A method can be overridden in two ways: firstly, inside one schema, a type can specialize another following the classical OO way and a method applied to this type can override an operation applied to one of its supertypes; secondly, different methods with the same name can be applied to the same object type in different schemas of the same working context. In such a case, the binding is done by by traversing schemas first (horizontally), then secondly by traversing the inheritance hierarchy (vertically). Thus by default, it is the first more specialized method which exists in the ordered list of schemas which is selected (see figure 3 (B)). We use *horizontal overriding* in opposition to classical *vertical overriding*.

Normal binding By default, the binding of an operation to a method is done by applying the horizontal overriding rule. As an example, in the working context of figure 3 (B), in any call *mysource.edit* () issued from any schema, where *mysource* is an instance of `source`, *edit* is bound to `edit 2` of `schema 1`.

When developers wish to escape from this rule, they can either use the classical *super* operator of OO languages, the *next* operator, or an *explicit binding*.

The *next* operator In the implementation of a method, one can explicitly ask the system to skip the current schema and to search for the method in the tail

[5] P-RooT is implemented on top of the Emeraude implementation of the PCTE interfaces [7], and provides an OO interface of these interfaces.

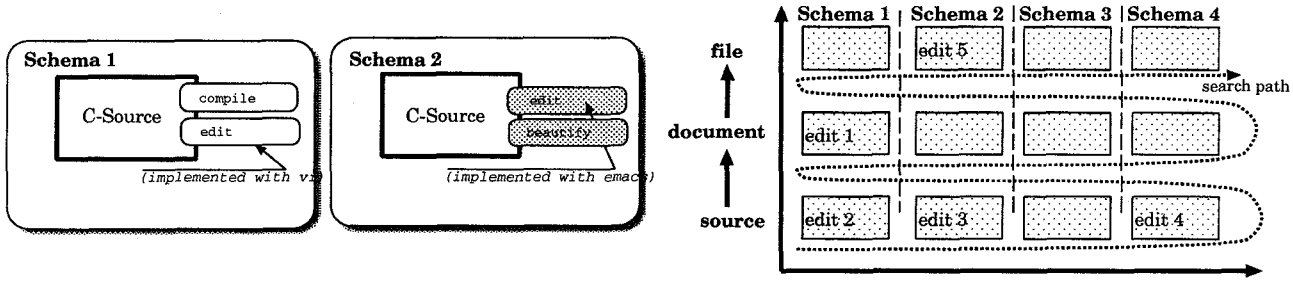


Figure 3: A. Schemas and Types –B.Working context and Name Binding

of the schema list. In the working context of figure 3 (B), in any call *mysource.edit next* issued from `schema 1` , *edit* is bound to `edit 3`, while the same call issued from `schema 2` is bound to *edit4*. This operator has been introduced to allow a service (i.e. a method) to be defined by reusing and extending an existing one implemented in another schema. *Next* can be related to *super* in OO languages.

Explicit binding The user may explicitly provide the name of the schema and/or the ancestor type in which the method has to be searched for when an operation is called. This mechanism can be useful but its use reduces the level of flexibility of the environment and must remain exceptional.

Horizontal overriding, the *next* operator and working context dynamic composition allow object types to be dynamically refined. The next section shows how this can be used to define and integrate services in a very flexible and powerful way.

4.3 Using subjectivity to build and tailor environments

4.3.1 Building a *COO* environment

A particular *COO* environment is built by combining together different services belonging to the different layers pointed out in section 3.4. These different services are provided by the framework in the form of schemas. Thus, a *COO* environment is defined as a working context in which schemas are installed in the layers' order in the corresponding conceptual architecture: the most external first and the most internal last. Thus, in reference to architecture in figure 2-(B) the *Interface layer* schema is the first and the *Repository layer* schema is the last. This does not mean that the idea of layer corresponds exactly to this of schema. Indeed, a layer can be implemented through several schemas. As an example, in the current *COO* implementation, the "*predefined strategy*" layer can correspond to three schemas, as illustrated in figure 4A.

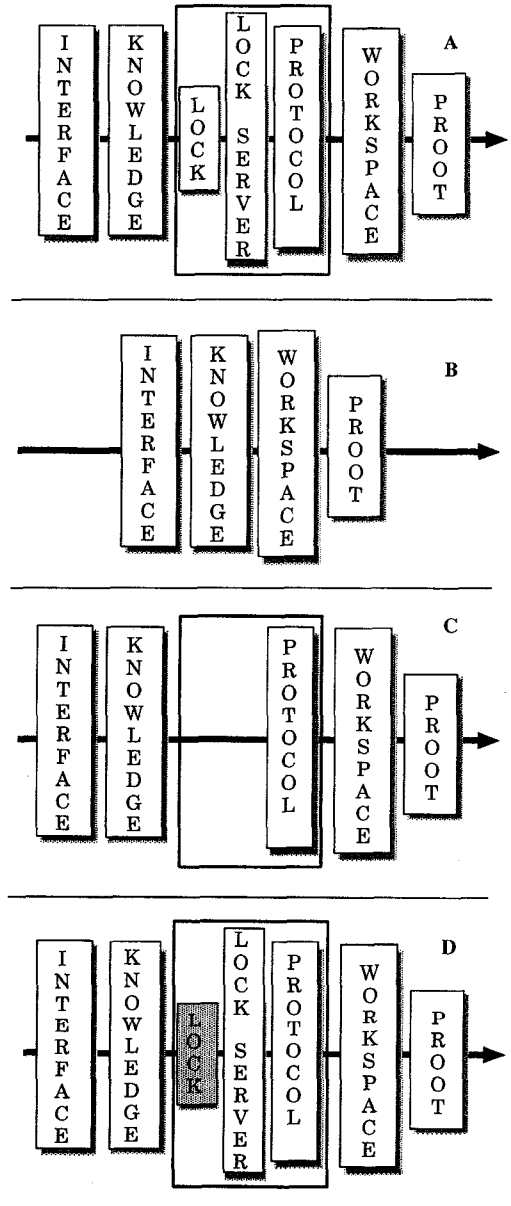


Figure 4: The COO services

To illustrate more precisely the way *COO* environments are built, let us detail some services provided by different schemas.

Basic transfer operations between workspaces, i.e **check_in**, **check_out** and **update** are implemented by the **WORKSPACE** schema. **LOCK SERVER** permits the definition of locking models and provides operations to set and unset locks, say **set_lock** and **unset_lock**.

Several other schemas add semantics to these basics services, mainly by setting additional constraints in order to maintain consistency with regards to the current cooperation policy. Thus, all these schemas *reuse* the basics services provided by **WORKSPACE** and **LOCK SERVER**, but override them to add their own semantic. Concretely, each schema redefines the transfer operations but reuse the existing service by issuing a call with the *next* operator.

In this way, **PROTOCOL** provides for intermediate result management; it introduces the *upward_commit* method and overrides *check_in* and *check_out* to manage consistency in presence of intermediate results. **LOCK** extends transfer operations with locking capabilities, by overrinding them so that they call the **set_lock** or **unset_lock** basic services. Finally, the **KNOWLEDGE** schema also override transfer operations to check additional conditions and constraints corresponding to the process model.

From these schemas, different environments with different behavior can be built. In figure 4B, the environment is built with only four schemas. A transfer operation (e.g. check_out) issued in this context is bound to the one defined in the **KNOWLEDGE** schema. This one checks different conditions and makes a call with the *next* operator. This call is then bound to the operation defined in the **WORKSPACE** schema, which will do the effective transfer. This environment thus does not use any predefined strategy. This is not the case of this depicted in figure 4C, where the **PROTOCOL** schema has been added. In this case, the call issued from **KNOWLEDGE** is bound to the operation defined for **PROTOCOL** which again issues a call bound to the one in **WORKSPACE**. In 4C, intermediate results will be managed. Finally, figure 4d illustrate a case in which a locking strategy has been added to the environment.

4.3.2 Inserting a schema–*extension, evolution, tailoring*

A schema can be inserted either in the head, the middle or in the tail of the list. In general, insertion in head is used to extend the environment with new functionalities. A method in the new schema can override another in an internal schema, indeed the last schema itself. In other terms, some operation calls previously bound to a method of an existing schema can be bound now to a method in the new schema.

Insertion in the middle or in the tail is generally used to add or make an internal service evolve. A process will be able to use this new service without modification. Any operation call, previously bound to a method which is now overridden by a method in the inserted schema, is now bound to this overriding method.

For example, the installation of a LOCK schema is optional. If a LOCK schema is missing, cooperation is controlled only by PROTOCOL (which allows intermediate result visibility and shared writing without restriction). Nevertheless, a LOCK schema can be dynamically added (see figure 4, c and d). It will apply in the current development to all the following concurrent accesses without requesting the KNOWLEDGE schema to be modified.

4.3.3 Removing a schema–*tailoring, evolution*

A schema can be dynamically suppressed without no consequence under two conditions: firstly, that there is no explicit call to a method of this schema, and that the other operation calls, which are currently bound to a method of the schema being removed, can be bound to a more core method previously overridden by a method of the removed schema. Secondly, that the process definition allows it: if subjectivity technically simplify code evolution, (process) semantic considerations can prevent or delay this evolution. As an example, if subjectivity allows to transform a strict policy (isolated work) to a more permissive policy (shared writing), what to do with documents which has been declared as not accessible in shared writing. The first condition is in the scope of this paper, not the second.

Similarly to the previous example, the LOCK schema can potentially be removed (see figure 4, c). If no process is using the LOCK services, this can be done immediately because the LOCK *check_out, check_in, ...* methods override the *check_out, check_in, ...* methods of PROTOCOL and will be replaced by these methods. If a process is running, this is related to dynamic evolution of a process and is somewhat more complicated: a decision must be made about the objects currently constrained by locks.

4.3.4 Replacing a schema–*evolution, tailoring*

Replacing a schema can be seen as the combination of both the suppression and the insertion of a schema.

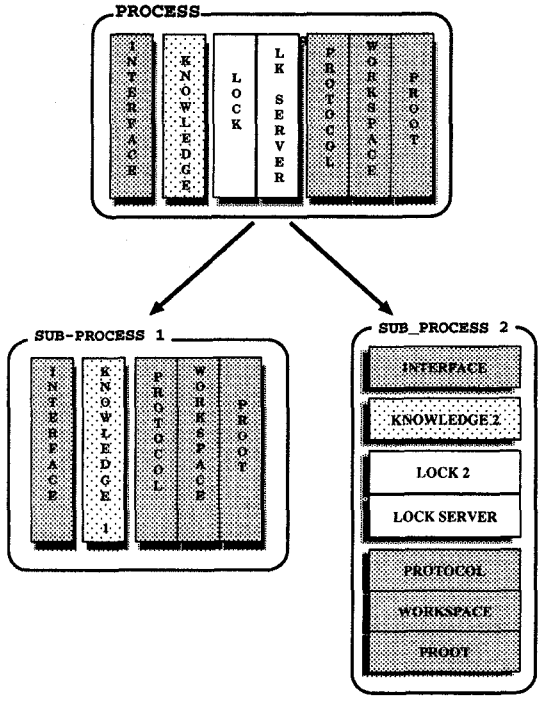


Figure 5: Hierarchical organization and heterogeneousprotocol

As an example, a cooperation policy can be changed simply by replacing a LOCK schema by another (see figure 4, a and d) or a PROTOCOL schema by another. This is easy if the previous policy is extended by the newer one. If not, the problem is the same as when removing a LOCK schema.

4.3.5 Support to hierarchical organization of processes

As described in the previous section, a cooperation policy is built in *COO* by aggregating and ordering several distinct schemas in the same working context. Thus, different sets of schemas define different cooperation policies. This is of the highest importance to support the requirement of integrating different cooperation policies in the same environment, as stated in section 2.1.

Applied to *COO*, each sub-process executes in its proper working context (its proper ordered set of schemas) which defines (especially) its proper cooperation policy. Thus, different cooperation policies can to be attached to the different nodes of a *COO* process tree (as introduced in 2.3). This is illustrated in figure 5 where a root process and two sub-processes are described. Each one has its own set of schemas and, as a consequence, its own cooperation policy. Indeed, each one has its proper "Knowledge", defined in the `KNOWLEDGE`, `KNOWLEDGE 1`, and `KNOWLEDGE 2` schemas. In addition, while `SUB_PROCESS 1` uses a basic cooperation protocol defined by schema `PROTOCOL`, `PROCESS` and `SUB_PROCESS 2` refine this protocol by adding their own locking strategy through the `LOCK SERVER`, `LOCK` and `LOCK 2` schemas. This allows us to install heterogeneous cooperation policies as introduced in figure 1.

With this organization, when an object is transferred from a workspace to another, the policy which governs its evolution changes also and the policy in the source workspace and the target workspace must respect some compatibility rules. Clearly, not all combinations of cooperation policies to build a global heterogeneous policy are consistent. But this goes out of the scope of the paper.

5 Lessons learned and conclusion

5.1 Evaluation of the *LS* style

Advantages

1. *LS* assumes a natural continuity between design and implementation,

2. schemas formalize and document components: objects and relationships types allow a graphical description of components and describe the interface of the service,

3. subjectivity provides a powerful way to connect components. It allows a schema in a Working Context to impact, by method overriding and especially usage of the *next* operator, one of the subsequent ones. This is very powerful and connecting different schemas in a Working Context is a particularly constructive task. As an example, we derived, from an initial browser, different browsers by simply exchanging the order of some schemas in a working context with very few interventions in the code of methods. With regards to *specialization*, subjectiviyu allos to manage classes in a more modular way,

4. schemas definition and insertion in a working context guide architectural design. The behavior of an environment can be (more) simply tested and changed.

Drawbacks The main drawback is that it is not always easy to evaluate the complete impact of a schema

insertion or suppression in a working context: an inserted schema impacts all the operations which override now one of its. A schema suppression can suppress a basic service on which an upper layer schema is based. These dependencies should be automatically managed. Especially, the hypothesis of a schema concerning the existence of a lower layer method should be explicit.

One can also argue that our approach is more suited to the building of new environments from scratch rather then to integration of different heterogeneous components into one environment. Our response taken from the literature is that: it is currently difficult to integrate pre-existing tools and that new integration technologies need to be developed [12]. Our proposition goes in this direction because, on the one hand, the way we model services is based on two standards (PCTE and CORBA), and on the other hand, some technical choices are directly related to integration (as an example, the workspace concept has been introduced with heterogeneity and distribution support in mind, and subjectivity itself is such an integration technology).

Design guidelines This work reports on our experience when developing *P-RooT* and *COO* which is a 130 000 lines of C code system. We can give the following advice:

Separate concerns. This is a well known means to facilitate software reuse and maintenance. We experimented it. For example, several transaction protocols are based on version management. Thus, we developed an independent version management service that we reused for different protocols and which can be used independently of transactions.

Prefer multiple short schemas. We demonstrated empirically that this is a better guarantee to limit the coupling between schemas and to support and favorize reuse. This is related to the previous item.

Concentrate on lower layers. The lower layers are the foundation of the environment and need a particular focus. We experimented that the higher we went in the layer, the shorter and the quicker the development was. This is probably due to the expertise which resides in the lower levels and which can be reused.

Manage dependencies. It is important to document dependencies between schemas. Currently, this is done by hand but it can and should be partially automated. This is related to our first drawback.

5.2 Conclusion

In this paper, we have reported on our experience when developing the *COO* system.

We have demonstrated how the *LS* style allowed us to fulfill our requirements and to implement heterogeneous and hybrid cooperation policies. Although our examples where in the context of cooperation, our discourse about the *LS* style can be generalized. In fact, it is well adapted to system developments for which a layered design is appropriate. We think this it is the case for most SDE developments and it applies for most software process oriented applications.

The *COO* project is continuing, with the objective to, on the one hand, provide a more active support for the safe definition of heterogeneous cooperation protocols and for recovery in case of failure, and, on the other hand, to formalize a methodology for designing and maintaining *LS style* software.

Acknowledgements To members of the *Promoter* Esprit Working Group, especially whose involved in chapter 4 and 5 of the book in preparation. To the colleagues who participate to the implementation of the prototype, and to Ian Robertson and for the English review of the paper.

References

[1] F. Bancilhon, W. Kim, and H. Korth. A Model for CAD Transactions. In *Proceedings of the 11th international conference on VLDB*, pages 25–33, Stockholm, august 1985.

[2] S. Bandinelli, S. Ceri, and M. Felder. A notation for describing advanced transaction models. Technical report, Politechnico di Milano, 1994.

[3] S. Bandinelli, A. Fuggetta, C. Ghezzi, and L. Lavazza. SPADE: An Environment for Software Process Analysis, Design and Enactment. In J. Kramer A. Finkelstein and B. Nuseibeh, editors, *Software Process Modelling and Technology.* Research Study Press, 1994.

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

[5] N. Belkhatir, J. Estublier, and W. Melo. Adèle 2: A Support to Large Software Development Process. In *Proceedings of the First International Conference on the Software Process*, 1991.

[6] B. Berliner. CVS II : Parallelizing software development. In *Proceedings of USENIX*, Washigton D. C., 1990.

[7] I. Campbell. Portable Common Tool Environment. *Computer Standard and Interfaces*, 8, 1988.

[8] F. Charoy. An Object Oriented Layer on PCTE. In Ian Campbell, editor, *PCTE'93 conference*, November 1993.

[9] F. Charoy and P. Molli. Experimenting PCIS Triggers on P-RooT. In *PCTE'94 conference*, pages 389–398, November 1994.

[10] R. Conradi and C. Malm. Cooperating transactions against the EPOS database. In *Software Configuration Management Workshop 3*, Trondheim, 1991.

[11] J. Estublier and R. Casallas. Three Dimensional Versioning. In *Software Configuration Management Workshop 5*, Seatle, 1995. LNCS.

[12] D. Garlan, R. Allen, and J. Ockerbloom. Architectural Mismatch or Why it's hard to build systems from existing parts. In *17th ICSE*, Seatle, USA, April 1995.

[13] C. Godart. COO: a Transaction Model to support COOperating software developers COOrdination. In *4th European Software Engineering Conference, Garmisch, LNCS 717*, 1993.

[14] C. Godart and D. Dietrich. Stepwise specification of interactive processes in *COO*. In *4th European Workshop on Software Process Technology, Noordwijkerhout, The Nederlands, LNCS 913*, April 1995.

[15] W. Harrison and H. Ossher. Subject-Oriented Programming (a critique of pure objects). In *Proceedings of the 8th ACM Conference on Object-Oriented Programming Systems, Langages and Applications*, pages 411–428, Wahington, D.C., September 1993. ACM.

[16] M.I. Kellner and al. Ispw6 software process example. In M. Dowson, editor, *First International Conference On Software Process*, pages 176–186, California, 1991. IEEE Press.

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

[18] WG. Promoter. *Software Process: Principles, Methodology, Technology.* Ed. J.C. Derniame and A. Fuggetta, Wiley, to appear, 1996.

[19] W. Schaefer and A. Wolf. Cooperation Patterns for Process-centred Software Development Environment. In *7th International Conference on Knowledge Engineering and Software Engineering*, Rockville, 1995. Knowledge Systems Inst, pp 454–463.

International Journal of Software Engineering
and Knowledge Engineering
Vol. 14, No. 1 (2004) 61–78

World Scientific
www.worldscientific.com

COO-FLOW: A PROCESS TECHNOLOGY TO SUPPORT COOPERATIVE PROCESSES

DANIELA GRIGORI

Prism Laboratory, 45 avenue des Etats-Unis,
78035 Versailles Cedex, France
daniela.grigori@prism.uvsq.fr

FRANCOIS CHAROY* and CLAUDE GODART†

Université Henri Poincaré & INRIA, Campus Scientifique, BP 239,
54506 Vandoeuvre Cedex, France
**francois.charoy@loria.fr*
†*claude.godart@loria.fr*

In this paper we present a process management technology for the coordination of creative and large scale distributed processes. Our approach is the result of usage analysis in domains like Software Development, Architecture/Engineering/Construction, and e-Learning processes. The basic conclusions of these experiments are the following: (1) cooperative processes are described in the same way as production processes, but these descriptions are interpreted in a different way depending on the nature of the process, (2) the interpretation of process description depends mainly on the required flexibility of control flow and of data flow, and on the relationship between them, (3) the management of intermediate results is a central feature for supporting the cooperation inherent to these processes. COO-flow is a process technology that results from these studies. It is based on two complementing contributions: anticipation that allows succeeding activities to cooperate, and COO-transactions that allows parallel activities to cooperate. This paper introduces COO-flow characteristics, gives a (partial) formalization and briefly discusses its Web implementation.

Keywords: Workflow flexibility; cooperative processes modelling; coordination; cooperation.

1. Introduction

This paper describes a software contribution to the management of creative processes in co-design and/or co-engineering applications. The approach developed can be placed in the workflow initiative, but with particular attention paid to the flexibility requested by the nature of creative interactions. In other words, our objective is to develop a system that is as efficient for design processes as current workflow management systems are for administration and production processes. In addition,

it supports team work distributed on a large scale network, typically the Internet.

There have been many experiments on applying current workflow technology in design domain, but they resulted in a quick dismissal of this approach on working sites. This is due to the characteristics of creative processes that have been pointed out for a long time in the literature (long duration, uncertain development, important variability, interactivity, etc.) and that are badly supported by conventional workflow management systems. Yet, if an efficient support for process description and enactment is a strong request for design and engineering domains, the current usage is still often limited to project management tools. As a consequence, the management of the link between planning offices and working sites is quite weak and insufficient.

