Sphères de comportement pour la modélisation et l'exécution de procédés flexibles

Adnene Guabtni

Nancy-Université
Université Henri Poincaré

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

Sphères de comportement pour la modélisation et l'exécution de procédés flexibles

THÈSE

présentée et soutenue publiquement le 07 mai 2007

pour l'obtention du

Doctorat de l'Université Henri Poincaré – Nancy 1
(spécialité informatique)

par

Adnene Guabtni

Composition du jury

Président :	Ye Qiong Song	Professeur, Institut National Polytechnique de Lorraine, France
Rapporteurs :	Fethi Rabhi	A/Professor, University of New South Wales, Australia
	Wojciech Cellary	Professor, Poznan University of Economics, Poland
Examinateurs :	Claude Godart	Professeur, Université Henri Poincaré - Nancy I, France
	François Charoy	Maître de conférences, Université Henri Poincaré - Nancy I, France
	Zohra Bellahsene	Professeur, Université de Montpellier II, France

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

Loria

Résumé étendu

L'objectif de la thèse consiste à proposer une nouvelle approche, appelée "Sphères de comportement" pour la modélisation et l'exécution de procédés flexibles. Il s'agit d'introduire une flexibilité, du point de vue de la modélisation et de l'exécution, dans la gestion de propriétés comportementales avancées telles que les propriétés transactionnelles, opérationnelles ou organisationnelles.

Nous avons constaté que le manque de flexibilité dans la spécification de telles propriétés comportementales est dû principalement à deux raisons majeures. En premier lieu, beaucoup de propriétés comportementales expriment souvent un comportement de groupe et non uniquement un comportement individuel et, bien souvent, les systèmes de gestion de procédés métiers les réduisent à être uniquement d'ordre individuel, ce qui réduit considérablement la flexibilité des procédés. En second lieu, l'architecture des systèmes de gestion de procédés présente, dans la plupart des cas, un modèle de procédé intégrant en son sein la spécification de propriétés comportementales. Cela contraint les concepteurs de procédés métiers à se conformer à la structure et aux contraintes du modèle de procédé existant dans la spécification de propriétés comportementales avancées, ce qui engendre un manque de flexibilité.

C'est grâce à de telles constatations que nous avons emprunté la voie de la séparation des préoccupations entre la spécification du procédé et celle des propriétés comportementales. La séparation claire entre la spécification des éléments de base d'un procédé (les activités le constituant et leurs dépendances fonctionnelles, informationnelles ou temporelles) de celle des propriétés comportementales, et particulièrement celles qui concernent des groupes d'activités, serait garant d'une flexibilité et d'une expressivité accrue des procédés métiers.

La solution que nous proposons dans cette thèse tente de constituer un modèle unifié de représentation de propriétés comportementales de groupe. Elle tire son origine de la notion de "sphère de contrôle" introduite par Davies en 1978 pour la gestion du traitement de données (Data Processing). Depuis, la notion de sphère a été adaptée et utilisée comme entité encapsulant plusieurs activités d'un procédé, pour exprimer de façon plus flexible et assurer de façon plus fiable des propriétés transactionnelles telles que l'atomicité et la compensation, donnant ainsi lieu aux notions de sphères d'atomicité et sphères de compensation. Nous nous proposons dans cette thèse de généraliser la notion de sphère pour introduire celle que nous appelons "sphère de comportement" qui englobe un certain nombre d'activités d'un procédé qui expriment, ensemble, une propriété comportementale de groupe. La sphère de comportement devra en assurer la réalisation fiable et flexible. Nous définissons les principes fondamentaux d'une sphère de comportement afin d'avoir une référence permettant de vérifier si une propriété comportementale de groupe est réalisable ou non en faisant appel à l'approche des sphères de comportement.

Nous nous focalisons ensuite sur le cas particulier des sphères d'isolation qui s'inscrivent dans la lignée des sphères d'atomicité et des sphères de compensation d'ores et déjà définies dans la littérature. Nous détaillons les caractéristiques d'une sphère d'isolation qui s'orientent vers deux dimensions, la cohésion et la cohérence, donnant lieu à deux nouveaux critères de sérialisabilité : l'extra-Sphère-Sérialisabilité et l'intra-Sphère-Sérialisabilité. Nous étudions ensuite l'apport, en

termes de flexibilité et d'expressivité des sphères d'isolation ainsi définies, particulièrement à l'exécution d'activités coopératives.

Enfin, nous proposerons une démarche pour la mise en oeuvre des sphères de comportement dans les plateformes de procédés à base de Web Services, à travers leur modélisation dans la notation BPMN et leur spécification dans le langage BPEL ainsi que la génération de contextes WS-Coordination. L'approche de mise en oeuvre sera dite "de bout en bout" en ce sens que nous partirons de la notation jusqu'à l'exécution du procédé.

Remerciements

Plus que par tradition, je voudrais ici exprimer ma reconnaissance à tous ceux, du laboratoire LORIA ou de l'extérieur qui m'ont aidé par leurs conseils judicieux, par leurs discussions enrichissantes, et l'intérêt qu'ils ont porté à mon travail.

Ma sincère gratitude va tout d'abord à M. Claude Godart, mon directeur de thèse, Professeur à l'Université Nancy I - ESSTIN, pour la confiance qu'il m'a témoignée en m'acceptant dans l'équipe de recherche ECOO qu'il anime, pour avoir encouragé chez moi la prise d'initiative et pour avoir cru en moi.

J'adresse mes vifs remerciements à M. François Charoy, mon co-encadrant, Maître de conférences à l'Université Nancy I - ESIAL, pour son appui dans le choix de mes orientations et son suivi permanent. Par son expérience pointue, il a guidé mes premiers pas dans la recherche, je le suis profondément reconnaissant d'avoir toujours su m'aider, m'encourager et me conseiller en toutes circonstances avec gentillesse et efficacité. Ce fut un réel plaisir de collaborer avec lui.

Je tiens à remercier vivement :

M. Ye Qiong SONG, Professeur à l'Institut National Polytechnique de Lorraine, qui me fait l'honneur de présider ce jury.

M. Fethi Rabhi, A/Professor à l'Université New South Wales en Australie, qui a accepté d'être rapporteur de ma thèse.

M. Wojciech Cellary, Professeur à la Poznan University of Economics, en Pologne, pour avoir accepté d'être rapporteur de ma thèse.

Mme Zohra Bellahsene, Professeur à l'Université de Montpellier II, en France, pour m'avoir fait l'honneur de participer à ce jury.

Je remercie de tout cœur tous les membres de l'équipe ECOO du LORIA, chercheurs permanents, doctorants, ingénieurs, stagiaires et assistantes, pour l'amabilité avec laquelle ils m'ont intégré dans l'équipe, pour l'aide aussi précieuse qu'efficace qu'ils m'ont apportée régulièrement et pour la sympathique attention dont ils m'ont entouré jusqu'à l'achèvement de cette thèse.

Je ne voudrais terminer sans remercier tout le personnel enseignant et administratif de l'école d'ingénieurs ESSTIN où je suis intervenu, durant les quatre dernières années, en tant qu'enseignant moniteur puis en tant qu'ATER, et qui n'ont jamais cessé de m'aider, par leurs conseils, à développer mes capacités pédagogiques.

C'est un grand plaisir pour moi de remercier ici mes collègues et amis, au LORIA et ailleurs, qui m'ont soutenu tout au long de la thèse et plus chaleureusement Mlle Chedia Dhaoui pour ses encouragements incessants et son soutien permanent.

Enfin, je remercie profondément ma famille de m'avoir soutenu et encouragé à aller toujours de l'avant.

Table des matières

Chapitre I Introduction, problématique et objectifs de la thèse **1**

1 Introduction générale et contexte de la recherche **3**

2 Problématique générale et objectifs de la thèse **7**

2.1 Les propriétés comportementales dans les procédés métiers 7

2.2 La flexibilité de la modélisation et de l'exécution de procédés métiers 10

2.3 La séparation des préoccupations 12

2.4 Objectifs de la thèse 13

3 Organisation du mémoire **17**

Chapitre II État de l'art **19**

1 La modélisation de procédés **21**

1.1 Introduction historique et évolution de la modélisation de procédés métiers . 21

1.1.1 Les origines de l'automatisation de tâches dans les systèmes d'information 22

1.1.2 Les procédés de workflow pour la modélisation et l'exécution de procédés métiers 23

1.2 L'ère du Business Process Management BPM 28

1.3 Les besoins en termes de BPM 30

1.4 Conclusion 32

2 Les approches à base de sphères **33**

2.1 L'approche des sphères de contrôle 33

2.2 Du traitement de données aux procédés métiers : différences et nouveaux enjeux 34

2.3 Les propositions existantes à base de sphères pour la gestion de procédés métiers 35
2.3.1 Les sphères d'atomicité . . . 35
2.3.2 Les sphères de compensation . . . 37
2.3.3 Les "poches" de flexibilité (Pocket of Flexibility) . . . 37
2.3.4 Synthèse . . . 38

3 La propriété transactionnelle d'isolation dans les procédés métiers 39

3.1 Les mécanismes de gestion de concurrence d'accès . . . 39
3.1.1 Le concept de transaction . . . 39
3.1.2 Impact de l'exécution concurrente de transactions sur la cohérence des données . . . 40
3.1.3 Les solutions existantes aux problèmes de concurrence d'accès . . . 42
3.2 Les transactions dans les procédés métiers . . . 47
3.2.1 Enjeux de la gestion transactionnelle des procédés . . . 47
3.2.2 Le workflow transactionnel . . . 48
3.2.3 Les transactions à longue durée d'exécution . . . 56
3.2.4 Les MTA : tolérance aux fautes et flexibilité des contraintes transactionnelles dans les procédés métiers . . . 57
3.3 Contraintes sur la coordination d'activités coopératives . . . 61
3.3.1 Qu'est qu'une activité coopérative ? . . . 61
3.3.2 Impact de l'isolation sur la coordination d'activités coopératives . . . 61
3.3.3 Flexibilité de l'isolation d'activités coopératives . . . 62
3.4 Synthèse et conclusion . . . 62

Chapitre III Contributions de la thèse 63

1 Sphères de comportement : vers une nouvelle façon de modéliser les procédés 65

1.1 Contexte de base d'exécution de procédés métiers . . . 66
1.1.1 L'élément de base d'un procédé métier : l'activité (ou tâche) . . . 66
1.1.2 Structure des procédés métiers . . . 67
1.2 Notre vision des sphères de comportement . . . 67
1.2.1 Premier principe : une séparation des préoccupations . . . 69

1.2.2 Deuxième principe : un comportement d'ensemble 70
1.2.3 Troisième principe : une flexibilité et une expressivité accrues 71
1.3 Identification de propriétés inappropriées à notre approche 71
1.4 Identification de propriétés favorables à notre approche 72
2 Proposition d'une solution d'isolation flexible et orientée procédé 75
2.1 Contexte, enjeux et objectifs de l'étude de l'isolation dans les systèmes de gestion de procédés 76
2.1.1 L'isolation dans les systèmes de bases de données transactionnelles . . 76
2.1.2 L'isolation dans les systèmes de gestion de procédés 77
2.2 L'approche des sphères d'isolation 77
2.2.1 Présentation du formalisme utilisé et des concepts de base 77
2.2.2 Éléments de base d'une sphère d'isolation 83
2.3 La cohésion des sphères d'isolation 85
2.3.1 Le niveau de cohésion à lecture non validées (Read Uncommitted) . . 85
2.3.2 Le niveau de cohésion à lectures validées (Read Committed) 86
2.3.3 Le niveau de cohésion à lecture répétables (Repeatable Read) 86
2.3.4 Le niveau de cohésion sérialisable 87
2.4 La cohérence des sphères d'isolation 89
2.4.1 Le niveau de cohérence coopérative 90
2.4.2 Le niveau de cohérence d'activité 90
2.4.3 Le niveau de cohérence de sphère 91
2.5 Gestion avancée des sphères d'isolation : l'emboîtement de sphères 99
2.5.1 Enjeux de l'emboîtement de sphères d'isolation et règles de base . . . 100
2.5.2 Cas de l'Intra-Sphère-Sérialisabilité et l'emboîtement de sphères . . . 101
2.5.3 Cas de l'Extra-Sphère-Sérialisabilité et l'emboîtement de sphères . . . 102
2.5.4 Discussion à propos des possibilités de chevauchement de sphères . . . 103
2.6 Synthèse 104
3 Apports des sphères d'isolation aux procédés coopératifs 105
3.1 Les procédés coopératifs 105
3.2 La controverse de la sérialisabilité dans les procédés coopératifs 106
3.3 Les besoins en isolation flexible dans les procédés coopératifs 107

3.4 Identification des anomalies de coopération 108
3.5 Apports des sphères d'isolation à la gestion flexible des anomalies de coopération 110
3.6 Application des sphères d'isolation à un exemple pratique de procédé coopératif 111
3.7 Synthèse 113

Chapitre IV Mise en œuvre des sphères d'isolation 115

1 De la théorie à la pratique : gestion de la dualité Cohésion/Cohérence 117

1.1 Le protocole CCDP (Cohesion/Coherence Duality Protocol) 117
1.1.1 Les verrous du protocole CCDP 118
1.1.2 Algorithme de gestion de la cohésion et de la cohérence des sphères lors d'une tentative de lecture ou d'écriture de données 118
1.2 synthèse 121

2 Enjeux de la mise en oeuvre des sphères de comportement / d'isolation 123

2.1 Les enjeux de la mise en œuvre 123
2.2 Choix de l'environnement de mise en oeuvre 124
2.3 Les technologies web services "WS-Technologies" 125
2.3.1 WS-Coordination / WS-Transactions 127
2.4 Synthèse 128

3 Sphères de comportement pour la coordination de services web 129

3.1 Proposition d'une nouvelle organisation des types de WS-Coordination 129
3.2 Sphères emboîtées et services d'inscription / désincription du WS-Coordination 131

4 Mise en oeuvre des sphères d'isolation dans une plateforme de web services 133

4.1 La démarche de mise en oeuvre 133
4.2 Le volet notation et spécification 134
4.3 Vers le volet exécution 135
4.4 WS-Sphère, WS-IsolationSphere et le protocole CCDP dans une implémentation de WS-Coordination 139
4.4.1 Architecture de Kandula avec prise en charge des WS-Sphere 139
4.4.2 Implémentation du protocole CCDP 141

4.5 Synthèse 141

Conclusion et perspectives **145**

Annexe **149**

L'évolution de la modélisation et l'exécution de procédés métiers en fonction des standards de BPM **151**

Liste des symboles **157**

Liste des définitions **159**

Bibliographie **161**

Table des figures

1 Un exemple de procédé de workflow pour la gestion des publications scientifiques 4

2 L'opérateur "Multi Merge" pour exprimer l'itération dans les procédés de workflow 9

3 Un procédé de vente de voiture auprès d'un concessionnaire : exemple illustratif de l'atomicité comme propriété comportementale de groupe 11

4 Un exemple de propriété comportementale de type instanciation multiple appliqué à une seule activité 13

5 Une vue historique des systèmes de gestion de workflow 22

6 Le modèle de référence du workflow selon la WfMC 24

7 Le méta-modèle de définition de procédés de workflow selon la WfMC [Wfmc 99] 25

8 Diagramme de changement d'état d'un procédé de workflow 26

9 Diagramme de changement d'état d'une activité d'un procédé de workflow 26

10 Utilisation du formalisme des réseaux de petri pour l'exécution de procédés métiers 26

11 Évolution des standards de la modélisation et de l'exécution de procédés métiers 29

12 Exemple de procédé en notation BPMN 29

13 Exemple de procédé en réseaux de petri annoté par des sphères d'atomicité [Derk 01] 36

14 Équivalent en petri net d'un procédé annoté par des sphères d'atomicité [Derk 01] 37

15 Exemple de sphère de compensation, selon [Leym 95] 38

16 Les conflits entre opérations sur un même objet 43

17 Taxonomie du workflow transactionnel selon Grefen (2002) [Gref 02b] 49

18 L'architecture MWF/MTR 53

19 L'architecture MTR/MWF 53

20 L'architecture MTR+MWF 54

21 L'architecture MTRWF 54

22 L'architecture MWF 54

23 L'architecture MWF 55

24 Répartition des différents travaux sur le workflow transactionnel par rapport aux six classes de la taxonomie proposée [Gref 02b] 56

25 Utilisation de l'approche des sphères de comportement 68

26 Architecture de prise en charge de sphères de comportement 69

27 Cas de l'instanciation multiple respectant le deuxième principe des sphères de comportement 70

28 Cas de la distribution des rôles non conforme au deuxième principe des sphères de comportement 72

29 Sphère de séparation des fonctions 73

30 Sérialisation des opérations de $\mathcal{OP}$ 80

31 Cycle Intermédiaire/Final d'une donnée 81

32 Une sphère d'isolation 84

33 Histoire d'exécution ne respectant pas le niveau de cohésion à lectures non validées 86

34 Histoire d'exécution ne respectant pas le niveau de cohésion à lectures validées . . 87

35 Histoire d'exécution ne respectant pas le niveau de cohésion à lectures répétables 87

36 Histoire d'exécution ne respectant pas le niveau de cohésion sérialisable 87

37 Exemple d'exécution Intra Sphère Sérialisable 88

38 Dualité Cohésion/Cohérence 90

39 Exemple de sphère avec un niveau de cohérence coopérative 91

40 Exemple de sphère avec un niveau de cohérence d'activité 92

41 Un procédé P donnant place à la concurrence d'exécution d'activités 92

42 Exécution sérialisée H_1 : pas de concurrence entre activités 93

43 Introduction d'une sphère d'isolation dans le procédé P 93

44 Exécution sérialisée H_1 et ses limites face aux contraintes d'isolation de la sphère ω. 94

45 Histoire d'exécution coopérative : cycle de conflits entre des parties coopératives et non coopératives du procédé 94

46 Histoire d'exécution avec une coopération effective grâce à un haut niveau de cohérence et à un bas niveau de cohésion 95

47 Exécution concurrente accrue et pas de cycles de conflits dans une exécution coopérative : cohérence maximale 95

48 Exemple d'exécution Extra Sphère Sérialisable 96

49 Exemple d'exécution Intra et Extra Sphère Sérialisable 96

50 L'emboîtement de sphères d'isolation : quelle portée pour la cohésion et la cohérence d'une sphère ? 100

51 La portée de la cohésion et de la cohérence des sphères d'isolation dans le cas de sphères emboîtées 101

52 Intra-Sphère-Sérialisabilité avec des sphères emboîtées 102
53 Sphères emboîtées 102
54 Extra-Sphère-Sérialisabilité avec des sphères emboîtées 103

55 Un exemple de procédés coopératif 106
56 Dualité Cohésion/Cohérence : comportement coopératif à la carte 111
57 Application des sphères d'isolation à un exemple pratique de procédé coopératif . 112

58 Standards utilisés avec BPEL 126

59 Intégration de la spécification WS-Sphere dans la spécification WS-Coordination 130
60 Utilisation de l'extensibilité de la spécification WS-Coordination pour ajouter les propriétés des sphères d'isolation 131
61 Algorithme pour l'enregistrement à une WS-Sphere 132
62 Algorithme pour la désinscription d'une WS-Sphere 132

63 Architecture de mise en oeuvre du passage de BPMN à BPEL 133
64 Les paquetages de formes dans Dia : le cas de BPMN 135
65 Création d'une nouvelle forme pour les sphères d'isolation dans le paquetage de formes BPMN 135
66 Exemple de procédé contenant une sphère d'isolation créée à l'aide de l'outil Dia et du paquetage de formes modifié 136
67 Format d'enregistrement des fichiers Dia 136
68 Format XML représentant les procédés BPMN accepté par l'outil BPMN2BPEL 137
69 Structure résumée du fichier XSL permettant de transformer les fichiers Dia en fichiers au format XML requis par BPMN2BPEL 138
70 Codes dans le fichier XSL permettant de générer des nœuds de type activité (Task) 138
71 Codes dans le fichier XSL permettant de générer des nœuds de type sphère d'isolation 138
72 Architecture de Kandula avec prise en charge des WS-Sphere et WS-IsolationSphere 140
73 WSCoordination et la gestion des lectures/écritures 141
74 Diagramme de séquence suite à une opération de lecture ou d'écriture dans une sphère d'isolation 142
75 WSCoordination et la gestion des lectures/écritures 142

Chapitre I

Introduction, problématique et objectifs de la thèse

1. **Introduction et contexte de la recherche**
2. **Problématique générale et objectifs de la thèse**
3. **Organisation du mémoire**

1

Introduction générale et contexte de la recherche

Les procédés métiers ont été introduits au sein des entreprises afin d'automatiser les procédures de travail. C'est à partir des années 1975 que les systèmes dits de workflow[1], permettant de gérer des procédés métiers, ont connu un essor important. Un workflow [Haye 91, Koul 95] est représenté sous forme d'un graphe, souvent sans cycle, dont les activités (ou tâches) forment les noeuds. La relation entre les activités est exprimée selon deux types d'arcs. Un arc de type "flot de contrôle" permet d'exprimer l'ordonnancement à respecter entre deux activités du procédé tandis qu'un arc de type "flot de données" permet d'exprimer les échanges de données entre deux activités. Quelques éléments supplémentaires peuvent former des noeuds particuliers exprimant des opérateurs logiques (AND split, AND join, OR split ...). Cette vision d'un procédé métier peut, de notre point de vue, constituer une vision basique de ce qu'un procédé représente. La figure 1 illustre un exemple concret de workflow.

Les premiers prototypes de systèmes de workflow étaient basés sur l'automatisation de tâches pour l'entreprise et nous pouvons principalement citer Officetalk [Elli 80], développé par kip Ellis et Gary Nutt chez Xerox, ou encore SCOOP [Zism 77] développé par Michael Zisman chez Wharton. Ces derniers datent des années 70 et constituent les précurseurs des systèmes évolués qui sont apparus plus récemment. La mise sur le marché de systèmes primitifs de workflow fut réalisée dans les années 80. La prolifération de systèmes de workflow de plus en plus évolués a été intense durant les années 90 qui peuvent être considérées comme les grandes années du workflow.

Depuis, plusieurs systèmes de gestion de procédés métiers sont apparus donnant lieu à plusieurs représentations plus ou moins complexes, plus ou moins fiables, et plus ou moins expressives de procédés métiers permettant l'exécution d'activités au sein de l'entreprise. Beaucoup de travaux sur les systèmes de workflow continuent, jusqu'à ce jour, à enrichir ce domaine [Aals 02, Elli 93, Geor 95a, Jabl 96, Koul 95, Leym 00, Mend 05].