The approach developed in this paper is the result of a research including many usage analyses in domains like Architecture/Engineering/Construction (AEC), Software Development Processes, and Cooperative Learning. These studies lead us to the following assumptions:

- *in process books*, *creative and production processes are described in the same way* using boxes and arrows, but the interpretation of such descriptions changes depending on the nature of the application,
- in this interpretation, the *relationship* established *between control and data flow is determinant* and management of intermediate results (drafts, sketches ...) is a key point of this relationship,
- *process description and management* is only a contribution to creative project coordination *that must be combined with other contributing components* (object sharing, awareness, communication, etc.).

This paper describes the COO-flow model that results from these studies. It is based on two complementing contributions: anticipation allowing for succeeding activities to cooperate, and COO-transactions allowing for parallel activities to cooperate and for large scale distribution.

The rest of the paper is organized as follows. The next section motivates and gives overviews of our approach. Section 3 deals with cooperation between succeeding activities and Sec. 4 with cooperation between parallel activities; Sec. 5 discusses integration of both. Section 6 considers some implementation work. Section 7 presents related work and the last section discusses future directions and concludes.

2. Motivations and Overview

2.1. *General considerations*

The work described in this paper has been motivated by several experiments in co-design and co-engineering applications. A first application domain is software process. In this domain, software process modelling and enactment is considered as one of the more important elements for judging the capability of

a company to develop software (see levels of the Capability Maturity Model http://www.sei.cmu.edu/cmm/cmm.html). This request is even more important in the context of cross-organizational software development (we experimented in the AEE project (http://aee.inria.fr/en/index.html).

We also found similar requirements in a project where the objective was to develop services to host virtual teams in AEC domain (for Small and Medium Enterprises). Even if current workflow technology was not efficient, we also understood the need for some process technology to support project management. But we also understood that this process technology would only be a contribution, a component of a larger infrastructure [12] (see also Sec. 6).

These motivations were confirmed by the interest taken by the Hitachi company in some of our preliminary work concerning cooperative transactions and their request to experiment the integration of their WorkCoordinator workflow management system with our transactional technology in order to model and enact cooperative processes. This experiment is related in Sec. 6.

During these experiments, we were convince of the possibility to develop a process technology for creative applications but with the following assumptions:

1. A process model for the coordination of a group of people implied in a creative process does not have to describe all the processes in full details, but only at a gross grain level;
2. In a real application, there is a lot of chance that there will not exist only one such process, but several. In addition these gross grain processes will have to co-habit with fine grain (traditional) processes specialized in well defined tasks;
3. Process management is only one contribution to cooperative process management that must co-habit with other dimensions of coordination like object-sharing and versioning, awareness providing, group decision support, etc.;
4. Creative processes modeling and enactment must be simple, as simple as traditional workflow modeling and enactment. Furthermore, we think that the modeling is the same, but that interpretation changes, depending on the nature of the application concerned.

COO-flow model has been inspired by these considerations.

2.2. *Same description, but different interpretations*

The basic idea comes from the following observation. Concretely, when we look at "paper descriptions" of processes, administrative or production processes on the one hand, and co-design or co-engineering processes on the other, we note that both classes are depicted in the same way using the same basic formalism: typically annotated boxes, arrows and so on.

However, it is clear that when we look at such a description, depending on the context in which it applies, we do not interpret it in the same way. For example, in an administrative context, we interpret two boxes separated by an arrow as a strict

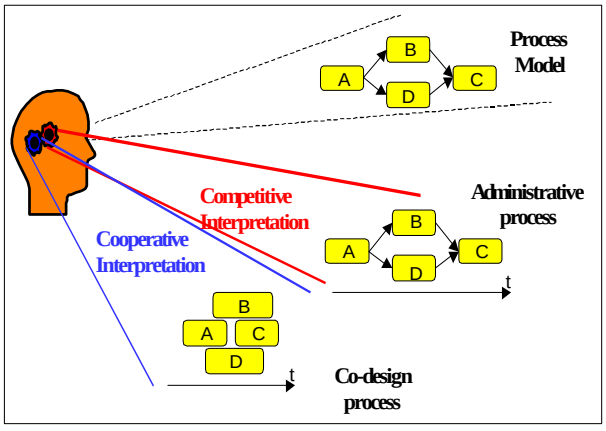


Fig. 1. Same description but different interpretations.

sequence of the two tasks represented by the two boxes and in our mind, the second task cannot start its execution before the first one has terminated. However, when we consider the same schema in the context of a co-design process, our interpretation is much more flexible. We know that this schema is a general indication, but that tasks will probably overlap, iterate. More generally, in the case of an administrative or production process, we see tasks as black boxes executing serially and sharing data only when they start and terminate. In the case of a creative process, we see tasks as white boxes making visible some results when executing, overlapping each other and even iterating their executions.

That is what Fig. 1 sketches. Starting from the same process model, and depending on the context, the mind interprets the same schema in different ways. As an example, making reference to Fig. 1, if we consider a typical administrative process and we take "*select a book*" for A, "*select a payment mode*" for B and "*verify payment capacity*" for C, we understand that these three activities will execute one after the other (see Fig. 1, competitive interpretation). If we consider a cooperative process and we take "*edit a plan*" for A, "*review a plan*" for B and "*validate plan*" for C, our mind infers the possibility for these activities to execute "cooperatively". As an example, "*edit plan*", "*review plan*", and "*validate plan*" activity executions will probably overlap and share successive versions of the plan (see Fig. 1, cooperative interpretation).

2.3. *Difference of interpretation is matter of control flow flexibility and dataflow flexibility*

The question discussed here is: what makes the difference between a competitive interpretation and a cooperative interpretation? The difference is a matter of control flow and dataflow flexibility. A typical competitive process is a process where

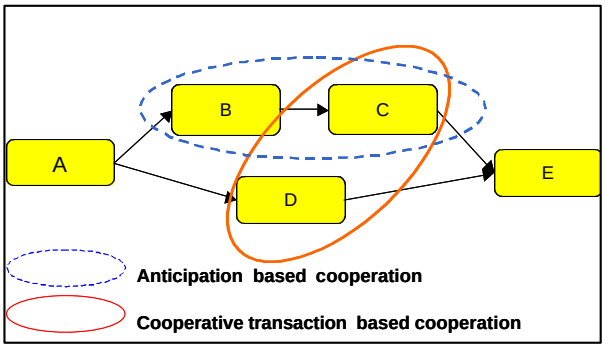


Fig. 2. Dimensions of flexibility.

activities are executed as ACID transactions[a], synchronized by typical workflow operators (operators of the WfMC model there are a good reference: sequence, and-join, and-split, or-join, or-split, etc.). Thus, relaxing the constraints of workflow operators and allowing activities to execute in a non isolated way will provide support for a flexible interpretation of process model.

More flexible is the control flow and the data flow, more cooperative is the interpretation. In COO-flow, flexibility is based on two complementing contributions: "anticipation" and "COO-transaction" (see Fig. 2).

Anticipation is the principle that allows an activity to start its execution even if all its conditions are not "completely" fulfilled. It contributes to control flow and data flow flexibility. This concept is developed in the next section.

COO-transactions allow activities to exchange (intermediate) results in a consistent way by providing data flow flexibility. This concept and its application is developed in Sec. 4.

2.4. *Discussion*

One question that arises at this point is: "as by definition creative activities are not isolated, why do we encapsulate them in transactions?" The response is that in most creative activities, if people interact by exchanging some "semi-consistent" data, nevertheless they want to be assured of a certain degree of security, of consistency, and with a minimum of programming. That is exactly the role of a transactional correctness criteria that defines some general consistency properties, the transactional protocol enforcing them in a transparent way.

In [2] authors demonstrate that the advanced transactions models, needed for modelling the advanced applications they were concerned with, could be modelled with basic workflow control structures and ACID transactions, or in other words

[a]Traditional WFMS does not impose activities to be executed as ACID transactions, but we assume this restriction to support the consistency requested by our applications, in the vein of [2].

that workflow systems are a superset of advanced transaction models.

If this is theoretically true, our feeling is that the creative processes that we are concerned with cannot be modelled in the same way. This is due to the uncertain development (atomic activities are not initially known) and combinatory nature of creative process interactions [13].

To take into account these characteristics of creative processes, we developed the COO-transaction concept [5] that provides consistency guaranties in the presence of intermediate results exchange. A supplementary difficulty was the need for integration of this model with the ability to anticipate the execution of an activity.

3. Anticipation: Cooperation between Succeeding Activities

This section introduces the concept of "anticipation". Intuitively, anticipation means that an activity can start its execution even if all its execution conditions are not fulfilled. Traditionally, an activity can start its execution only if control flow operators allow it (the preceding activity is terminated in case of a sequence, at least one of the preceding activities is terminated in case of an or-join, all the preceding activities are terminated in case of an and-join and so-on) and if data flow based conditions (traditional condition based on data values) allow it. On the contrary, anticipation allows an activity to start its execution even if these conditions are not yet fulfilled (the preceding activity is not terminated in case of a sequence, none of the preceding activities is terminated in case of an or-join, not all its preceding activities are terminated in case of an and-join and so-on) and if data flow based conditions are fulfilled, but only to a minimum. In practice, activities can start if they have input enough intermediate results, typically a configuration of drafts, that allows work to start.

Anticipation impacts the execution model of an activity in the following way. It is necessary:

- to introduce two new states in the activity model: *ready_to_anticipate* and *anticipating states*. Consequently, it is necessary to assign activities in *ready_to_anticipate* state to agents who are able to start their anticipated execution,
- to modify the process execution algorithm: while in traditional workflow, the only event that triggers a new step for electing executable activities is the *termination of an activity*, in the modified model, *activity start* and *data production* events can change the state of succeeding activities to the *ready to anticipate* state,
- to modify data flow to integrate data exchange between anticipated activities (to manage exchange of intermediate results).

3.1. *Adding ready_to_anticipate and anticipating states*

In traditional workflow, activity life cycle is roughly the following (see Fig. 3). When a process instance is created, all activities are in the *initial* state (except

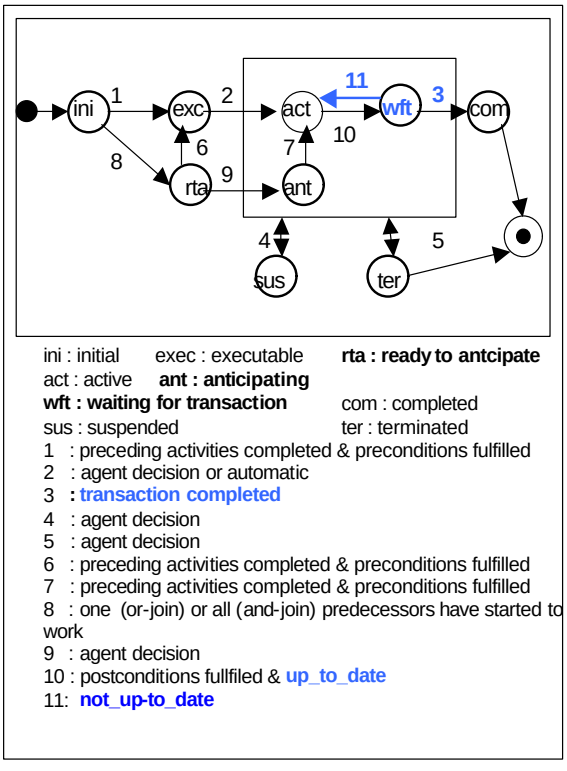


Fig. 3. Activity states.

the start activities that are directly *executable*). An activity enters the *executable* state as soon as the preceding ones have *completed* and its preconditions are fulfilled. As soon as an activity becomes executable, it is scheduled, i.e. it is assigned to all agents who actually qualify under its associated staff query. When chosen by an agent (or automatically), the activity becomes *active*. When its execution terminates normally, the activity reaches the *completed* state (final state). An *active* activity can be *suspended*, i.e. quiescent until returned to the executable state, or *terminated*, i.e. stopped before its normal completion without the possibility to return to the *executable* state.

Anticipation impacts this model as follows: if its anticipation conditions are fulfilled, an activity can enter the *ready to anticipate* state coming from the *initial* state. When entering the *executable* state, the activity is scheduled, i.e. is assigned to all agents who actually qualify under its associated staff query. A *ready to anticipate* activity can become, either executable if all its activation conditions are fulfilled with final results of preceding activities, or *anticipating* when one agent chooses the activity and starts the work (starts to anticipate). An activity in the *anticipating* state enters the *active* state when it is in a situation where it would

have been allowed to start its execution if it was not anticipating, in other words, when previous activities have completed.

3.2. *Conditions for anticipation*

We have analyzed several strategies. The three main ones are sketched below (and formalized in [15]:

1. *Free anticipation* — an activity in an *initial* state may anticipate at any moment. This allows activities to start their work earlier. In other words, control flow dependencies defined in the process model are interpreted at execution time as end-end dependencies; i.e. *an activity can finish its execution only when the preceding ones have.*
2. *Control flow dictated anticipation* — an activity may anticipate when, in the case of an *or-join* like operator, one of its predecessors has started to work, i.e. is in *anticipating* or *executing* state, and, in the case of an *and-join* like operator, all its predecessors have started their execution. With this strategy, the traditional start-end dependency between activities is relaxed, being replaced with a start-start dependency; i.e. *an activity can start its execution as soon as the predecessor has started.*
3. *Control flow and data flow dictated anticipation* — an activity can anticipate when one of its predecessors has started to work (is in *anticipating* or *executing* state) and for all its mandatory inputs, one intermediate result is available. This is like control flow anticipation with additional constraints concerning input existence. In addition, intermediate results can be requested to satisfy a minimal set of constraints.

3.3. *Formalization of anticipation strategies*

In this section we define the conditions for an activity to enter the "ready to anticipate" state for the control flow and data flow dictated strategies presented above. This specification extends the specification introduced in [22] for a traditional worflow. For a complete specification of COO-flow, see [15].

First, we need to introduce the definitions for *Activity state map* and *Data state map* [22], the applications that give the state of an activity (resp. a data element) at any moment during the execution of a process instance.

Definition 1. (***Activity state map***) Let N be the set of all activities of a process model and $\aleph$ the set of natural numbers representing the time axis for each process instance ($0 \in \aleph$ represents the point in time when the associated instance of the process model is created). The activity state map

$$\omega : \aleph \times N \rightarrow S$$

associates at any point in time $i \in \aleph$ each activity A with its actual state ω (i, A).

S includes all activity states relevant to execution: *initial*, *executable*, *active*, *ready to anticipate*, *anticipating*, *terminated*, *completed*, *suspended*.

Definition 2. (***Data state map***) Let V be the set of all process data elements (data needed as input by the activities, data required by conditions and the data to be exchanged between activities). The data state map

$$\delta : \aleph \times V \rightarrow S$$

associates at any point in time $i \in \aleph$ each data element v with its actual state $\delta\ (i, v)$.

The relevant states of a data element are *initial, intermediate and final*. An output data element of an activity enters the *intermediate* state when the activity executes a *Write* operation on this data; it enters the *final* state when activity completes.

Now we can define the conditions for an activity to enter the "ready to anticipate" state for the control flow and data flow dictated strategies presented in the previous section (for the free anticipation strategy, *initial* and *ready to anticipate* state are merged).

Definition 3. (***Activities in State "Ready to Anticipate", control flow strategy***) Activity A becomes *ready to anticipate* at time $i \Leftrightarrow$

1. $\omega(i-1, \mathrm{A}) = \mathit{initial}$
2. $\forall\ X \in \mathrm{A}^{\leftarrow}$, $\omega(i, X) \in$ {executing, anticipating} if A is an AND-join node or a regular node
   $\exists X \in \mathrm{A}^{\leftarrow}$, $\omega(i, X) \in$ {executing, anticipating} if A is an OR-join node

where $\mathrm{A}^{\leftarrow}$ denotes the set of all direct predecessors of activity A.

A regular or an AND-Join activity in initial state enters the *ready to anticipate* state when its predecessors are activated for execution or anticipation (*executing* or *anticipating* state). An OR-Join activity enters the *ready to anticipate* state when one of its predecessors is activated for execution or anticipation (active or anticipating state).

Definition 4. (***Activities in State "Ready to Anticipate", control and data flow strategy without conditions on data***) The control and data flow strategy imposes supplementary conditions concerning the availability of input data as follows. Activity A becomes *ready to anticipate* at time i $\Leftrightarrow$

1. condition 1 of Definition 3
2. condition 2 of Definition 3
3. $\forall\ v \in i(\mathrm{A}) : (w, v) \in \Delta^{\mathrm{in}}(\mathrm{A}) \Rightarrow \delta(i, w) \neq \mathit{initial}$ and $\exists v \in i(\mathrm{A}) : (w, v) \in \Delta^{\mathrm{in}}(\mathrm{A}) \Rightarrow \delta(i-1, w) = \mathit{initial}$

where $i(\mathrm{A})$ is the input container of activity A (the set of all its input data elements), (w, v) is a data connector connecting a data element w of the output container

of an activity ($w \in o(X)$) with a data element v in the input of another activity ($v \in i(\mathrm{A})$), $\Delta^{\mathrm{in}}(\mathrm{A})$ denotes the set of all data connectors (v, w) having as destination an element in the input container of activity A ($v \in i(\mathrm{A})$)

The third condition requires for all input elements ($v \in i(\mathrm{A})$) being destination of a data connector, the source of the data connector to be in *intermediate* or *final* state.

4. Cooperative Transactions: Cooperation between Parallel Activities

As introduced above, cooperation between parallel activities is mainly supported by COO-transactions. COO-transactions model relaxes the isolation property of ACID transactions in order to support consistent exchange of intermediate results between activities. For the sake of brievety, it is satisfying enough here to explain how its protocol provides an active support to cooperation while maintaining at the same time a sufficient level of consistency. Then we explain how process activities are encapsulated in COO-transactions.

4.1. *COO protocol*

The principle of COO-transactions is that a COO-transaction can read an intermediate result of another COO-transaction. A COO-transaction that has read an intermediate result of another transaction depends on it and must read the corresponding final result before to complete. COO-transactions implicated in a cycle of dependencies are grouped together and must agree on the final value of the shared data before to complete simultaneously.

A transaction that starts its execution enters the *isolated* state (called isolated with reference to ACID transaction executing in isolation), see Fig. 4. A cooperative transaction can read an intermediate result of another cooperative transaction. When a transaction T_0 reads an intermediate result (a result produced by a transaction during its execution before it completes) of an object x produced by a transaction T_1, it becomes dependent on T_1 (see Fig. 5).

When the transaction T_0 updates its value of x with the final value of x produced by T_1, the dependency is removed. A transaction which has read one or several intermediate results, but that is not implicated in a cycle (T_0 is not depending on itself) is in the *dependent* state (see Fig. 4). A transaction which is the *dependent* state and which releases its last dependency enters the *isolated* state again.

Transactions involved in a cyclic dependency graph form a group of transactions: each transaction of the group enters the *grouped* state. A transaction cannot try to terminate if it is still dependent on another transaction (not *up_to_date*). If a group-member transaction tries to terminate (it is *up_to_date* with other transactions), and they are still transactions of the group in the *grouped* state, it enters the *ready to commit* (RTC) state. If a group-member transaction wants to terminate and

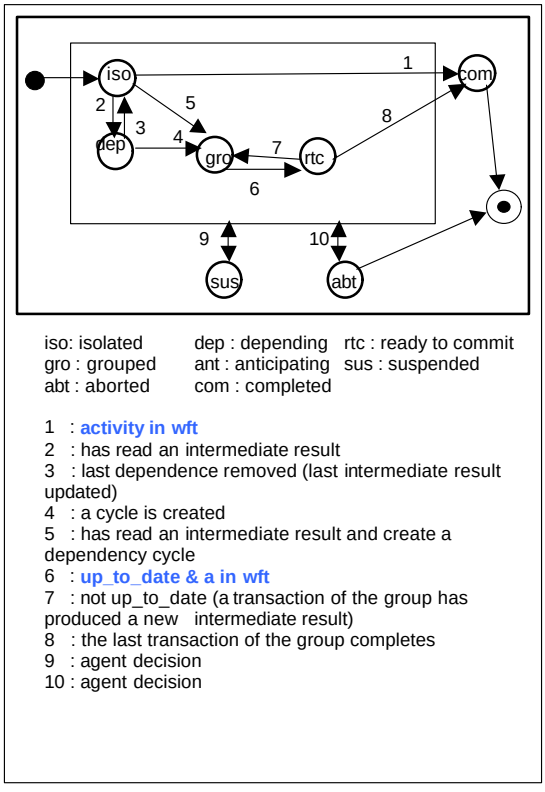


Fig. 4. Transaction states.