De nos jours, un workflow est souvent considéré comme la version opérationnelle d'un procédé métier. En effet, la spécification du workflow, réalisée par la WfMC[2] [WfMC], intègre plusieurs aspects pratiques, jouant un grand rôle dans l'exécution des procédés, ce qui a fait son succès auprès des entreprises. Nous pouvons citer, par exemple, la gestion des rôles, permettant d'affecter

[1] Workflow : traduit littéralement en "flux de travail"
[2] WfMC : Wotkflow Management Coalition

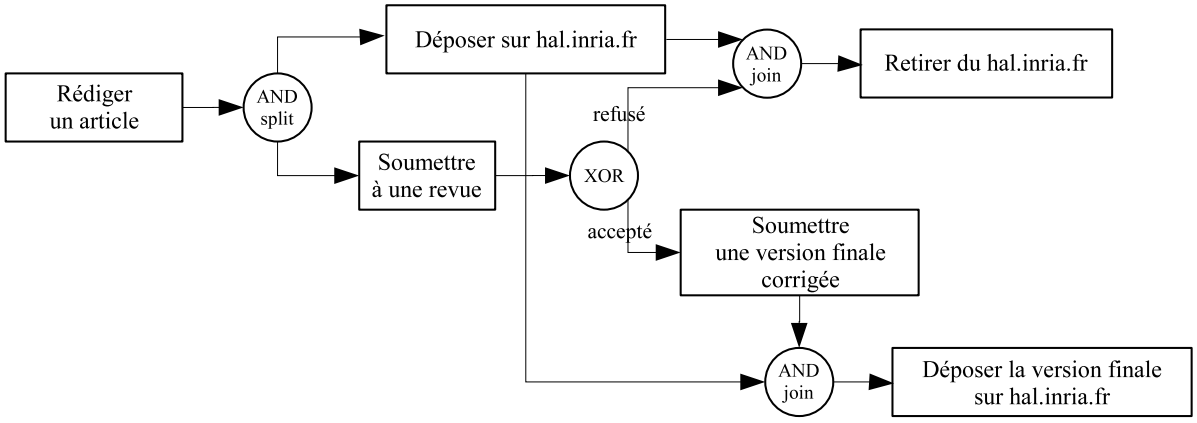


FIG. 1 – Un exemple de procédé de workflow pour la gestion des publications scientifiques

convenablement, et selon leurs compétences, auprès de chaque acteur de l'entreprise les tâches qu'il devra réaliser.

La période d'émergence des systèmes de workflow coïncide avec celle durant laquelle les entreprises ont commencé à diversifier leurs domaines d'activités (65-75). Des besoins nouveaux, correspondant à des situations nouvelles ont vu le jour. Les procédés de production qui ont constitué les exemples typiques des premiers systèmes de workflow se sont vus progressivement remplacés par des procédés beaucoup plus orientés service. Les préoccupations en termes de propriétés à garantir sur l'exécution d'un procédé métiers ont alors connu un revers important. En effet, un procédé de production classique était généralement orienté vers un objectif clair, le produit à fabriquer, et les propriétés à garantir pour le procédé étaient principalement sa terminaison, c'est-à-dire son exécution de bout en bout ou encore les délais de fabrication à respecter. Avec l'avènement d'une nouvelle génération de procédés, appelée d'ailleurs "Business Process" au lieu de "Workflow", principalement orientés services, de nouvelles préoccupation sont apparues. Nous pouvons citer les propriétés d'atomicité pour garantir qu'un ensemble de tâches ou de services doivent être exécutés en entier (la notion du tout ou rien) ou encore la compensation que l'on retrouve souvent dans des exemples de procédés de réservation ou de commerce électronique.

C'est à travers la nouvelle vague de procédés "orientés services" que l'engouement, ces dernières années, pour la gestion de procédés métiers au sein des entreprises a pris de l'ampleur. En effet, nous constatons une montée en puissance de la notion de BPM (Business Process Management) et plusieurs nouvelles normes et standards ont vu le jour ces dernières années, particulièrement orientés web services [XLANG01, WSFL01, ebXML01, WSCL01, WSCI02, BPML02, WS BPEL, WS CDL05]. La technologie web services a attiré beaucoup d'entreprises qui cherchaient une plateforme d'exécution présentant peu de contraintes et des coûts moindres qu'avec les solutions traditionnelles. C'est pour cela que les solutions de gestion de procédés métiers dans de tels environnements d'exécution ont assisté à un développement important. Cet engouement des entreprises et des chercheurs a favorisé la généralisation des procédés métiers dans des domaines et des champs d'application très diversifiés. Ainsi, de nouvelles contraintes à prendre en compte et de nouvelles propriétés à exprimer sont apparues. La fiabilité de l'exécution a été l'une des exigences du domaine bancaire, la réutilisabilité des procédés a été celle des domaines de l'ingénierie, l'interopérabilité des procédés a été celle des entreprises multi-partenaires, la gestion

du travail coopératif et l'anticipation des tâches dans le domaine du TCAO[1], etc.

Le contexte actuel est régi par une multitude de besoins découverts au fur et à mesure de la maturation des procédés métiers et de leur application à des domaines variés. Ce sont des besoins de comportements particuliers souhaités par les utilisateurs de procédés métiers. Ainsi, plusieurs propriétés exprimant des comportements nouveaux ont été introduites dans les procédés métiers. Nous désignons de telles propriétés par l'expression "propriété comportementale". Ce terme rend parfaitement compte de la nature de ces propriétés à répondre à des besoins liés aux applications des procédés métiers. Un comportement revient à une exécution d'activités conformément à certaines contraintes d'ordre opérationnel, informationnel ou encore transactionnel. C'est ainsi que nous avons classifié les propriétés comportementales en trois classes principales.

Les propriétés transactionnelles (atomicité, isolation, compensation ...)

Dans de nombreuses situations critiques, il est nécessaire d'assurer un certain degré de sûreté de fonctionnement, une cohérence des données manipulées ou tout simplement une exécution correcte. C'est souvent le cas des procédés bancaires durant lesquels une erreur peut avoir des conséquences fatales. Par exemple, un procédé accomplissant un virement bancaire d'un compte à un autre est généralement composé de deux activités, la première activité débite le premier compte d'une somme s et la deuxième activité crédite le deuxième compte de la même somme s. on ne peut pas débiter le premier compte sans finir par créditer l'autre compte et inversement.

L'exemple que nous venons de citer illustre bien le besoin critique d'un *comportement atomique*. La propriété d'atomicité exprime, en se basant sur le principe du "tout ou rien", le fait qu'une activité, étant composée de plusieurs opérations, soit exécutée en entier, c'est-à-dire que toutes les opérations de l'activité, sans exception, soient réalisées. Sinon, l'activité ne devra pas du tout s'exécuter. Ainsi, on ne peut pas débiter le premier compte sans finir par créditer l'autre compte et inversement.

L'isolation est une autre propriété transactionnelle qui consiste à protéger une activité d'accès concurrents aux données qu'elle manipule. Nous verrons pour la suite qu'une telle propriété est peu prise en compte dans les systèmes de gestion de procédés métiers et de façon relativement rigide, souvent associée à l'atomicité.

Nous citons également comme propriété transactionnelle, la propriété de compensation qui est prise en compte dans certains modèles de procédés. Pour une activité compensable, l'échec d'une de ses opérations conduit à l'annulation des effets provoqués par toutes celles qui ont déjà été réalisées (compensation). Cette propriété est très utilisée dans le cas de procédés de réservation tels que les procédés de composition de voyages. Dans les modèles de procédés, une activité compensable est généralement associée à une autre activité, dite de compensation et la compensation s'effectue souvent activité par activité.

Les propriétés opérationnelles (instanciation multiples, itérations, délais ...)

Les propriétés opérationnelles traitent d'aspects portant sur l'exécution des activités d'un procédé. Par exemple, le fait de dire que l'exécution d'une activité obéit à une durée d'exécution maximale correspond à une propriété opérationnelle de délai (*deadline*). De même, l'exécution multiple d'une activité constitue une propriété opérationnelle puisqu'elle consiste à définir le nombre d'instances d'une activité. Cette dernière propriété rejoint également celle de l'itération de l'exécution d'activités. Généralement l'instanciation multiple est définie comme un paramètre de l'activité, ce qui permet de spécifier une telle propriété

[1] TCAO : Travail Coopératif Assisté par Ordinateur

activité par activité ou, en cas d'ensembles d'activités, en les unissant dans une activité dite "bloc" qui représente un sous-procédé.

Les propriétés organisationnelles (affectation de rôles, sous-procédés ...)
En ce qui concerne les propriétés organisationnelles, nous pouvons citer l'affectation de rôles aux activités. Cette propriété est typique des modèles de workflow. Un rôle, tel que "graphiste" caractérise généralement une catégorie d'acteurs. Une activité associée au rôle "graphiste" pourra être réalisée par tout acteur graphiste. Dans les systèmes de workflow, chaque activité est caractérisée par un rôle et la distribution des rôles est effectuée activité par activité. On peut cependant imaginer que cette affectation se fasse par groupe d'activités.

Le contexte de notre recherche est donc celui de spécifications multiples de modèles de procédés, exprimant de différentes manières des propriétés avancées de plus en plus nombreuses. Néanmoins, exprimer une propriété comportementale n'est pas aussi évident que cela pourrait le paraître et une solution donnée pose toujours le problème de sa flexibilité d'une part et de son expressivité d'autre part. Notre travail se place donc par rapport à la problématique de la modélisation et de l'exécution de propriétés comportementales dans les procédés métiers que nous détaillons dans la section qui suit.

2

Problématique générale et objectifs de la thèse

Sommaire

2.1 Les propriétés comportementales dans les procédés métiers . . 7
2.2 La flexibilité de la modélisation et de l'exécution de procédés métiers 10
2.3 La séparation des préoccupations 12
2.4 Objectifs de la thèse 13

2.1 Les propriétés comportementales dans les procédés métiers

La problématique générale de la thèse porte sur l'expression de propriétés, dites comportementales, dans les procédés métiers. Une propriété comportementale représente la description d'un comportement particulier requis par une ou plusieurs activités d'un procédé métier. Nous pouvons citer à titre d'exemple l'exécution itérative, l'exécution multiple, l'atomicité, la compensation, etc.

Beaucoup de travaux portant sur l'introduction de nouvelles propriétés, au fur et à mesure de l'évolution des systèmes de gestion de procédés, sont partis de la même vision de base des procédés métiers. Cette vision considère un procédé métier sous forme d'un graphe sans cycle dont les activités (ou tâches) forment les noeuds et les dépendances fonctionnelles, temporelles ou organisationnelles forment les arcs de type "flot de contrôle" ou "flot de données". Comme nous l'avons indiqué dans l'introduction du contexte de recherche, des éléments supplémentaires peuvent former des noeuds particuliers exprimant des opérateurs logiques (AND split, AND join, OR split ...). Si la référence à un modèle de procédé commun et cohérent est un point fort des différents travaux réalisés pour introduire des propriétés comportementales nouvelles, nous critiquons la manière dont les propriétés proposées ont été introduites.

En effet, l'obstacle auquel ont affaire les concepteurs de procédés métiers est souvent celui des contraintes imposées par le modèle de procédé utilisé. Par exemple, lorsque nous voulons exprimer le comportement itératif, nous sommes toujours confrontés à la contrainte selon laquelle le procédé est un graphe sans cycle. Ainsi, la façon triviale de réaliser une itération dans un

procédé, consistant à faire une boucle de flots de contrôle, n'est pas toujours possible, ce qui est compréhensible du fait qu'un cycle de flot de contrôle exprimera, selon une analyse superficielle du problème, le fait que la première activité du cycle ne pourra commencer que lorsque la dernière sera terminée (d'où un blocage). Cet exemple résume parfaitement l'obstacle structurel que forment les modèles de procédés métiers. Dans les travaux introduisant des comportements nouveaux dans les procédés métiers actuels, il existe deux approches permettant de dépasser, ou du moins de minimiser les obstacles imposés par les modèles de procédés.

La première approche tente de réaliser le comportement souhaité en l'intégrant au modèle de procédé comme une propriété d'un des éléments du procédé. Ces éléments sont l'activité, en tant qu'entité élémentaire, ou le procédé en lui-même. Ainsi, si nous prenons l'exemple de l'atomicité, il est facile de définir une propriété sur les activités permettant de spécifier qu'une activité, individuellement, est atomique. Ou encore, il est facile de définir une propriété sur les procédés permettant de spécifier qu'un procédé tout entier est atomique. Grâce à la notion de sous-procédés, il devient également possible d'avoir une atomicité de parties (sous-procédés) d'un grand procédé.

Cette première approche, applicable à beaucoup de propriétés comportementales autres que l'atomicité, est critiquable en quelques points. En premier lieu, une propriété comportementale est, dans sa définition même, soit individuelle, soit collective (ou de groupe). c'est-à-dire que si un comportement implique plusieurs activités, il sera dit de groupe et devra donc être mis en oeuvre en tenant compte de cela. Ainsi, à partir du moment où une propriété comportementale est affectée à l'activité, en tant qu'entité, il n'est plus possible de faire le lien entre les différentes activités impliquées dans un comportement de groupe. Chaque activité réalisera le comportement en question et ce à l'échelle de l'activité elle-même. En second lieu, si une propriété comportementale est affectée à un sous-procédé, il est nécessaire de savoir qu'un sous-procédé présente une structure bien définie, c'est-à-dire qu'il comporte une activité de début et une activité de fin. Cette structure assure à tout procédé une terminaison, cela étant le fruit de l'application de la théorie des réseaux de petri (petri-nets) [Elli 93, Aals 98]. Ainsi, si les activités impliquées dans un comportement particulier ne forment pas, ensemble, une structure de procédé valide, le comportement en question ne sera pas possible à exprimer.

Nous voyons clairement, à partir de ces constatations, que l'approche consistant à affecter la propriété comportementale à l'activité comme entité ou au procédé (ou sous-procédé) comme entité également, n'est pas une approche fructueuse pour exprimer le comportement souhaité dans tous les cas de figure.

La deuxième approche permettant de dépasser les obstacles imposés par les modèles de procédés consiste à contourner le problème en introduisant de nouveaux opérateurs particuliers. L'approche consiste à introduire un nouvel opérateur, prenant le rôle de noeud dans le procédé métier et permettant un contrôle particulier de l'exécution. Nous pouvons citer comme exemple concernant le comportement itératif celui de l'opérateur "Multi Merge". Comme illustré dans la figure 2, dans le cas de l'opérateur "Multi Merge", un cycle de flot de contrôle ayant pour point de départ et pour point d'arrivée le même opérateur "Multi Merge" sera toléré par le système de gestion de procédé.

Si cette solution est pratique pour la réalisation d'itérations dans les systèmes de workflow, celle-ci restreint la sémantique de l'itération car, les activités dont l'exécution sera itérée devront être obligatoirement liées par un flot de contrôle. Ainsi, il n'est pas possible d'exprimer le fait que deux ou plusieurs activités parallèles s'exécutent, ensemble, de façon itérative. Au delà

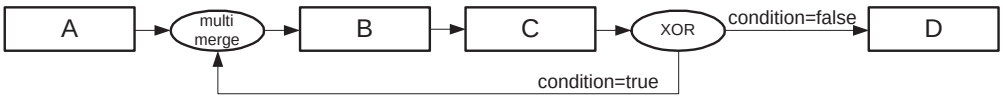


FIG. 2 – L'opérateur "Multi Merge" pour exprimer l'itération dans les procédés de workflow

de l'expressivité réduite d'une telle approche, il est nécessaire de signaler que, si pour chaque comportement possible, un opérateur particulier était créé, le modèle de procédé serait submergé d'opérateurs en tout genre, dont l'utilité et l'utilisation devrait être maîtrisée par le concepteur de procédé. C'est pour ces raisons que cette approche nous semble inadaptée à l'expression de nouvelles propriétés comportementales.

À partir de l'analyse des recherches actuelles portant sur l'introduction de nouvelles propriétés comportementales dans les procédés métiers, nous avons pu clarifier leurs limites. Au delà des limites constatées, nous avons tenté d'établir un diagnostic précis des causes de ces limites. Nous avons conclu que le coeur du problème porte sur deux aspects.

La première cause d'échec des solutions existantes d'introduction de propriétés comportementales est le fait que les propriétés que nous avons appelé "de groupe" sont peu prises en charge et sont souvent assimilées à des propriétés à l'échelle de l'activité (individuelle), au sens où la propriété est intégrée à la définition même de l'activité, ou à l'échelle du procédé tout entier (ou sous-procédé). Il existe clairement un contraste important entre ces deux approches extrêmes dans la prise en charge des propriétés comportementales. Par exemple, lors de la modélisation d'une activité, on est souvent amené à apporter des informations concernant la compensation (activité compensable ou non), l'atomicité (activité atomique ou non), etc. A l'opposé, une propriété est également souvent intégrée à la définition du procédé en général et non de l'activité. Par exemple, un procédé peut être entièrement atomique ou entièrement compensable. Dans la réalité, les propriétés comportementales, en plus d'être à l'échelle de l'activité ou à l'échelle du procédé, peuvent concerner des ensembles d'activités. De telles propriétés comportementales expriment un comportement impliquant un ensemble d'activité, ne constituant pas forcément un procédé à part entière. Il est donc nécessaire qu'une solution introduisant des propriétés comportementales dans les procédés métiers tienne compte du fait qu'une propriété comportementale puisse être "de groupe" et lui permette d'être spécifiée pour un ensemble d'activités quelconque.

Le deuxième aspect entrainant l'échec des solutions existantes d'introduction de propriétés comportementales est celui de l'architecture des modèles. Si le modèle de procédé intègre les propriétés comportementales, cela aboutit à un modèle unique et intégré, loin de fournir les conditions idéales d'expression de nouveaux comportements puisque les propriétés introduites sont sujettes aux contraintes du modèle de procédé. Il est donc nécessaire d'opter pour une séparation des préoccupations, entre la spécification du procédé (selon le modèle de base d'un procédé) et celle des propriétés comportementales à réaliser. Cette séparation des préoccupations est, selon nous, la clé d'une solution plus flexible et plus expressive.

Nous allons dans la section qui suit, poser la problématique de la relation entre la prise en compte des propriétés comportementales en tant que propriétés de groupes et la problématique de la flexibilité de la modélisation et de l'exécution des procédés métiers. Ensuite, nous montrerons la corrélation entre la problématique de séparation des préoccupations et celle de l'expressivité

des procédés métiers.

2.2 La flexibilité de modélisation et d'exécution de procédés métiers

Comme évoqué dans l'introduction, les propriétés comportementales sont souvent exprimées dans les multiples spécifications de procédés métiers soit en les adoptant au niveau de l'activité de façon individuelle soit en les adoptant au niveau du procédé entier. Une telle pratique n'admet pas un juste milieu qui prendrait en compte les propriétés comportementales impliquants un sous ensemble d'activités. Nous citerons ci-dessous quelques exemples de propriétés comportementales d'ordre transactionnel, opérationnel ou organisationnel et la façon dont elles sont réalisées dans les modèles de procédés existants. Nous pourrons alors déceler les problèmes que le contraste important entre le niveau de l'activité et le niveau du procédé influence la flexibilité de la modélisation et de l'exécution de telles propriétés comportementales dans les procédés métiers.

Nous prendrons d'abord l'exemple de la propriété transactionnelle d'atomicité dont le principe est l'exécution du "tout ou rien". Une activité dite atomique ne peut pas s'exécuter partiellement : soit elle s'exécute entièrement, soit elle ne s'exécute pas du tout. Dès les premiers pas des systèmes de gestion de procédés métiers, l'atomicité des activités a été prise en compte. Dans les procédés métiers, nous avons eu l'habitude de considérer chaque activité comme une transaction ACID (Atomique, Consistance, Isolée et Durable) ce qui permettait d'écarter tout problème de sûreté de fonctionnement. Néanmoins, la pratique était de considérer l'ensemble des 4 propriétés transactionnelles de base ACID comme un seul comportement et les activités sont souvent considérées comme des transactions ACID. Cela n'est pas tout à fait justifié car, selon le besoin, nous pouvons exiger uniquement l'isolation et pas d'atomicité ou bien l'atomicité et pas l'isolation ou toute autre combinaison.

Au delà des problèmes explicites de manque de flexibilité tels que ceux des propriétés transactionnelles, nous pensons que, pour beaucoup de propriétés comportementales envisagées, le niveau auquel elle sont appliquées (au niveau de l'activité ou à celui du procédé tout entier) influence directement la flexibilité de la modélisation et de l'exécution de telles propriétés. En effet, les propriétés comportementales concernent généralement un comportement de groupe et cela n'est pas compatible avec le fait que le comportement soit modélisé au niveau de l'activité (comportement individuel d'une activité) ou au niveau du procédé (comportement général du procédé dans son ensemble).

Si nous reprenons l'exemple de l'atomicité, nous pouvons constater que la plus grande part de manque de flexibilité ne vient pas uniquement du mélange des genres entre les propriétés ACID de base, mais repose également sur le choix de ne pas considérer la propriété d'atomicité comme une propriété de groupe. En effet, l'atomicité concerne, dans beaucoup de situations, un groupe d'activités. Plusieurs activités, s'exécutant autour d'un objectif commun seraient susceptibles d'exprimer le besoin de s'exécuter en totalité (toutes les activités du groupe) et non partiellement. L'atomicité est donc clairement une propriété comportementale de groupe. Nous illustrons nos propos à travers l'exemple de la figure 3 de procédé de vente de voiture auprès d'un concessionnaire.

Dans l'exemple de la figure 3, le concessionnaire veut garantir le fait qu'un contrat signé soit

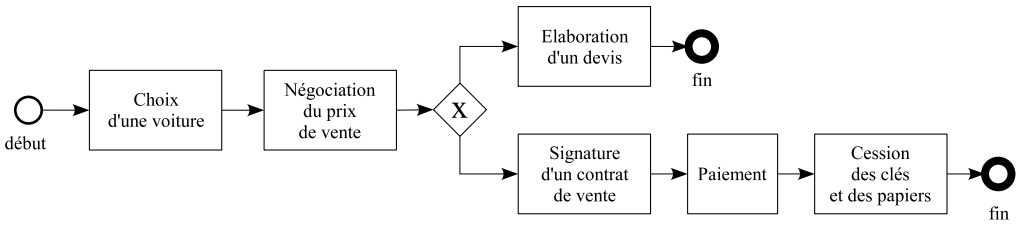


FIG. 3 – Un procédé de vente de voiture auprès d'un concessionnaire : exemple illustratif de l'atomicité comme propriété comportementale de groupe

toujours payé et que les papiers nécessaires au client acheteur soient fournis correctement. Ce besoin exprime clairement une propriété d'atomicité d'un groupe d'activités. Avec les systèmes de gestion de procédés actuels, seule l'atomicité d'une activité ou du procédé dans son ensemble peut être réalisée. Toutefois, le procédé peut très bien s'arrêter à la phase de négociation du prix de vente ou encore à l'étape d'élaboration d'un devis.

Ainsi, l'atomicité de l'ensemble du procédé n'est pas un choix plausible et celle d'activités individuelles ne répond pas non plus au besoin du concessionnaire. En effet, garantir l'atomicité de chacune des trois activités "Signature d'un contrat de vente", "Paiement" et "Cession des clés et des papiers" ne permettra pas de garantir la réalisation des trois activités ensemble.