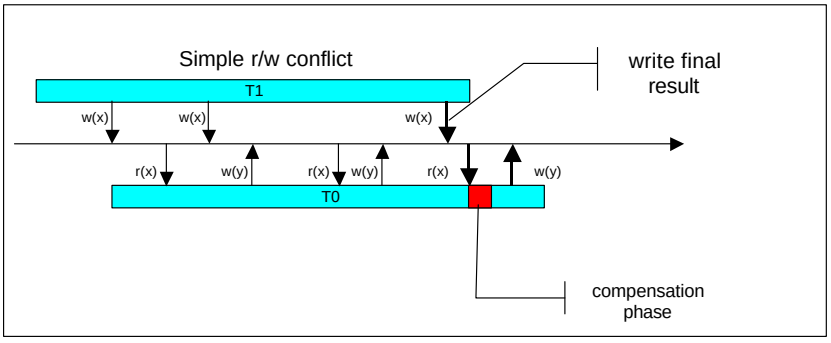


Fig. 5. T_0 uses intermediate results of T_1.

the other transactions of the group are in the RTC state, all the transactions of the group terminate simultaneously, i.e. enter the *completed* state. However, if a group member produces a new intermediate result during the group termination phase, then this termination is aborted and all group-members re-enter the *grouped*

state. A transaction can be *suspended* or forced to abort (*aborted* state). For a formalization of COO-transactions, see [5], [27].

The originality of the COO-transaction model [27], with regard to other advanced transaction models, is that it supports concurrent writing (including writing of two copies of the same document) while preserving some properties (COO-serializability). Especially this allows, thanks to intermediate results management, large scale process management, including disconnected work.

4.2. *Activities as COO transactions*

To support cooperation in a consistent way, each activity is encapsulated in a transaction. The problem is that a user can try to terminate an activity while the corresponding transaction is not *up_to-date.* To prevent this case, we have added in the activity model a new state, i.e. *waiting for transaction* (*wft*) and modified some transition conditions (see Figs. 3 and 4). The principle is that, when a user wants to complete an activity, before this completion can effectively take place, the system must verify that this activity is consistent with the others, or in other words that the corresponding transaction can also complete. When an activity enters the *wft* state, i.e. its post-conditions are fulfilled, a command to complete is sent to the corresponding transaction; if the corresponding transaction can complete, the transaction and the activity pass simultaneously in the *completed* state. If the transaction is implied in a group, the activity has to wait in the *wft* state for the group termination (this will be triggered by the last transaction of the group wanting to terminate) and all activities of the group will complete simultaneously. If the transaction still has to work before to complete (for instance, it has not read the last final value present in the database, or a new result has been recently published by a group member), the activity re-enters the *active* state.

The introduction of a new activity state for the synchronization of activity and transaction termination, as introduced in this paper, is one possible solution. We studied another symmetric solution in the Corvette project where we extended the transaction model in a symmetric way. We prefer the solution deepened in this section for usage reasons, but in both cases, orthogonality of activity management and transaction management is not completely achieved (in the sense that we have to modify either the activity model or the transaction model to integrate them together).

Encapsulation of activities in COO-transactions allows the following supplementary types of data exchange:

— Data can be accessed on the user initiative from a shared workspace which control the access (following a pull principle).
— Activities can exchange intermediate results. Moreover, in order to support opportunist styles of work, activities situated in parallel braches can exchange data and even work concurrently on shared copies.

These types of data exchange are not possible in traditional workflow systems where

activities are transactional black boxes and data is transmitted directly between activities (push mode).

New activity operations are introduced, *Write* and *Read*. These operations can be used by users (or even special tools) to manage publication of data during activity execution. *Write* operation publish a data element and makes it available to the other activities of the process instance. *Read* operation is used to read the current version of the data from the shared workspace.

5. Integration of Anticipation and COO-Transactions

In the previous sections, we have introduced anticipation and COO-transactions as complementary contributions to support flexible interpretation of creative processes. The problem we are concerned with in this section is that these two contributions are not completely orthogonal and each have an influence on the other.

Thus, we have pointed out two potential points of conflict:

- the termination of an activity (explained in the previous section),
- the risk of contradiction between the activity order implied by control flow and the activity order implied by data flow.

5.1. *Risk of contradiction between control flow and data flow*

Anticipation allows successive activities to execute partly in parallel, thus increasing the degree of parallelism. These activities can exchange data, generating dynamic dependencies between them. Although anticipation allows succeeding activities to run in parallel, the principle is that, at the end, things appear as if activities had not anticipated. Especially, if B succeeds A, A should start and terminate before B and exchanges that conduct to contradict such an order should be forbidden.

Reading intermediate results generates dependencies between activities dynamically; these dependencies are managed by the transaction manager. If A precedes B and A reads an intermediate result of B, directly or indirectly, the two activities will be grouped and forced to terminate simultaneously, what can be considered as contradicting the order relation defined by the control flow model. As example, in Fig. 6, A precedes B from the point of view of the workflow, and B has probably read an intermediate result of A, but A has also read an intermediate result of C that has read an intermediate result of B: A, B and C form a group of transactions.

We think that in a large number of cases, the fact that activities which were defined as successive in the process model are forced to terminate synchronously by the transaction manager is not really a problem. However, if it is, this situation can be prevented by enforcing the following rule: *an activity cannot read an intermediate result of another activity if this induces a dynamic dependency between activities that contradict the flow dependency* (*static dependency*) *between them.* This can be simply implemented as dependences are known and managed by the transaction manager.

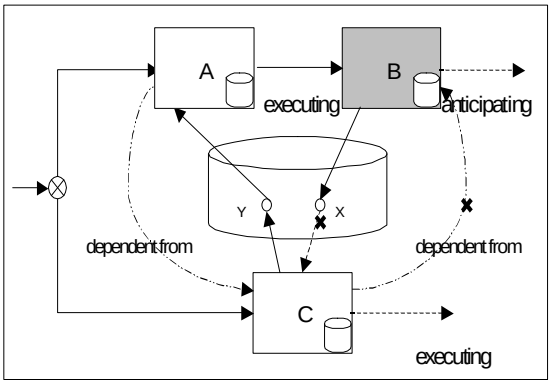


Fig. 6. Contradiction between control flow and data flow.

6. Implementations

This section reports on two implementations related to COO-flow model. Both of them have been deployed in an Internet context, exploiting capabilities of intermediate results management.

The Corvette project http://www.loria.fr /equipes/ecoo/corvette/ studied the feasibility of the COO-flow concept by the integration of the Hitachi Workcoordinator WFMS with COO-transactions, one important design constraint of the project being the impossibility to modify WorkCoordinator code. As WorkCoordinator does not support *anticipation*, cooperation between succeeding activities was not feasible without changing the WFMS. Fortunately, we were able to demonstrate cooperation between parallel activities, with only some smooth intervention in the transaction model and a small intrusion in WorkCoordinator for defining triggers in its relevant database in order to capture important events (especially activity state changes). This experience is reported in [3].

We have implemented a new version of the COO-flow model by taking advantage of the *Enterprise Java Beans* technology and the *J2EE* (*Java*^{TM} *2 Platform*, *Enterprise Edition*) for security, transaction management, connection with databases systems, portability. This implementation uses the Jboss application server (http://www.jboss.org/). It is called Bonita and can be downloaded from (http://woinville.loria.fr/nToxic/Project/bonita/Download/bonita-src.tar.gz).

Also, Bonita is being tailored for different means in LibreSource (http://www.libresource.org), a project dedicated to distributed software development, and in Coopera (http://woinville.loria.fr/coopera), a project dedicated to e-learning for children. COO-flow prototype is being integrated also in ToxicFarm (http://woinville.loria.fr), a platform hosting services for distributed teams. ToxicFarm will be experimented in the context of cooperative learning and of software design and development at the scale of France.

7. Related Works

Recently, there have been much research on workflow flexibility [4, 20]. Based on the role and the use of process model, we distinguish three main approaches for flexibility.

The first approach considers *the process model as a resource for action* ([1, 23]). Basically, it means that the process is a guide for users upon which they can build their own plan. It is not a definitive constraint that has to be enforced. Thus, users have all the initiative to execute their activities. They are not constrained by the predefined order of activities but are inspired by it and encouraged to follow it. In this vein, [23] proposes to enhance the workflow model with goal activities and regions in order to allow its use as a resource for action. A goal node represents a part of the procedure with an unstructured work specification; its description contains goals, intent or guidelines. If this approach provide a lot of flexibility on workflow execution, it also lack control on what effectively happens on working site (even if combined with user awareness).

The second approach uses the process as a constraint for the flow of work, but it is accepted that it may change during its lifetime. In other words, the process can be ally adapted during its execution. ADEPTflex [24], Chautauqua [9], WASA [30] and WIDE [6]] provide explicit primitives to dynamically change running workflow instances. These primitives allow to add/delete tasks and to change control and data flow within a running workflow instance. Constraints are imposed on the modifications in order to guarantee the syntactic correctness of the resulting process instance. [28] [26] [10] propose methods to migrate instances from the old process definition to the new one without introducing errors.

The third approach for flexibility consists of evolving the process model itself to allow for more flexible execution. In this case, flexibility has to be modelled and is anticipated during the process modelling step. In Mobile [21], authors define several perspectives (functional, behavioural, informational, organizational, operational) for a workflow model, the definitions of perspectives being independent of one another. Descriptive modelling is defined as the possibility to omit irrelevant aspects in the definition of a perspective. In MCI [11], authors propose simple constructs that better represents real work patterns (optional activities, group activities, synchrony) to be used instead of a complex composition of elementary workflow operators.

The first two approaches consider flexibility by allowing to escape from the model itself. In one case, the model is a guide to reach a goal, in the other case, the model is a path to reach a goal that may change during its course. In the third approach, it is the model that evolves to provide the requested flexibility. The approach introduced in this paper is quite different. It is not based on the way the process model is used or instantiated, neither on the way it can be evolved or modelled: it adds flexibility in the workflow management system execution engine itself. This flexibility is based on the study of the relationships between control and

data flow: anticipation of activity execution combined with exchange of intermediate results provide for a very effective support for cooperation between activities. In addition, as an activity can be prevented to anticipate and to publish intermediate results, different levels of flexibility can be defined and can co-habit in the same process.

Another possibility to provide a support for cooperative processes that was investigated in the literature is to merge workflow which supports formal and well-structured processes with groupware, which offers communication and cooperation tools for unstructured processes. [17, 29] integrates into a cooperative hypermedia system with many communication and cooperation features and process support capabilities. However, the correction of data exchange is not assured but is left to user responsibility.

During the past decade, traditional workflow systems we been extended with transactional features. Examples of these are TriGSflow [19] workflow system, which is based on an active database system using a multi-parent nested transaction model, and the Exotica [2] system, which shows how advanced transactions models, such as Linear Sagas and Flexible Transactions, can be implemented on top of existing workflow systems such as FlowMark. The basic ideas of other transactional workflows (ConTracts [25], WAMO [7, 8], WIDE [14], CREW [18]) are the transactional bracketing of parts of a process, attaching compensation and contingency activities to the activities of the process, declaring some of the activities to be vital (or critical), and defining points in the process up to which rollback occurs on failure, followed by forward execution. In these models, the cooperation is limited to passing results between activities in a programmed, predefined manner. They relax the atomicity property of a process, by dividing it in sub-transactions. However, these sub-transactions are still atomic. For these reasons, these models defined for a given application domain are difficult to adapt for cooperative processes that are dynamic, unpredictable and less structured.

8. Conclusion and Future Directions

This paper has introduced a new process model that fits very well the requirements of creative processes coordination by introducing control flow flexibility and dataflow flexibility without changing process description. Its technical feasibility has been demonstrated and its usability in distributed teams has been partially tested. In addition, this approach is not presented in opposition with existing ones, but on the contrary, as our experiments have demonstrated the need, as a complementary one.

As an example, COO-transactions can be introduced transparently as a set of cooperation patterns at the same level as MCI patterns [11], as we started to do in [12]. More generally, we understood during different experiments in different application domains that the approaches for workflow flexibility are complementary and that the better we would be able to make several of them co-habit, the better

we would provide support for co-design and co-engineering processes. In fact, in a co-design and co-engineering application, it does not exist one single process but several ones. In addition, depending on the nature and granularity of processes, these processes follow different models. As a consequence, there is a need for a framework to reason about these different models and their interactions: this is our front line objective.

Anticipation strategies can be enhanced by adapting the algorithms developed in [16] to decide on the opportunity to anticipate an activity by calculating a probability for this activity to become active in the future.

Another objective is to study the relationships between process-based coordination and awareness based coordination. Especially, process knowledge can be used to provide awareness (process awareness) and other sources of awareness can provide a good feedback of process acceptance on working sites on which process evolution can be based. This work is developed in the context of Toxic Farm (http://woinville.loria.fr). The same preoccupation exists in MCI [11].

In the continuation of this study, an ongoing objective is to make usage analysis in realistic situations by taking advantage of the applicative projects introduced above.

We are also concerned with the integration of the COO-flow model with other models, in the context of an intranet process or a multi-enterprises process.

References

1. A. Agostini and G. De Michelis, *Modeling the Document Flow within a Cooperative Process as a Resource for Action*, CTL-DSI, University of Milano, 1996.
2. G. Alonso, D. Agrawal, A. El Abbadi, M. Kamath, R. Gunthoer, and C. Mohan, "Advanced transaction models in workflow context", *Proc. 12th Int. Conf. on Data Engineering, ICDE*, New Orleans, 1996, pp. 574–581.
3. K. Baina, F. Charoy, C. Godart, D. Grigori, H. Skaf, S. Akifuji, T. Sakaguchi, T. Seki and M. Yoshioka, "Corvette: A cooperative workflow experiment", *3rd IFIP Working Conf. on Infrastructure for Virtual Enterprises*, Sesimbra, Portugal, 2002.
4. A. Bernstein, C. Dellarocas, and M. Klein, "Towards adaptive workflow systems (CSCW-98 workshop report)", *ACM SIGMOD Record* **28**(3) (1999) 518.
5. G. Canals, C. Godart, P. Molli, and M. Munier, "A criterion to enforce correctness of indirectly cooperating applications", *Information Sciences* **110** (1998) 279–302.
6. F. Casati, S. Ceri,, B. Pernici and G. Pozzi, "Workflow evolution", in *Data and Knowledge Engineering*, Elsevier Science, January 1998.
7. J. Eder and W. Liebhart, "The workflow activity model WAMO", in *Proc. 3rd Int. Conf. on Cooperative Information Systems*, Vienna, Austria, 1995.
8. J. Eder and W. Liebhart, "Contributions to exception handling in workflow management", *Proc. 6th Int. Conf. on Extending Database Technology*, Valencia, Spain, 1998.
9. C. Ellis and C. Maltzahn, "Chautaqua workflow system", *30th Hawaii Int Conf. on System Sciences*, Information System Track, 1997.
10. A. Fent, H. Reiter and B. Freitag, "Design for change: Evolving workflow specifications in ULTRAflow", *CAISE 2002*, LNCS 2348, 2002, pp. 516–534.
11. D. Georgakopoulos, H. Shuster, A. Cichoki, D. Baker and M. Rusinkiewicz, *Collaborative Management Infrastructure: Final Report*, 2000, MCC.

12. C. Godart, C. Bouthier, P. Canalda, F. Charoy *et al.*, "Asynchronous coordination of virtual teams in creative applications: Requirements and design criteria", *Australian Conference Workshop on Information Technology for Virtual Enterprises*, ed. M. Orlowska, IEEE Press, 2001.
13. C. Godart, O. Perrin, and H. Skaf, "COO: A workflow operator to improve cooperation modeling in virtual processes", *9th Int. Workshop on Research Issues in Data Engineering Information Technology for Virtual Entreprises* (*RIDE VE'99*), 1999.
14. P. Grefen, J. Vonk, E. Boertjes, P. Apers, "Two-layer Transaction management for Workflow Management applications", *Proc. 8th Int. Conf. on Database and Expert Systems Administration*, Toulouse, France, 1997.
15. D. Grigori, *Workflow Elements for Cooperative Processes Definiton and Enactment*, PHD thesis in *Computer Sciences*, University Henri Poincaré Nancy1, Nancy, 2001, p. 135.
16. D. Grigori, F. Casati, U. Dayal and M. C. Shan, "Improving business process quality through exception understanding, prediction, and prevention", 27^{th} *Int. Conf. on Very Large Data Bases* ($VLDB$), Roma, 2001.
17. J. M. Haake and W. Wang, "Flexible support for business processes: Extending cooperative hypermedia with process support", *Information and Software Technology* **41**, 6 (1999).
18. M. Kamath and K. Ramamritham, "Failure handling and coordianted execution of concurrent workflows", *Proc. 14th Int. Conf. on Data Engineering*, Orlando, Florida, February 1998.
19. G. Keppel, S. Rausch-Schott, W. Retschitzegger, and S. Vieweg, "TriGS: Making a passive object-oriented database system active", *JOOP7(4)* **63** (1994) 40–51.
20. M. Klein, C. Dellarocas and A. Bernstein (eds.), "Computer Supported Cooperative Work (CSCW)", *Journal of Collaborative Computing* **9** (2000).
21. S. Jablonski, "Mobile: A modular workflow model and architecture", *4th Int. Working Conf. on Dynamic Modeling and Information Systems*, Noordwijkerhout, NL, 1994.
22. F. Leymann and D. Roller, *Production Workflow*, Prentice Hall, 1999.
23. G. Nutt, "The evolution toward flexible workflow system", *Distributed Systems Engineering*, 1996.
24. M. Reichert and P. Dadam, "ADEPTflex — Supporting dynamic changes of workflows without losing control", *Journal of Intelligent Information Systems* **10** (1998).
25. A. Reuter, K. Schneider and F. Schwenkreis, *Contracts revisited*, in S. Jajodia and L. Kerschberg (eds.), *Advanced Transaction Models and Architectures*, Kluwer Academic Publishers, NY, 1997.
26. S. Rinderle, M. Reichert, and P. Dadam, "Evaluation of correctness criteria for dynamic workflow changes", *Proc. Int. Conf. on Business Process Management* ($BPM'03$), Eindhoven, The Netherlands, 2003.
27. S. Tata, G. Canals and C. Godart, "Specifying interactions in cooperative applications", *11th Int. Conf. on Software Engineering and Knowledge Engineering, SEKE'99*, Kaiserslautern, Germany.
28. W. van der Aalst, "Exterminating the dynamic change bug. A concrete approach to support workflow change", *Information Systems Frontiers* **3**(3) (2001) 297–317.
29. W. Wang and M. J. Haake, "Flexible coordination with cooperative hypermedia", *Proc. ACM Hypertext'98*, 1998, pp. 245–255
30. M. Weske, "Flexible modeling and execution of workflow activities", *31st Hawaii Int. Conf. on System Sciences*, Software Technology Track (Vol VII), 1996.

CORVETTE: a cooperative workflow for virtual teams coordination

K. Baïna, F. Charoy, C. Godart*, D. Grigori, S. el Hadri and H. Skaf

ECOO Team, LORIA/INRIA, Campus Scientifique, BP 239,
54506 Vandœuvre Cedex, France
E-mail: Claude.Godart@loria.fr
http://www.loria.fr/equipes/ecoo
*Corresponding author

S. Akifuji, T. Sakaguchi, Y. Seki and M. Yoshioka

Hitachi SDL, Kawasaki Lab., 1099, Ohzenji, Asao-ku,
Kawasaki-shi, Kanagawa-ken, 215-0013, Japan
http://www.sdl.hitachi.co.jp/

Abstract: Workflow management systems are now widely deployed for handling administrative and production applications. In order that the workflow management systems keep supporting a larger range of applications, several research works have been launched to improve the workflow technology. This paper reports on the CORVETTE [1] project, an experiment in developing a cooperative workflow management system by integrating a workflow management system commercialised by Hitachi Ltd. with a cooperation technology proposed by INRIA. A cooperative workflow management system is a system gathering workflow management functionalities and having capabilities to manage cooperative behaviours characteristics of creative application processes (e.g. in codesign and coengineering). In this kind of applications, interactions between activities are characterised by being more subtle than in traditional applications (i.e. non-scheduled, unpredictable, dynamic, etc.). More precisely, CORVETTE was targeted to support coordination workflow processes of a virtual team working over the internet.

Keywords: workflow management systems; cooperative processes; coordination of virtual teams; cooperative transaction models.

Reference to this paper should be made as follows: Baïna, K., Charoy, F., Godart, C., Grigori, D., el Hadri, S., Skaf, H., Akifuji, S., Sakaguchi, T., Seki, Y. and Yoshioka, M. (2004) 'CORVETTE: a cooperative workflow development experiment', *Int. J. Networking and Virtual Organisations*, Vol. 2, No. 3, pp.232–245.

Biographical notes: Karim Baïna is an Assistant Professor at "Ecole Supérieure d'Informatique et d'Analyse des Systèmes" (ENSIAS), Rabat, Morocco. He obtained his PhD in Computer Sciences from University Henri Poincaré, Nancy 1, France, 2003. His research interests include cooperative systems, workflow management systems, group negotiation and decision systems.

François Charoy is an Assistant Professor of Information Systems at the University of Nancy 2. He received his PhD in Computer Science from the University Henri Poincaré of Nancy, France in 1992. His research interests include issues related to process modelling, workflow, long term transactions and cooperative work. He has published papers in many conference proceedings and information systems journals. He has also contributed to important software.

Claude Godart is Full-Time Professor at University Henri Poincaré, Nancy 1, France. He is scientific leader of the ECOO project of INRIA interested in developing middleware for supporting cooperative work. His proper interests includes consistency of data in distributed systems, workflow models and systems, web services and virtual enterprises.

Daniela Grigori received her masters and PhD from the University Nancy I, France in 1998 and 2001, respectively. Since 2002 she is an Assistant Professor at the University of Versailles, France. Her main research interests are in process modelling, business process intelligence, web services, coordination and cooperation models and systems.