Ainsi, à partir du moment où seule l'activité ou le procédé peuvent acquérir la propriété d'atomicité, il devient très difficile d'exprimer le fait qu'un groupe d'activités devra s'exécuter de façon atomique sans que ce sous-ensemble ne constitue lui-même un sous-procédé. En étant considéré comme un sous-procédé, le sous-ensemble d'activités est contraint d'avoir la structure d'un procédé, c'est-à-dire bien souvent, une activité de début et une activité de fin clairement identifiées. Cela est clairement en déphasage avec les besoins réels des entreprises car, souvent, les activités d'un groupe qui doit s'exécuter de façon atomique, n'ont pas un lien fonctionnel systématique et ne dépendent pas forcément de flots de contrôle spécifiques pouvant entraîner une structure de sous-procédé.

Après avoir identifié la problématique de la prise en compte des comportements de groupe et de la flexibilité des propriétés comportementales, nous avons naturellement focalisé notre réflexion sur les propriétés comportementales particulières qui impliquent un groupe d'activités, présentant donc une granularité médiane entre l'activité individuelle et le procédé global lors de leur spécification. Nous avons décidé de désigner de telles propriétés par l'expression "propriétés comportementales de groupe". Il est alors devenu indispensable d'identifier de telles propriétés et d'étudier les aspects relatifs à leur mise en œuvre dans les systèmes de gestion de procédés métiers.

Nous partons du constat que toute propriété comportementale de groupe a un champ de définition et une sémantique associée qui ne se limite pas au niveau de l'activité individuellement mais concerne des groupes d'activités. Il est évident que l'expression d'une propriété particulière n'a pas toujours la même signification ou sémantique lorsqu'elle est exprimée pour une activité individuellement et lorsqu'elle est exprimée pour un groupe d'activités en tant qu'entité globale. Par exemple, exprimer le fait qu'un groupe de trois activités s'exécute de façon atomique (les trois s'exécutent sinon aucune ne s'exécute) est différent du fait que chacune des trois activités s'exécute de façon atomique individuellement (chaque activité s'exécute totalement sinon elle ne

s'exécute pas), le résultat n'est évidemment pas le même.

Peu de travaux ont étudié la modélisation de propriétés comportementales de groupe dans les procédés métiers ([Leym 95] pour la compensation, [Derk 01] pour l'atomicité ou encore [Guab 04] pour l'instanciation multiple). Pourtant un point commun réunit toutes ces propositions : c'est la séparation des préoccupations entre la spécification du procédé métier et celle des propriétés comportementales, comme nous l'expliquons dans la section qui suit.

2.3 La séparation des préoccupations

Plusieurs travaux ont été réalisés afin de permettre l'expression de propriétés comportementales telles que l'atomicité, la compensation, l'instanciation multiple, etc. Ces travaux ont, pour certains, introduit des modifications dans le modèle de procédé utilisé. Le modèle de procédé correspond à la définition des différentes briques composant un procédé et à la définition de leurs interconnexions et leurs interactions. Un modèle de procédé de base pourrait définit l'activité (entité élémentaire de travail), le flot de contrôle (arc entre deux activités exprimant une interconnexion de type précédence) et le flot de données (arc entre deux activités exprimant une interconnexion de type échange de données).

Afin d'exprimer des propriétés comportementales, certains concepteurs de procédés essayent d'utiliser les briques de base d'un procédé (activités, flots de contrôle, flots de données, opérateurs logiques de base, opérateurs de synchronisation ...). Cette approche est certes fastidieuse mais elle fournit tout de même des résultats satisfaisants et réalise les propriétés comportementales souhaitées. Néanmoins, les procédés s'en trouvent beaucoup plus chargés et compliqués et la lisibilité du procédé ainsi construit est grandement diminuée. Comprendre de tels procédés relève parfois de l'exploit. Nous proposons dans la figure 4 un exemple concret pour illustrer de telles situations. Il s'agit d'un procédé d'évaluation d'articles de recherche soumis à une conférence. La propriété comportementale d'instanciation multiple y est exprimée à travers le fait que l'activité "Évaluer Article" devra s'exécuter en plusieurs instances dont le nombre total n'est pas défini à l'avance. Nous avons supposé pour cet exemple que le nombre d'évaluations peut aller jusqu'à trois. Un article est ainsi évalué par un certain nombre de relecteurs pouvant aller jusqu'à trois. Il est supposé que pour chaque article, le nombre de relecteurs soit choisi arbitrairement en cours d'exécution du procédé.

Il est clair qu'une telle solution engendre une structure assez compliquée à modéliser et ne peut être utilisée que pour des propriétés et des situations relativement simples. En effet, beaucoup de propriétés comportementales demeurent difficiles, voire impossibles à exprimer sans recourir à l'ajout de nouvelles briques, telles que des opérateurs exotiques (nous pensons ici aux opérateurs MERGE ou MultiMerge utilisés pour l'instanciation multiple de groupes d'activités), ou bien des paramètres additionnels pour les activités ou les procédés (l'atomicité a été introduite en tant que paramètre de l'activité ou du procédé). Cette approche revient tout simplement à intégrer les propriétés comportementales dans le modèle de procédé lui-même. Le problème engendré par une telle approche est principalement l'hétérogénéité des modèles de procédés puisque chaque modèle se différenciera par certaines propriétés non prises en charge par un autre. En outre, deux modèles peuvent prendre en charge de façon différente une même propriété. Enfin, face au nombre important de variantes d'opérateurs exotiques proposées, le concepteur de procédés n'est pas supposé tous les connaître, ni maîtriser leur utilisation qui n'est pas sans difficulté.

L'intégration des propriétés comportementales dans le modèle de procédé conduit à rajouter

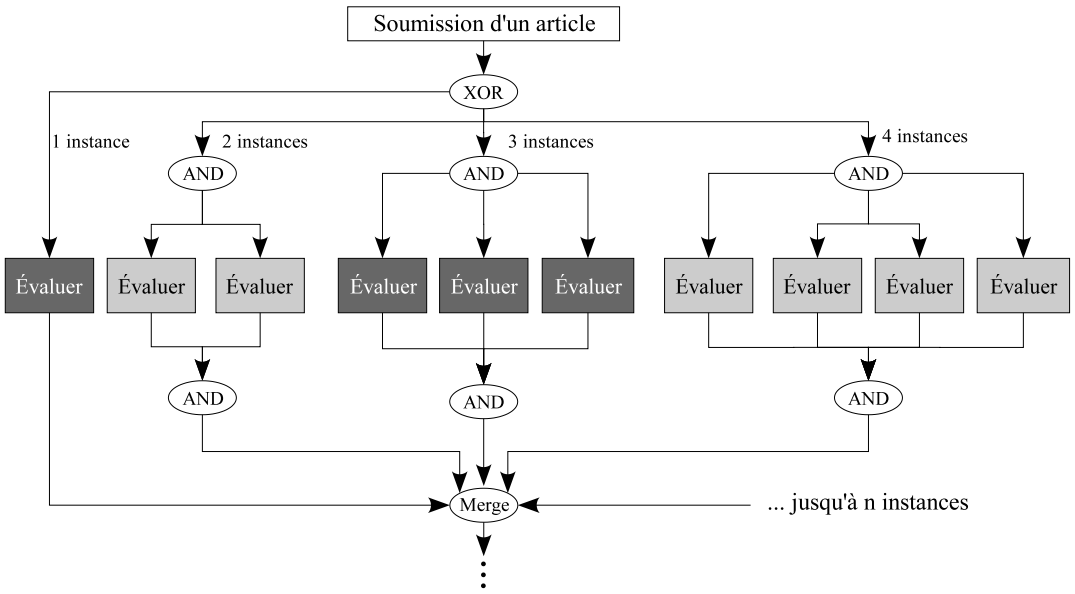


FIG. 4 – Un exemple de propriété comportementale de type instanciation multiple appliqué à une seule activité

une contrainte structurelle à la gestion des propriétés comportementales, qui est celle du procédé lui-même. En effet, une propriété est souvent exprimée à travers un paramètre additionnel affecté à l'activité ou au procédé. Cette contrainte est à l'origine du manque d'une prise en charge satisfaisante des propriétés comportementales de groupe.

Face à de tels problèmes inhérents à l'intégration de la gestion des propriétés comportementales dans le modèle de procédé, il est judicieux d'envisager la séparation entre la spécification du procédé et celle des propriétés comportementales. Cette séparation des préoccupations permettra de disposer d'un modèle de procédé de base, répondant à la problématique primaire des procédés métiers, c'est-à-dire l'ordonnancement avancé d'activités et d'un autre modèle pour la modélisation et l'expression des propriétés comportementales de groupe. Notre premier souci c'est le fait que la séparation des préoccupations sera particulièrement propice à l'expression des propriétés comportementales de groupe et permettra leur plus grande flexibilité.

2.4 Objectifs de la thèse

L'objectif de la thèse consiste à proposer une nouvelle approche pour la modélisation de propriétés comportementales de groupe dans les procédés métiers. Comme nous l'avons précédemment évoqué, le point commun entre les rares travaux sur la gestion de propriétés comportementales de groupe ([Leym 95] pour la compensation, [Derk 01] pour l'atomicité ou encore [Guab 04] pour l'instanciation multiple), est la séparation du modèle de procédé de celui de la propriété comportementale étudiée. La solution que nous mettons en avant dans cette thèse retient cet aspect et tente de constituer un modèle unifié de représentation de propriétés comportementales de groupe. L'approche que nous adoptons tire son origine de la notion de "sphère de contrôle" introduite par Davies en 1978 pour la gestion du traitement de données (Data

Processing).

Les sphères de contrôle ont constitué également l'inspiration principale des travaux précédents que nous avons cités. La notion de sphère a donc été utilisée comme entité encapsulant plusieurs activités d'un procédé, pour exprimer de façon plus flexible et assurer de façon plus fiable des propriétés transactionnelles telles que l'atomicité et la compensation, donnant ainsi lieu aux notions de sphères d'atomicité [Derk 01] et de sphères de compensation [Leym 95]. Nous avions, en 2004, entrepris d'utiliser le même principe d'encapsulation d'activités pour exprimer de façon plus flexible l'instanciation multiple de groupes d'activités [Guab 04].

Nous nous apprêtons dans cette thèse à généraliser la notion de sphère pour introduire celle que nous appelons "sphères de comportement". Une sphère de comportement englobe un certain nombre d'activités d'un procédé qui expriment, ensemble, une propriété comportementale de groupe. La sphère de comportement doit en assurer la réalisation fiable et flexible.

L'objectif premier est de définir les principes fondamentaux d'une sphère de comportement. Pour une propriété comportementale de groupe choisie, nous pourrons vérifier si les principes des sphères de comportement seraient valides ou non. Dans le cas positif, nous pourrons définir une sphère correspondant à une telle propriété et spécifier son fonctionnement particulier.

Ainsi, toute propriété comportementale de groupe n'est pas nécessairement exprimable à travers des sphères. Nous citerons certains exemples de sphères (sphère d'atomicité, sphère de compensation ...) qui ont déjà été développées et nous montrerons, pour ces exemples, que les principes des sphères de comportement sont respectés. Nous citerons également quelques exemples de propriétés comportementales de groupe ne répondant pas aux principes des sphères de comportement que nous aurons fixés. D'autres propriétés comportementales de groupe, assez pertinentes, sont, quant à elles, conformes aux principes des sphères de comportement et, parmi elles, nous avons choisi l'isolation comme cas d'étude approfondi pour la définition des sphères d'isolation.

Les sphères d'isolation s'inscrivent dans la lignée des sphères d'atomicité et des sphères de compensation d'ores et déjà définies. L'isolation, faisant partie des quatre propriétés transactionnelles typiques ACID[1], n'a jamais fait l'objet d'étude sur sa spécification pour un groupe d'activités. Pourtant, la sémantique de l'isolation d'une activité, considérée individuellement, est différente de celle de l'isolation d'un groupe d'activités face aux activités restantes d'un procédé. Nous analyserons cette différence en détail dans cette thèse et nous proposerons un mode de gestion de l'isolation, à base de sphères d'isolation, orienté vers deux dimensions, la cohésion et la cohérence, ce qui diffère de l'isolation classique d'activités individuelles, tournée uniquement vers la cohérence de l'exécution.

Nous étudierons ensuite l'apport, en termes de flexibilité et d'expressivité des sphères d'isolation ainsi définies et nous montrerons l'intérêt particulier de telles sphères pour l'exécution d'activités coopératives. En considérant le travail coopératif comme la manipulation active et concurrente d'objets partagés, l'isolation était jusque là considérée comme l'ennemi du travail coopératif car elle empêchait toute concurrence d'accès. Avec les sphères d'isolation, une dimension supplémentaire est introduite puisque les activités d'une même sphère disposent d'un environnement plus confiné, celui de la sphère, qui permet un travail coopératif alliant flexibilité des accès aux données partagées et fiabilité de l'exécution.

Enfin, nous établissons une démarche pour la mise en œuvre de toute sphère de comportement possible et nous concrétiserons la mise en œuvre des sphères d'isolation dans les systèmes de

[1] ACID : Atomicité, Consistance, Isolation, Durabilité

gestion de procédés métiers et particulièrement dans les plateformes de procédés à base de Web Services, à travers leur modélisation dans la notation BPMN et leur spécification dans BPEL. L'approche de mise en œuvre est dite "de bout en bout" en ce sens que nous partirons de la notation (BPMN prenant en charge la notion de sphères) jusqu'au langage d'exécution du procédé (BPEL).

3

Organisation du mémoire

Après avoir introduit, dans le premier chapitre, la problématique de la thèse et présenté la démarche de notre recherche, nous allons traiter, dans le deuxième chapitre, l'état de l'art. Nous proposons d'organiser l'état de l'art en trois sections principales. La première concerne l'étude historique et l'évolution de la modélisation de procédés, l'avènement de l'ère du Business Process Management (BPM) et l'analyse des besoins actuels des entreprises en termes de BPM. Nous mettrons en évidence le besoin d'une plus grande flexibilité et fiabilité de l'exécution des procédés métiers. La deuxième section de l'état de l'art concerne l'étude des travaux portant sur la modélisation de sphères dans les procédés métiers. Nous partons de l'origine de cette approche qui est celle des sphères de contrôle, puis nous montrons les liens et les différences entre le domaine d'application des sphères de contrôle et celui des procédés métiers. Nous présentons plusieurs exemples de sphères que nous avons identifiés dans la littérature. La troisième et dernière section du premier chapitre concerne le cas de la propriété d'isolation dans les procédés métiers. Nous y exposons les différents mécanismes de gestion de la concurrence d'accès, l'état de l'art sur la gestion transactionnelle des procédés métiers et la flexibilité de l'isolation dans les systèmes transactionnels. Nous aurons ainsi proposé un aperçu des solutions existantes en termes de gestion transactionnelle des procédés et nous constatons les lacunes en termes de gestion de l'isolation dans les procédés transactionnels, d'où l'intérêt de proposer une approche à base de sphères pour la gestion flexible et fiable de l'isolation.

Dans le troisième chapitre, celui de la contribution, nous proposons de définir la notion de sphère de comportement et nous identifions clairement ses principes fondamentaux. Dans ce chapitre, nous reprenons les différentes sphères existantes et nous validerons leur respect des principes fondamentaux des sphères de comportement. Nous évoquons certaines propriétés candidates à la constitution de sphères. Parmi ces propriétés candidates, nous développons en détail celle de l'isolation et la notion de sphère d'isolation. Nous y proposons une description du fonctionnement de telles sphères ainsi que leurs apports à l'exécution d'activités coopératives dans un procédé. Nous analysons comment les sphères d'isolation permettent de répondre à la controverse du critère de sérialisation dans les procédés coopératifs. Pour cela, nous détaillons les aspects opérationnels des sphères d'isolation afin de définir formellement la stratégie d'isolation développée ainsi que le critère de sphère-sérialisation qui constitue une évolution du critère de sérialisation classique vers la prise en compte des procédés d'exécution coopératifs. Enfin, nous identifions, à travers les dimensions de cohésion et de cohérence, un ensemble d'anomalies dites d'isolation dans un contexte d'exécution de procédés coopératifs et nous montrons l'apport des sphères d'isolation afin de gérer l'ensemble de ces anomalies.

Le quatrième chapitre est consacré à la mise en œuvre des sphères d'isolation. Nous effectuons d'abord une brève description du contexte d'exécution des web services et de leur coordination. Nous établissons à travers cette description l'intérêt particulier de mettre en œuvre les sphères d'isolation dans un tel contexte. Ensuite, sur la base des protocoles de coordination existants, nous élaborons une nouvelle architecture du "WS-Coordination framework" qui correspond à la plate forme de coordination de web services, développée par le groupe OASIS (Microsoft, BEA ...). L'architecture de la plateforme de coordination de web service que nous avons développé intègre, dans une vision d'extensibilité, la notion de sphères de comportement en général, et celle de sphère d'isolation en particulier. Cette dernière approche a permis d'intégrer la spécification des sphères d'isolation dans la partie exécution des procédés web services (à travers les contexte de coordination et le langage d'exécution BPEL). Nous proposons, également dans ce chapitre, l'intégration des sphères d'isolation dans la notation des procédés web services (dans la notation BPMN). Cette prise en charge en termes de notation nous a conduit à développer les outils nécessaires permettant de réaliser la chaîne de bout en bout, de la modélisation en notation BPMN, à l'exécution d'un code BPEL généré automatiquement et des contextes de coordination des sphères d'isolation.

Pour terminer, nous concluons par un rappel des objectifs, un bilan des contributions réalisées et nous proposons des perspectives pour de futures recherches.

Chapitre II

État de l'art

1. **La modélisation de procédés**
2. **Les approches à base de sphères**
3. **La propriété transactionnelle d'isolation dans les procédés métiers**

1

La modélisation de procédés

Sommaire

1.1 Introduction historique et évolution de la modélisation de procédés métiers 21
1.1.1 Les origines de l'automatisation de tâches dans les systèmes d'information 22
1.1.2 Les procédés de workflow pour la modélisation et l'exécution de procédés métiers 23
1.2 L'ère du Business Process Management BPM 28
1.3 Les besoins en termes de BPM 30
1.4 Conclusion 32

1.1 Introduction historique et évolution de la modélisation de procédés métiers

Dans cette première section du chapitre de l'état de l'art, nous proposons de décrire l'évolution de la modélisation de procédés métiers depuis le tout début des systèmes d'information et l'automatisation de tâches, en passant par la prolifération de systèmes de workflow jusqu'aux actuelles plateformes d'exécution de procédés de web services. Notre objectif est de clarifier les approches existantes de modélisation de procédés métiers et les relations les unes avec les autres.

Nous commencerons par présenter les travaux réalisés sur la gestion de l'automatisation de l'exécution de tâches dans les systèmes d'information et nous verrons que l'orientation principale qui a été adoptée est celle des systèmes de gestion de Workflow. Cette approche est l'une des premières à concrétiser les systèmes de gestion de procédés métiers dans les systèmes d'information et, nous allons le voir, elle a influencé la voie qu'a pris ce domaine, notamment en se basant sur les réseaux de petri.

Enfin, nous tenterons de dévoiler les besoins contemporains en termes de gestion de procédés métiers à travers la nouvelle donne du BPM (Business Process Management). Nous verrons ensuite qu'un détachement de plus en plus marqué est en train de s'opérer dans la modélisation de procédés métiers optant davantage pour la séparation des préoccupations entre la spécification du procédé métier et celle des propriétés comportementales qu'il exprime. Nous verrons également comment de nouveaux besoins encouragent davantage l'ouverture sur de telles manières de

modéliser les procédés métiers.

1.1.1 Les origines de l'automatisation de tâches dans les systèmes d'information

Au tout début de l'élaboration des systèmes d'information, durant les années 60, leur utilisation était principalement orientée applications. Elle consistait à permettre aux applications diverses dans un système d'exploitation de manipuler les données brutes stockées, souvent, sous forme basique de fichiers. Cela constitue les premiers systèmes d'information informatiques.

Ce n'est que plus tard, durant les annnées 70, à travers l'introduction de Systèmes de Gestion de Bases de Données (SGBD), que l'utilisateur pouvait utiliser directement les données du système d'information. Cela consistait à proposer un langage d'interrogation de données, tel que le langage SQL, dont l'objectif premier est la recherche d'informations ou l'introduction de nouvelles informations. Après la délégation de la gestion des données à un système indépendant, l'interface utilisateur a suivi la même vague et des Systèmes de Gestion d'Interfaces Utilisateurs (SGIU) ont vu le jour durant les années 80 (voir en figure 5).

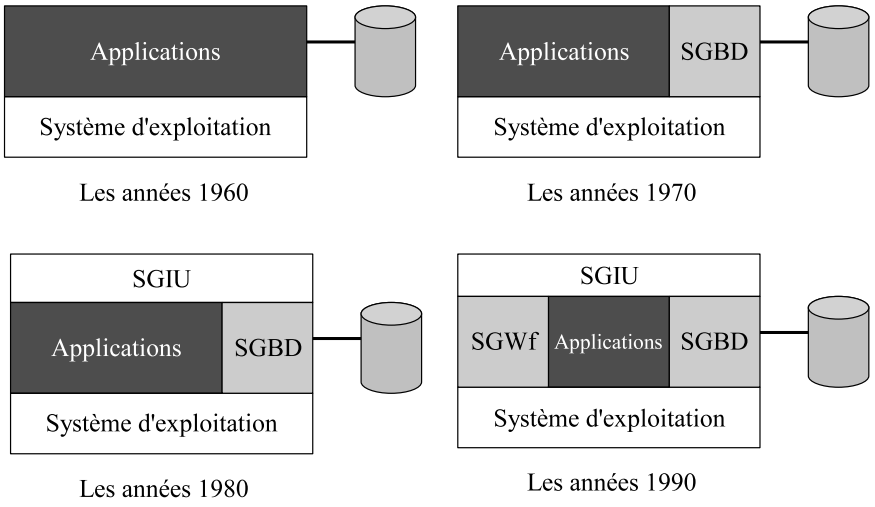


FIG. 5 – Une vue historique des systèmes de gestion de workflow

L'introduction de tâches, appelées à l'époque procédures, dans les SGBD permettait d'automatiser un traitement de données. Cela consistait à décrire, selon un langage procédural, les opérations à effectuer sur une base de données. Les procédures étaient appelées par des applications ou d'autres procédures. Ainsi, l'exécution de procédures dans les systèmes d'information était, selon nous, la première forme d'automatisation de tâches et cela correspondrais à une forme très primitive de modélisation de procédés métiers.

Cette forme de modélisation était construite autour d'une logique selon laquelle une application connait l'existence de procédures à appeler. Ainsi il n'existe pas réellement de gestion de l'exécution puisque chaque application fait appel aux procédures qu'elle connait de façon unilatérale. La conséquence d'une telle logique est qu'aucun contrôle de l'exécution n'est réellement possible.

C'est dans le sens d'un contrôle de l'exécution que les systèmes dits de workflow[1] ont été utilisées pour la gestion de l'exécution d'un ensemble de tâches, notamment dans un système d'information. Il ne s'agit plus d'exécuter des requêtes basiques sur un système de base de données mais putôt d'élements applicatifs, appelés tâches ou activités, qui s'exécutent selon un procédé aboutissant à un ensemble étudié d'opérations sur les données de la base. Les opérations sur la base de données obéissent à une logique plus ou moins complexe qui reflète celle du procédé métier. Les systèmes d'information s'orientent donc de plus en plus vers une gestion des procédés.

La figure 5 illustre l'évolution entre les années 60 et les années 90 des systèmes d'information et de leur prise en compte des procédés métiers, particulièrement à travers les systèmes de workflow. Nous détaillerons dans la section qui suit l'émergence et l'évolution des Systèmes de Gestion de Workflow (SGWf) et nous étudierons plus en profondeur leurs implications dans la gestion de procédés métiers.

1.1.2 Les procédés de workflow pour la modélisation et l'exécution de procédés métiers

Les Systèmes de Gestion de Workflow (SGWf) ont pris une grande ampleur durant les années 90. Dans de tels systèmes, un procédé métier est généralement perçu comme une séquence de tâches effectuées au sein d'une entreprise (banque, compagnie d'assurance ...). Ces tâches sont généralement réalisées par différentes personnes ou applications en vue d'atteindre des objectifs clairs. Un exemple de procédé métier dans une banque est celui qui permet à un client de demander un prêt, au banquier de vérifier sa solvabilité, au responsable de débloquer les fonds ...

L'intégration de Systèmes de Gestion de Procédés Métiers (SGPM)[2] dans les systèmes d'information des entreprises date en réalité de plus de trois décénnies et a aboutit à la prolifération de systèmes de Workflow principalement durant les années 90. La spécification de Workflow a été standardisée par la coalition WfMC[3] et faisait partie des outils de groupware largement utilisés dans les entreprises depuis les années 80 et plus particulièrement durant les années 90. Beaucoup de chercheurs et de praticiens considèrent le workflow comme la représentation informatique d'un procédé métier. A travers son aspect pratique, plusieurs notions émergent dans un workflow dont principalement la notion de rôle, d'activité ou de tâche, de flot de contrôle ou de règles qui conditionnent les relations entre activités. La WfMC présente les composants et leurs interfaces d'un modèle de workflow de référence. Un tel modèle, représenté en figure 6, fournit cinq interfaces :

- L'interface de définition de procédés
- L'interface utilisateur
- L'interface applicative
- L'interface de contrôle de l'exécution
- L'interface d'administration et de monitoring

Le succès des systèmes de workflow revient principalement au fait que l'architecture choisie couvre autant les aspects modélisation, d'utilisation (par l'utilisateur humain ou l'application informatique), de contrôle et d'administration de procédés métiers. Sans vouloir rentrer dans les détails du modèle de la WfMC, longuement détaillé dans la spécification officielle parue

[1] Workflow : traduit littéralement en "flux de travail"
[2] Business Process Management Systems (BPMS)
[3] WfMC : Workflow Management Coalition

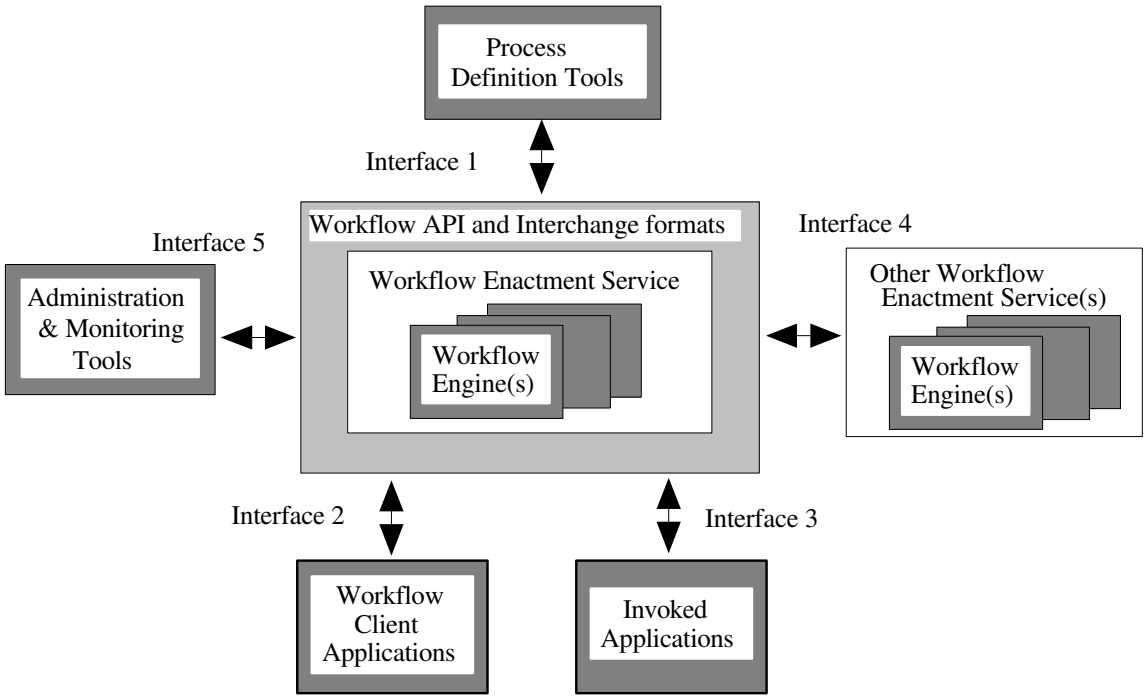


FIG. 6 – Le modèle de référence du workflow selon la WfMC

dans [WfMC, Wfmc 99], nous proposons de commenter l'interface de définition de procédés qui s'engage sur la voie de la modélisation de procédés métiers, et l'interface de contrôle de l'exécution qui s'engage dans la voie de l'exécution des procédés métiers. Ces deux voies forment les deux piliers de notre problématique et nous nous proposons d'en débattre dans ce qui suit.

Le workflow pour la définition de procédés métiers

La gestion de workflow a été introduite dans les entreprises en vue d'éclairer la sémantique de leurs procédés. Plusieurs travaux ont été réalisés afin de définir clairement certains comportements et certaines structures de procédés. Cela consistait à introduire de nouvelles propriétés aux procédés métiers classés selon trois axes : la dimension organisationnelle (hiérarchie, rôles ...), la dimension procédé (séquence, parallélisme, choix multiple, instanciation multiple ...) et la dimension informationnelle - *Information Technology infrastructure* - (consistance de données, récupération de données, compensation, isolation, atomicité).

Ces trois dimensions constituent les trois piliers des systèmes de Workflow. Les systèmes de gestion de workflow sont principalement axés sur la spécification de la structure des procédés en introduisant un ensemble d'éléments structuraux (Activités, flots de contrôle, flots de données, opérateurs logiques ET, OU, OU Exclusif ...). Lorsque le concepteur d'un procédé de workflow désire exprimer un comportement particulier, il utilise ces éléments de la façon la plus ingénieuse possible afin de construire un procédé répondant à ses besoins. C'est grâce à cette pratique que des patrons de Workflow ont été introduits dans [Aals 03] en proposant 20 patrons exprimant chacun la manière d'exprimer certains comportement et besoins dans les procédés métiers. Plusieurs systèmes de gestion de workflow ont implémenté ces patrons afin d'exprimer certains

comportements dans leurs procédés. Mais plusieurs problèmes sont impossibles à exprimer en utilisant les éléments de workflow classiques et ont conduit les concepteurs de systèmes de gestion de Workflow à introduire des éléments exotiques tels que des opérateurs spécifiques permettant de remédier à certains manques dans la spécification standardisée par le Wfmc (Workflow Management Coalition) [Wfmc 99].

La définition de procédés de workflow se base sur un méta-modèle, illustré dans la figure 7. La coalition WfMC exprime dans sa spécification de référence de workflow, plusieurs types d'objets que les entreprises peuvent, toujours selon la WfMC, étendre en introduisant leurs propres objets additionnels. Le modèle de workflow exprime donc un modèle de procédé de base dont les éléments peuvent être étendus afin d'exprimer plus de possibilités des procédés métiers.

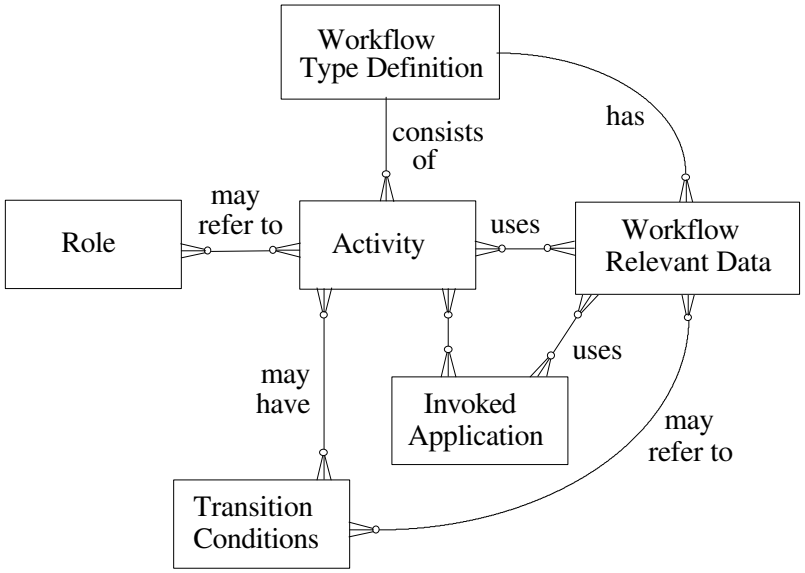


FIG. 7 – Le méta-modèle de définition de procédés de workflow selon la WfMC [Wfmc 99]

L'exécution des procédés de workflow : au delà d'un simple ordonnancement de tâches

La coordination des activités d'un procédé revient à en contrôler l'ordonnancement en respectant les dépendances d'ordre fonctionnel, à l'aide du flot de contrôle, et informationnel, à l'aide du flot de données. L'ordonnancement d'activités a pour objectif de contrôler l'ordre d'exécution des activités. Le contrôle de l'exécution se fait à travers une approche d'états-transitions. En effet, un procédé de workflow a un état qui évolu selon le diagramme de changement d'état de la figure 8 et l'activité a également un état qui évolue selon le diagramme de changement d'état de la figure 9.

Il est courant de voir des systèmes de gestion de workflow choisir une formalisation de l'exécution en utilisant des réseaux de petri (petri net) [Elli 93, Aals 98]. Ce formalisme a permis à beaucoup de chercheurs de construire des systèmes capables d'assurer une exécution sûre et ce grâce au bagage important de preuves automatisables à l'aide de la théorie des réseaux de

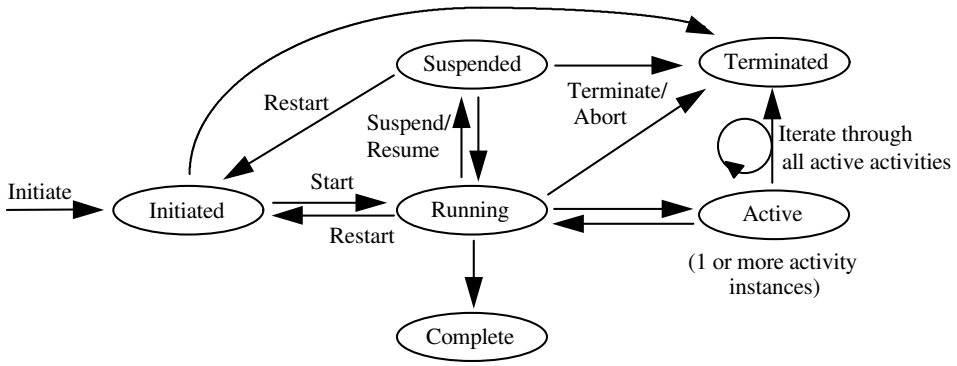


FIG. 8 – Diagramme de changement d'état d'un procédé de workflow

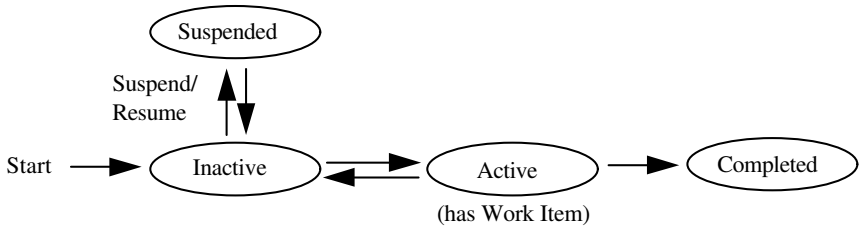


FIG. 9 – Diagramme de changement d'état d'une activité d'un procédé de workflow

petri. La figure 10 illustre la représentation en réseaux de petri de l'exécution d'un procédé de workflow.

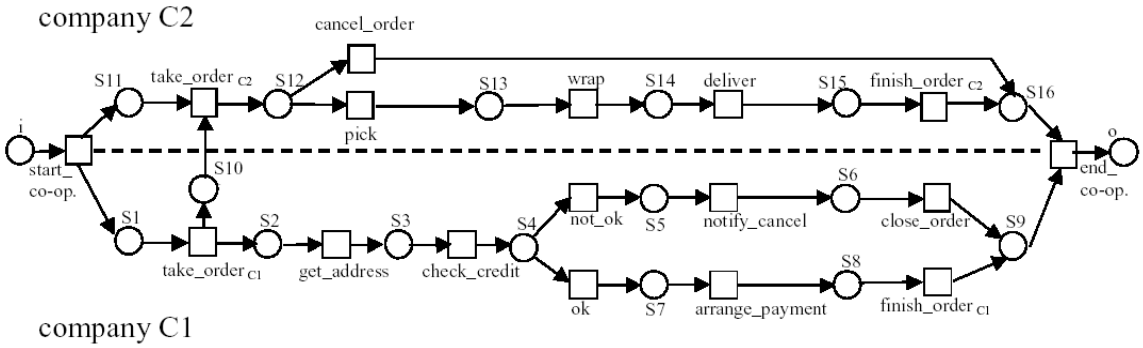


FIG. 10 – Utilisation du formalisme des réseaux de petri pour l'exécution de procédés métiers

Néanmoins, la coordination d'activités ne se résume pas en une gestion d'ordonnancement d'activités mais va bien au delà. En effet, nous considérons que la coordination d'activités revient à la gestion de leur exécution pour satisfaire trois contraintes. La première est d'ordre temporel, induisant l'exécution d'une activité avant ou après celle d'une autre pour des raisons de "timing" ou même à un instant bien précis. La deuxième est d'ordre fonctionnel, induisant qu'une acti-

vité s'exécute après une autre car son exécution en dépend, soit parce qu'elle doit récupérer ses données résultats, soit parce qu'elle constitue la poursuite d'un travail que la première a commencé. Enfin, la troisième contrainte est d'ordre comportemental, exprimant des propriétés plus complexes comme les propriétés transactionnelles de compensation par exemple.

Le workflow est aussi inter-organisationnel

Dès le début des années 90, plusieurs standards ont vu le jour face à la nature propriétaire des systèmes de workflow de l'époque. La standardisation répondait également et surtout au besoin, largement répandu à l'époque, d'une intégration des procédés métiers, appelé BPI pour "*Business Process Integration*". Le but était de fournir aux entreprises des systèmes de workflow intégrés aux applications et favorisant ainsi l'interaction, l'échange et l'interopérabilité entre les applications de l'entreprise. Cela a contribué à la prolifération de projets d'EAI "*Enterprise Application Integration*", traduit littéralement "intégration des applications de l'entreprise".

Grâce aux workflow inter-applications, l'EAI va au-delà de l'interopérabilité entre les applications. En effet, à travers une approche plus modulaire (garder les applications hétérogènes de l'entreprise et ne développer que des "*middleware*"), et une gestion de l'interaction entre ces applications au travers de workflows, l'EAI constitue une alternative aux ERP "*Enterprise Resource Planning*", aussi appelés Progiciels de Gestion Intégrés (PGI) qui ont pour conséquence le développement de nouveaux outils intégrés pour tous les services de l'entreprise et donc des coûts de développement, de mise en place et de formation plus lourds.

Les nouvelles pistes du workflow : workflow transactionnel, workflow dynamique ...

A l'origine, les systèmes de workflow permettaient de représenter des procédés de production, répétitifs et supportant peu de changement dans le temps. Une telle vision était efficace en terme de volume de procédés modélisés dans l'entreprise et de gestion des cycles d'exécution du travail. Malgré les avancées considérables dans le domaine de l'intégration des procédés métiers, plusieurs obstacles relatifs à l'adaptabilité des workflow et à leur aptitude à évoluer plus facilement selon les besoins ont commencé à poser problème.

Les procédés ad-hoc, dont la définition complète n'est pas déterminée à l'avance ou bien qui sont en perpétuelle évolution [Char 06], ou encore les procédés coopératifs dont la nature du travail induit le partage de résultats intermédiaires évoluant assez fréquemment [Grig 01b] font l'objet de recherches récentes. La flexibilité de l'exécution d'un workflow a également été étudiée afin d'introduire des techniques telles que l'anticipation d'exécution des activités [Grig 01a]. Une autre évolution de la gestion de procédés métiers est également enclenchée, celle de la généralisation des procédés métiers à toute activité dans l'entreprise afin de garantir, dorénavant des procédés dits de bout-en bout et la promotion de la réutilisabilité.

Le WfMC a également fait un effort de standardisation allant vers l'interopérabilité des systèmes de workflow en proposant le format XPDL (XML Processing Description Language) [XPDL]. Le but de XPDL est de stocker et d'échanger le diagramme représentant un procédé. Il permet à un outil de conception de procédés d'écrire le diagramme, et à d'autres de le lire, de façon parfaitement interopérable et standardisée. Ainsi, la conception d'un procédé sera indépendante de la plateforme d'exécution de procédés à utiliser, sous réserve qu'elle accepte l'importation de fichiers XPDL.

Au même moment, une évolution majeure dans les systèmes de gestion de procédés est en train de se mettre en place. Cette évolution est directement liée à la prolifération d'architectures orientées services (SOA) [Bieb 05, SOAref06]. Plus particulièrement, les technologies web services constituent un environnement propice à une nouvelle vision des procédés métiers et des exécutions de tâches, orientées services, et donc, obéissants à de nouvelles contraintes et de nouveaux enjeux. C'est le nouvel ère du Business Process Management BPM, que nous développons dans la section qui suit.

1.2 L'ère du Business Process Management BPM

Durant les dernières années, l'abréviation BPM, ou encore BPMS, s'est largement répandue au sein des entreprises, exprimant la notion de "*Business Process Management (System)*", traduite littéralement "(système de) gestion de procédés métiers". Pour tenter de définir ce qu'est le BPM, nous pouvons citer les propos de David McCoy, du Gartner Group, qui l'a défini en mars 2001 comme suit :

> " *... blending of process management/workflow with application integration technology ... to support rich human interaction and deep application connectivity*".

L'entreprise dépasse alors les clichés de flux de travail qu'exprimaient les systèmes de Workflow et considère l'organisation et l'interaction d'acteurs humains et de composants logiciels (humain à humain, humain à application, application à application). La réutilisabilité, l'extensibilité, la flexibilité sont des briques de base des BPMS. On est alors très loin des considérations initiales du workflow des années 80.

L'architecture orientée service constitue l'une des architectures les plus adaptées à cette nouvelle donne du BPM. Une telle architecture considère une collection de services communiquants les un avec les autres et pouvant être coordonnés afin de satisfaire une certaine exécution requise. Cette architecture n'est pas nouvelle. En effet, une telle architecture existait déjà à travers l'utilisation des DCOM (Distributed Component Object Model) basés sur la spécification COM de Microsoft ou des ORB (Object Request Brokers) basés sur la spécification CORBA de l'OMG[1].

De nos jours, la technologie des web service a pris le relais de part son aspect non propriétaire et de part ses protocole ouverts et simples que sont HTTP et SOAP[2] basé sur XML. Dans les plateformes de web services plusieurs standards ont vu le jour donnant lieu à un ensemble complémentaire et interdépendant de spécifications assurant flexibilité, réutilisabilité, interopérabilité et extensibilité des systèmes de BPM.

La modélisation de procédés métiers a fait l'objet de plusieurs études impliquant les principaux acteurs dans ce domaine tant au niveau des entreprises (IBM, Microsoft, bea ...) qu'au niveau des organismes de standardisation (OMG, WfmC ...). Un tel engouement a donné lieu à une multitude de tentatives visant à proposer des standards (Xlang, BPML, WSFL, ebXML, WSCL, WSCI, BPEL4WS, WS-Chroeography, WS-BPEL) pour la modélisation et l'exécution de procédés métiers répondant au mieux à la nouvelle donne des architectures orientées services. La chronologie de tels standards est présentée dans la figure 11 et nous proposons en annexe A un récapitulatif des apports de chaque spécification ainsi que la relation entre les différentes spécifications.

[1] Object Management Group
[2] SOAP : Simple Object Access Protocol

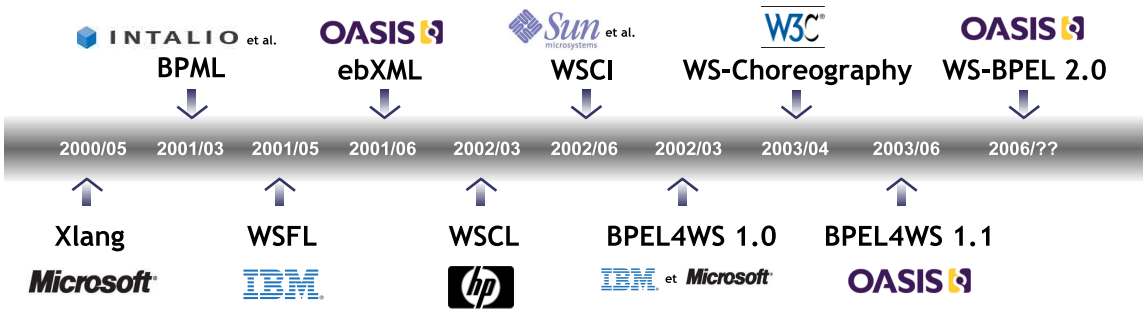


FIG. 11 – Évolution des standards de la modélisation et de l'exécution de procédés métiers

Parmi toutes les propositions existantes, nous pouvons citer celles qui nous semblent les plus pertinentes dans le contexte de l'évolution actuelle des BPM. Citons d'abord, le standard BPEL "*Business Process Execution Language*", souvent appelé BPEL4WS ou encore WS-BPEL, selon la version de la spécification. BPEL est un langage permettant d'exprimer l'exécution d'un procédé web services. Ce language a été le premier à être définit par les organismes de standardisation. La raison est simple : c'est le langage "bas niveau" pour exprimer l'exécution d'un procédé métier.

Plus tard, la spécification BPMN[1] fut proposée par l'OMG[2] et le BPMI[3] en 2005. Cette notation, purement graphique, a été adoptée, jusqu'à ce jour par plus de 42 BPMS (conformément aux statistiques de la BPMI http ://www.bpmn.org/ du 26 février 2007). La notation BPMN reprend les principes du Business Process Management et des workflow afin d'apporter une représentation visuelle exhaustive de tous les aspects conceptuels du BPM. Un tel standard permet aux concepteurs de procédés métiers de se comprendre mutuellement à travers une représentation graphique unifiée. La figure 12 illustre un exemple de procédé en notation BPMN.