Hala Skaf-Molli is an Assistant Professor at the University Henri Poincaré, Nancy 1, France. She received her PhD in Computer Science from the same university in 1997. Her research interests include issues related consistency maintenance of the data mediating the cooperation between several partners, and combination of transactional techniques and transformational techniques for the synchronisation of copies.

Shunsuke Akifuji received his MS (Engineering) degree from University of Tokyo, Japan in 1988. He joined Hitachi, Ltd. in 1988 and is currently working for Hitachi (China) Investment Ltd., Research & Development Center. His research interests include issues related to workflow management system, enterprise applications and web services.

Toshiaki Sakaguchi received his MS (Science) degree from Kyoto University, Japan in 1990. He joined Hitachi, Ltd. in 1990, and is engaging in the development of a Collaboration Ware "BOXER" since 2001. His interests include issues related to workflow management system, collaboration software and web services. He has participated actively in the Workflow Management Facility standardisation in OMG.

Yoko Seki received her MS (Engineering) degree from Nagoya University, Japan in 1997 and is currently a Researcher at Hitachi, Ltd., Systems Development Laboratory. Her research interestsare in the areas of business process management, chiefly workflow management systems, and web services.

Masaichiro Yoshioka received his MS (Science) degree from Kyoto University, Japan in 1982. He joined Hitachi, Ltd. in 1982, and is engaging in Information and Communication Technology (ICT) business planning and R&D management since 2000. His business relates all the aspects of ICT areas and trends. He has established several research collaborations with research institutes including INRIA.

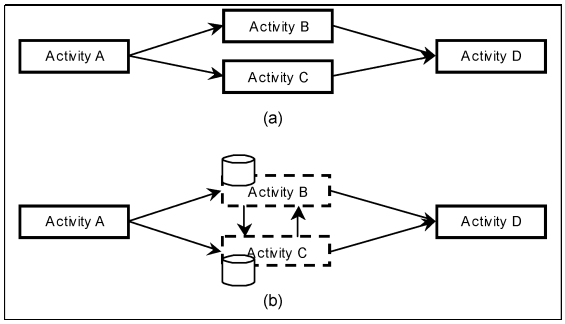
1 Introduction

Workflow management systems are now widely deployed for handling administrative and production applications. To address a larger range of applications, a lot of research has been launched to surpass the limits of current workflow technology [2–15]. Among them, some researches [4,5,16] concentrate on supporting creative processes, typically codesign and coengineering processes, where interactions between activities are more subtle than in traditional applications (i.e. non-scheduled, unpredictable, dynamic, etc.). We call a system that gathers workflow management functionalities and has capabilities to manage cooperative behaviours characteristic of creative applications as a cooperative workflow management system. This paper, written on the basis of [17], reports on the CORVETTE project. CORVETTE is an experiment in developing a cooperative workflow management system by integrating a workflow management system commercialised by Hitachi Ltd. with a cooperation technology developed by INRIA. More precisely, CORVETTE was targeted to support coordination processes of a virtual team working over the internet. It is organised as follows. Section 2 describes the motivation for cooperative workflow. Section 3 describes the CORVETTE system design. Section 4 sketches the implementation. Finally, Section 5 synthesises the experiment and discusses some conclusions.

2 Why cooperative workflow?

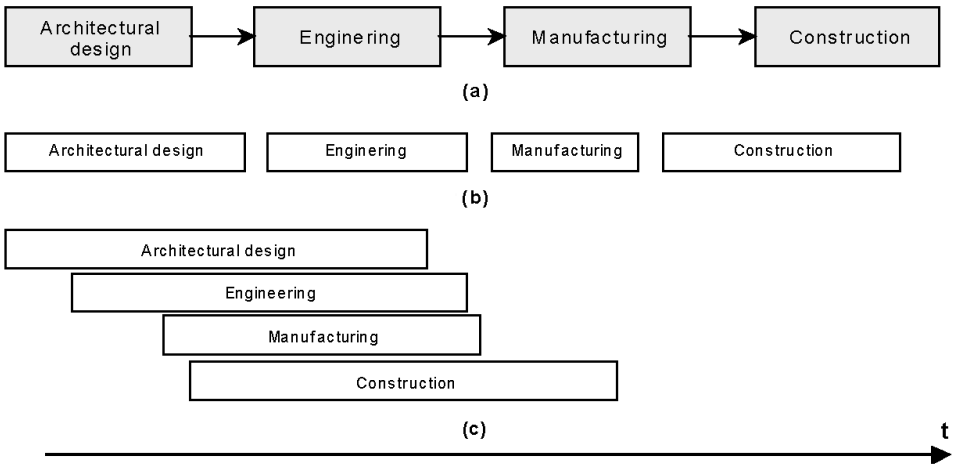
Current workflow models are mainly concerned with the automation of administrative and production processes. These processes coordinate well-defined activities which execute in isolation, i.e. synchronise only at their start/terminate states. Though these models work efficiently for a class of applications, their limitations become evident when one wants to model the subtlety of cooperative interactions as they occur in more creative processes, typically codesign and coengineering processes. Cooperative workflow (Figure 1) has been mainly introduced to overcome these limitations. A cooperative workflow is a workflow where some activities being executed in parallel can share some intermediate results during their execution [18,19]. A cooperative workflow has the capability to synchronise activities at any point of their executions and thus ensures the coordination between designers [19].

Figure 1 From classical to cooperative workflow (a) classical work flow: activities B and C are closed and execute in isolation and (b) collaborative workflow: activities B and C are porous and can share intermediate results when executing



To illustrate this, let us consider three partners cooperating within a building trade process (an architect, a research engineer and a building contractor). The architect has the responsibility of producing plans corresponding to a set of requirements given by a customer. Based on the plans of the architect and on his own expertise, the research engineer makes the main technological choices. The building contractor has the responsibility of manufacturing wood components and finally of assembling them on building site. This process is roughly described in Figure 2(a). A traditional (i.e. sequential) execution of this process is depicted in Figure 2(b). However, real processes do not execute really in this way. Processes corresponding to the three partners are more intricate and execute in parallel rather than in isolation. Actually, they exchange documents when executing with the objective to obtain a fast feedback and to point out risks in the construction as soon as possible. In general, definition of a model as in Figure 2(b) can delay a design problem and contribute to poor acceptance of the workflow management system on working sites. A better way to do things is to allow the architect, the engineer and the contractor to cooperate by exchanging intermediate results early (e.g. draft documents) when operating. This idea is illustrated in Figure 2(c) besides the fact that this organisation can dramatically decrease the total duration of a process.

Figure 2 Building trade example (a) building trade process model, (b) building trade serial execution and (c) building trade collaborative execution



Enabling interactions between parallel activities is a typical cooperation characteristic within creative applications that we are concerned with. One important question at this point is: can a traditional workflow management system (WFMS) model and enact such interactions efficiently? In more accepted definitions of a workflow management system [20–22], an activity is a black box with an input container that is filled before the activity starts its execution and an output container that is filled when the activity ends. Thus, in classical workflow management systems, visibility of intermediate results is categorically prohibited.

Therefore, such a WFMS cannot easily support such cooperative interactions. One can try to model it with iterations, knowing that the limits of this solution are easily reached. Interactions are difficult to forecast and their number increases exponentially with the number of activities [19]. Moreover, for practical opportunity, some products

allow activities to interact when executing, but in this case, semantics of parallel executions become quite unclear, or rather is delegated to the applications, under the responsibility of the programmer. As far as current systems go, they cannot easily model cooperative behaviours. There is, therefore an obvious need for improving workflow management systems with cooperative workflow technology.

3 CORVETTE design

The objective of the CORVETTE project is to examine the development of a cooperative workflow management system to coordinate a virtual team cooperating through a wide network, typically the internet. Cooperation has to be taken here in the sense of the ECOO team cooperation model that we will present below. The final system must be implemented by 'connecting' the *Hitachi WorkCoordinator* [23] workflow management system and the *ECOO MOTU cooperative system* [24], which provides complementary cooperation services, and especially COO [25] cooperative transactions. This achievement must consider the *WorkCoordinator* system as a black box and thus must not touch its source code. Thus, the implementation has to enrich the *WorkCoordinator* with new cooperation services that interact with workflow services only by invocating the *WorkCoordinator* CORBA application programmable interface or by monitoring its embedded Oracle database.

This section is organised in three steps. First, we introduce the contributing technologies, followed by presentation of the important design decisions, and finally, we illustrate the detailed architecture of CORVETTE.

3.1 Contributing technologies

We start with the ECOO cooperation technology represented by the *MOTU cooperative system* and then introduce the Hitachi workflow technology within the *WorkCoordinator* system.

3.1.1 ECOO cooperation technology: Motu system

The ECOO cooperation model is based on the ability for an activity to publish an intermediate result, as introduced in Section 2. More precisely, the ECOO cooperation model is characterised by four generic cooperation patterns as depicted below:

- *Producer/consumer*. Two activities follow a producer/consumer cooperation pattern; one has the responsibility to create/modify an object and the other reads this object to integrate it in its own work. Producers and consumers can momentarily see different versions of the same object, but the consumer must retrieve the final producer version.
- *Redactor/reviewer*. Two activities follow a redactor/reviewer cooperation pattern; the redactor creates/modifies an object and the reviewer reads one or several successive versions of this object with the objective of reviewing it. The corresponding review objects are also shared in their turn.

- *Cooperative write*. Two activities follow a cooperative write pattern; they develop a common object, work on two different versions and frequently merge their modifications. Finally, they have to agree on a common final version.
- *Concurrency.* Two activities follow a concurrency cooperation pattern if they must execute in isolation. In cooperative processes, some activities must be allowed to share some results, while other activities must be allowed to execute in isolation with respect to others.

Based on the ECOO cooperation model, the *MOTU cooperative system* [24] is a computer supported cooperative (or CSCW) tool on the lines of BSCW [26], TeamScope [27] or Sourceforge [28]. It proposes, among other cooperation services, workspace management, communication, coordination, group awareness [29] etc. The main *Motu* cooperation service components used in CORVETTE project are the workspace manager and the COO-transaction manager that have been streamlined to manage consistency of non-linear versioned document exchanges.

The COO-transaction protocol asserts *concurrency atomicity* of cooperative processes (i.e. correctness of interactions between processes exchanging results when executing). For this purpose, each process is encapsulated in a COO-transaction. Each COO-transaction executes in its private workspace and COO-transactions cooperate by transferring intermediate results between their workspaces. The COO-transaction protocol is a set of rules that sets constraints on these transfers. They are intuitively depicted in the following:

- A result produced before the end of a COO-transaction is always an intermediate result of this transaction. Users can produce an intermediate result (operation *IR-write*).
- We call as the final result, a result produced at the end of a COO-transaction. All final results are produced atomically during the execution of the *terminate* operation of the transaction. A COO-transaction that produces an intermediate result must produce a corresponding final result. The protocol collects all the objects that were *IR-written* by the activity and automatically produces a final result for each of them during the termination phase of the activity.
- If a transaction has read an intermediate result, then it must read the corresponding final result. The system maintains a *dependency relationship* between activities to memorise the fact that a transaction reads an intermediate result of another one. When a transaction A_1 reads the intermediate value of an object x produced by a transaction A_0, a dependency $A_1 \rightarrow A_0$ is created. When the transaction A_1 reads a value of x and A_0 is terminated (i.e. when it has produced its final result), the dependence is removed.
- A transaction cannot terminate if it is still dependent on another one. If a transaction tries to terminate without reading the final value of some object after a previous access to an intermediate value of this object, the *terminate* operation is aborted and the transaction remains active.
- Transactions involved in a cyclic dependence graph form a group of transactions.

- A group-member transaction can start a group termination by trying to terminate itself. The *terminate* operation in this case produces a set of potentially final results and changes the state of the transaction from *active* to *ready to commit* (RTC).
- When a group-member transaction tries to terminate and all the other group-members are in the RTC state, then all the transactions are terminated simultaneously. Potentially final results are definitely promoted to final results.
- When a group-member transaction tries to terminate and another group-member is still active, then the former produces new potentially final results and enters the RTC state.
- If a group member produces a new intermediate result during the group termination phase, then this termination tentative is aborted, and all the group-members re-enter the active state. This is the way for a transaction to clearly indicate its disagreement with the object values produced by the group, and to ask for more work on the shared object.

More formal definitions of COO-transaction model and relationships between concurrency, atomicity and consistency is given in [30,31].

3.1.2 Hitachi workflow technology: WorkCoordinator system

Based on the Work*f*low Management Coalition (or W*f*MC) workflow specifications [20], the *WorkCoordinator* workflow management system (or *WCO*) considers a workflow entity as a control flow graph with *activities* as nodes and inter-activities *transition*s as edges (i.e. and-join, and-split, or-join, or-split, sequence, etc.). An activity has a description (textual field), a deadline (in days), and a post condition (an SQL statement) which defines when the activity can be completed. An activity entity is associated to a non-empty set of *work item*s. A work item entity is an atomic job symbolising the place where effective work is done. Contrary to activities, there is no control flow scheduling the work item execution. A work item has a description (a textual field), a casting rule (an SQL statement) that identifies the work item performer (en identifier user), a pre-condition and a post-condition (SQL statements) which define, respectively, when it can start and complete. The parallel execution of activities or work items enables data to be shared when executing (objects, rows and files) and provides flexibility and cooperation. However, parallel execution consistency control concerning data access is the responsibility of programmers. *WCO* workflow model emphasises control flow but does not describe data flow between either activities or work items. *WCO* separates the process definition model (including process control flow definition) and the process execution context (including casting rules definitions, used application information, condition definitions and rule definitions, etc.). While the former is encapsulated in the *WCO* framework, the latter is handled by external database management systems (Hitachi or Oracle RDBMS). Thanks to this flexibility of *WCO* architecture, the modification of a process execution context is possible without process definition model alteration. Moreover, *WCO* maintains the possibility of modifying process definitions at runtime. Thus, cohabitation of several versions of the same processes can be ensured. Besides these capabilities of definition and enacting processes, *WCO* offers facilities to monitor life cycles of process instances, activities and work items. Finally, it integrates a CORBA Workflow Application Programming Interface (or WAPI) which is *WfMC*

Interface 2 compliant. This interface allows external applications to invoke some of the workflow methods and functionalities. For this purpose, the *WorkCoordinator* is based on a *Visigenic broker Architecture for C++*.

3.2 CORVETTE software design

The constraint of not modifying the *WorkCoordinator* system source code has seriously directed CORVETTE design choices. This accelerated our choice for cooperative process modelling and simplified discussions about architecture. However, the scope of cooperation model to be implemented in CORVETTE, has been enough limited. Otherwise, we have clearly distinguished between what is the content of workflow services and what is the content of data flow and concurrency services, which was an important policy matter in middleware software programming.

CORVETTE cooperative process modelling. The question is: how to model cooperative processes? In the ECOO project, we study two complementary approaches. The first approach consists in providing new workflow operators to explicitly point out where cooperation is possible [18]. These 'cooperative operators' extend the set of traditional operators [32]. The second consists in modelling a cooperative process in the same way as traditional production processes, using the same set of operators, but interpreted in another way (i.e. a cooperative way). In other terms, in a traditional interpretation, activities are seen as black boxes, while in a cooperative interpretation they are seen as white boxes (being allowed to share intermediate results).

In CORVETTE, we chose the second alternative, which has two qualities:

- First, it corresponds to (a) reality in process books, administration processes are described in the same way as design processes. Actually, the workflow process reader interpretation changes depending on his/her know-how of application domain (idea depicted in Figure 2).
- Second, introducing cooperation has no impact on process modelling and allows us to reuse *WCO* process modeller, to model CORVETTE cooperative processes, as it is. Thus, this respects the constraint of not modifying the *WCO* source code, but transfers the problem to process management (i.e. enactment, monitoring, etc.).

Another decision was, without losing generality, to have *one work item per activity*.

CORVETTE cooperation patterns modelling. Based on the decision of keeping the *WCO* model for cooperative process modelling and the constraint of not modifying the *WCO* source code, a question that arises is: which cooperation, or which cooperation patterns, is still possible to model with this constraint? In fact, and fortunately, *WCO* is mainly concerned with control flow and is very permissive regarding data flow. In fact, it imposes no constraint on data flows and cooperation between two work items, and in our context between two parallel activities, is allowed. An important restriction came, however, from the fact that a *WCO* activity can execute only when its preceding activities have terminated their executions. This means that only activities in parallel branches can cooperate, which has an impact on cooperative process modelling. Moreover, cooperation patterns *Producer/consumer* and *Redactor/reviewer*, referring by nature to succeeding activities, cannot effectively be implemented. In other terms, either these patterns are not allowed, or their implementation implies some contra-nature modelling.

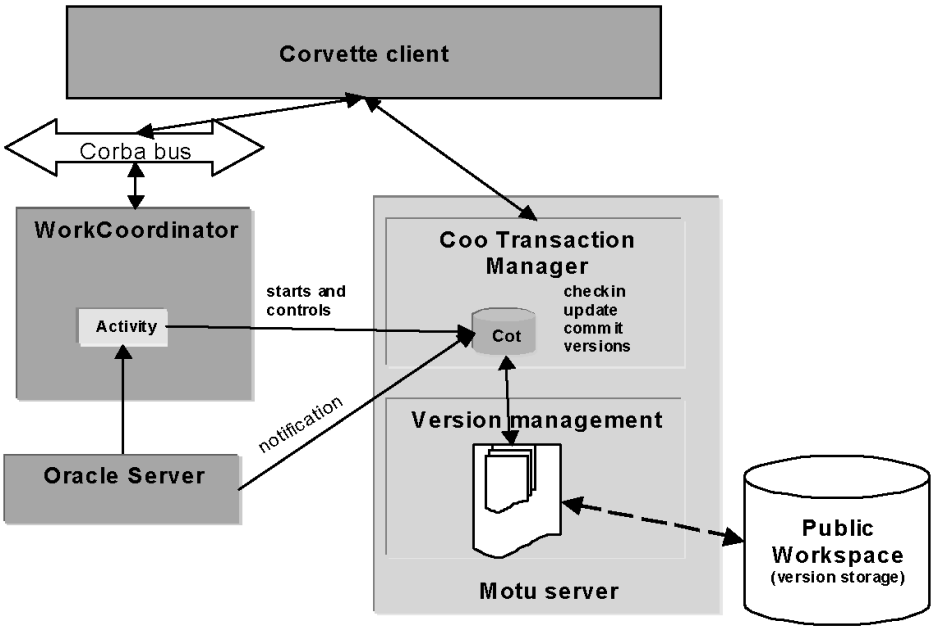
3.3 CORVETTE software architecture

CORVETTE software components. The *WorkCoordinator* workflow management system handles workflow process modelling, enactment and coordination of workflow activities delegating data management services to the programmer responsibility. The *MOTU cooperative system* supports COO-transactions entities and manages a versioned document workspace for each COO-transaction. The CORVETTE architecture, thus, integrates two main components: a workflow management component handling control flow between workflow activities and a data management component handling data versioning and data flow between COO-transactions.

In this architecture, it was not possible to modify the *WorkCoordinator*, so the decision was quickly oriented towards the definition of a mediator between *WorkCoordinator*, *Motu* and users without intervening in contributing components source code. However, this was not the single reason for choosing this architecture. We think that clearly distinguishing between a component for control flow management and another for data flow management involves considerable software architecture in middleware programming context.

Figure 3 depicts the overall CORVETTE architecture. A *CORVETTE Client* component assumes the mediator role between *one user*, *WorkCoordinator Server, and Motu Server* (embedding *COO-transactions and workspaces Managers*). In other words, there is one *CORVETTE Client* per user. This latter can simultaneously be performer of several work items, executing in different workspaces. Concurrency management between work items that are interfaced by one or several *CORVETTE Client*s leads to *Motu COO-transaction Manager*. More information on CORVETTE implementation details are given in [33].

Figure 3 CORVETTE general design



CORVETTE plugging rules. The main issue in defining *CORVETTE Client* was the ability to establish a correspondence and coherency between *WorkCoordinator* process definition entities (handled by *WorkCoordinator Server* and *Oracle Server*) and COO-transactions entities (handled by *MOTU Server*). In other terms, how does one encapsulate a *WorkCoordinator* unit of work in a COO-transaction entity?

As, in *WorkCoordinator*, real work is executed in work items, the decision was to associate each work item to a COO-transaction. As work items can run in parallel, cooperation in the sense of intermediate result sharing can occur. And encapsulating work items in COO-transaction asserts consistency of cooperation in the sense of the COO protocol. As work items are started automatically by the Work Coordinator execution engine as soon as their activation conditions are fulfilled (e.g. the preceding work item has completed), this must be detected in order to create the corresponding COO-transaction. Reciprocally, termination of a work item must be done in coordination with its corresponding COO-transaction. This means that termination condition of the work item and of the COO-transaction must be fulfilled at the same time. Another issue concerns termination of a group of COO-transactions. When several COO-transactions are grouped due to cyclic dependences between them, they must terminate simultaneously, following a kind of two phases termination protocol. That means that all corresponding work items must also terminate simultaneously. To implement this capability, one new state was introduced (*WaitingCommit*) in the COO-transaction model. Table 1 depicts this mapping.