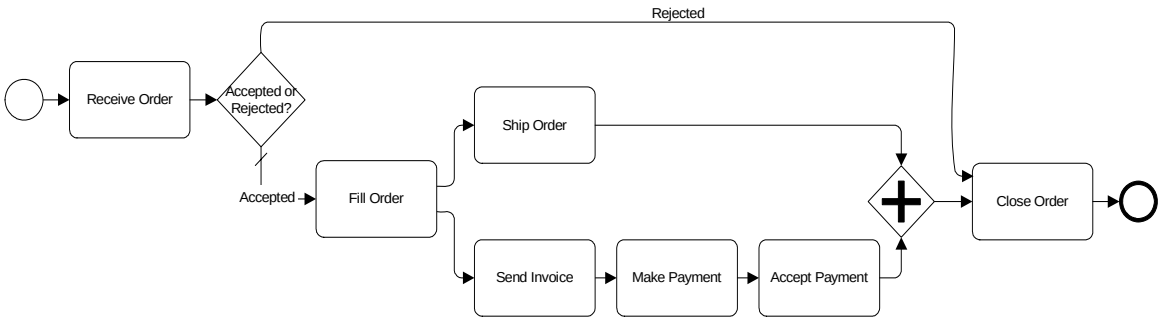


FIG. 12 – Exemple de procédé en notation BPMN

Au delà de l'aspect graphique offrant une notation de référence permettant la traduction automatique d'un procédé modélisé graphiquement dans un format compréhensible par un moteur d'exécution de procédés, la notation BPMN supporte tous les "*workflow patterns*, recensés

[1] BPMN : Business Process Modeling Notation
[2] OMG : Object Management Group
[3] BPMI : Business Process Management Initiative

principalement dans [Aals 03]. Ces patrons de workflow consistent en micro-modèles typiques de processus capables, une fois combinés, d'exprimer toute situation ou tout besoin. En supportant tous les patrons de workflow, la notation BPMN permet une grande expressivité.

Les techniques de gestion de procédés métiers, de la conception (BPMN) à l'exécution (BPEL) offrent d'importants moyens pour la coordination d'activités. Par exemple, les éléments de base de BPMN sont de quatre types : les "couloirs" (*Swimlanes*), les "objets de flux" (*Flow Objects*), les "objets de relation" (*Connecting Objects*), et les "objets symboliques" (*Artifacts*). La coordination d'activités dans les procédés métiers, en l'occurrence à travers la notation BPMN, consiste à connecter entre eux des "objets de flux" à laide d' "objets de relation". BPMN fournit trois types d'objets de flux : des activités, des événements et des portes (*Gateways*), et trois types d' "objets de relation" : des flux séquence, des flux message et des associations.

En proposant la notation BPMN, l'OMG et le BPMI se devaient d'assurer sa compatibilité avec le langage d'exécution de procédés web services, le BPEL. La norme BPMN spécifie la correspondance entre BPMN et BPEL. Grâce à BPMN et BPEL, le BPM peut garantir à une organisation la parfaite adéquation et synchronisation des modèles de procédés décrits et exécutés. Un diagramme en notation BPMN pourra être exporté en BPEL et être déployé sur une plateforme d'exécution de web services (Moeteur d'exécution BPEL).

Plusieurs travaux ont été effectués sur la correspondance entre BPMN et BPEL [Ouya 06b, Ouya 06a], néanmoins, les outils logiciels qui permettent une telle conversion ne peuvent pas, à ce jour, effectuer la conversion dans les deux sens. Cela reste admissible si nous comparons le BPMN à un langage de programmation avancée et BPEL à l'exécutable résultant de la compilation de ce langage.

Certes ces technologies sont assez récentes et les outils qui les mettent en œuvre ne sont pas encore parfaitement matures, mais elles incarnent un changement majeur dans la façon de définir et d'exécuter les procédés métiers dans les entreprises et héritent de la maturité de nombreuses autres technologies qui les ont précédées.

1.3 Les besoins en termes de BPM

Les différentes spécifications proposées tout au long des sept dernières année avaient, à chaque fois, pour objectif de mieux répondre aux besoins en matière de gestion des procédés métiers, au niveau de sa modélisation, de son exécution et de son maintien. Ces besoins n'ont cessé d'évoluer et nous pouvons les résumer, à l'état actuel des choses, comme suit :

- Modéliser des procédés collaboratifs
  L'expérience en modélisation de procédés dans des environnements d'EAI ou de B2B a mené à l'adoption croissante des modèles de procédés collaboratifs. Dans les modèles de procédés collaboratifs, un procédé est décrit comme un ensemble de collaborations entre divers participants, y compris des organismes, applications, employés, et d'autres procédés. Le modèle de procédé doit ainsi non seulement se focaliser sur les activités à l'intérieur d'une entreprise, mais également s'ouvrir sur les activités inter-entreprises, plus communément appelées B2B (Business to Business).
- Modéliser des procédés de workflows
  Le workflow définit comment les participants à un procédé agissent ensemble tout au long de l'exécution, en orchestrant leurs travaux respectifs. Habituellement, une abstraction des

participants est opérée en utilisant des rôles, notamment dans les workflows. À chaque rôle est affecté un certain nombre de tâches à réaliser. La plupart des normes de workflow prennent également en considération la décomposition en sous-procédés, ce qui permet à des activités dans un workflow d'être réalisée à travers un autre workflow.

- Modéliser des procédés transactionnels
  Les transactions doivent figurer en tant que piliers pour la construction de n'importe quel procédé métier et les modèles de procédés métiers doivent en tenir compte en fournissant des moyens de spécification et de contrôle de transactions. De plus, ces modèles doivent prendre en compte les transactions longues dont l'exécution peut prendre des heures ou des semaines. Si une transaction échoue, des actions compensatrices peuvent être nécessaires. Par exemple si une réservation d'hôtel est annulée après qu'un paiement ait été effectué, une action compensatrice peut être exigée pour annuler le paiement. D'autres contraintes telles que les délais d'exécution ou la notification peuvent également être exigées.
- Gérer les exceptions
  Si une exception est provoquée pendant l'exécution d'un procédé, alors il est important que le modèle de procédé permette, d'une part sa détection et d'autre part le déclenchement d'actions appropriées pour rétablir un état cohérent du procédé.
- Modéliser des procédés orientés "service"
  La technologie web services permet de fournir un environnement de communication entre les participants à un procédé collaboratif. L'architecture orientée service permet de faciliter une telle communication et d'aboutir à une orchestration facilitée. Certaines normes de modélisation de procédés, proposées récemment, telles que WSFL ou XLANG emploient des interfaces WSDL pour décrire les services exposés par chacun des participants aux autres.
- Gérer la sécurité et la fiabilité des communications entre procédés
  Pour des processus ayant une nature stratégique ou critique, l'envoi fiable et sécurisé de messages est une exigence absolue. Ainsi, des messages de B2B peuvent devoir être électroniquement signés et authentifiés. Cette qualité de service peut varier selon le type de transaction.
- Gérer l'audit de procédés
  Pour des raisons légales, il est généralement très important dans des processus de B2B que l'audit de certaines transactions soit effectué afin de garder une trace. Ceci signifie qu'un partenaire commercial ne peut pas réclamer qu'une transaction ne soit pas acceptée quand en réalité elle l'est ; c'est-à-dire, qu'il assure le non-reniement de la transaction par l'associé. L'audit des procédés doit être techniquement et légalement irrévocable.
- Gérer les contrats
  La notion de contrat dans les environnements d'exécution de procédés est propre aux procédés B2B. Un contrat est défini entre deux ou plusieurs entreprises (ou participants) pour garantir aux différents partis la réalisation de fonctions spécifiques (identifiées par des rôles) dans un procédé métier distribué.

1.4 Conclusion

Après avoir survolé l'architecture orientée services et les plateformes d'exécution de web services, nous pouvons clairement constater qu'une nouvelle façon de modéliser et d'exécuter les procédés métiers est dores et déjà en place. Il s'agit en définitive de séparer les préoccupations entre la modélisation du procédé et son exécution. Cette séparation des préoccupation (*Separation of Concerns*) est finalement le résultat de plusieurs années de "réformes", optant de plus en plus pour le détachement d'un modèle de procédé dit intégré, comme c'est le cas des workflow. Dans la section qui va suivre, nous allons présenter les différents travaux qui ont opté pour une telle séparation des préoccupations en se basant sur une approche à base de sphères. Nous essayerons de montrer l'intérêt d'une telle approche et les différentes tentatives réalisées dans ce sens donnant lieu aux sphères d'atomicité, de compensation ou encore de flexibilité.

2

Les approches à base de sphères

Sommaire

2.1 L'approche des sphères de contrôle 33
2.2 Du traitement de données aux procédés métiers : différences et nouveaux enjeux 34
2.3 Les propositions existantes à base de sphères pour la gestion de procédés métiers 35
2.3.1 Les sphères d'atomicité 35
2.3.2 Les sphères de compensation 37
2.3.3 Les "poches" de flexibilité (Pocket of Flexibility) 37
2.3.4 Synthèse 38

Dans cette thèse, nous proposons d'adopter une approche utilisant des sphères pour la spécification des comportements particuliers dans les procédés métiers et nous l'appliquons au cas des contraintes d'isolation. Cette approche n'est pas nouvelle puisque plusieurs travaux ont abordé des aspects divers, transactionnels ou non, à l'aide de sphères.

Nous proposons non seulement d'utiliser des sphères pour spécifier les propriétés transactionnelles, en l'occurrence l'isolation, dans les procédés métiers, mais nous voulons également éclairer cette approche en la formalisant et en fournissant au lecteur les moyens d'identifier son utilité dans d'autres cas de figure. Dans ce chapitre, nous présenterons l'état de l'art sur les approches à base de sphères, des origines incarnées par les sphères de contrôle [Davi 78] en 1978, aux différentes sphères (d'atomicité, de compensation, etc.) qui ont été développées depuis.

2.1 L'approche des sphères de contrôle

Notre vision des sphères s'est inspirée de la notion de "sphère de contrôle" proposée par Davies citer Dav78 à la fin des années 70. Le contexte principal dans lequel le concept de sphère de contrôle a été défini est la réutilisation des traitements dans les systèmes de traitement de données. Il s'agit de définir des frontières autour d'ensembles de sous-processus. Cette encapsulation de sous-processus était introduite pour assurer leur réutilisabilité mais bien plus encore.

En effet, les sphères de contrôle ont été employées à plusieurs fins dans des systèmes répartis multi-nœuds de traitement de données. Parmi ces usages, les sphères de contrôle permettent d'éviter l'interférence entre processus. Elles permettent aussi d'assurer le retour d'un processus à

un stade précédent où l'état des données est acceptable. Elles permettent de prévenir l'utilisation de données d'un processus avant qu'il ne soit sûr qu'il ne va pas être annulé. Aussi, les sphères de contrôle permettent de contrôler des processus répartis sur plusieurs nœuds ou de préserver l'ordre d'exécution dans un environnement de programmation et de traitement multiples. Elles permettent également de mémoriser les résultats pour l'audit afin de réduire la probabilité d'erreurs ou encore d'assurer la consistance des résultats d'un processus. Il est donc clair que les enjeux des sphères de contrôle sont riches et multiples. Voyons d'abord en quoi une telle approche est originale.

Les sphères de contrôle se basent sur l'encapsulation d'un ensemble de sous-processus et l'exercice d'un certain contrôle sur de tels groupes. Il existe une multitude de sphères de contrôle différentes, chacune permettant de maintenir une certaine relation entre un ensemble de sous-processus et un type particulier de contrôle. Le contrôle en question est classé par l'auteur en cinq types :

- Contrôle de processus (atomicité, validation *(commitment)*, validation à nœuds multiples, dépendances, allocation de ressources).
- Contrôle de recouvrement (recouvrement dit in-process, post-process ou système)
- Contrôle de l'audit (audit intra processus et inter processus)
- Contrôle de l'intégrité relationnelle (consistance)
- Contrôle de l'environnement d'exécution (exécution à la chaîne, exécution dite on-ligne, exécution dite in-line)

Les solutions basées sur les sphères de contrôle définissent des regroupements de processus et leur attribuent des propriétés particulières adaptées au comportement souhaité du groupe. Néanmoins, les apports d'une telle solution se sont focalisés sur l'exécution de processus de traitement de données et plusieurs chercheurs ont tenté d'étendre le raisonnement à base de sphères aux procédés métiers et au BPM : c'est le sujet de la section qui suit.

2.2 Du traitement de données aux procédés métiers : différences et nouveaux enjeux

Une différence majeure existe entre les procédés métiers dans les BPMS et les processus de traitement de données dans les DPS (Data Processing Systems). En effet, Les procédés dans les BPMS consistent en activités manuelles ou automatiques ayant divers objectifs (traitement, prise de décision, synchronisation ...) et organisés autour d'une structure de procédé intégrant des propriétés beaucoup plus structurelles que les DPS traditionnels.

Certes, les enjeux de la gestion des données, leur cohérence et leur consistence, font partie tout autant des préoccupations des DPS que des BPMS puisque tout les deux aboutissent à une manipulation de données. La différence réside dans le fait que, d'une part, la manipulation n'est pas uniquement orientée traitement de données dans les BPMS mais inclut également la prise de décision ou encore la synchronisation, et d'autre part, de nouveaux problèmes sont apparus dans le contrôle de procédés métiers tels que des besoins transactionnels ou plus généralement comportementaux. Les banques ont besoin d'exprimer des propriétés d'atomicité en cas, par exemple, d'opérations critiques de virement d'argent, les entreprises d'e-commerce ou les compagnies aériennes ont besoin d'exprimer des propriétés de compensation en cas, par exemple, d'annulation

de vol, etc.

La notion de sphère ne répond plus seulement aux problèmes évoqués en 1979 par Davies [Davi 78]. Elle intègre maintenant des besoins nouveaux, découverts au fur et à mesure de l'utilisation des procédés métiers dans de plus en plus de domaines d'activités. La notion de sphère de contrôle a donc été l'inspiration de plusieurs chercheurs qui ont adapté ce concept pour la résolution de nouveaux problèmes dans les BPMS. Nous nous proposons dans la suite de présenter quelques travaux sur de nouvelles sphères pour les procédés métiers.

2.3 Les propositions existantes à base de sphères pour la gestion de procédés métiers

Dans cette section, nous allons présenter quelques solutions existantes, inspirées de l'approche des sphères de contrôle, permettant de résoudre des problèmes particuliers rencontrés dans les procédés métiers.

Jusqu'à ce jour, les sphères de comportement ont été utilisées afin d'exprimer un comportement transactionnel dans les procédés métiers. Cela est compréhensible au regard du besoin important de fiabilité et de solidité des procédés métiers. C'est ainsi que nous allons présenter quelques sphères de comportement permettant de résoudre des problèmes d'atomicité, d'isolation ou encore de compensation.

Malgré cette tendance vers les propriétés transactionnelles, nous avons également montré qu'il est judicieux d'utiliser les sphères de comportement pour toute autre propriété et nous avons validé ce choix en ce qui concerne l'instanciation multiple d'activités.

Ajouter la flexibilité à la définition de comportements particuliers au sein des procédés n'est pas chose nouvelle. Plusieurs travaux se sont fondés sur une telle séparation entre la définition du procédé et celle des comportements requis pendant son exécution. Nous proposons de décrire brièvement trois tentatives réussies de solutions basées sur la notion de sphère que nous avons relevées dans la littérature. Deux d'entre elles, connues en tant que "sphères d'atomicité" et "sphères de compensation", ont été proposées pour exprimer de façon plus flexible des propriétés transactionnelles dans des procédés métiers. La troisième est connue sous le nom de "pocket" de flexibilité et concerne la flexibilité du contrôle de l'exécution de groupes d'activités.

2.3.1 Les sphères d'atomicité

L'une des propriétés majeures dans la gestion transactionnelle des procédés métiers, l'atomicité, a été gérée à travers différentes techniques, comme par exemple la validation à deux phases et a fait l'objet d'une prise en charge à l'aide de sphères de comportement [Leym 00][Heuv 02][Derk 01]. L'atomicité dans les procédés métiers exprime le principe du "tout ou rien" sur l'exécution d'une activité ou de tout le procédé. Les systèmes de gestion de workflow actuels ont banalisé l'expression de l'atomicité, mais seulement au niveau de l'activité seule ou au niveau de tout le procédé. C'est pour introduire une certaine flexibilité de l'atomicité et sortir du clivage des niveaux (activité individuelle ou procédé entier) que Reuter [Reut 89] a permis d'introduire une spécification de workflow transactionnel appelé ConTract. Un tel modèle a permis l'exécution d'activités (ou de groupes d'activités) assurant des propriétés ACID strictes. Le modèle ConTract gère le flot de contrôle et la compensation d'unités atomiques composées de plusieurs tâches. Néanmoins,

comparé aux ambitions de l'approche des sphères de comportement, le modèle ConTract définit les propriétés transactionnelles d'atomicité explicitement dans le modèle de workflow ce qui est contraire à l'objectif de séparation de la spécification du workflow de la spécification des propriétés comportementales, en l'occurrence l'atomicité, propre à l'approche des sphères de comportement.

Une autre approche proposant une flexibilité de l'atomicité et basée sur la vision des sphères de comportement a été proposée dans Derks et al. [Derk 01]. Les auteurs proposent la notion de sphère d'atomicité qui permet de séparer la spécification du procédé de workflow de celle des besoins en atomicité. L'approche proposée permet de réaliser une atomicité partielle de groupes d'activités. L'auteur présente l'utilisation de sphères d'atomicité à travers une annotation des procédés métiers, qu'il représente sous forme de réseaux de petri (petri net). La figure 13 illustre un exemple, proposé par l'auteur, annoté par des sphères sous forme de carrés encapsulant des tâches du procédé. La figure 14 illustre l'équivalant d'un tel procédé, sous forme de réseaux de petri, respectant les contraintes d'atomicité requises dans la figure 13.

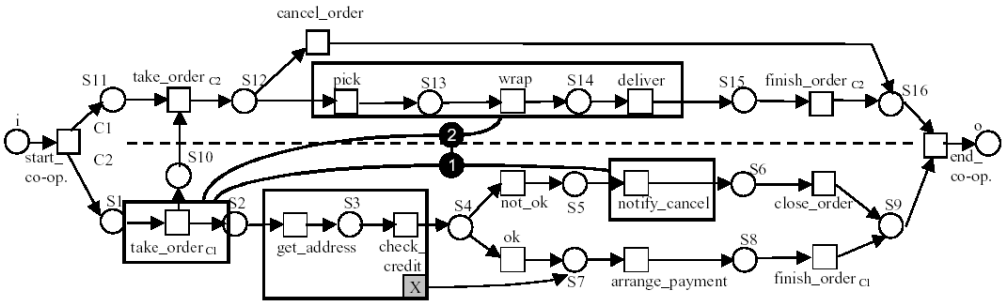


FIG. 13 – Exemple de procédé en réseaux de petri annoté par des sphères d'atomicité [Derk 01]

Le travail réalisé dans [Derk 01] permet de montrer très clairement la problématique de séparation des préoccupations entre la spécification du procédé et la spécification des propriétés telle que celle de l'atomicité dans ce cas bien précis. Derks et al. proposent de transformer les procédés annotés par des sphères d'atomoicité pour y exprimer, directement dans le modèle du procédé lui même, le comportement désiré. Cela a été possible pour l'atomicité car, en fin de compte, l'atomicité peut être réalisée en contrôlant l'agencement des activités et la préparation d'alernatives lors d'échec de certaines activités.

Néanmoins, la solution proposée impose une façon d'implémenter l'atomicité d'une sphère à l'aide de réseaux de petri. Une contrainte pèse lourd sur la modélisation de procédés annotés par des sphères d'atomicité est celle qui exige qu'une sphère ait un point d'entrée (en flot de contrôle) et un point de sortie. Cette contrainte est typique des modèle à base de réseaux de petri car ceux ci sont basés sur une gestion de l'exécution à l'aide du passage de jetons d'une activité à l'autre et cela implique que le nombre de jetons à l'entrée (en principe un seul pour le début du procédé) est égal au nombre de jetons en sortie (en principe un seul pour la fin du procédé). Nous voyons donc bien que les tentatives d'implémenter des propriétés comportementales, telle que l'atomicité, s'avèrent toujours peu fructueuses du fait que le modèle de procédé lui même handicape la sémantique ou l'expressivité de la propriété en question.

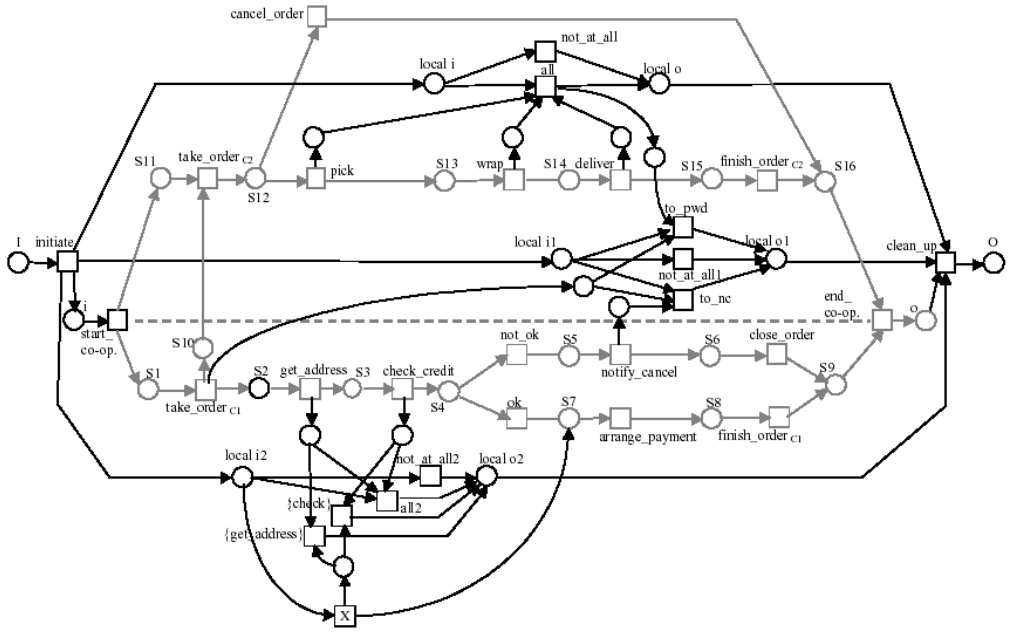


FIG. 14 – Équivalent en petri net d'un procédé annoté par des sphères d'atomicité [Derk 01]

2.3.2 Les sphères de compensation

L'approche des sphères de compensation [Leym 95] considère la compensation de groupes d'activités et non uniquement celle d'activités individuelles. De ce fait, cette approche rejoint celle des sphères d'atomicité que nous avons présenté précédamment. L'apport de telles sphères réside dans la possibilité de disposer d'un "procédé" de compensation pour l'ensemble des activités du groupe. Les activités d'une même sphère de compensation expriment naturellement le fait qu'en cas de compensation de toute la sphère, un procédé de compensation peut réaliser une telle compensation. Ce procédé de compensation est associée directement à la sphère comme illustré dans la figure 15.

2.3.3 Les "poches" de flexibilité (Pocket of Flexibility)

Dans [Sadi 05] les auteurs partent du constat que la rigidité des modèle de procédés métiers pose de sérieux problèmes en termes de flexibilité dans plusieurs champs potentiels d'application d'une telle technologie (santé, *e-learning*, CRM sont quelques exemples couramment cités). L'idée des auteurs est de définir les modèles de procédé de façon classique avec les mêmes constructeurs (activités, flot de contrôle ...). Puis, ils proposent d'ajouter des "poches" de flexibilité (Pocket of Flexibility) dans la définition du modèle de procédé. Une poche de flexibilité est un sous-ensemble d'activités du procédé où le modèle d'exécution est moins rigide car il n'est plus régi par les constructeurs classiques des procédés. L'exécution des activités d'une poche de flexibilité est alors régie à travers de nouvelles contraintes s'appliquant uniquement sur les activités de la poche (par exemple des contraintes de séquence ou encore des contraintes de bifurcation). Une poche de flexibilité peut être vue comme une sphère où le comportement du moteur d'exécution est régi par des contraintes plutôt que par les dépendances habituelles des procédés métiers. La

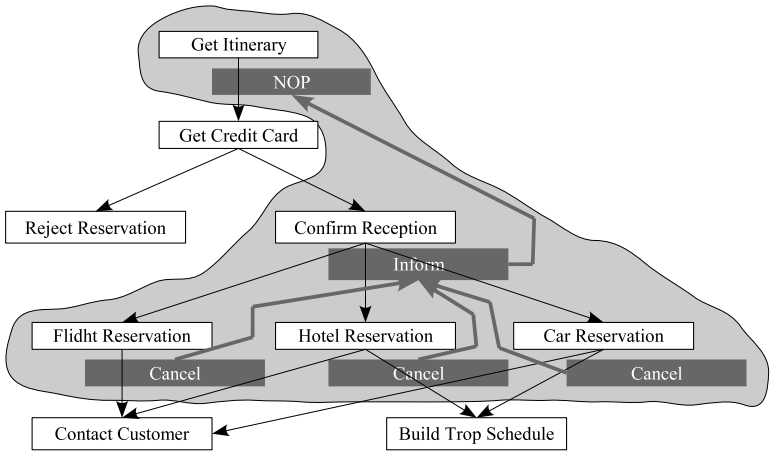


FIG. 15 – Exemple de sphère de compensation, selon [Leym 95]

définition des contraintes particulières des poches de flexibilité est indépendante de la définition du procédé lui-même, d'où une séparation des préoccupations entre la spécification du procédé et celle des transactions.

2.3.4 Synthèse

En synthèse, nous pouvons dire que les différentes approches d'encapsulation de sous ensembles d'activités pour leur affecter un comportement particulier se sont basés sur un principe simple, celui de la séparation des préoccupations entre la spécification du procédé en lui même et celle des comportements que ses activités doivent assurer. Notre travail consistera à unifier l'ensemble de ces approches pour en former une seule, que nous nommerons "sphères de comportement" et qui seront un candidat à une modélisation unifiée des comportements dans les procédés métiers. Nous préciserons comment une telle approche pourra être utilisée dans le cas particulier de l'isoaltion qui jouera le rôle d'une étude de cas dans cette thèse. C'est pour cette raison que nous nous proposons, dans la suite, de présenter une étude bibliographique de la gestion de la propriété transactionnelle d'isolation dans les procédés métiers.

3

La propriété transactionnelle d'isolation dans les procédés métiers

Sommaire

3.1 Les mécanismes de gestion de concurrence d'accès 39
3.1.1 Le concept de transaction 39
3.1.2 Impact de l'exécution concurrente de transactions sur la cohérence des données 40
3.1.3 Les solutions existantes aux problèmes de concurrence d'accès . . . 42
3.2 Les transactions dans les procédés métiers 47
3.2.1 Enjeux de la gestion transactionnelle des procédés 47
3.2.2 Le workflow transactionnel 48
3.2.3 Les transactions à longue durée d'exécution 56
3.2.4 Les MTA : tolérance aux fautes et flexibilité des contraintes transactionnelles dans les procédés métiers 57
3.3 Contraintes sur la coordination d'activités coopératives 61
3.3.1 Qu'est qu'une activité coopérative ? 61
3.3.2 Impact de l'isolation sur la coordination d'activités coopératives . . 61
3.3.3 Flexibilité de l'isolation d'activités coopératives 62
3.4 Synthèse et conclusion 62

3.1 Les mécanismes de gestion de concurrence d'accès

À l'origine, les mécanismes de gestion de concurrence d'accès ont été introduits dans les systèmes de gestion de bases de données. Ces mécanismes étaient définis autour du concept de transaction dont nous précisons le sens dans la section qui suit.

3.1.1 Le concept de transaction

La notion de transaction a pris naissance à partir de la volonté de n'appliquer à une base de données que des suites d'opérations qui en maintiennent la cohérence et la consistance. C'est dans ce but que le concept de transaction est introduit comme suit :

Définition 1 ***(Transaction)***
Une transaction est une suite d'opérations (lectures - écritures) effectuées sur une base de données

Les principes d'utilisation du concept de transaction sont les suivants :

1. L'interaction avec la base de données se fait exclusivement par l'intermédiaire de transactions qui en maintiennent la cohérence.
2. Le système de base de données gère les transactions de façon à en garantir les caractéristiques d'atomicité, d'isolation, de cohérence et de durabilité.

Les quatre caractéristiques de base souhaitées pour l'exécution d'une transaction, souvent résumées par l'acronyme A.C.I.D. correspondent à :

- L'atomicité : si une transaction est exécutée, elle doit l'être entièrement, c'est-à-dire de façon, en apparence, indivisible.
- La cohérence : l'exécution d'une transaction doit respecter les contraintes d'intégrité définies pour la base de données. Par exemple, dans le cas où la somme de données doit toujours rester constante, si la transaction décrémente l'une, elle devra incrémenter l'autre pour respecter la contrainte d'intégrité.
- L'isolation : il n'y a pas d'interaction entre transactions exécutées en parallèle. L'exécution de transactions parallèles devra paraître comme si elles étaient exécutées en série, c'est-à-dire sans aucune interaction.
- La durabilité : une fois une transaction exécutée, son résultat ne peut pas être perdu. Cela est nécessaire lorsqu'une transaction produit un résultat qui se retrouve écrasé par une autre transaction avant même qu'il ne soit exploité.

Généralement, les systèmes de gestion de base de données gèrent toutes les transactions en tant que transactions ACID, c'est-à-dire qu'elles respectent l'ensemble des quatre caractéristiques présentées ci-dessus. L'exécution de transactions ACID aboutit à une exécution dite sérialisable (ou séquentialisable), c'est-à-dire que les effets de leur exécution sont équivalants à leur exécution en série. L'intérêt de la sérialisabilité est d'autant plus important que l'exécution d'un ensemble de transactions $\{T_1, T_2, ..., T_n\}$ donne lieu à un ordonnancement des opérations de chacune d'entre elles. Dans un tel ordonnancement, les opérations de deux transactions peuvent être alternées, c'est-à-dire que leurs exécutions sont chevauchées.

Sans réel contrôle, notamment à travers des protocoles de gestion de concurrence d'accès, l'ordonnancement des transactions risque d'engendrer des incohérences au sein de la base de données. Ces incohérences ont été répertoriées en trois anomalies qui décrivent ce qui peut se produire quand deux transactions ou plus opèrent sur les mêmes données : l'anomalie de lectures impropres, de lectures non répétables et de lectures fantômes que nous nous proposons de présenter dans la section qui suit.

3.1.2 Impact de l'exécution concurrente de transactions sur la cohérence des données

Les anomalies dites SQL, inhérentes à une isolation insuffisante de transactions, sont étudiées en détail dans la spécification ANSI SQL-92 [ANSI SQL 92]. Pour illustrer ces anomalies,

prenons l'exemple de deux applications clientes séparées utilisant leurs propres instances d'un gestionnaire de voyage pour accéder aux mêmes données, en l'occurrence ceux de la disponibilité des places à bord d'un avion d'une compagnie aérienne. Le gestionnaire de voyage permet d'exécuter, entre autres, la méthode *réserver_siège()* permettant de réserver un siège dans un avion et la méthode *lister_sièges_libres()* permettant d'avoir la liste des sièges disponibles. Cet exemple manipule une table *RESERVATION* d'une base de données, à laquelle on accède par la méthode de *réserver_siège()* ainsi que la méthode *lister_sièges_libres()*. Les deux applications clientes consistent en deux transactions concurrentes.

Lectures impropres

Une lecture impropre se produit lorsque la première transaction lit les changements non validés effectués par la deuxième transaction. Si, entre temps, la deuxième transaction est annulée, les données lues par la première transaction deviennent non valides parce que l'annulation a défait les changements. La première transaction ne se rendra pas compte que les données qu'elle a lues sont devenues non valides.

Il s'agit du cas où une transaction utilise une liste non valide de sièges disponibles parce qu'un siège, par exemple le n°59, est disponible mais n'a pas inclu dans la liste. Cela pourrait causer des conséquences sérieuses si le siège n°59 était le dernier disponible car, dans un tel cas, la transaction rapporterait inexactement que le vol est complet. Le client essaierait vraisemblablement de changer de vol, voire d'abandonner la compagnie choisie pour se tourner vers la concurrence.

Lectures non répétables

Une lecture non répétable se produit quand la première transaction lit une donnée qui, avant la fin d'exécution de la transaction, est sujette à un changement opéré par une deuxième transaction. Les données lues par la première transaction ne sont alors plus valides et la poursuite de son exécution risque d'engendrer des incohérences. La première transaction, ayant lu la valeur d'une donnée, requiert alors la garantie que la donnée ne sera pas modifiée pendant sa durée d'exécution.

L'anomalie de lectures non répétables engendre une incohérence dans la base de données car, dans ce cas, deux voyageurs pouraient réserver le même siège. Ainsi, tout comme l'anomalie des lectures impropres, l'anomalie des lectures non répétables est à proscrire pour un système fiable.

Lectures fantômes

Une lecture fantôme se produit quand de nouvelles données sont insérées dans la base de données et que ces données supplémentaires ne sont pas discernables immédiatement par les transactions qui ont commencé avant l'insertion. Le scénario le plus classique est celui d'une première transaction qui exécute une requête sur une table de la base de données et récupère ainsi le résultat de la requête. Ensuite, une deuxième transaction insère dans la même table des données supplémentaires qui pourraient faire partie du résultat de la requête. La première transaction ne pourra pas prendre en considération les données supplémentaires nouvellement insérées.

L'anomalie de lectures fantômes tient son nom du fait que les nouvelles données insérées

passent inaperçues pour l'une des transactions alors que cela peut engendrer une incohérence de la base de données. Par exemple, il peut s'agir d'une part de l'application cliente de réservation de vols et d'autre part de l'application gérant l'affichage des places disponibles dans une agence de voyage. Le système devra s'assurer que l'occupation d'une place devra se répercuter sur le nombre de places affichées.

Il est important de proscrire les trois anomalies que nous venons de citer, autrement dit l'anomalie des lectures impropres, l'anomalie des lectures non répétables et l'anomalie de lectures fantômes. C'est pour cela que plusieurs solutions ont vu le jour pour arriver à éviter de telles anomalies et nous allons les discuter dans la section qui suit.

3.1.3 Les solutions existantes aux problèmes de concurrence d'accès

Vers la consistance transactionnelle

La satisfaction de ce qui est communément appelé "consistance transactionnelle" suppose que les résultats obtenus après l'exécution d'un ensemble de transactions concurrentes soient équivalants à ceux obtenus avec un ordonnancement série de ces mêmes transactions. Il est donc important de répondre à la question suivante : Existe-il un critère permettant de savoir si un ordonnancement d'un ensemble de transactions concurrentes est équivalant à leur ordonnancement série ? En répondant à cette question, les recherches en bases de données ont introduit le critère de sérialisabilité qui caractérise de tels ordonnancements. Il est ainsi commun de les appeler "ordonnancements sérialisés". Dans la section qui suit, nous allons présenter brièvement un tel critère et les différentes formulations qui existent dans la littérature.

Sérialisabilité de l'exécution

Lors de l'exécution d'un ensemble de transactions, la satisfaction de la consistance transactionnelle suppose que les résultats obtenus soient équivalants à ceux obtenus avec un ordonnancement série et donc que cet ordonnancement soit "sérialisable". La "sérialisabilité" est définie autour du concept d'opérations conflictuelles. Nous définissons d'abord ce dernier concept.

> **Opérations conflictuelles** : Chaque transaction est supposée être constituée d'opérations, généralement de type lecture ($R(x)$) ou écriture ($W(x, v)$), sur les objets (x) du système d'information sur lequel elle s'exécute. $R(x)$ renvoie la valeur de x et $W(x, v)$ affecte une nouvelle valeur v à l'objet x. Deux opérations a et b, exécutées respectivement par deux transactions T_1 et T_2, sont dites conflictuelles si l'exécution de a suivie de b est susceptible de conduire à des résultats différents de l'exécution de b suivie de a. La nature de l'opération (lecture ou écriture) influence la relation conflictuelle comme illustré dans la figure 16.

L'exécution d'un ensemble de transactions aboutit à un ordonnancement d'opérations conduisant à un mélange des opérations en provenance de ces différentes transactions. La sérialisabilité d'une exécution, c'est-à-dire celle de l'ordonnancement des opérations d'un ensemble de transactions, peut être définie à travers la notion d'opérations conflictuelles comme suit :

		Opération de la transaction T_1	
		lecture de x	écriture de x
Opération de la transaction T_2	lecture de x	pas de conflit	conflit
	écriture de x	conflit	conflit

FIG. 16 – Les conflits entre opérations sur un même objet

Définition 2 ***(Sérialisabilité (contraintes sur l'ordonnancement d'opérations conflictuelles))***
Un ordonnancement est sérialisable si et seulement si pour tout couple (T_i,T_j) de transactions, toutes les paires d'opérations conflictuelles ($a_{i,h}$,$b_{j,k}$) de ces transactions ($a_{i,h}$ étant la h^{ème} opération de la transaction T_i et $b_{j,k}$, la k^{ème} opération de la transaction T_j) sont exécutées dans un même ordre (toutes les opérations $a_{i,h}$ précèdent les $b_{j,k}$ ou l'inverse), l'ordre des opérations de transactions différentes n'entrant pas dans une paire conflictuelle étant sans importance.

Une deuxième formulation, basée sur la notion d'ordonnancement série, est également couramment utilisée dans la littérature :

Définition 3 ***(Sérialisabilité (l'équivalence à un ordonnancement série))***
Un ordonnancement est sérialisable si et seulement s'il peut être transformé en un ordonnancement série par permutations successives d'opérations ne constituant pas une paire d'opérations conflictuelles. Une telle transformation, si elle est possible, devra fournir un ordonnancement série équivalant. Un ordonnancement série correspond à celui où les transactions s'exécutent les unes après les autres sans aucun chevauchement de leurs exécutions.

Enfin, une troisième formulation, basée sur la notion de graphe de précédence est également couramment utilisée dans la littérature :

Définition 4 ***(Sérialisabilité (graphe de conflits acyclique))***
Un ordonnancement est sérialisable si et seulement si le graphe de conflits des transactions est un graphe sans cycle. Les transactions sont les sommets du graphe et un arc ayant a comme origine la transaction T_i et comme extrémité la transaction T_j s'il existe une paire conflictuelle d'opérations ($a_{i,h}$,$b_{j,k}$) respectivement de T_i et T_j et que $a_{i,h}$ est exécutée avant $b_{j,k}$ dans l'ordonnancement.

La sérialisabilité est une contrainte souvent difficile à vérifier, particulièrement dans le cas d'un grand nombre de transactions distribuées. Il est donc nécessaire de disposer de techniques évoluées permettant de garantir un tel critère de cohérence. C'est l'objet de la section qui suit.

Les techniques de verrouillage

La technique de verrouillage fait partie des techniques la plus anciennes et les plus couramment utilisées pour contrôler la concurrence d'accès à des objets partagés. Rappelons simplement les concepts de base attachés à cette technique. Une transaction utilisant une donnée d peut poser un verrou sur celle-ci afin d'en protéger l'accès. Un verrou appartient à l'une des deux classes suivantes :

- Verrous exclusifs : une fois un verrou exclusif posé sur une donnée, aucun autre verrou ne pourra être posé sur la même donnée tant que celui-ci ne sera pas libéré.
- Verrous partagés : autorise toute obtention d'autres verrous partagés sur la même donnée mais pas de verrous exclusif tant que tous les verrous partagés ne seront libérés.

Les bases de données, et pluparticulièrement les bases de données relationnelles, emploient généralement différentes techniques de verrouillage. Les plus communs sont les verrous en lecture (partagés), les verrous en écriture (exclusifs). Ces dispositifs de verrouillage commandent la manière dont les transactions accèdent, de façon concurrente, à des données partagées. Les règles définies dans le tableau 3.1 sont ainsi appliquées pour les demandes d'acquisition de verrous sur une donnée en fonction des verrous qui y sont actuellement posés par d'autres transactions.

	Verrou demandé	
verrou déjà posé	verrou en lecture	verrou en écriture
aucun	possible	possible
verrou en lecture	possible	impossible
verrou en écriture	impossible	impossible

TAB. 3.1 – Règles d'attribution de verrous de lecture et d'écriture

Dans le tableau 3.1, la mention "impossible" exprime le fait que la transaction effectuant la demande sera bloquée jusqu'à ce que les verrous l'empêchant soient libérés. Ainsi, la technique de verrouillage induit le risque d'interblocage, ce qui pose un problème non seulement dans les systèmes de bases de données mais également dans tout système de gestion de processus distribués. De plus la technique de verrouillage que nous venons de présenter ne suffit pas pour garantir une sérialisabilité de l'exécution d'un ensemble de transactions. Une méthode de verrouillage était nécessaire pour assurer la propriété de sérialisabilité aux exécutions concurrentes. Nous citerons d'abord la méthode de verrouillage strict à deux phases puis la méthode de verrouillage rigoureux à deux phases [Bern 87].

Verrouillage strict à deux phases : Cette méthode de verrouillage est basée sur les deux règles suivantes :

1. Si une transaction veut lire/écrire un objet, elle doit poser un verrou partagé/exclusif sur cet objet.

2. Tous les verrous exclusifs posés par une transaction doivent être libérés à la fin de l'exécution de la transaction et pas avant.

C'est grâce à ces deux règles que deux phases sont clairement identifiables, la première est celle de l'acquisition des verrous et la deuxième est celle de leur libération qui s'effectue uniquement à la fin de l'exécution de la transaction.

La méthode de verrouillage strict à deux phases permet d'empêcher la lecture de valeurs non validées, l'écrasement de données non encore validées, ainsi que les lectures non répétées. Ainsi, cette méthode permet d'éviter les annulations en cascade en cas d'annulation d'une transaction. Néanmoins, la méthode de verrouillage strict à deux phases n'assure pas la sérialisabilité de l'exécution des transactions. Pour cela, une autre méthode dite méthode de verrouillage rigoureux à deux phases, introduit des règles encore plus "rigoureuses" que nous présentons ci-dessous.

Verrouillage rigoureux à deux phases : Cette méthode de verrouillage est basée sur deux règles plus rigoureuses qui sont les suivantes :

1. Si une transaction veut lire/écrire un objet, elle doit poser un verrou partagé/exclusif sur cet objet.
2. Tous les verrous, à la fois exclusifs et partagés, posés par une transaction doivent être libérés à la fin de l'exécution de la transaction et pas avant.

Bien plus stricte que la méthode de verrouillage strict à deux phases, la méthode de verrouillage rigoureux à deux phases permet aux transactions d'être arrangées en série par l'ordre dans lequel elles sont validées. Sous 2PL rigoureux, tous les verrous (partagés et exclusifs) posés par une transaction doivent être maintenus jusqu'à ce qu'elle soit validée. Il est important de noter que la plupart des systèmes de base de données se contentent d'employer la méthode de verrouillage strict à deux phases.

La technique d'estampillage

Dans la section précédente, nous avons souligné que dans l'approche par verrouillage des données, l'ordre de sérialisation des transactions se construit "dynamiquement" au cours de l'exécution de ces transactions selon l'enchevêtrement des demandes de verrous. Avec l'approche par estampillage [Bern 80], un ordre de sérialisation est défini statiquement en se basant sur l'ordre de la date de création de chaque transaction. Pour ce faire, lorsqu'une transaction est démarrée, une estampille lui est attribuée. L'estampillage choisi devra garantir un ordonnancement total des transactions concurrentes. Il s'agit généralement de la date de lancement de la transaction (s'il existe un temps absolu de référence) ou d'une estampille sur la base d'une horloge logique partielle fournie par l'algorithme de Lamport [Lamp 78]. Ainsi, l'approche d'estampillage permet de garantir que deux transactions différentes ont des estampilles différentes. Les estampilles sont utilisées pour garantir que l'ordonnancement des transactions concurrentes est équivalant à l'ordonnancement série défini par l'ordre de leurs estampilles. Les opérations réalisées par les transactions (lecture ou écriture) sont elles-mêmes estampillées par l'estampille de la transaction correspondante.

L'estampillage est alors utilisé pour régler les conflits entre les opérations selon les règles suivantes :

- une lecture d'une donnée ne peut être satisfaite que si la dernière transaction ayant modifié cette donnée est plus ancienne ;
- une écriture d'une donnée ne peut être satisfaite que si les dernières transactions ayant modifié ou lu cette donnée sont plus anciennes.

L'approche optimiste ou par certification

L'approche optimiste, présentée dans [Kung 79], fait l'hypothèse optimiste qu'il n'y aura pas de conflit et donc suppose qu'en laissant les transactions s'exécuter sans aucun contrôle, on obtiendra une exécution sérialisable. Les dépendances sont mémorisées durant l'exécution. Ce n'est que lorsqu'une transaction se termine que le contrôle est réalisé lors d'une phase de certification de l'opération de validation "*COMMIT*". Si un conflit est détecté, la transaction est rejetée et devra être redémarrée. Cette approche autorise le maximum de concurrence et ne conduit à aucun interblocage mais elle est susceptible de provoquer de nombreux abandons après de coûteux calculs qui doivent être recommencés. Elle n'est raisonnable que si l'on sait que les conflits seront peu nombreux.

L'approche à niveaux d'isolation

L'approche à niveaux d'isolation est définie par rapport aux anomalies d'isolation (lectures impropres, lectures non répétables et lectures fantôme) que nous avons évoquées dans la section 3.1.2 de ce chapitre. Les niveaux d'isolation sont utilisés généralement dans des systèmes de base de données pour décrire comment la technique de verrouillage devra être appliquée aux données d'une transaction [ANSI SQL 92]. L'objectif est d'apporter, selon le niveau choisit, une certaine souplesse des contraintes d'isolation.