Table 1 WorkCoordinator work item/COO-transaction states mapping

Work item state	*COO-transaction state*
Initial	Initial
Performing	Executing
	Waiting Commit
	RTC
Completed	Terminated

The role of CORVETTE client is to manage this mapping. This is mainly performed in the *create work it*em, *perform work item, open workspace* and *terminate work item* CORVETTE commands as follows:

- *CORVETTE create work item*: this command overwrites the *WorkCoodinator create work item* command to create a work item, and its associated COO-transaction structures, including the corresponding private workspace
- *CORVETTE perform work item*: this command overwrites the *WorkCoordinator perform work item* command to manage the associated COO-transaction (pushing it in *executing* state)
- *CORVETTE open workspace*: this command allows to create and populate the private workspace associated to the COO-transaction with necessary work item enactment artefacts
- *CORVETTE terminate work item*: this command overwrites the *WorkCoordinator terminate work item* command to manage the associated COO-transactions. Let us explain the algorithm of this command:

Let t_w this work item associated COO-transaction.

1 if t_w is not in a group of COO-transactions:
  1.1 if t_w is dependent on another one (has read an intermediate result of another active COO-transaction), it must wait for this other transaction to terminate
  1.2 if not, t_w enters the *waiting commit* state and asks its encapsulated work item to terminate
    1.2.1 if the work item terminates (all its termination conditions are fulfilled), the COO-transaction t_w commits
    1.2.2 if not, the COO-transaction t_w returns to the *executing* state

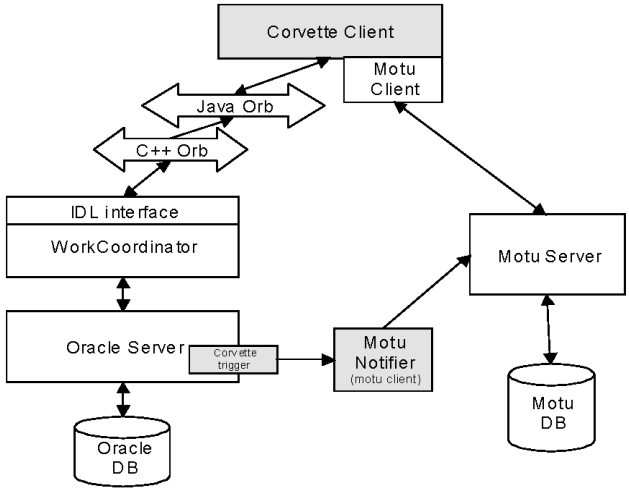
2 if t_w is member of a COO-transaction group:
  2.1 if there are still COO-transaction(s) of the group *executing*, the COO-transaction t_w enters the *ready to commit* (*RTC*) state
  2.2 if all other grouped COO-transactions are in *RTC* state, the COO-transaction t_w triggers the termination of all encapsulated work items of the group and enters the *waiting commit* state
    2.2.1 if all group work items terminate, all their associated COO-transactions are committed
    2.2.2 if not, as a work item cannot go back to the *performing* state, a human group decision session has to be launched to insure a manual recovery of the work item group.

As a direct consequence of cooperation cycles detection, the problem with a 'work item group' termination is that all the group work items, associated to grouped COO-transactions, have to terminate simultaneously. A termination conflict occurs when a sub-set of such a group have terminated and that one other work item of this group cannot. As other work items are not able to return from *terminated* to *performing* state, the system is blocked. In the context of our applications, *abort* is not acceptable and direct human intervention for manual recovery of the work item group is necessary. As in other computer supported cooperative applications, human decision is needed to help the system to solve conflicts (e.g. exception recovery, resolution of indeterminism, negotiation of needed set of values [34], etc.). *WorkCoordinator* experts affirm, however, that the case where a work item fails is extremely rare.

4 CORVETTE implementation

As depicted in Figure 4, CORVETTE is a client for, on the first hand, the *WorkCoordinator Server*, and on the other hand, the *Motu Server*. Concerning the interface between *CORVETTE client* and *Motu server*, as both are written in Java, *CORVETTE client* is simply a special Java RMI (Remote Method Invocation) *Motu Client*. While, the interface between *CORVETTE Client* and *WorkCoordinator Server* (written in C++), it is based on *WorkCoordinator IDL interface* and on interoperability between *Visigenic ORB for C++* and *Visigenic ORB for Java*.

Figure 4 CORVETTE implementation overview



Otherwise, in order to detect work items creation and termination, as necessary for transaction management, triggers and Java stored procedures have been added to the *WorkCoordinator* workflow relevant data managed by an *Oracle Server* (*CORVETTE trigger* component). Each time a new work item is created or deleted, a message is sent to the *Motu notifier* component. Note that the installation of these triggers in the *WorkCoordinator* relevant data database is the only intervention done in some *WorkCoordinator* structures.

Finally, the *Motu* notifier is a special *Motu Server* that monitors *WorkCoordinator* events and creates a transaction each time a work item is started. It manages also notification information for users awareness support.

5 Synthesis and conclusion

Globally, this experiment is a success. We demonstrate the feasibility of defining a cooperative workflow management system by 'plugging' (in the sense of integration software component without modification) together a 'traditional' workflow management system and an advanced cooperative transaction model. The main reason for this success is the absence of data flow consideration in the *WorkCoordinator*. Thus, we did not have to manage the integration of *WorkCoordinator* data flow model with our transaction model. Another success is the demonstration of the ability to model cooperative processes as traditional processes, but to interpret them in a cooperative way corresponding to cooperative behaviours.

This success is limited in the sense that not all cooperation capabilities, initially forecasted, have been implemented in such a flexible way (due to the inability to share intermediate results between succeeding activities). To overcome this limitation, it is necessary to provide activities with the capability to anticipate:

> "anticipation is the weakening of strict sequential execution of activity sequences in a process by allowing intermediate results to be used as preliminary input of succeeding activities."

For more about anticipation, see [35]. Anticipation allows the implementation of *Producer/consumer* and *Redactor/reviewer* between succeeding activities, thus providing support for the full ECOO cooperation model.

As a conclusion of this experiment, we think that, if a workflow manager component does not impose constraints on data flow, and if it provides the capabilities introduced in the previous paragraph (anticipation, events and group termination), it will be possible to completely develop a cooperative workflow management system by simply plugging together this workflow component and a cooperative transaction manager. In addition, if all the activities of the process are concurrent (i.e. execute in isolation) as the process model does not change, this workflow management system has the behaviour of a traditional (competitive) one.

The experience gained in the Corvette project deeply influences our current developments, namelythe *Bonita* [36] flexible workflow management system and the *Toxic Farm* [37] portal for virtual team hosting.

References and Notes

1 CORVETTE stands for Coordination of a Virtual Team.

2 Agostini, A. and De Michelis, G. (1996) *Modeling the Document Flow within a Cooperative Process as a Resource for Action*, University of Milano.

3 Baker, D., Georgakopoulos, D., Schuster, H. and Cichocki, A. (2002) 'Awareness provisioning in collaboration management', in *International Journal of Cooperative Information Systems.*

4 Schuster, H., Baker, D., Cichocki, A., Georgakopoulos, D. Rusinkiewicz, M. (2000) 'The collaboration management infrastructure', in *International Conference on Data Engineering* (ICDE), San Diego.

5 Grigori, D., Charoy, F. and Godart, C. (2001) 'Flexible data management and execution to support cooperative workflow: the COO approach', in *The Third International Symposium on Cooperative Database Systems for Advanced Applications (CODAS)*, IEEE Press, Beijing, China.

6 Baïna, K., Benali, K. and Godart, C. (2001) 'A process service model for dynamic enterprise process interconnection', in *9th Int. Conf. on Cooperative Information Systems (CoopIS)*, Springer Verlag, Trento, Italy, LNCS 2172.

7 Baïna, K. and Dustdar, S. (2002) 'Web-services coordination models', in *Second International Workshop on Cooperative Internet Computing (CIC)*, Kluwer Academic Publishers, Hong Kong.

8 Reuter, A.a.S., F. (1995) 'Contracts – a low-level mechanism for building general-purpose workflow management systems', *IEEE Data Engineering Bulletin*, Vol. 18, No. 1.

9 Suchmann, L.A. (1987) *Plans and Situated Action. The Problem of Human-Machine Communication*, Cambridge University Press.

10 Nutt, G.J. (1996) 'The evolution toward flexible workflow systems', in *Distributed Systems Engineering*.

11 Reichert, M. and Dadam, P. (1998) 'ADEPTflex – supporting dynamic changes of workflows without losing control', *Journal of Intelligent Information Systems*, Vol. 10.

12 Weske, M. and Schneider, B. (2002) 'An XML-centred system architecture for flexible electronic services', in *International Journal of Information Technology and Decision Making*, Vol. 1, No. 3, pp.525–540.

13 Ellis, C. and Maltzahn, C. (1997) 'Chautaqua workflow system', in *30th Hawaii Int Conf. On System Sciences, Information System Track.*

14 Han, Y., Sheth, A. and Bussler, C. (1998) 'A taxonomy of adaptive workflow management', in *Towards Adaptive Workflow Systems,* CSCW Workshop, Seattle, USA.

15 Casati, F. and Pozzi, G. (1999) 'Modeling exceptional behaviors in commercial workflow management systems', in *4th IFCIS Int. Conf. on Cooperative Information Systems (CoopIS)*, IEEE Computer Society Press, Edinburgh, Scotland.

16 Joeris, G. (1999) 'Defining flexible workflow execution behaviors', in *Enterprise-wide and Cross-enterprise Workflow Management – Concepts, Systems, Applications*, *GI Workshop Proceedings – Informatik*, Ulmer Informatik Berichte Nr. 99-07, University of Ulm.

17 Baïna, K. *et al.* (2002) 'CORVETTE: a cooperative workflow development experiment', in *3rd IFIP Working Conference on Collaborative Business Ecosystems and Virtual Enterprises (PRO-VE'02)*, Kluwer Academic Publishers, Sesimbra, Portugal.

18 Godart, C. (1993) 'COO: a transaction model to support cooperating software developers', in *European Software Engineering Conference*, Garmisch, Germany, LNCS 717.

19 Godart, C., Olivier, P. and Skaf, H. (1999) 'COO: a workflow operator to improve cooperation modeling in virtual processes', in *Research Issues in Data Engineering*, Sydney, Australia.

20 WfMC, *Workflow Management Coalition*, http://www.wfmc.org/

21 Jablonski, S. and Bussler, C. (1996) *Workflow Management – Modeling Concepts, Architecture and Implementation*, International Thomson Computer Press.

22 Leymann, F. and Roller, D. (1999) *Production Workflow*, Prentice Hall.

23 WCO, *Hitachi-WorkCoordinator,* http://www.hitachi.co.jp/Prod/comp/soft1/wco/eng/index.html

24 Motu, http://motu.sourceforge.net

25 COO stands for COOperation transaction model

26 BSCW, *Basic Support for Cooperative Work*, http://bscw.gmd.de/

27 Teamscope, http://www.teamscope.com/

28 Sourceforge, http://www.sourceforge.net

29 Molli, P., Skaf-Molli, H. and Bouthier, C. (2001) 'State tree map: an awareness widget for nultisSynchronous groupware', in *7th International Workshop on Groupware*, CRIWG, IEEE Computer Society, Darmstadt, Germany.

30 Canals, G., Charoy, F., Godart, C., Molli, P. and Munier, M.A. (1998) 'Criterion to enforce correctness of indirectly cooperating applications', in *Information Sciences: an International Journal,* Vol. 110*,* pp.279–303.

31 Skaf, H., Charoy, F. and Godart, C. (1999) 'Maintaining shared workspaces consistency during software development', *Software Engineering and Knowledge Engineering Journal*, Vol. 9, No. 5.

32 Typically and-join, and-split, or-join, or-split, sequence as defined by W*f*MC.

33 CorvetteManual (2001) http://www.loria.fr/equipes/ecoo/corvette/CorvetteUserManual.htm.

34 Munier, M., Baïna, K. and Benali., K. (2001) 'A negotiation model for CSCW', in *5th International Conference on Cooperative Information Systems (CoopIS)*, Springer Verlag, Eilat, Israel, LNCS 1901.

35 Grigori, D., Charoy, F. and Godart, C. (2001) 'Anticipation to enhance flexibility of workflow execution', in *DEXA Conference*, Munich, LNCS 2113.

36 Bonita (2003) http://bonita.debian-sf.objectweb.org/.

37 Toxic Farm (2003) http://woinville.loria.fr.

AI & Soc (2006) 20: 384–402
DOI 10.1007/s00146-005-0025-2

OPEN FORUM PAPER

Nicolas Grégori · Jean-Charles Hautecouverture
François Charoy · Claude Godart

Combining ergonomics, culture and scenario for the design of a cooperation platform

Received: 1 October 2004 / Accepted: 10 August 2005 / Published online: 30 November 2005

Abstract Analyzing the way computer technologies are used is crucial for their development. Such analyses make it possible to evaluate these technologies and enhance their evolution. The present article presents some ideas drawn from the development of a cooperation platform for elementary school children (10–11 years old). On the basis of an obvious ergonomic requirement, we worked on two other dimensions: cultural aspects and the teaching scenario. The goal was to set up observation situations and analyze the conversations produced during those situations, in order to understand what using the platform meant to both the pupils and their teachers.

Keywords Cooperation · Usability · Ergonomics · Teaching scenario · Culture · Analysis of interactions

1 Introduction

There is a large body of research on the development of computer technologies for schools. The idea is not only to teach children with computers, but also to enable them to learn remotely or to work on joint projects. This last case is our focus here. The objective is twofold: first, to teach children to cooperate with each other, mainly in an asynchronous way, and second, to teach them how to cooperate by means of an Internet technology.

So, we are interested in a collective and distant production, and this production is mediatized by a computer device. However, the implementation of such cooperative activities mediated by computer environments does not rest

N. Grégori (✉) · J.-C. Hautecouverture
LabPsyLor/Codisant, Université Nancy 2, BP 33-97,
54015 Nancy Cedex, France
E-mail: gregori@univ-nancy2.fr · Tel.: +33-3-83913167 · Fax: +33-3-83913175
E-mail: jhauteco @univ-nancy2.fr

F. Charoy · C. Godart
Université Henri Poincaré and INRIA, LORIA, BP 239,
54506 Vandoeuvre Cedex, France
E-mail: charoy@loria.fr
E-mail: godart@loria.fr

solely on the technical and ergonomic properties of those environments. A key element is the educational devices themselves. It is hypothesized here that three factors have an impact on the usability and acceptability of computer tools. Of course, the first is the ergonomic factor, which is the facet most often brought to bear. The second is the cultural factor. The third is the teaching scenario. We contend that these three factors must be simultaneously taken into account in order to analyze a computer tool from the standpoint of its use. Indeed, it is useless and hazardous to concentrate solely on ergonomic aspects because one runs the risk of producing a perfectly efficient but useless tool. These three factors must also be considered early in the design process because they have a strong impact on the definition of the observation protocol.

This is the point of view defended in this paper, based on the design of a platform for cooperating via the Internet called Coopera.[1] The platform allows pupils (fifth graders of age 10 or 11) from several elementary schools to carry out joint projects. The originality of the platform is that users are able to run their own cooperative activities by means of the file sharing and viewing capabilities of the model underlying the design. The viewing feature allows the pupils to know at any time who created or modified a file and what modification was made. This allowed us to study the role of the educational device along with the appropriation of the platform.

So, a cooperation platform is not only a technical support for the management of the activity and the exchange of information but also an instrument to be mastered by the users and a resource for collective action. We defend then that we better talk about an ''instrumental device'' and not just about a ''technical device.'' This instrumental device is composed of three devices: the teaching scenario device, the computer device and the animation device. This results from our dialogical approach for use analysis. Since we are interested in the appropriation of the platform by the children, we have set up some work sessions during which the children have produced some actions and some speech acts. In addition, since they have to master the techniques proposed (the cooperation platform), we have elaborated two animations during which the children have physically handled the concepts of the platform.

To defend our point of view, we will start by presenting our approach to use analysis. Our aim was to observe natural situations in which the tool is used. In other words, we do not rely on an experimental psychology framework or an ergonomic testing method, but rather work as would an ethnographer. This approach to use analysis is important because it guides us in capturing and defining the three factors we discuss here. We will describe these factors in Sect. 4 using an excerpt of a conversation between a pair of children working on the platform. Before that, we will describe the platform technically (Sect. 3). We will finish with a discussion that points out the merits of our approach.

[1] Coopera is a RIAM project (Network for Multimedia and Audiovisual Innovation) financed by the Centre National du Cinema (French National Center for Cinema), 2002–2004.

2 Theoretical frame for usage analysis

This section presents our theoretical background. We include our works in the situated-action paradigm, which allowed us to observe natural work situations and examine the role of users in the development of the cooperation platform. We also present the participants of the three projects that occurred during this 2-year program.

2.1 Two reasons for working like ethnographers

The method we use to analyze the cooperative work is an ethnomethodologically informed ethnography (Crabtree 1998), for two main reasons. The first is epistemological. Any human activity is instrumentalized by objects on one hand and language on the other. With objects and words (Vygotsky 1978), we control and transform the environment and our own behaviors and hence our relationships with others. It is therefore important that our analysis take into account the weight of the environment and the social relations in the activity under study. Enquiries about technology-mediated activity (Kaptelinin and Nardi 2003; Kuuti and Bannon 1993) are useful because it describes an action as a chain of operations carried out by individuals who are not thinking about it. The observer's objective is to clarify this chain of operations.

The second reason is practical. The situations studied in the Coopera project are ones with strong social interactions in which the context deeply influences the activity. Therefore, the variables are multiple and complex, and we do not attempt to control them. On the contrary, our aim is to grasp this complexity. Moreover, our objective is not to compare groups of pupils under contrasted experimental conditions, which is why an ethnomethodological approach is more suitable than an experimental one (Nardi et al. 1993).

In addition to taking into account the context, we look at the role of the users themselves. Educational devices involve at least two types of users, each having different things at: pupils and teachers. In a certain way, a participative design process is taking place because our observations are centered on the users' actual behavior in a situation that allows them to express their understanding of the cooperative situation and their needs. Finally, behind our ethnographical approach, we are promoting a way of building and analyzing situations of technology-mediated cooperation.

2.2 Setting up of natural situations

Human activity is complex. It transforms the environment at the same time as it is transformed by it, in a kind of action–environment coupling (Hutchins 1995; Suchman 1987). To observe and study such a complex activity, it is illusory to try to characterize it using experimental parameters. On the contrary, it is necessary to set up situations in which natural practices can be carried out. Vygotsky (1978) proposed an experimental genetic method which meets this requirement. In line with this method, we set up pairs of pupils working in their own classrooms on projects designed by their teachers. This has two main advantages. First, it reflects the true use of technologies in France. Second, the

discussions generated between pairs of pupils can be analyzed. Of course, we adapted the Vygotskian method to current visual audio technologies, which allowed us to obtain linguistic and material recordings of the activity. Moreover, we were physically present during the work sessions, as observers.

2.3 Analysis of conversational interactions

The social interactions recorded consisted of language use and the handling of objects (mainly handling of the mouse and keyboard and events displayed on the screen). To analyze the linguistic part of the data, we relied on speech act theory (Searle and Vanderveken 1985; Vanderveken 1990), even if, theoretically, the principles of conversational analysis based on ethnomethodology do not go hand in hand with those of discourse analysis (Levinson 1983). However, we can use these two opposing theories and tools because the speech act category is dialogized (Trognon and Brassac 1995; Brassac and Grégori 2001).

The basic premise is that it is possible to "grasp" human cognitive processes by analyzing the speech produced by subjects in an interlocutionary situation. Better yet, a fine-grained description of the chain of conversation, for modeling purposes, is a reliable way of gaining insight into the mechanisms of human cognition. We are acting here as theorists of social interaction, viewed at the "micro" level.

The concept of speech act, in its original definition, will serve as our starting point. When a subject in a conversational context performs an utterance, he/she is accomplishing what is called a speech act. Each speech act is an elementary link in the conversational chain. Speech act theory was first axiomatized in the form of illocutionary logic (Searle and Vanderveken 1985), and then expanded into a formal general semantics (Vanderveken 1990). Granted, this theory has been and still is the subject of heated debate from all sides: first, because it bears the mark of a radically monologist attitude; secondly, because the role it grants to the speech act as the analysis unit of verbal interaction is often considered totally inadequate. We shall not dwell on this debate here, but it is clear that one of the major criticisms of this formal system, which meticulously models the expression and comprehension of language by human subjects through the in-depth study of their speech acts, is it completely fails to account for language usage in a dialogue situation. There are two main reasons for this: its omission of the non-literal dimension of conversation, despite how fundamental it is to intersubjectivity, and the static nature of any analysis that does not even address the processes at play in the dynamic progression of a conversation. The crux of this issue indeed lies here for anyone who hopes to use this theory to model interactions as they unfold, i.e., for anyone who wants to account for interaction as a process. The key is to take this general formal semantics and transform it by what one might call "dialogization," where the goal is to delineate and handle the non-literal and dynamic facets of the interlocutionary exchange.

The basic idea of dialogization is that, contrary to the classical theory, the initial utterance has no illocutionary status apart from that afforded by its processing by the actors as the conversation proceeds. Its status is neither the product of the emitting speaker alone nor can it be accredited solely to the listener. It is built by means of a meaning-negotiation process carried out jointly by the two interlocutors. A given utterance, in a given conversation, does not

have just *one* meaning, the meaning its speaker attributed to it once and for all (whether literal or otherwise). It only acquires the meaning within the subtle interplay of a process of negotiation between two conversers; even then, its significance is only temporarily stabilized, and it belongs to neither of them.