Les différents niveaux d'isolation identifiés ainsi que leur répercussions sur les anomalies d'isolation, illustrés dans le tableau 3.2, sont les suivants :

- Niveau 0 : niveau de lecture non validée (Read Uncommitted)
  La transaction peut lire des données non validées (données écrites par une transaction différente qui est encore en cours d'exécution).
  Les lectures impropres, les lectures non répétables et les lectures fantômes peuvent se produire.
- Niveau 1 : niveau de lecture validée (Read Committed)
  La transaction ne peut pas lire des données non validées (les données écrites par une transaction différente qui est encore en cours d'exécution ne peuvent pas être lues).
  Les lectures impropres sont impossibles, les lectures non répétables et les lectures fantômes peuvent se produire.
- Niveau 2 : niveau de lecture répétable (Repeatable Read)
  La transaction ne peut pas modifier une donnée qui est lue par une transaction en cours d'exécution.
  Les lectures impropres et les lectures non répétables sont impossibles, les lectures fantômes peuvent se produire.
- Niveau 3 : niveau sérialisable
  La transaction a un accès exclusif sur la donnée qu'elle manipule (lecture et écriture).

Différentes transactions ne peuvent ni lire ni écrire la même donnée de façon concurrente. Les lectures impropres, les lectures non répétables et les lectures fantômes sont impossibles.

	Anomalies d'isolation		
Niveau d'isolation	lecture impropre	lecture non répétable	lecture fantôme
lecture non validée	possible	possible	possible
lecture validée	impossible	possible	possible
lecture répétable	impossible	impossible	possible
sérialisable	impossible	impossible	impossible

TAB. 3.2 – Les niveaux d'isolation et leur répercutions sur les anomalies d'isolation

Le comportement exact de ces niveaux d'isolement dépend en grande partie du dispositif de verrouillage employé par la base de données ou la ressource.

L'ensemble des techniques et approches que nous avons présentées dans cette section ont été apportés au domaine des bases de données. Avec l'avènement du Business Process Management, les contraintes transactionnelles ont très vite intéressé les chercheurs et ont conduit à une alliance entre gestion transactionnelle et gestion de procédés.

Dans la section suivante, nous allons présenter l'état de l'art sur la gestion transactionnelle de procédés métiers, plus connue sous le nom de "workflow transactionnel" ainsi que leurs enjeux reconnus dans un tel domaine.

3.2 Les transactions dans les procédés métiers

Dans les systèmes de gestion de procédés métiers, tels que les systèmes de workflow, chaque activité est souvent assimilée à une transaction puisqu'elle revient à effectuer des opérations sur une base de données ou sur tout autre système. Cela met en avant des enjeux important de gestion transactionnelle des procédés métiers donnant lieu aux workflows transactionnels. Nous verrons dans la section qui suit que plusieurs visions sont proposées dans la littérature permettant de mieux comprendre les enjeux de la gestion transactionnelle des procédés métiers.

3.2.1 Enjeux de la gestion transactionnelle des procédés

L'engouement des chercheurs autour des aspects transactionnels dans les procédés métiers, notamment le workflow, s'explique par le fait que les entreprises et utilisateurs de systèmes de workflow expriment sans cesse leur intérêt à avoir une fiabilité accrue de l'exécution de leur procédés. Cependant la fiabilité a été souvent confondue avec la complétude, ce qui réduit les objectifs des systèmes de procédés à veiller à l'exécution de toutes les activités d'un procédé. La fiabilité d'exécution d'un procédé va au delà de la simple complétude, notamment, en prenant en considération les aspects transactionnels.

3.2.2 Le workflow transactionnel

Le terme "workflow transactionnel" a été introduit dans [Shet 93] afin de montrer l'importance de la gestion transactionnelle de l'exécution des procédés de workflows. Plusieurs chercheurs ont utilisé le workflow transactionnel et contribué à son aboutissement [Brei 93, Geor 95b, Kris 95, Rusi 95, Tang 95, Bhir 05]. L'objectif du workflow transactionnel est la coordination d'activités multiples accédant aux mêmes systèmes d'informations en respectant certaines propriétés transactionnelles au niveau de l'activité ou au niveau du workflow.

La gestion transactionnelle, utilisée durant les dernières décennies dans les systèmes de gestion de bases de données, a été combinée aux notions de workflow afin de satisfaire une meilleure fiabilité de leur exécution. Comme nous l'avons signalé, un procédé métier est composé d'activités qui sont souvent considérées comme des transactions. Néanmoins, cela n'est pas la seule vision. Une classification des approches de combinaison des notions de workflow et de transaction a été proposée dans [Gref 02b] que nous allons détailler dans la section qui suit.

Taxonomie des approches de workflow transactionnel

La classification des différentes approches de workflow transactionnel présentée dans [Gref 02b] a porté sur deux niveaux, le niveau conceptuel et le niveau architectural. L'étude fait apparaître deux classes principales. Au niveau conceptuel, les deux classes correspondant aux cas où les modèles de workflow et de transactions sont séparées ou intégrées. De même, au niveau architectural, les deux classes correspondent aux cas où les modules de gestion de workflow d'une part, et de transaction d'autre part, sont séparés ou bien fusionnés en un seul module dans l'architecture du système de gestion de workflow transactionnel.

Du point de vue conceptuel, la séparation des préoccupations revient à une séparation des langages de spécification des workflows et des transactions. Dans la première classe nous disposons d'un langage de définition de workflow "*WorkFlow Definition Language*" (WFDL) et d'un langage de définition de transactions "*TRansaction Definition Language*" (TRDL) séparés l'un de l'autre. Dans la seconde classe, un seul langage est utilisé et qui serait un langage intégré pour la spécification des concepts de workflows et de transactions "*Transactional WorkFlow Definition Language*" (TRWFDL).

Les deux classes principales identifiées sont subdivisées en six sous-classes comme illustré dans la figure 17. L'analyse des classes identifiées a été effectuée par l'auteur en se référant aux objectifs visés, à la façon de réaliser ces objectifs et aux avantages et inconvénients que cela engendre.

Le workflow transactionnel à modèles séparés

La première des classes principales identifiées est la classe des procédés transactionnels à modèles séparés. Dans cette classe, les aspects transactionnels et les aspects procédés (workflow) sont traités séparément, et sont modélisés séparément pour être combinés dans le but d'obtenir un procédé transactionnel.

L'intérêt de séparer la modélisation des workflows et des transactions réside principalement dans la volonté de séparer le traitement des contraintes de type flot de contrôle de celles d'ordre transactionnel (*separation of concerns*).

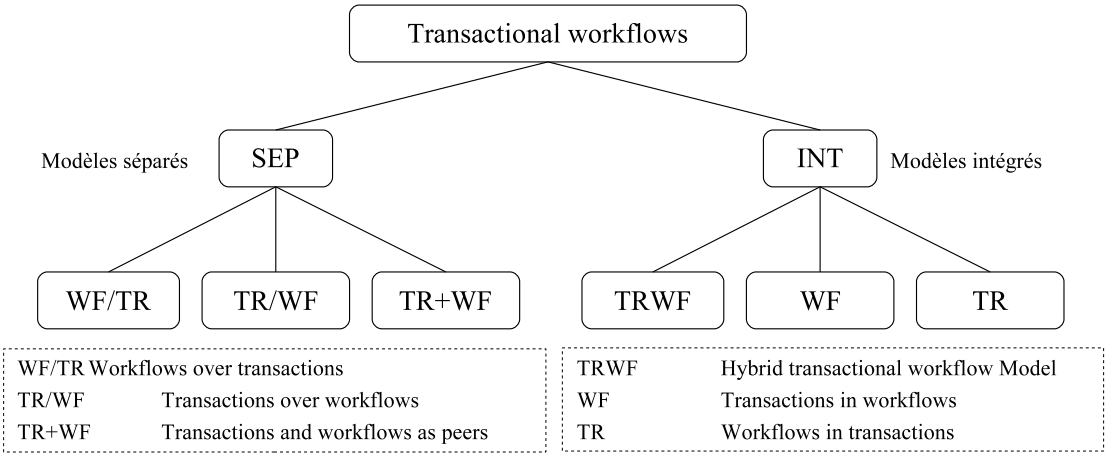


FIG. 17 – Taxonomie du workflow transactionnel selon Grefen (2002) [Gref 02b]

La distinction de trois sous-classes est basée sur la relation d'abstraction entre le modèle de procédé et le modèle de transactions. Du point de vue conceptuel, deux langages de spécification, l'un pour les workflows, l'autre pour les transactions, sont utilisés. L'auteur fait une distincion selon que l'un des langages est une extension de l'autre ou que les deux langages sont orthogonaux et qu'ils se complètent l'un l'autre. Ainsi, trois sous-classes sont identifiées, les deux premières correspondant au cas d'un langage extension de l'autre et la troisième correspondant au cas de deux langages mutuellement complémentaires. Les trois sous-classes identifiées dans [Gref 02b] sont décrites ci-dessous accompagnées d'exemples illustratifs de ce que les langages correspondant pourraient spécifier.

– *Workflows over transactions* (WF/TR) : les workflows sont plus abstraits que les transactions. Les modèles de transactions sont utilisés pour apporter une sémantique supplémentaire aux modèles de workflow. Ainsi, comme illustré dans l'exemple ci-contre, chaque activité (Task) est associée à une transaction. Les transactions permettent ainsi de réaliser les activités du workflow. La structure de flot de contrôle du workflow permet d'ordonnancer l'ensemble des transactions définies à travers l'ordonnancement des activités.

WFDL

```
TASK tsk1 TRANSACTION tr1
TASK tsk2 TRANSACTION tr2
SEQUENCE tsk1 tsk2
```

TRDL

```
BEGIN TRANSACTION tr1
USES d1,d2
READ d1, READ d2
WRITE d1, WRITE d2
IF statusOK
THEN COMMIT TRANSACTION
ELSE ABORT TRANSACTION
END TRANSACTION
...
```

– *Transactions over workflows* (TR/WF) : les transactions sont plus abstraites que les workflows. Les modèles de workflow servent de structure de procédé aux transactions. Ainsi l'exécution d'une transaction est régie par un workflow. Les propriétés transactionnelles, telles que l'atomicité dans l'exemple ci-contre, sont appliquées par la transaction sur l'exécution du workflow. Pour chaque transaction, il existera un workflow correspondant.

WFDL
```
WORKFLOW wf1
TASK tsk1, tsk2, tsk3, tsk4
SEQUENCE tsk1 tsk2
SEQUENCE tsk1 tsk3
SEQUENCE tsk2 tsk4
SEQUENCE tsk3 tsk4
END WORKFLOW
```

TRDL
```
TRANSACTION tr1
EXECUTE ATOMIC
IMPLEMENTATION wf1
END TRANSACTION
```

– *Transactions and workflows as peers* (TR+WF) : les transactions et les workflows existent à un même niveau d'abstraction. Workflows et transactions peuvent être vus comme deux sous-modèles séparés d'un modèle implicite de procédé. Dans l'exemple ci-contre, le modèle de workflows permet de définir la structure de procédé avec les activités (TASK) et le flot de contrôle alors que le modèle de transactions introduit d'autres activités, en l'occurrence les activités de compensation afin de garantir un retour à un point de cohérence (SAFEPOINT) en cas d'échec de l'exécution. Les deux modèles sont donc complémentaires et se positionnent au même niveau d'abstraction.

WFDL
```
WORKFLOW wf1
REFERS TRANSACTION tr1
TASK tsk1, tsk2, tsk3
SEQUENCE tsk1 tsk2
SEQUENCE tsk2 tsk3
END WORKFLOW
```

TRDL
```
BEGIN TRANSACTION tr1
REFERS WORKFLOW wf1
COMP ctsk1 tsk1
COMP ctsk2 tsk2
SAFEPOINT tsk1
END TRANSACTION
```

La syntaxe utilisée dans les exemples illustrant ci-dessus les différentes sous-classes ne représente pas un langage particulier mais offre une idée générale sur la structure de tels langages. Cette remarque est également valable en ce qui concerne les exemples pour le workflow transactionnel à modèle intégré présentés dans la section qui suit.

Le workflow transactionnel à modèle intégré

La deuxième des classes principales identifiées est celle où un seul modèle, dit intégré, permet de spécifier les aspects workflow et transactions. Dans une telle classe il n'y a pas de relation entre modèle de transactions et modèle de workflows. Le critère permettant de distinguer trois sous-classes est la nature même du modèle intégré. Ainsi, trois sous-classes sont distinguées comme suit :

– *Hybrid transactional workflow model* (TRWF) : un seul modèle hybride contient les concepts de transactions et de workflows. La définition d'une activité, comme dans l'exemple ci-contre, inclut la définition des concepts transactionnels tels que l'activité de compensation (COMP) et le point de cohérence (SAFEPOINT).

TRWFDL
```
WORKFLOW wf1
TASK tsk1 COMP ctsk1 SAFE-
POINT
TASK tsk2 COMP ctsk2
TASK tsk3 COMP none
SEQUENCE tsk1, tsk2
SEQUENCE tsk2, tsk3
END WORKFLOW
```

– *Transactions in workflows* (WF) : un seul modèle contenant les concepts de workflows et à travers lesquels les concepts de transactions sont exprimés. Aucun concept de transactions n'est explicite dans un tel modèle et l'utilisation adéquate des concepts de workflows permet d'exprimer ceux de transactions. Généralement, dans un tel modèle, l'expression de comportements transactionnels avancés est assez complexe à réaliser uniquement à l'aide des concepts de workflows (activité et flot de contrôle).

WFDL

```
WORKFLOW wf1
TASK tsk1, tsk2, tsk3
TASK ctsk1, ctsk2
SPLIT or1, or2
SEQUENCE tsk1 or1
SEQUENCE or1 tsk2
SEQUENCE or1 ctsk1
SEQUENCE tsk2 or2
SEQUENCE or2 tsk3
SEQUENCE or2 ctsk2
SEQUENCE ctsk2 ctsk1
END WORKFLOW
```

– *Workflows in transactions* (TR) : un seul modèle contenant les concepts de transactions à travers lesquels les concepts de workflows sont exprimés. Dans ce cas, aucun concept de workflow n'est explicite. Ils sont exprimés à travers les concepts de transactions en introduisant les dépendances ou le parallélisme entre transactions.

TRDL

```
TRANSACTION tr1
SUBTRANSACTION s1
action1 ; action2
END SUBTRANSACTION
SUBTRANSACTION s2
action3 ; action4
END SUBTRANSACTION
PARALLEL s1, s2
END TRANSACTION
```

Analyse des différentes classes de workflow transactionnel

Les six classes de workflow transactionnel que nous avons présentées dans la section précédente répondent toutes à des objectifs précis et ont chacune des avantages et des inconvénients. À chaque type d'application, une classe sera la plus adaptée en offrant certains avantages adéquats dans tel ou tel cas de figure. Il est donc nécessaire de récapituler les objectifs, les moyens pour les réaliser, les avantages et les inconvénients de chacune des classes de workflow transactionnel comme cela est fait dans [Gref 02b] : voir le tableau ci-dessous.

Classe	Objectif	Moyen	Avantages	Inconvénients
WF/TR	Robustesse du workflow	Gestion de données dans les workflow	Séparation des *concerns*, flexibilité, support	Manque d'intégration
TR/WF	Sémantique de type flot de contrôle pour les transactions	Gestion de procédés dans les transactions	Séparation des *concerns*, flexibilité	Manque d'intégration
TR+WF	Intégration des workflows et des transactions	Gestion couplé de données et de procédés	Séparation des *concerns*, flexibilité	Manque d'intégration et de cohérence
TRWF	Intégration des workflows et des transactions	Gestion hybride de données et de procédés	Intégration, cohérence	Formalisme complexe, manque de flexibilité
WF	Robustesse du workflow	Gestion avancée de procédés	Simplicité du formalisme, cohérence, support	Expressivité limitée
TR	Sémantique de type flot de contrôle pour les transactions	Gestion avancée de transactions	Simplicité du formalisme, cohérence	Expressivité limitée

TAB. 3.3 – Comparaison des classes de workflow transactionnel, selon [Gref 02b]

Ainsi, la flexibilité et la séparation des études entre d'un côté l'aspect procédé et de l'autre l'aspect transaction constituent la force des classes à séparation de modèles. Le revers de la médaille dans de telles classes est la difficulté à conjuguer les deux modèles, vu qu'ils sont séparés.

Les classes à modèle unique quant à elles, offrent l'avantage d'une cohérence entre aspects procédés et aspects transactionnels à travers notamment l'intégration dans un même modèle. Comme toute solution d'intégration, le manque d'expressivité est un inconvénient majeur. En effet, dans la classe TR par exemple, il est question d'exprimer des aspects transactionnels en utilisant uniquement les aspects procédé, ce qui est relativement délicat à réaliser en cas de sémantique transactionnelle complexe, si ce n'est impossible dans certains cas.

Il est important de noter que, du point de vue pratique, les classes de workflow transactionnel les plus facilement réalisables sont la classe TR+WF, car elle combine l'existant (WFMS[1] et TRMS[2]) et la classe WF car elle consiste en une utilisation judicieuse du système de gestion de workflow existant.

Après l'analyse d'une telle taxonomie, nous constatons que le workflow transactionnel est loin d'être unanimement défini car il vise principalement deux objectifs divergents.

Le premier objectif est de fournir des caractéristiques transactionnelles aux workflows afin d'en garantir une meilleure fiabilité. Dans un tel cas, la fiabilité est atteinte en introduisant des concepts transactionnels tels que l'atomicité, la compensation, etc, qui permettent de garantir un fonctionnement correct du point de vue transactionnel. Ce premier objectif répond efficacement à la demande de fiabilité de l'exécution des workflows. Il regroupe le "workflows over transactions" (WF/TR) et le "transactions in workflows" (WF).

Le deuxième objectif est d'apporter aux systèmes de gestion transactionnelles les fondations nécessaires en termes de structures de procédés afin d'en réaliser la fiabilité. Dans ce cas, la fiabilité recherchée est celle de l'exécution des transactions et non des workflows. Les concepts de workflows sont utilisés afin de garantir un tel objectif. Cet objectif regroupe le "transactions over workflows" (TR/WF) et le "workflows in transactions" (TR). Les deux classes "transactions and workflows as peers" (TR+WF) et "hybrid transactional workflow model" (TRWF) répondent aux deux objectifs car les modèles de workflow et de

[1] WFMS pour système de gestion de workflow
[2] TRMS pour système de gestion de transactions

transactions y sont complémentaires.

Du point de vue architectural, l'étude dans [Gref 02a] a révélé une classification similaire à celle que nous venons de présenter mais exprimant une sémantique différente. En effet, les classes d'architectures de systèmes de gestion de workflow transactionnel SGWFT sont au nombre de six, dont les trois premiers séparent le module de gestion de workflow MWF de celui de gestion de transactions MTR alors que les trois derniers fusionnent ces deux modules pour n'en faire qu'un seul. Ces différentes classes sont présentées comme suit :

Modules séparés :

– MWF/MTR : le MWF emploie l'interface du MTR pour créer un nouveau contexte de transaction puis effectue des manipulations de données dans un tel contexte. Dans cette classe, le MTR et le SGWFT sont souvent intégrés dans un environnement d'application de base de données. L'architecture MWF/MTR est utilisée dans la plupart des systèmes commerciaux.

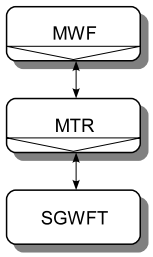


FIG. 18 – L'architecture MWF/MTR

– MTR/MWF : le MTR emploie l'interface du MWF pour créer un nouveau contexte de workflow et invoque les primitives de workflow. Cette classe d'architecture est particulièrement utile dans des environnements de commerce électronique où un moteur de transaction appelle des procédés soutenus par une technologie de gestion de workflow.

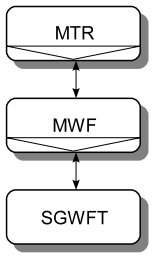


FIG. 19 – L'architecture MTR/MWF

– MTR+MWF : le MTR se comporte comme un serveur de transactions et le MWF se comporte en tant que serveur de procédé dans une relation pair-à-pair. L'interface entre le MWF et le MTR est strictement fonctionnelle : le MWF communique les états de procédé au MTR, le MTR communique au MWF les contextes de transactions et des commandes de workflow permettant d'effectuer les effets transactionnels sur les procédés.

Modules intégrés :

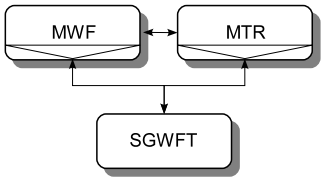


FIG. 20 – L'architecture MTR+MWF

– MTRWF : un tel module offre une prise en charge intégrée de la gestion de transactions et de workflows. Un état hybride de transaction et de workflow y est exprimé. Le module supporte des langages de type TRWF.

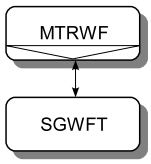


FIG. 21 – L'architecture MTRWF

– MWF : l'état d'un workflow transactionnel est entièrement maintenu par le module de workflow. Les attributs transactionnels de cet état sont traduits à travers les attributs de workflow. Le modèle de workflow MWF peut interpréter uniquement les caractéristiques dans la classe WF. C'est pour cette raison que d'autres caractéristiques transactionnelles d'autres classes (typiquement TRWF) doivent être traduites à travers les attributs de workflow.

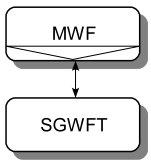


FIG. 22 – L'architecture MWF

– MTR : l'état d'un workflow transactionnel est entièrement maintenu par le module de transactions. Les attributs de workflow de cet état sont traduits à travers les attributs de transactions. Le modèle de transactions MTR peut interpréter uniquement les caractéristiques dans la classe TR. C'est pour cette raison que d'autres caractéristiques de workflow d'autres classes (typiquement TRWF) doivent être traduites à travers les attributs de transactions.

Dans la suite nous allons présenter les principaux travaux sur le workflow transactionnel en suivant la classification que nous venons d'étudier afin de mieux comprendre le contexte d'utilisation de chacune des classes évoquées.

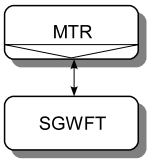


FIG. 23 – L'architecture MWF

Les principaux travaux sur le workflow transactionnel

En se référant à la taxonomie proposée dans [Gref 02b], nous pouvons citer, pour chaque classe, les principaux travaux réalisés. L'initiative Mercurius [Gref 98] se positionne dans le cadre de la classe "workflows over transactions" en plaçant la gestion des fonctionnalités de workflow au dessus de celle des fonctionnalités transactionnelles.

L'approche CrossFlow [Gref 00, Vonk 00] représente un excellent exemple de la classe "transactions over workflows". Dans l'architecture CrossFlow, la spécification des concepts transactionnels inter-organisations est exprimée à travers un contrat électronique qui est traduit en définitions de workflows. La perspective transactionnelle est alors placée à un niveau supérieur à celles du workflow.