Being co-responsible for stabilizing the interlocutionary significance of each utterance in the sequence, the actors in the exchange participate in the co-construction of the meaning of the linguistic forms that weave the fabric of the conversation. As it hinges on the key idea that meaning is co-constructed in a process-based and radically dialogical fashion, this way of modeling conversation takes a constructivist approach. It is not necessary to postulate the existence of a predefined meaning that precedes the expression or comprehension of the linguistic form produced in context. All that is needed is the simple idea that the conversers jointly mould the still-negotiable meaning in a process-driven way. This is the view of conversational exchange that will be used in our analyses.

Speech act category is therefore an interesting unit of analysis because it goes back to both cognitive and social dimensions of conversations. It makes it possible to simultaneously take into account the meanings built by the subjects and the social relations that emerge during the activity.

But we are not just interested in the language produced. We also observe the objects handled (Vinck and Jeantet 1995), because they are means for coordinating actions and producing shared knowledge. Bringing such ''intermediary objects'' to bear in our analyses of the actors' schemes and actions is the outcome of much more recent and much less polished, theorizing efforts on our part. The importance of such objects to exchanges between actors in a cooperative situation has become very obvious to us and has forced us to recognize the merits of including object manipulation in our theoretical account.

2.4 Role of the actor-user in the design process

The above points (setting up natural situations and analyzing situated social interactions) lead us to reflect upon the role of users (pupils and teachers) in the design of the cooperation platform. If we agree that meaning is produced in context (and more specifically in a natural context), then we must also agree that users who produce that meaning play an active role in the design process. They are not testers or evaluators, but prescribers. That is why the situations are designed to promote our understanding of how users go about cooperating in a context that requires complex synchronizations, at the operative (concerning the platform itself), cognitive (concerning the knowledge produced) or social (concerning group awareness for example) level. This goal is very important because the concerned users are 10 to 11-year-old children, who are not accustomed to working this way, and teachers who are not used to working in such a context either, even though they volunteered to participate.

2.5 Ergonomics, scenario and culture

The development of a cooperation platform for pupils must meet ergonomic requirements. Of course, the platform must be efficient and usable. The study of its contextual use must make it possible to work on this dimension. It must also

meet the requirements of a teaching scenario. It is the teaching scenario that situates the action and therefore allows us to construct a natural observation setting, for at least three reasons. First, the teaching scenario does or does not support cooperation; secondly, it does or does not motivate the children; and thirdly, it involves the teachers in the project. Thus, generating good work situations means paying attention to this requirement. Moreover, the act of cooperating is not meaningless. It is based on social relations between individuals, by means of particular procedures. It is thus inscribed in a cultural environment that is complex. The development of the cooperation platform is therefore necessarily tied to this third requirement.

The analyses produced must make it possible to characterize the platform according to these three essential dimensions: ergonomics, teaching scenario and culture. This is why we defend an approach centered on social interactions.

2.6 Participants

As stated above, the study involved two types of participants: pupils (fifth grade) and teachers. The pupils were the direct users of the platform. They worked in cooperation across schools. The teachers could be considered as the administrators of the platform rather than direct users. They were important actors too, since they had to define the teaching scenarios. The computers were located in dedicated rooms that can be found in most French elementary schools today. Mostly, two children shared a computer. Three projects were carried out during 3 school years using three successive versions of the platform.

Two schools were involved in the first project, entitled Poems. In an asynchronous way, the pupils exchanged their activities in order to write, illustrate and format poems (May–June 2002). The cooperation gave rise to a final product that gathered all poems. The second project (Operette: school year 2002–2003) involved a third school. The task was to create a Web site about the opera house in Nancy. In addition to creating the documents (computer files and web pages), the objective for the pupils was to learn how to share skills, share knowledge and articulate ideas in order to achieve a common result. The third project, an online magazine, took place during the 2003–2004 school year. A fourth school joined the first three. Note that each year we worked with the same schools but new groups of children.

3 Coopera: technical aspects

In any computer technology, the technical aspects are important. It is because the tool should function and offer new possibilities that it becomes useful.

In this article, we consider only a specific part of cooperation, that occurring between a group of persons working together to produce a set of documents in an asynchronous way, i.e., distributed in space and time. We relied on the cooperation model of a platform developed locally, the *Toxic-Farm* (Godart et al. 2004; Fig. 1). One of the goals of the project was to produce a version of this platform adapted to users who are not familiar with cooperative work.

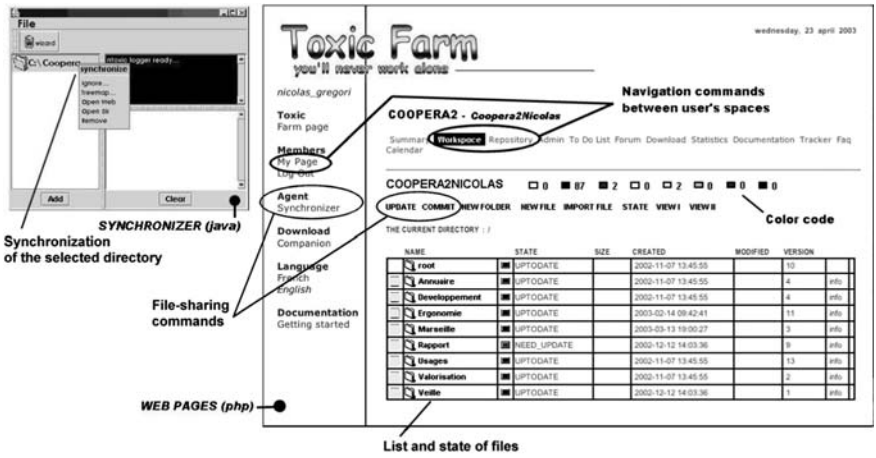


Fig. 1 ToxicFarm. Each Web page contains a lot of information. Most of the commands for space navigation, coordination, communication, etc., can be launched from nearly anywhere. This is disruptive for children who still have difficulty viewing the organization of spaces and need a clear structure for features and commands

3.1 Cooperation model

The cooperation model implemented in *ToxicFarm* is widely used in cooperative work, in a large variety of applications (co-authoring, co-designing, co-engineering, etc.). It is rooted in the nature of cooperation itself, but has been equipped with tools and popularized in the software engineering domain through the copy/modify/merge paradigm[2]. and in the groupware domain, typically in the diverge/ merge model of *Prospero* (Dourish 1995).

This approach emphasizes multi-synchronous work where ''Working activities proceed in parallel (multiple streams of activities), during which the participants are disconnected (divergence occurs) and periodically their individual efforts are integrated, by means of a synchronization, in order to achieve a consistent state and advance the activity of the group.'' Here, cooperation consists in creating alternative contributing versions of a base object, merging them back into the base object, creating alternative versions and so on.

Privacy between partners is generally achieved by maintaining a multi-space: a central directory contains the up-to-date versions of shared objects and each partner is associated to a private workspace where he/she can check out objects to modify them. Before committing his/her own changes to the common directory, the user must merge those changes with other changes committed to the directory since his/her last check. This can lead to conflicts if two people have modified the same file in their own private space; in such a case, conflicts must be resolved before the file can be committed to the common directory (this can be done automatically or may require some communication between the two people).

[2] http://www.cvshome.org/docs/manual/

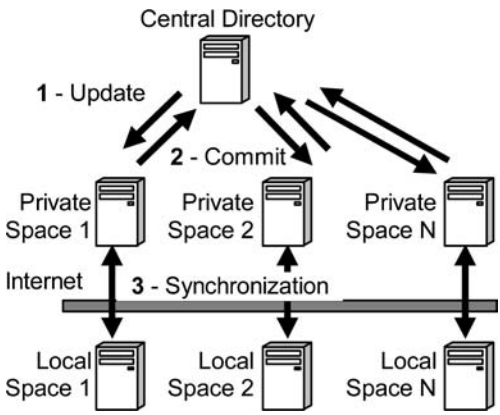


Fig. 2 Space management in ToxicFarm

3.2 Initial system: *ToxicFarm*

Object sharing in *ToxicFarm* is based first on a workspace management system that implements a long-transaction model (Feiler and Downey 1990). For each project, there is a shared object space (called the central directory) where all objects in the projects including their versions are stored. Each user in the project can create a private workspace. Initially, a private workspace contains a copy of the last version of all files from the shared space, i.e., a copy of the whole project. Users can update their private workspace with new versions from the central directory or publish new versions from their private workspace to the central directory. This system is similar to the widely used systems like CVS found in software engineering, except that this version of the management system has been simplified and generalized (updated version of directories). In addition, the long-transaction model requires the complete set of changes done in the directory to be published each time a 'commit' is performed. In order to avoid lost updates, a user can only 'commit' if no changes have been made in the directory since the user's last update.

Classically, the central directory is stored on a server, and private spaces are directly supported by the user's computer. But in ToxicFarm, the users' private workspaces are also kept on the central server, and an additional space level, called the local workspace, has been added. A local workspace is a copy of the private workspace located on the user's own machine (Fig. 2). This allows for advanced functionalities like awareness and mobility support (Dourish and Bellotti 1992; Gutwin et al. 1996). To maintain consistency between the local workspace and its corresponding private workspace, the user can periodically synchronize them. All interactions between private and local spaces are managed by a tool called the Synchronizer.

A typical usage scenario for this platform can be described as follows: a new user wants to participate in a project and starts by creating a private workspace for the project. To be able to read and modify the project objects, he synchronizes this private workspace on his computer, which creates a local workspace for the private one. Then he works with his usual tools, modifying files in his local workspace (possibly disconnected from the network). He can

periodically synchronize his work, which transfers his changes to his private workspace.

While this user is working, someone else may publish some changes in the common directory. To see changes in his local workspace, the user must, on his own initiative, update his private workspace and then synchronize. This may create conflicts that will have to be resolved. When he stops working, the user can synchronize and then publish his work. When he reconnects after a break, he can update his private workspace and synchronize it in order to have knowledge of any work done by his partners during his absence.

The availability of the central directory and private workspaces on the central server provides users with state awareness capabilities. Users are notified of changes made on files in other users' private workspaces before they are published. States are seen from the point of view of the user. A file can:

- Be up-to-date (no one has changed it)
- Be locally modified (changed in the current user's private workspace)
- Be modified (a new version has been published in the central directory)
- Be remotely modified (a user has changed it in his private workspace but has not published it yet)
- Cause a potential conflict (it has been locally and remotely modified)
- Be in conflict (it has been locally modified and there is a new version in the central directory)
- Be in version conflict (there are two copies of the same file in the private directory, the locally modified one and the updated one from the shared directory)

In classical systems, conflicts are detected only when they occur, whereas in our system, thanks to state awareness, they can be detected earlier and even avoided because users are aware of modifications that have not yet been published. Our assumption is that providing this kind of awareness will help project participants improve coordination. We use colors associated with files to represent file states and provide a synthetic view of the states of files.

4 Three factors for use analysis: ergonomics, culture and teaching scenario

Once again, we were working at the same time on the evaluation and development of the platform, so psychologists and developers were always closely working together. In this section, we present the evolution of the cooperation platform. Then we describe the three usage factors we defend. This is done by analyzing an excerpt of a conversational interaction that occurred during the use of the second version of the platform (Coopera 1).

4.1 Evolution of the Coopera interface

The Coopera interface has evolved during the two years of the project. In both its initial state (ToxicFarm) and as Coopera 1, it was composed of two very different elements: a set of Web pages developed in php and Java synchronizer (Figs. 1, 2, 3). The final development combined all the functionalities into a java tool (Fig. 4).

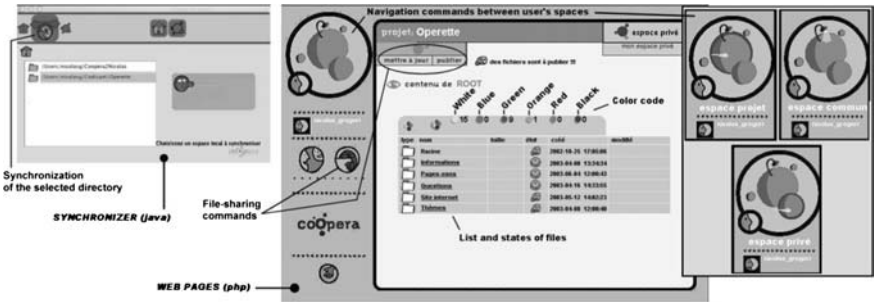


Fig. 3 Coopera 1. Navigation can be done from feature to feature and colors are used to denote change. Only the subset of commands useful in the current context is displayed. Only one private and local space is allowed per user. The planet/satellite metaphor was introduced in order to enhance their visual presentation of spaces. But this metaphor did not suffice. The particularization of the synchronizer remained a problem. In addition, refreshing Web pages in order to view modifications was difficult to manage for children

As shown in Figs. 1, 3 and 4, the evolution of the platform between versions Coopera 1 and Coopera 2 is the clearest. The functionalities remain the same, but the tool is very different. Indeed, Coopera 1 ended up being just a graphic evolution of ToxicFarm. Coopera 2 represents a major evolution of the platform because it takes the cultural dimension into account.

Thus, the evolution was not just ergonomic, an aspect which was also studied of course. But the redesign of the two environments (Web pages and synchronizer) into only one was the fundamental step for Coopera 2. The concept of cooperation is now salient in the interface. It is expressed via the appearance of the principle of cooperation. Indeed, the file-sharing model is present and used as a support for action, in two ways. First, its state changes according to the

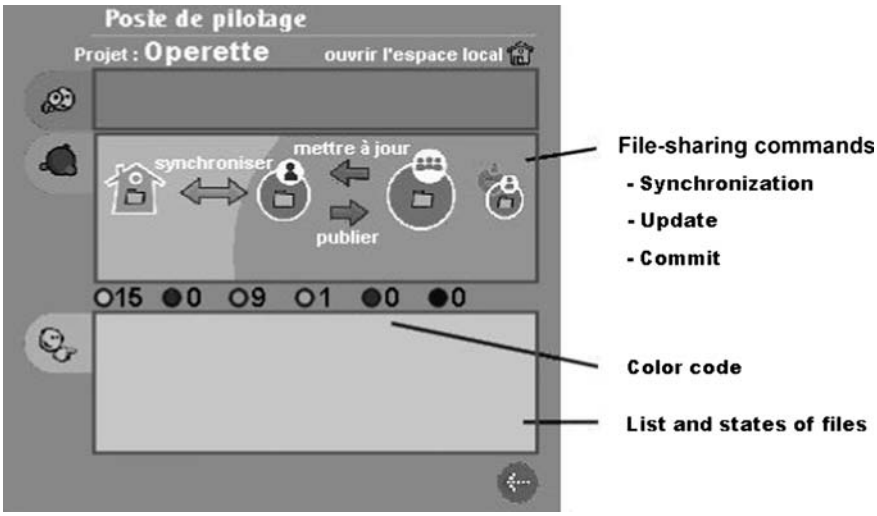


Fig. 4 Coopera 2. On the same page, the cockpit (a Java-rich client) shows the different levels of workspaces, handles a visual transfer of files between spaces and provides real-time awareness of space changes

actions to be carried out. For example, if I must update my private space, the ''update'' arrow (''mettre à jour'' on Fig. 4) is active, informing me of the need to perform this action. Secondly, this representation makes it possible to carry out actions. If I want to update my private space, I click on the ''update'' arrow.

Now we will discuss the three factors in greater detail using a conversational sequence. In addition to the ergonomic and cultural aspects just mentioned, we will present some preliminary ideas on the teaching scenario, which does not appear directly in the interface.

4.2 A sequence analysis: color code interpretation difficulties

We now present a work sequence carried out by pupils. This will allow us to point out important parts of the cooperation dynamic, including how the state of cooperative activity was perceived and how a situation was understood.[3]

The excerpt studied here is part of the third work session out of seven on the Coopera 1 version. During this step, two girls were working together to answer a question sent by another pair from another school. Eleven minutes after starting to use the platform, the pair (J and S), supported by a teacher (T), opened a page showing their private space. J said (01) ''You always have to look at the color circles there.'' The children started reading the color code. The main events are related to two errors they made at this time.

Fig. 5 shows that the excerpt can be divided into two main times, that is two errors made by the children: a misidentification of the spaces and an action erroneously anticipated. From these two sequences, we will see that the children have a correct understanding of the file-sharing model (the names of the spaces, how files move from space to space, operations to move files). They also understood the principles of the color code (circles are related to a space, the number refers to the number of files in the space and provides an overall view of the project).

Misidentification of a space. After T's question about *''what are the color circles saying,''* S (03) answered interpreting the meaning of the blue circle: ''*We have nine in blue.''* In other words, there were nine things in the corresponding space, named the common space (J 05a). Here really started the misidentification of the spaces since J, pointing to an orange circle with mouse (07), said *''One in our space.''*

A process of stabilization is now going ahead, following T's intervention about the meaning of the green circle (13). This question breaks with the representations expressed by the children about the meaning of the circles. Knowing that the green circle is associated with the private space of J and S, the children now deduce that the orange circle represents the private space of their partners. That is what is done in J 20a. So T's initial question (T 2) is now satisfied, i.e., the children are now able to say *''what are the color circles saying.''* We have represented this question/answer pair with E1 (Fig. 5).

An action erroneously anticipated. What the green circle means is now stabilized. But the two girls are now going to make a second error answering to T who questions the meaning of the number associated to the green circle (23b).

[3] For reasons of language comprehension, we have chosen to translate the sequence in English, even if this process poses some problems.

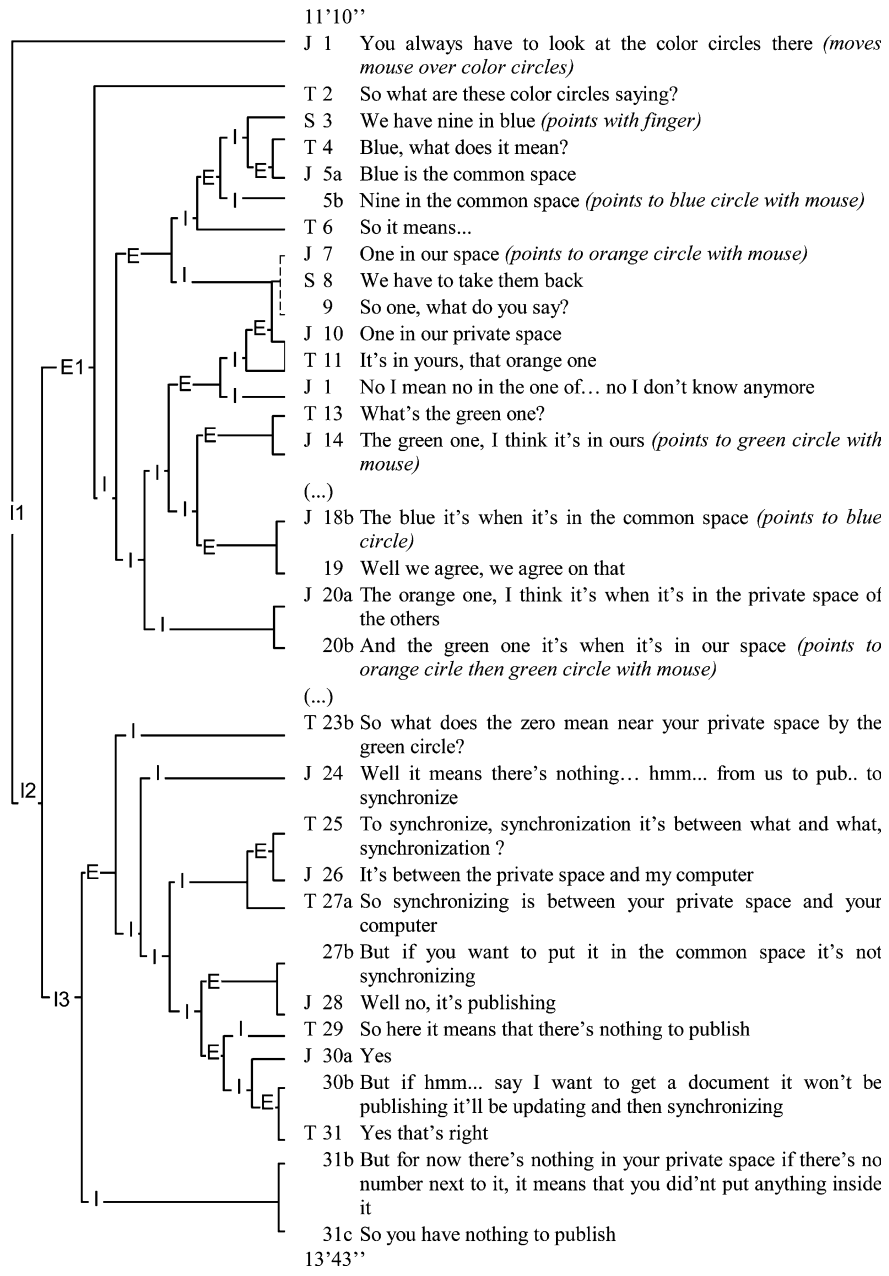


Fig. 5 Functional and hierarchical aspects of the conversational sequence

The action associated to this circle is to commit but not to synchronize as J said (24). Our hypothesis is that J's error is not really due to a misunderstanding of the action to be realized but due to a misunderstanding of the situation itself. More precisely, J's error is that she believes that the files in her private space are

coming from her partners. When J evokes the case in which she would get a document from others (30b), she confirms our hypothesis because this highlights the fact that she has well understood the file-sharing model. This utterance is important because between T 26 and J 30a, J has just answered to T's questions. But in 30b, she has really expressed that she well understood how the cooperation platform is functioning.

For the first time, the situation expressed by the green circle and the number associated to it has been explicitly argued. It is because this situation was implicit till now that the two girls and T misunderstood each other.

Viewing the file-sharing model in order to facilitate the cooperation. The children have made two main errors in this sequence. The analysis of the sequence made it possible to state that the file-sharing model did not pose difficulty for young children but that they misunderstood the situation in which they were and, consequently, that they were unable to anticipate what to do.

The children's first error regarding the state of the cooperative activity was not the result of a lack of knowledge, but of difficulty relating that knowledge to the properties of the color code. This difficulty expresses an ergonomic deficiency in the way the cooperative dynamic could be visualized, which was too abstract. The cognitive work required was too great for the fifth grade children. A better view of the file-sharing model, one combining spaces and states, would certainly facilitate this understanding.

The children's second error, regarding the state expressed by the green circle and the number, raises the question of the model of relationship with others and the self-appropriation induced by the system involving the different colors and files represented. Ergonomics is very important in the appropriation process and thus in understanding the state of the cooperative activity. Here again, vizualization of the file-sharing model was very important. This is what was achieved in Coopera 2 (Fig. 4).

4.3 Usability factors

4.3.1 Micro- and macro-levels analysis

Let us present the three factors of usability. The excerpt presented here points out some ergonomic difficulties that must be taken into account. The analysis of the conversation made it possible to understand the emergent meanings and make ergonomic recommendations. But difficulty anticipating future actions is not just an ergonomic problem. It also depends on the teaching scenario (it is necessary to motivate and engage the children in a cooperative activity) and on cultural aspects (it is necessary to know what "to cooperate" means, to have an idea of "who I am" and of "who the partner is" in the cooperative system). In other words, we were working at both the micro- and the macro-levels.

At the micro-level, we were interested in the ease and difficulty of use. Bastien and Scapin's (1993) criteria are used to interprete the ergonomic value of the various versions of the platform. In the excerpt (Coopera 1), the criteria for failure were meaning, incentive and density of information (Fig. 7).

Thus, the micro-analyses allowed us to work on both the ergonomic and cultural factors. J's second error was both ergonomic and cultural: J focused on

the use of the platform but not really on the cooperation with others. In contrast, the teaching scenario was analyzed at a macro-level. It was conducted via discussions with the children and their teachers. Let us note, however, that this scenario was strongly related to the other two factors. Indeed, the utility of a platform depends on the scenario, which determines how much it will be used. In other words, even if the ergonomics is correct and the cultural framework is good, the platform will not be useful unless the teaching scenario provides the incentive to cooperate. These three factors are interdependent. They are presented below.

4.3.2 Ergonomics

The ergonomic facet of an interface is fundamental. Interfaces are the bases upon which users will build the meanings needed to manage their cooperative activity. One of our main objectives was to allow the children to grasp the dynamics of cooperation. This had to require little cognitive effort when children looked at the interface icons. In the first two experiments, the pupils have trouble interpreting the icons and understanding the situations they encountered. Then difficulties arose primarily from ergonomic problems with the first two versions, ToxicFarm and Coopera 1 (Figs. 1, 2, 3). Neither of these versions provided a concrete view of the file-sharing model. The only visual artifact that showed the cooperative activities was the color code. It was too abstract and required a lot of cognitive effort to be interpreted. This was further complicated by the complex architecture of the pages and by an organization of icons that was not very explicit, although this point had been improved in Coopera 1. However, the user-friendliness of Coopera 1 was worse than that of ToxicFarm. A planet metaphor was used for navigation, but it did not help in understanding the cooperative activity. This confused the users. They were not the ones moving from space to space since the moving objects were files.

The modifications made to ToxicFarm to produce Coopera 1 were mostly translations and changes in the interface so that it would appear more attractive to children. Coopera 2 was the result of deeper changes in the interfaces. Homogeneity was enhanced. The new view of the cooperative activity was much more meaningful for children, and the use of the system was greatly simplified: all commands appeared on the same interface, in a Java- rich client that had only been used for synchronization in the previous systems.

4.3.3 Cultural aspects

The cooperation dynamics cannot be separated from the emergence of group awareness, which results mainly from the ability of actors to think about the place, role, and activities of all members of the project. More precisely, group awareness is characterized by the ability to discern intentions and expectations of others' actions toward oneself, and by intentional behaviors toward others. Putting children in this relational dynamic situation requires translating the system properties into signs. This translation was expressed when the children read the interface icons. They had to not only understand their primary meanings (common space for blue) but also relate them to their meanings in the

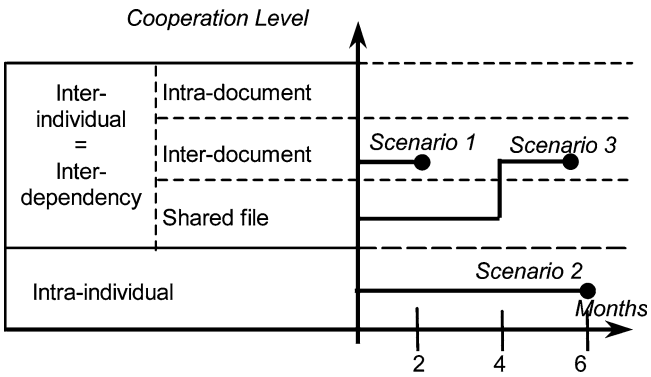


Fig. 6 Cooperation level of the teaching scenarios

cooperation dynamics. When they converted properties into signs, the users were not interacting with the system but with other users through the system.

This translation process was a problem for Coopera. It was observed during the first experiment with ToxicFarm, which led us to set up animations. The purpose of the animations was to allow the children to appropriate the relationship to themselves and to others through Coopera and to grasp the file-sharing model that supported this relational model. The animations took place in the classroom with all the children. The concepts in the system were imitated with real physical objects such as tables (representing spaces) or sheets of paper (representing documents), which the children could manipulate during the animation. This setup (inspired from the Vygotskian theories) was chosen based on the fact that concept acquisition depends on how the concept is experienced. This experience must be not only cognitive, but also physical and emotional.

4.3.4 Teaching scenario

The teaching scenario is an important part of tool usability. The task to be accomplished has an impact on the motivation and satisfaction of users when they work on the system. This in turn has an impact on system appropriation. Other important aspects of teaching scenario are the time frame of the project and the level of cooperation induced by the scenario, which has an impact on the relationship between the users. This in turn affects the relational dynamic acculturation carried by the groupware system.

Regarding this last point, we defined several models that could be set up in the system, each with a different level of cooperation (Fig. 6). In the first model, called intra-individual, each user works alone and there is no cooperation between users. The second model, called inter-individual/shared file, is the first level of cooperation. Users are grouped into teams that share the same file but there is no production dependency. The third model is called inter-individual/inter-document. It is the second level of cooperation. Users are grouped into teams that share the same file, and there is a production dependency. The last level is called inter-individual/intra-document. In this highest level of cooperation, users in a class are grouped into teams. They share the same file and produce the same document.

The scenario of the third project (Coopera 2) was the most appropriate for cooperation among the fifth grade children. They had enough time to get accustomed to the system and do their job. They were interested in the goal of the project and were motivated to cooperate. Specific constraints related to the way the classes were organized had to be taken into account. The first one was the frequency of use. The children used the platform only once a week. The highest degree of cooperation that could be set up was the creation of the interdependency between the productions of groups on the same team. The second constraint was that all children had to use the system within a given time. Interdependency between productions could only be set up as a second step.

5 Discussion

The main difficulties faced by children were related to the execution of the cooperation dynamics. They did not perceive the others' intentions intuitively. Moreover, they did not generate intentional activities towards the others. These difficulties resulted mostly from the historical–cultural factor. The cooperation scheme proposed by the platform broke away from their general knowledge of group work. It is not common, at least not in France, to work on the same files as others and to allow others to work on a personal production. Even with the classroom animations, the children needed several sessions of working with the system to learn this relational mode. Indeed, they had to start working with the fundamental functions of the platform before intentional behavior emerged and group awareness was achieved. These functions were synchronize, publish (commit) and update.

At this level, ergonomics is important. It is fundamental that the pupils be able to use these functions with a minimal cognitive load. Besides, the teacher's role was crucial in the generation of group awareness. During the sequences, the teacher was the one to explain to the children the intentional behaviors they had to perceive and adopt. He would allow them to internalize these intentions. Regarding this point, the pedagogical merits of the platform are not limited to the support provided to produce a common result. It forced the children to be less centered on themselves and to think reflexively about their place and their actions in the course of a cooperative project. Thus, group awareness was not a sine qua non condition to use the platform. It was a goal to reach through its use. The platform allowed the children to get accustomed to the functions needed to set up a cooperative dynamic and to explain to them the relational mode supported by the system. They also helped to prepare teachers for their role in the children's appropriation of this relational mode and thus of cooperative behavior.

Furthermore, we observed a break between the mode of cooperation supported by the platform and the effective cooperation of the pupils. Then we state that the implementation of the dynamics of conversation does not rest solely on the technical and ergonomic properties of the computer environment, but on the whole instrumental device set up.

Three devices can be highlighted that make up this instrumental device (Fig. 7). The first is the ''teaching-scenario device.'' It is obviously composed of not only the teaching scenario, which has been co-elaborated with the teachers, but also the instruments used to introduce it to the pupils. The second is the

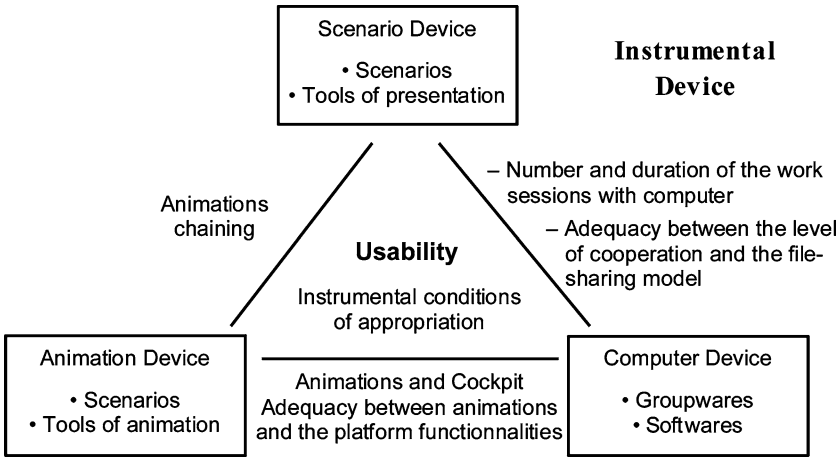


Fig. 7 Articulation between the three devices that compose the instrumental device

"computer device." It concerns the cooperation platform and the software used by the users during their activities. The third is the "device of animation," composed of two animations in the Coopera project. One is mainly concerned with the file-sharing model; the other is mainly centred on the appropriation of the mode of cooperation. This computer device also includes the instruments used for these animations.

In fact, it is the articulation between these three devices which is fundamental, more than the devices themselves. This is why our proposal is to extend the concept of usability to the instrumental device as a whole, i.e., the three devices and their relationships, so that the efficiency of the cooperation depends on the variables that link the three devices.

Two variables result from the relationship between the teaching scenario device and the computer device. The first concerns the number and the duration of the sessions of work with the computer and the second concerns the relevance of the mode of cooperation. The last point concerns the definition of the relationship between the level of cooperation of the teaching scenario (see Fig. 6) and the file-sharing model. Thus, the relevance of the mode of cooperation will be high if the pupils are strongly committed to cooperate.

The variable resulting from the relationship between the animation device and the computer device is related to the good adequacy between the form and the contents of animations on the one hand and between the form and the functionalities of the cockpit on the other. This relationship is very important since it allows the users to convert the properties of the cockpit into signs. In other words, the good adequacy between the animations and the properties of the cockpit allow the mobilization of the acquired knowledge during the animations, when the cockpit is used by the pupils.

Lastly, the variable resulting from the relationship between the teaching scenario device and the animation device is related to the sequence of animations in the scenario. According to the Vygotskian theories, any cognitive function must be handled in a certain way, to be acquired. Then we can state that the

animation relating to the file-sharing model must take place before the animation relating to the mode of cooperation.

When considered separately, each variable makes neither the appropriation of the mode of cooperation by the pupils nor the file-sharing model, which supports it, possible. The adequacy between the animations and the features of the cockpit are essential to support the conversion of the properties of this cockpit into signs. This conversion is what we call appropriation, and we highlighted it with our dialogical analysis of interactions, based on constructivist theories. This led us to program the animations during which the pupils have internalized two kinds of knowledge, first the properties and the functionalities of the file-sharing model and second the mode of cooperation. Co-intentionality depends on this knowledge.

6 Conclusion

In this paper, we have analyzed a conversational sequence that occurred during a work session with a cooperation platform for children under design. We have shown that the children knew how to use the platform. They knew how to execute the commands. They were able to participate in a group activity. However, they still had difficulty relating their actions to the file-sharing model. The methodology we used, based on a dialogical approach of analysis of conversations, helped us to identify these problems for three main reasons. First, the observation and analysis setup was centered on a real situated activity of children. Second, they were encouraged to talk and could therefore express their feelings. Third, we allowed them to physically experiment with the concepts of the platform during preliminary animations. These points showed us that to analyze a cooperative situation, we cannot limit ourselves to the ergonomic point of view. Ergonomics, culture and scenario are intricately connected. Understanding the cooperative activity is possible only if these three factors and their relationships are considered simultaneously.

In some ways, our work pertains to appropriation (Dourish 2003): first, because we took an interest in how the structure of the technology supported the cooperation between the children and second, because our methodology allowed us to analyze the social and cultural aspects of the platform and understand their consequences for its technical design. Of course, this methodology is costly and time consuming and requires lengthy analysis. It is not the most reactive method and we will have to convince industrial partners that time is not the enemy. But, combined with a flexible software development methodology, the gain certainly meets expectations. So we used a dialogical analysis of conversations to evaluate the computer environment under design and, then, make some recommendations for the development of this computer environment.

Finally, we are aware that our proposal requires more work and that further efforts are still needed to model the relationships between the three highlighted factors. However, we hope to have convinced the reader that our proposal for usage analysis is a relevant one. In future studies, we plan to (1) propose formalized criteria for the evaluation of such platforms, (2) contribute to the development of those platforms based on such evaluations and (3) promote the situated-action paradigm for use analysis. It seems difficult to simultaneously explore the three dimensions presented within a single experimental framework,

but actual use in a real situation must be observed if we want to contribute to the development of the usable and acceptable tools.

Acknowledgments Many thanks to Jeriko, our industrial partner, Pascal Pierre, IT advisor for our regional school council, and of course to the teachers, directors, children in the schools who agreed to contribute to these experiments (Louis-Pergaud, Laxou, Jules Ferry and St Pierre, Nancy, and La Sapinière, Toul), and to Didier Paquelin who worked with us at the University of Bordeaux 3.

References

Bastien C, Scapin D (1993) Ergonomic criteria for the evaluation of human–computer interfaces. Institut National de Recherche en Informatique et en Automatique, Paris

Brassac Ch, Grégori N (2001) Situated and distributed design of a computer teaching device. Int. J Des Sci Technol 8(2):11–31

Crabtree A (1998) Ethnography in participatory design. Proceedings of the participatory design Conference, Seattle, WA, 12–14 November, pp 93–105

Dourish P (2003) The appropriation of interactive technologies: some lessons from placeless documents. Comput Support Coop Work 12:465–490

Dourish P (1995) Parting of the ways: divergence, data management and collaborative Work. In: ECSCW'95

Dourish P, Bellotti V (1992) Awareness and coordination in shared workspaces. In: CSCW'92

Feiler PH, Downey GF (1990) Transaction-oriented configuration management: a case study. CMU/SEI-90-TR-23 ESD-90/TR-224

Godart C, Rabhi F, Oster G, Skaf-Molli H, Perrin O, Molli P, Ray P (2004) The ToxicFarm integrated cooperation framework for virtual teams. Distrib Parallel Databases 15(1): 67–88

Gutwin C, Roseman M, Greenberg S (1996) A usability study of awareness widgets in a shared workspace groupware system. In: CSCW'96

Hutchins E (1995) Cognition in the wild. MIT Press, Cambridge

Kaptelinin V, Nardi B (2003) Post-cognitivist hci: second wave theories (panel report). In: CHI 2003, Ft. Lauderdale, FL, 5–10 April

Kuuti K, Bannon LJ (1993) Searching for unity among diversity: exploring the ''interface'' concept. In: Proceedings of INTERCHI'93, 24–29 April, pp 263–268

Levinson S (1983) Pragmatics. Cambridge University Press, Cambridge

Nardi B, Gilbert N, Mantei M, McCarthy J (1993) Mixing oil and water? Ethnography versus experimental psychology in the study of computer-mediated communication. In: Proceedings of INTERCHI'93, 24–29 April, pp 3–6

Searle JR, Vanderveken D (1985) Foundations of illocutionary logic. Cambridge University Press, Cambridge

Suchman L (1987) Plans and situated actions. The problem of man–machine cooperation. Cambridge University Press, Cambridge

Trognon A, Brassac C (1995) Formalizing the theory of intentionality. J Pragmatics 23:555–562

Vanderveken D (1990) On the unification of speech-acts theory and formal semantics. In: Cohen P, Pollack M, Morgan J (eds) Intentions in communication. MIT Press, Cambridge

Vinck D, Jeantet A (1995) Mediating and commissioning objects in the sociotechnical process of product design: a conceptual approach. In: MacLean D, Saviotti P, Vinck D (eds) Management and new technology: design, networks and strategies. COST Social Sciences Serie, Bruxelles, pp 111–129

Vygotsky LS (1978) Mind in society: the development of higher psychological processes. Harvard University Press, Cambridge

Month 200X, Vol.21, No.X, pp.XX–XX *J. Comput. Sci. & Technol.*

Sphere Based Isolation for Cooperative Processes

Adnene Guabtni[1], François Charoy[1], and Claude Godart[1]

[1] *LORIA - INRIA - CNRS, BP 239, 54506 Vandouvre-lès-Nancy Cedex, France*

E-mail: {guabtni, charoy, godart}@loria.fr

Received MONTH DATE, YEAR.

Abstract Supporting cooperative work with business process technology is still a challenge nowadays. Processes are composed of activities performed by people distributed in time and space that concurrently access common data. Thus cooperative processes are faced to concurrency management problem. Isolation strategies developed in the database domain provide solutions to ensure correctness of concurrent manipulations of data. These solutions introduce constraints that are not compatible with a cooperative setting. Flexibility of isolation strategy has been introduced using SQL isolation levels. This solution is not adapted in our case. Isolation in SQL isolation levels concerns data and do not take into account process activities and their relation such as the cooperation between them. In this article we try to adapt isolation levels to the cooperative dimension of processes. The solution we propose is inspired from the sphere of control proposed by Davies (1976). First we identify different phenomena that happen during cooperation in cooperative processes. Then we propose a solution based on what we call "isolation spheres" to ensure correctness of cooperative processes and customise the exclusive control of the different cooperation phenomena.

Keywords Isolation spheres, isolation levels, cooperative processes, cohesion, coherence, serialisability

1 Introduction

For several years now, there are attempts to use process technology to coordinate cooperative activities. This requires adapting it to the specific needs of group work and to the coordination of very interactive, long running, goal oriented processes. This requires also providing some guarantees regarding the outcome of the work.

Cooperative activities or processes have specific characteristics and ¿needs that have been described many times in the literature [8, 6, 13, 11, 4]. They last longer, may change during there execution and communicate through data exchange. Current workflow systems do not support all these requirements.

In this work, we consider a specific dimension of process : the transactional dimension of processes. Considering a cooperative process as a long term transaction is an old idea that has lead to many work and propositions [17]. However, we argue that the results of this work has still drawbacks. Even if some work has been done to cover atomic-

ity and isolation in transactional process [16] but it requires to enforce some specific structural constraints on the process itself. Enforcing a given criteria for an entire process is not very flexible. It forces the process designer to take decision on the process design based on the transactional criteria rather than on the way the work should be done. Moreover, most of the time, only part of a cooperative process requires these kind of constraints.

In this paper, we consider a process as being the concurrent execution of activities with various requirements regarding atomicity and isolation. We propose to enforce isolation and/or atomicity properties to subset of the activities of the process.

2 Motivation

Advanced transactional models have been defined to cover the needs for correct business process execution [5]. In this context, a process is considered as being a transaction with a long execution time. Activities of the process are then considered as traditional transactions with a short execution time. Activities are usually considered as atomic database transactions with their associated properties (atomicity, isolation). Thus a business process is considered from a transactional point of view as a long term transaction composed of short duration activities. Transactional properties attached to the business process are the same during all the process execution. Moreover, the transactional nature of a process is often dependent on its structure and on the activities themselves [2]. This may not be useful and put many constraints on the process designer. In fact, if we consider the current transactional workflow models (see the transactional workflow taxonomy [7]), the process designers must take into account transactional requirements during the design of their workflow. We argue that the process should be defined for the users and not for the transactional monitor. Our motivation is that transactional behaviour should be defined separately from the workflow design and adapted to the process dimension.

Adaptation of transactional behaviour to workflow processes has been already done for atomicity [3], [18]. Isolation in workflow processes has been considered in a recent past [15] and flexibility was carried out on this matter (Contracts [14] and Coo [8]) but has never been generalised to cooperative workflow processes and cooperative parts of processes.

In this article we consider a process as being the concurrent execution of activities which can have various constraints regarding isolation. Usually, isolation in workflow systems is ensured by database systems. Those systems generally use standard ANSI SQL [1] to define the isolation's constraints of a transaction. The problem lies in the fact that these isolation's constraints cannot always satisfy those of workflow process activities. Isolation of process activities must take into account the process organisation and workflow transaction monitors don't permit that today. The following example describes the problem that may occur during a cooperative execution.

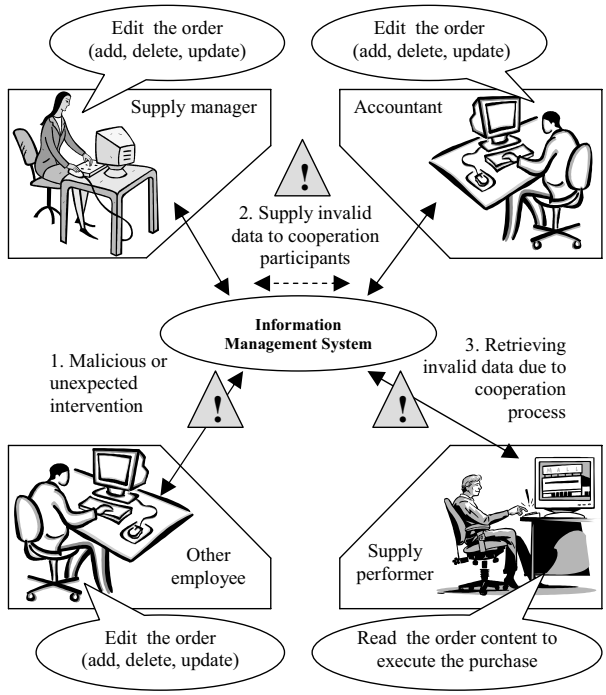


Fig.2. Motivating example

We assume a representation of a cooperative situation in a company concerning two persons working together on the elaboration of an order to purchase some products with some constraints on the number of each product, the total price of the order, the product types, the dependency relation between products etc. The order edition process consists in the work of the supply manager and the accountant. These two persons cooperate together and try to edit the order document with respect to dependencies constraints, types of products, quantities or prices. Problems that can happen while they cooperate are illustrated in figure 1 and consist of three main classes :

1. Malicious or unexpected action is performed by entities out of the cooperation : This is the case with an employee that introduces some modifications in the order document without being part of the cooperation between the supply manager and accountant. This action may induce a lack of correctness in the syntax or the semantic of the order document or simply induce a modification of the order.

2. Participant entities supply invalid data to cooperation participants : This is the case when one cooperation participant (the supply manager or the accountant in this example) delivers invalid data to the rest of the cooperation participants. For example, while the accountant is adding products to the order, the supply manager reads the current order content without worrying if the accountant has finished its edition or not. This kind of execution induces some lack of coherence in the order data.

3. Entities out of the cooperation retrieves invalid data produced by the cooperative activities : This is the case when cooperation is not clearly defined in terms of space and time. Space denote the relationship of the participants to the cooperation and time denotes the duration of the cooperation, its termination and the result publication time. The supply performer is outside the cooperation process and don't know what exactly happens between the supply manager and the accountant. In this case of problems, he retrieves the order content before the completion of the cooperation process and uses it in the purchase process. So the supply performer will not purchase the correct order.

We propose Isolation Spheres as a solution to

that problem. At design time and even at runtime, we can specify *a priori* which are the activities sharing the same data and which ones should be protected against concurrent access. We want to allow the workflow designer to decide on the level of isolation necessary and sufficient for these isolation spheres.

In this work, we first identify cooperation correctness needs in cooperative processes. Then we describe the isolation spheres approach to manage cooperation needs. In the next section, we develop the isolation sphere approach as a general isolation strategy for workflows, and then we identify concurrent data access problems as cooperation phenomena in cooperative processes. Finally we describe how our isolation spheres approach allows handling these problems.

3 Isolation Spheres

Isolation Spheres are inspired from Davies spheres of control [12]. A process is defined as a set of activities with start/end dependencies between these activities. An isolation sphere is defined as a subset of the activities of the process. For these activities we want to ensure some properties regarding data accessed by the activities (Cohesion property of a sphere) and data produced by the activities (Coherence property of a sphere).

Cohesion means that all activities of the sphere have the same view on the data they access. Updates done by activities outside of the sphere must not be visible by the activities of the sphere. This common view represents the basis for cohesion of a group of activities. Cohesion is expressed through different cohesion levels [9]: Read Uncommitted, Read Committed, Repeatable Read and Serialisable. These levels define the way the common view of the sphere on data is managed.

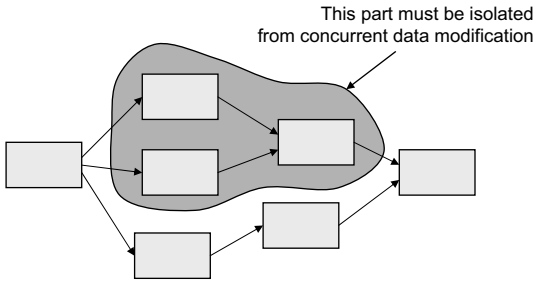


Fig.1. An isolation sphere

- **Read Uncommitted** level allows the sphere to use uncommitted values both at the start-up data view and the intermediate views during the execution of the sphere.
- **Read Committed** level allows only reading committed values also both at the start-up data view and the intermediate views etc.
- **Repeatable Read** level allows activities of the sphere to read values of data with the guarantee that during their use of the data, it will not be changed by an activity outside of the sphere.
- **Serialisable** level emulates a serial execution of the sphere with any other concurrent activity or sphere.

Coherence of a sphere represents how activities of the sphere share their data with activities outside of the sphere. In order to control the coherence between data used by activities of the sphere and

those by the rest of the processes including concurrent isolation spheres, it is essential to define a level of coherence of the sphere. Isolation spheres ensure some cohesion inside the group and also some coherence of the activities external to the sphere using the same data. The levels of coherence are the following:

- **Atomic coherence :** All values of data written by the activities of the sphere are visible outside of the sphere.
- **Selective coherence :** Only **validated** values written by the activities of the sphere are visible outside of the sphere.
- **Global coherence :** Only the **last validated** value written by the **last activity** of the sphere is visible outside of the sphere.

3.1 Cooperation phenomena

In order to identify phenomena induced by cooperative processes, we focus on the two main properties presented by isolation spheres that are cohesion and coherence. From a cohesion point of view, phenomena are those performing perturbations to the cooperation progress and then to the cohesion of the group cooperatinb. This is the Cohesion Problematic. From this point, we say that the two problems pointed out in the example are cohesion problems.

3.1.1 Cohesion

The first class of problems named 'Malicious or unexpected action' comes from one phenomenon that we call 'Disrupted Cooperation' as follows :

- **Disrupted Cooperation :** *Two activities A1 and A2 cooperate over an isolation sphere using concurrently data D. They use a value of D written during their execution by an activity not part of the sphere.*

  Activities inside the sphere reading values of the data on which they cooperate can be induced by mistake if that data is updated outside the sphere. In this case, the " outside cooperation " data update is not supervised.

The second class of problems untitled 'Supply invalid data between cooperation participants' provides three phenomena as follows :

- **Dirty Read Cooperation :** *Two activities A1 and A2 cooperate inside an isolation sphere using concurrently data D. A1 writes a value of D, A2 read it before the completion of A1 and A1 rollbacks.*

  This is similar to the Dirty Read phenomenon in Database world but in this case it is limited to the cooperation environment.

- **Fuzzy Read Cooperation :** *Two activities A1 and A2 cooperate over an isolation sphere using concurrently data D. A1 Reads a value of D, A2 write a new value of D before the completion of A1. So the work of A1 is wrong because it uses an out of date value.*

  This is similar to the Fuzzy Read phenomenon in Database world but in this case it is limited to the cooperation environment.

- **Phantom Read Cooperation :** *Two activities A1 and A2 cooperate over an isolation sphere using concurrently data of a database*

table. A1 requests the database with " where like " conditions. A2 adds a new row to the table before the completion of A1 so that A1 uses data not up to date.

This is similar to the Phantom phenomenon in Database world.

Each one of the cohesion phenomena is illustrated using examples of execution schedules in figure 3

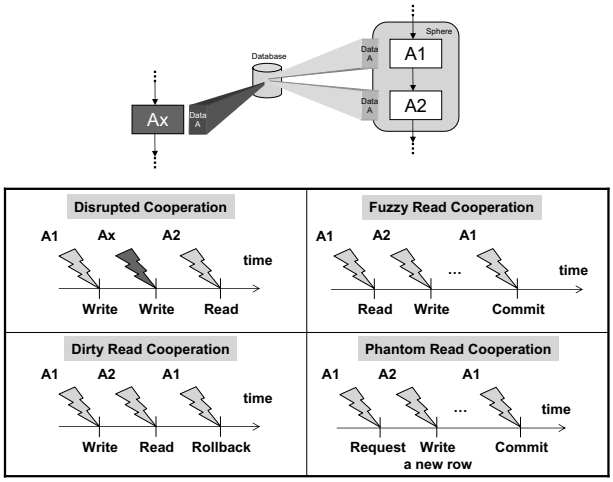


Fig.3. Cohesion phenomena and their corresponding execution schedules

3.1.2 Coherence phenomena

From a coherence point of view, phenomena are those performing perturbation of entities out of the cooperation but caused by the cooperation progress. This is the Coherence Problematic. We can realise that the third class of problems untitled 'Retrieving invalid data due to cooperation' and detected in the motivating example take part of the coherence phenomena and represents problems reverberated on entities out of the cooperation. The responsibility of cooperation participants in such problems consists in a lack of vigilance about delivery of invalid data or valid data but not permanent. Thus, this problem class provides two phenomena as follows :

- **External Dirty Read :** *A cooperation over a sphere inducing concurrent access to data D permits public access to uncommitted values of D written by one activity of the sphere.*

  Activities outside the sphere reading value not yet committed by an activity of the sphere can be induced in mistake if the activity that written the uncommitted value is rolled back. That's why we call this phenomenon as " External Dirty Read ".

- **External Misleading Read :** *Cooperation over a sphere inducing concurrent access to data D permits public access to each committed value of D written by one of the cooperative activities.*

  Activities outside the sphere reading values committed by activities of the sphere will consider that it's the result of the cooperation because it's committed. That's why we call this phenomenon as " External Misleading Read ".

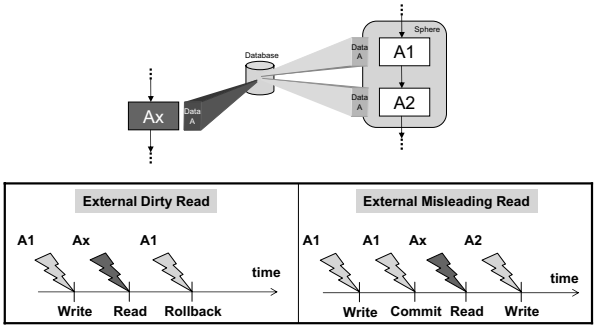


Fig.4. Coherence phenomenon and their corresponding execution schedules

Each one of the coherence phenomena is illustrated using examples of execution schedules in figure 4

4 Cooperative Process Correctness Using Isolation Spheres

Isolation spheres introduce isolation in the process management. Our study about this approach disclosed two dimensions : Cohesion and Coherence. The meaning of each level of these dimensions is described in [9]. The duality Cohesion/Coherence of an isolation sphere, as illustrated in figure 5, expresses the choice of each level (cohesion and coherence) depending on the needs expressed by the process designer. Depending on this choice, these levels induce more or less flexibility for the cooperative data exchanges (cohesion level) and more or less risk of divergence or incoherence (coherence level). In some cases, the process designer can accept the fact that some activities will use invalid data or not up to date data. That is the goal of this sphere level based design.

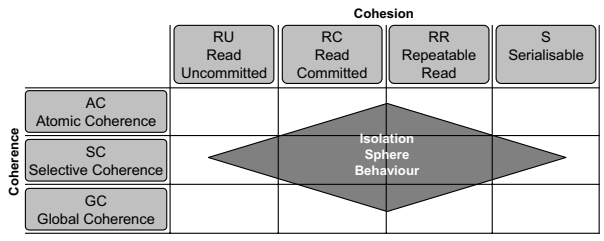


Fig.5. Duality Cohesion/Coherence : customised isolation behaviour

Applying these levels to cooperative processes allow the process designer to set the appropriate levels of cohesion and coherence for a given sphere. Figures 6 and 7 illustrate the matching between Cohesion and coherence levels with the phenomena based on cohesion and coherence problems. The process designer should take decisions about cohesion and coherence with a problem based approach build on the three main problem classes defined at the beginning of this section. This approach allows the designer to use the best adapted level to the situation.

	Phenomena	Disrupted Cooperation	Dirty Read Cooperation	Fuzzy Read Cooperation	Phantom Read Cooperation
No Isolation Sphere		Yes	Yes	Yes	Yes
Isolation Sphere	**Cohesion level**				
	Read Uncommited	No	Yes	Yes	Yes
	Read Commited	No	No	Yes	Yes
	Repeatable Read	No	No	No	Yes
	Serializable	No	No	No	No

Fig.6. Cohesion levels effects on cooperative phenomena

	Phenomena	External Dirty Read	External Misleading Read
Isolation Sphere	**Coherence level**		
	Atomic Coherence	Yes	Yes
	Selective Coherence	No	Yes
	Global Coherence	No	No

Fig.7. Coherence levels effects on cooperative phenomena

Cooperation phenomena →		Disrupted Cooperation	Dirty Read Cooperation	Fuzzy Read Cooperation	Phantom Read Cooperation	External Dirty Read	External Misleading Read
No Isolation Sphere		yes	yes	yes	yes	yes	yes
Cohesion level	**Coherence level**						
Read Uncommited	Atomic	no	yes	yes	yes	yes	yes
Read Commited	Atomic	no	no	yes	yes	yes	yes
Repeatable Read	Atomic	no	no	no	yes	yes	yes
Serializable	Atomic	no	no	no	no	yes	yes
Read Uncommited	Selective	no	yes	yes	yes	no	yes
Read Commited	Selective	no	no	yes	yes	no	yes
Repeatable Read	Selective	no	no	no	yes	no	yes
Serializable	Selective	no	no	no	no	no	yes
Read Uncommited	Global	no	yes	yes	yes	no	no
Read Commited	Global	no	no	yes	yes	no	no
Repeatable Read	Global	no	no	no	yes	no	no
Serializable	Global	no	no	no	no	no	no

Fig.8. Duality Cohesion/Coherence : customised cooperation behaviour

The duality Cohesion/Coherence allows us to customise the isolation strategy with twelve combinations as illustrated in figure 8. These combi-

nations illustrate all the flexibility introduced regarding isolation in general and especially to support cooperation. These possibilities can express the maximum degree of cooperation flexibility but without strong data safety (Read Uncommitted cohesion level with an Atomic coherence level) or a strong data safety without any cooperative behaviour due to serialisability (Serialisable cohesion level with Global coherence level). The intermediate combinations take into account at the same time, a level to ease the cooperation work and the data exchanges between cooperation participants and another level to ensure some execution correctness degree.

The contribution of isolation spheres in terms of correctness is the choice of which cooperation phenomena to allow and which to disallow. The process designer can be sure that the cooperation process will never accept what he disallowed using isolation spheres. This kind of correctness control introduces a high flexibility level. Also, the combination concerning a sphere can be adapted during the execution to new needs and constraints following two dimensions : the levels of isolation (cohesion and coherence) and the composition of the sphere (activities that join the cooperation group and those that leave it). Modifications performed at runtime need to be coherent : if some event occurs and has been accepted by the isolation sphere levels, new levels updated by the designer at runtime must at least accept this event. The goal of this constraint is to ensure compatibility between isolation sphere levels for cooperative correctness.

5 Conclusion

In this work, we have proposed to introduce process dimension in isolation strategy to ensure correctness of execution in case of cooperative activities (concurrent access to common data). Our approach is based on what we call Isolation Spheres and was inspired from the Spheres of Control [12]. Cooperative processes lead to many problems regarding their transactional behaviour. In this work we identified three main classes of problems and six phenomena that can happen during a cooperative process. Based on the Isolation Spheres approach, we tried to match what isolation spheres can ensure with what cooperative processes should ensure. The result is that isolation spheres deliver a complete support for cooperation phenomena exclusion with twelve levels in order to allow the designer to choose the balance between correctness of cooperation and flexibility in data access. Using isolation spheres for cooperative processes, the designer apply a separation of concerns between process definition and transaction requirements. Flexibility in transaction requirements definition is based on the multiple combinations of cohesion and coherence levels and can also be adapted during the execution.

We have already studied the problem of integration of isolation spheres to the architecture of BPEL (Business Process Execution Language) for web services platforms [10].

This work needs more advanced studies especially in terms of correctness criteria of execution schedules and some possible algorithms to correct the execution in fault cases at runtime. Such prob-

lems are important parts of the future work and will induce the elaboration of a effective integration of isolation spheres inside real workflow management systems and BPEL engines. A third problem concerns changing levels at execution time and changing sphere composition and their impact on execution.

References

[1] Ansi x3.135-1992, american national standard for information systems - database language - sql. November 1992.

[2] Gustavo Alonso, Divyakant Agrawal, Amr El Abbadi, Mohan Kamath, Roger Günthör, and C. Mohan. Advanced transaction models in workflow contexts. In Stanley Y. W. Su, editor, *Proceedings of the Twelfth International Conference on Data Engineering, February 26 - March 1, 1996, New Orleans, Louisiana*, pages 574–581. IEEE Computer Society, 1996.

[3] Wijnand Derks, Juliane Dehnert, Paul Grefen, and Willem Jonker. Customized atomicity specification for transactional workflow. In *Proceedings of the Third International Symposium on Cooperative Database Systems for Advanced Applications (CODAS'01)*, pages 140–147. IEEE Computer Society, 2001.

[4] Schahram Dustdar. Caramba - a process-aware collaboration system supporting ad hoc and collaborative processes in virtual teams. *Distributed and Parallel Databases*, 15(1):45–66, 2004.

[5] A.K. Elmagarmid, editor. *Database transaction models for advanced applications*. Morgan Kauffman, 1992.

[6] D. Georgakopoulos, H. Schuster, D. Baker, and A. Cichocki. Managing escalation of collaborative processes in crisis mitigation situations. *16th Int. Conference on data Engineering (ICDE'2000)*, 2000.

[7] Paul W. P. J. Grefen. Transactional workflows or workflow transactions? *DEXA*, pages 60–69, 2002.

[8] Daniela Grigori, François Charoy, and Claude Godart. Coo-flow: a process technology to support cooperative processes. *International Journal of Software Engineering and Knowledge Engineering - IJSEKE Journal*, 14(1), 2004.

[9] Adnene Guabtni, François Charoy, and Claude Godart. Spheres of isolation: adaptation of isolation levels to transactional workflow. In Wil M.P. van der Aalst et al., editor, *Business Process Management, 3rd International Conference, BPM 2005, Nancy, France*, volume 3649 of *Lectures Notes in Computer Science*, pages 458–463. Springer, September 2005.

[10] Adnene Guabtni, François Charoy, and Claude Godart. Concurrency management in transactional web services coordination. In S. Bressan, J. Küng, and R. Wagner, editors, *Database and Expert Systems Database and Expert Systems Applications, 17th International Conference, DEXA 2006, Krakow,*

Poland, volume 4080 of *Lectures Notes in Computer Science*, pages 592–601. Springer, Sep 2006.

[11] Jörg M. Haake and Weigang Wang. Flexible support for business processes: extending cooperative hypermedia with process support. In *GROUP '97: Proceedings of the international ACM SIGGROUP conference on Supporting group work : the integration challenge*, pages 341–350. ACM Press, 1997.

[12] Charles T. Davies Jr. Data processing spheres of control. *IBM Systems Journal 17(2): 179-198*, 1978.

[13] Peter Mangan and Shazia Sadiq. On building workflow models for flexible processes. In *Proceedings of the thirteenth Australasian conference on Database technologies*, pages 103–109. Australian Computer Society, Inc., 2002.

[14] Andreas Reuter and Friedemann Schwenkreis. Contracts - a low-level mechanism for building general-purpose workflow management-systems. *IEEE Data Eng. Bull.*, 18(1):4–10, 1995.

[15] Heiko Schuldt, Gustavo Alonso, Catriel Beeri, and Hans-Jörg Schek. Atomicity and isolation for transactional processes. *ACM Trans. Database Syst.*, 27(1):63–116, 2002.

[16] Heiko Schuldt, Gustavo Alonso, Catriel Beeri, and Hans-Jrg Schek. Atomicity and isolation for transactional processes. *ACM Transactions on Database Systems (TODS)*, 27(1):63–116, 2002.

[17] Amit P. Sheth and Marek Rusinkiewicz. On transactional workflows. *Data Engineering Bulletin*, 16(2):37–40, 1993.

[18] Willem-Jan van den Heuvel and Sergei Artyshchev. Developing a three-dimensional transaction model for supporting atomicity spheres. *International Workshop on Web Services Research, Standardization, and Deployment*, 2002.