Dans [Derk 01], un travail de séparation entre la spécification du procédé de workflow et celle des caractéristiques transactionnelles a été réalisé et l'approche proposée entre dans le cadre de la classe "transactions and workflows as peers" puisque les deux modèles se retrouvent au même niveau d'abstraction. Nous reviendrons plus loin sur ce travail de séparation des préoccupations car il pose les bases de ce que nous appellerons sphères de comportement.

Concernant les travaux sur les procédés transactionnels ayant choisi un modèle unique pour la spécification des concepts de workflow et de transactions, nous pouvons citer d'abord le travail réalisé dans [Leym 95] dans lequel l'auteur propose d'utiliser une structure de sphères transactionnelles pour la compensation partielle de procédés. La spécification des sphères transactionnelles s'effectue alors dans le procédé de workflow ce qui le classe parmi les approches "hybrid transactional workflow model" de la taxonomie. D'autres travaux peuvent faire partie de la classe "hybrid transactional workflow model" tels l'approche ObjectFlow [Hsu 98], l'approche Exotica [Alon 96] ou encore le projet FlowBack [Kiep 98].

Dans la classe "transactions in workflows", nous pouvons citer principalement les travaux dans lesquels les caractéristiques transactionnelles sont exprimées à travers des réseaux de petri [Dehn 01]. Le travail présenté dans [Derk 01] a également porté sur la spécification des caractéristiques transactionnelles, en l'occurrence l'atomicité, en utilisant des réseaux de petri. À travers les réseaux de petri, les caractéristiques transactionnelles se retrouvent exprimées à travers des concepts de workflows et rentrent parfaitement dans le cadre de la classe "transactions in workflows".

Enfin, dans la classe "workflows in transactions", nous citons le modèle ConTracts [Reut 95, Reut 97] qui propose un environnement particulièrement adapté aux transactions à longue durée d'exécution. ConTracts est un langage de type script permettant de regrouper un ensemble de transactions. Chaque transaction est considérée comme ACID[1]. Un script de planification de l'exécution des transactions est alors défini. L'approche TSME (*Transaction Specification and Management Environment*) [Geor 94, Geor 96] est une approche similaire rentrant également dans le cadre de la classe "workflows in transactions".

Une analyse assez élaborée de la place que prennent les différentes approches du workflow transactionnel dans la taxonomie présentée a été réalisée dans [Gref 02b]. Cela a permis d'aboutir à un récapitulatif schématique de la répartition des différents travaux sur le workflow transactionnel par rapport aux six classes de la taxonomie comme illustré dans la figure 24

[1] ACID : Atomique, Consistante, Isolée et Durable

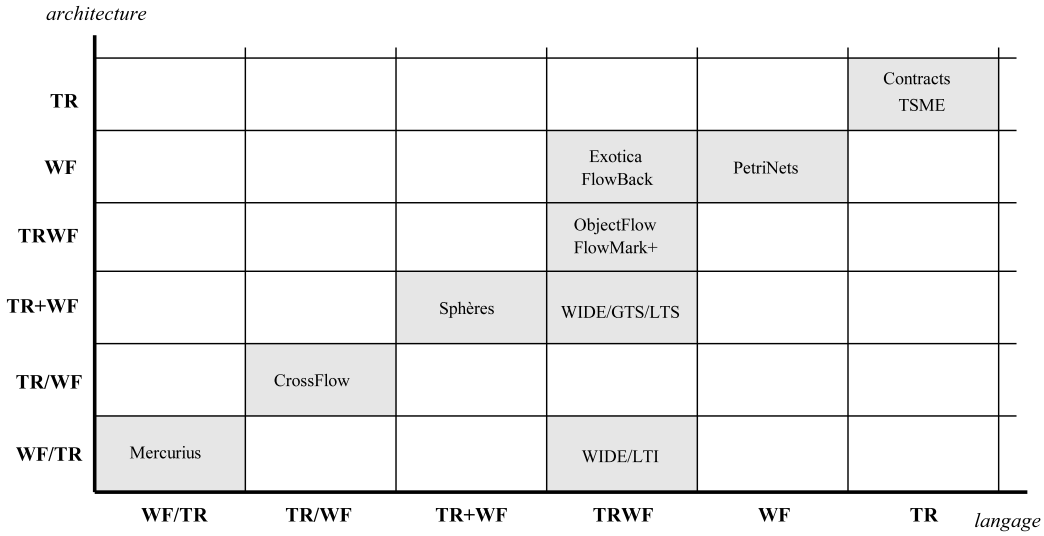


FIG. 24 – Répartition des différents travaux sur le workflow transactionnel par rapport aux six classes de la taxonomie proposée [Gref 02b]

Comme nous venons de le voir à travers les différentes utilisations de workflow transactionnels, un des enjeux émergeants est celui de la gestion de transactions à longue durée d'exécution. Nous allons voir plus en détail pourquoi ce type de transactions attire particulièrement l'attention des chercheurs et comment le workflow transactionnel, à travers les différents modèles de transactions avancés fournit différentes solutions aux problèmes qu'elles posent.

3.2.3 Les transactions à longue durée d'exécution

Les applications de workflow sont typiquement à longue durée d'exécution comparées aux transactions de bases de données. Dès le commencement du workflow transactionnel, un workflow est considéré comme une activité à longue durée d'exécution [Daya 90, Daya 91]. Une activité à longue durée d'exécution consiste alors en un ensemble d'opérations (unités d'exécution) qui peuvent consister récursivement en d'autres activités. Cette vision rappel les transactions emboîtées (nested transactions) des modèles transactionnels avancés. Dès lors, les modèles de workflow transactionnel proposaient la spécification statique (dans le script de l'activité) ou dynamique (sous forme de règles évènement-condition-action ECA) du flot de contrôle et de données.

La compensation, consistant à annuler l'effet d'une activité si besoin est, a été l'une des premières fonctions que les chercheurs en workflow transactionnel ont introduites. Néanmoins, face à la fonctionnalité devenue présente, il manquait un modèle de transaction avancé permettant de gérer l'exécution transactionnelle d'activités à longue durée d'exécution dans le procédé de workflow.

Motivés par les besoins des applications avancées en systèmes de workflow transactionnel, plusieurs Modèles de Transactions Avancés (MTA) ont été proposés, étudiés et comparés. Les principales références dans la littérature sont [Chry 91, Geor 94] pour les canevas[1] de définition et de comparaison des MTA, ou [Elma 92] pour disposer d'un ensemble représentatif de MTA, et aussi [Brei 93, Rusi 95] où des études sur la relation entre les MTA et le workflow sont présentées.

Les modèles de transactions avancés permettent de dépasser la traditionnelle transaction ACID prise

[1] Canevas du terme "framework" en anglais.

en compte dans la plupart des systèmes de gestion de bases de données. L'apport des MTA réside dans le fait qu'ils permettent la prise en charge des fonctionnalités exigées par les applications avancées telles que la collaboration entre activités, la coordination d'activités dans les workflows ad hoc, etc, tout en réduisant les délais de blocage dus à la synchronisation de transactions. Néanmoins, beaucoup de MTA sont proposés pour des applications spécifiques et offrent ainsi une réponse adéquate aux contraintes transactionnelles d'une application particulière et demeurent inefficaces dans d'autres cas.

Nous nous proposons dans la section suivante de présenter les principaux modèles de transactions avancés et de montrer leurs apports à la tolérance aux fautes et la flexibilité des contraintes transactionnelles dans les procédés métiers. Ensuite nous montrerons comment de tels MTA ne répondent pas de façon adéquate à la problématique de l'exécution transactionnelle des activités à longue durée d'exécution et pluparticulièrement à l'exécution d'activités coopératives.

3.2.4 Les MTA : tolérance aux fautes et flexibilité des contraintes transactionnelles dans les procédés métiers

Comme nous venons de le signaler, les modèles de transactions avancés ont été élaborés en réponse à un besoin de flexibilité dans l'exécution transactionnelle de procédés. En effet, les systèmes transactionnels étaient construits autour de transactions ACID dont l'exécution est très rigide. Durant l'exécution d'une transaction ACID, le système risque de bloquer l'exécution de toute autre transaction, de quelque nature qu'elle soit, si celle-ci utilise les mêmes données que la première, c'est le problème de blocage transactionnel. D'un autre côté, une transaction ACID peut s'exécuter pendant une longue durée puis échouer. L'échec d'une transaction ACID induit de façon systématique l'annulation de tout effet qu'elle aurait produit. Cela n'est pas très souhaitable si la durée d'exécution a été assez conséquente. Pour ces raisons, des modèles de transactions avancés ont été élaborés permettant une meilleure tolérance aux fautes et une flexibilité, plus ou moins grande, des contraintes transactionnelles dans les procédés métiers, et particulièrement les procédés d'activités à longue durée d'exécution.

Le modèle SAGA

La plupart des transactions dans les SGBD courants sont considérées comme des transactions de courte durée. Cette hypothèse, peu réaliste, consiste à considérer que l'exécution ou la compensation d'une transaction ne prendrait, au pire, que quelques secondes. Les transactions longues sont des transactions qui peuvent prendre des minutes, des jours, voire des semaines avant que les résultats de la transaction ne soient connus. Cette extension de la durée d'exécution d'une transaction est de plus en plus courante surtout quand des compagnies multiples participent à une transaction multi-parties. Elle se produit également quand la transaction attend un événement externe (comme une saisie de la part de l'utilisateur).

Le modèle SAGA [Garc 87] a été l'un des premiers modèles à traiter ce problème en apportant une certaine flexibilité de l'atomicité des transactions longues. Une SAGA est un ensemble de transactions compensables. Chaque transaction peut être exécutée puis validée et ses résultats sont rendus disponibles aux autres transactions. Si une des transactions d'une SAGA échoue, le reste des transactions validées de la même SAGA doit être compensé. Le modèle de transactions SAGA convient particulièrement aux scénarios de réservation car, généralement, l'utilisateur voudrait réserver l'ensemble vol, voiture, hôtel ou rien du tout. Si l'une des transactions (parmi "réserver vol", "réserver voiture" et "réserver hôtel") échoue, l'ensemble des transactions déjà validé doit alors être compensé.

Dans sa version la plus simple, une SAGA est une séquence de transactions T_1,T_2,...,T_n dont chacune peut être compensée respectivement à l'aide d'activités de compensation C_1,C_2,...C_n. L'exécution d'une SAGA est réalisée en suivant l'une des deux séquences suivantes :

- (T_1,T_2,...,T_n) si toutes les transactions T_i ($1 \leq i \leq n$) s'exécutent avec succès.

- (T_1,T_2,...,T_i,C_{i-1},C_{i-2},...,C_1) si T_i échoue et, au préalable, T_1,T_2,...,T_{i-1} ont été validées.

Un environnement d'exécution prenant en charge le modèle SAGA exécute les transactions d'une SAGA l'une après l'autre afin de respecter les contraintes définies ci-dessus. Ainsi, dès l'échec d'exécution d'une transaction, la compensation des transactions déjà validées est effectuée dans le sens inverse de leur exécution initiale.

Le modèle ConTracts

Le modèle ConTracts [Reut 97, Reut 89] permet de combiner les propriétés ACID et la fonctionnalité de compensation de transactions. Ce modèle est particulièrement destiné aux transactions à longue durée d'exécution. Dans le modèle ConTracts, une transaction à longue durée d'exécution consiste en un ensemble d'opérations ou d'étapes (*steps*). Chaque étape peut consister en n'importe quel programme ou action. Le modèle ConTracts fournit une variante du flot de contrôle pour définir explicitement la coordination des étapes.

Le problème auquel le modèle ConTracts est dédié consiste dans le fait qu'en cas d'échec d'une longue transaction, seules les étapes ayant fait défaut doivent être réexécutées et qu'il n'est pas adéquat de ré-exécuter l'ensemble des étapes d'une transaction si une partie a été exécutée avec succès. L'objectif est donc de préserver les étapes exécutées avec succès et de tenter de ré-exécuter uniquement celles qui ont causé un échec. D'un autre côté, une telle optimisation de la récupération de transactions longues en cas d'échec permet d'éviter la perte de travail, résultat des étapes réussies. La solution proposée est de définir des groupes d'étapes ayant des contraintes de type ACID. Nous verrons plus loin qu'une telle approche ressemble à l'approche des sphères d'atomicité [Leym 00, Derk 01, Heuv 02].

Toutefois, le modèle ConTracts ne reflète pas réellement une sémantique de workflow car il ne considère aucunement des instances de procédé, les flots de données, la répartition de rôles, ou encore l'élimination de chemins dits morts dont nous parlerons dans la section qui suit.

Le modèle de transactions flexibles

Le modèle de transactions flexibles [Elma 90, Mehr 92, Zhan 94] a été défini dans le contexte des transactions distribuées dans des environnements multi-bases de données hétérogènes. Dans de tels environnement, contrairement à une seule base de données locale où la compensation de transactions est unilatérale, les transactions distribuées agissent indépendamment les unes des autres et le contrôle de la sémantique de leur terminaison/compensation pose un problème majeur. Les transactions flexibles ont été proposées comme une solution possible à un tel problème en définissant explicitement la sémantique de terminaison de chaque activité individuellement. Chaque activité peut être compensable, rejouable ou pivot. Les effets produits par une activité compensable peuvent être sémantiquement annulés par l'exécution d'un service ou d'une activité de compensation. Cela constitue une garantie de compensabilité de l'activité. Une activité est dite rejouable lorsqu'elle offre la garantie de se terminer correctement au bout d'un nombre fini de tentatives. Ainsi, l'échec d'une telle activité ne doit pas remettre en question le bon déroulement de l'exécution du procédé car la réexécution de l'activité plusieurs fois s'il le faut aboutit avec garantie à un succès de celle-ci. Une activité est dite pivot quand elle n'est ni compensable ni rejouable. Une activité pivot offre la garantie de s'exécuter correctement dès la première tentative. A travers cette garantie, elle constitue une étape marquée du procédé car elle joue le double rôle de point de non retour, puisqu'elle n'est pas compensable et de point de cohérence puisqu'elle aboutit à un état cohérent sûr puisque sans risque d'échec.

En prenant en compte les garanties, la sémantique de terminaison de chaque activité et les possibilités de réexécution ou de compensation de chacune d'entre elles, il est possible de vérifier, à la phase de définition du procédé ou encore au cours de son exécution, qu'un procédé peut être exécuté correctement. C'est effectivement le cas quand tous les échecs pouvant se produire pendant l'exécution d'un procédé peuvent être résolus par la compensation complète ou par la compensation partielle jusqu'à un point du procédé d'où un chemin alternatif d'exécution peut être emprunté. Ces points du procédé correspondent

aux activités dites pivot qui ne peuvent être ni compensées ni réexécutées.

Ainsi, à l'aide du modèle de transactions flexibles, l'exécution de chemins représentant des alternatives possibles permet d'apporter une grande flexibilité de l'exécution. Si un chemin échoue, un autre est susceptible de prendre le relais et s'exécuter correctement. Cette vision est très utile : elle propose, par exemple, des exécutions alternatives, de la plus difficile à réaliser à la plus facile ou même la plus évidente. L'exécution tente les chemins les plus difficiles et en cas d'échec passe aux moins difficiles. Souvent, en utilisant le modèle de transactions flexibles, nous garantissons qu'il existe au moins un chemin où on est sûr de réussir ; il s'agit en pratique du chemin correspondant à la solution de secours.

Le modèle unifié de contrôle de concurrence et de recouvrement

Dans [Schu 02a], les auteurs ont proposé un modèle, inspiré du modèle de transactions flexibles [Elma 90, Mehr 92, Zhan 94]. Leur objectif était de proposer un critère de sérialisabilité d'exécution de procédés concurrents. Les auteurs ont, pour cela, proposé une architecture de système de gestion transactionnelle de procédés. Ils ont beaucoup contribué, dans le domaine, pour la formalisation des procédés transactionnels. C'est d'ailleurs ce travail qui nous a inspiré pour la réalisation du formalisme utilisé dans le chapitre de contribution de cette thèse. Le travail réalisé dans [Schu 02a] a permis d'étudier non seulement l'exécution de procédés mais également et surtout l'exécution de leur recouvrement. Le critère de sérialisabilité proposé (Process-Serialisability) permet d'assurer une exécution correcte d'un ensemble de procédés concurrents sur un même système de gestion de procédés.

Si le modèle proposé dans [Schu 02a] a permis de concrétiser un système de gestion transactionnelle de procédés métiers ainsi que leur recouvrement, il n'a pas fournit un niveau d'expressivité et de flexibilité dépassant les modèles précédents. Ce travail reste tout de même l'une des références en matière de gestion de la concurrence de procédés métiers tout en garantissant leur sérialisabilité.

Le modèle de transactions coopératives

Dans le modèle de transactions coopératives [Nodi 92], chaque activité se déroule au sein d'une transaction. Le domaine d'application particulier de ce modèle a été celui d'activités de conception. Ainsi les transactions sont structurées de manière hiérarchique afin de représenter la décomposition des activités en tâches et sous-tâches. De ce fait, les transactions coopératives rappellent les transactions emboîtées (*Nested Transactions*) [Moss 82] car elles présentent une vision à base d'arborescence. Dans le cas des transactions coopératives, l'arborescence est limitée à trois niveaux principaux, la racine, un ou plusieurs groupes transactionnels, et plusieurs transactions coopératives. Les transactions coopératives correspondent ainsi aux feuilles de l'arbre et sont regroupés en groupes transactionnels. Les transactions coopératives ne sont pas obligatoirement sérialisables car des *patterns* ou règles sur comment les opérations peuvent être intercalées sont définies dans chaque groupe transactionnel ainsi que des *conflicts* sous forme de règles qui spécifient quelles sont les opérations qui ne sont pas autorisées à s'exécuter de façon concurrente. Les *patterns* et les *conflicts*, totalement paramétrables en fonction de l'application ou de la situation, sont utilisés comme critère de correction des transactions coopératives.

Toutefois, le modèle des transactions coopératives oblige le designer du procédé à définir, par avance, les contraintes de type *patterns* et *conflicts*. De plus, l'ensemble des opérations possibles doit être à priori connu et fixe. Cela n'est pas toujours le cas de procédés coopératifs où le travail ainsi que les opérations à effectuer ne sont pas tout à fait structurées.

Le modèle des COO-transactions

Le modèle de transactions coopératives [Goda 96, Moll 96, Cana 98b, Grig 04] se propose d'étudier les problèmes de synchronisation ayant lieu lorsque des transactions manipulant les mêmes données sont exécutées de façon concurrente. L'environnement COO permet la synchronisation d'activités de développement en les encapsulant dans ce que les auteurs ont appelé COO-transactions. Les COO-transactions

permettent de rendre plus flexible le respect de la propriété d'isolation imposée par les transactions traditionnelles de type ACID tout en maintenant les autres propriétés.

L'environnement COO permet à des développeurs de partager leurs résultats intermédiaires. Ainsi chaque développeur travaille sur une copie locale de l'objet partagé, effectue des changements sur la copie locale puis place une version dite "intermédiaire" de l'objet ainsi modifié dans un espace de dépôt partagé avec les autres développeurs qui pourront alors utiliser ses résultats intermédiaires. Selon l'ordre d'exécution des transactions et grâce au respect d'un critère de correction, appelé COO-Sérialisabilité [Cana 98a], il est possible d'aboutir à une version stable et finale de l'objet manipulé. Le critère de COO-sérialisabilité consiste en une extension du critère classique de sérialisabilité en introduisant la prise en compte de la notion de résultat intermédiaire.

Pour un tel critère, une exécution est considérée comme correcte si elle respecte les règles suivantes :

- Si une transaction produit un résultat intermédiaire alors elle se doit de produire le résultat final correspondant lorsqu'elle est validée.
- Si une transaction T_1 utilise un résultat intermédiaire produit par une transaction T_2, alors T_1 doit lire le résultat final correspondant, produit lorsque T_1 est validée, avant de pouvoir produire ses propres résultats finaux.
- En cas d'échanges bidirectionnels, c'est-à-dire lorsqu'une transaction T_1 utilise un résultat intermédiaire produit par une transaction T_2 et la transaction T_2 utilise un résultat intermédiaire produit par la transaction T_1, les transactions impliquées, dans notre exemple T_1 et T_2, sont regroupés en une entité indivisible dont les transactions devront se terminer simultanément.

Trois politiques de coopération sont ainsi possibles à l'aide de la COO-sérialisabilité entre plusieurs transactions :

- Politique client/serveur : une transaction joue le rôle de serveur en produisant différentes version intermédiaires d'un objet dont certaines sont utilisées par d'autres transactions, jouant le rôle de clients. Cela permet aux transactions clientes de commencer leur travail avec des versions intermédiaires de l'objet utilisé. Le serveur est dans l'obligation de produire un résultat final et les clients sont dans l'obligation d'en tenir compte avant de produire à leur tour leurs propres résultats finaux.
- Politique d'écriture coopérative : deux ou plusieurs transactions s'échangent des résultats intermédiaires successifs qu'ils produisent mutuellement d'un même objet. Le résultat final de l'objet doit être atteint à l'aide d'un consensus entre les transactions impliquées.
- Politique rédacteur/relecteur : Une transaction joue le rôle de rédacteur en fournissant un résultat intermédiaire d'un objet et une autre transaction. Cette dernière joue le rôle de relecteur et utilise ce résultat intermédiaire pour produire un résultat intermédiaire d'un autre objet. Ce nouveau résultat sera lui-même utilisé par la première transaction. Cette politique est en fait similaire à la politique d'écriture coopérative.

Si le modèle des COO-transactions permet d'apporter une meilleure flexibilité de l'isolation dans les procédés transactionnels, il ne permet d'exprimer que les trois situations coopératives que sont le client/serveur, l'écriture coopérative ou le rédacteur/relecteur. Cela ne constitue qu'une partie des situations coopératives possibles. De plus, les interactions et la coordination entre les différentes transactions en présence sont régies par un même critère de correction qui est la politique d'écritures coopératives de la COO-sérialisabilité. Vue sous cet angle, dans un ensemble de transactions, la coopération entre deux d'entre elles ne peut pas se distinguer des autres en étant client/serveur ou rédacteur/relecteur.

Synthèse sur les MTA

En étudiant les différents modèles de transactions avancés, nous sommes arrivés à la conclusion suivante : les modèles de transaction avancés se sont focalisés essentiellement sur le respect d'une seule propriété transactionnelle qui est celle de l'atomicité. Certes la cohérence des données représente un

objectif primordial dans toute gestion transactionnelle mais les solutions évoquées sont toutes basées sur le principe de compensation qui découle naturellement de la volonté de réaliser un travail en sa totalité ou pas du tout, d'où le besoin de compenser les parties de ce travail ayant déjà été validé. Nous pouvons clairement distinguer dans les différentes approches que nous avons présentées dans cet état de l'art un penchant vers la gestion de l'atomicité au détriment de celle des autres propriétés ACID et en particulier de l'isolation.

Nous nous proposons, dans la section qui suit, d'étudier la question de l'isolation dans les procédés d'exécution d'activités coopératives. Ce champ d'application des procédés métiers est assez sensible à la question de l'isolation. En effet, nous verrons que la flexibilité de l'isolation serait une valeur ajoutée particulièrement pertinente pour de tels procédés.

3.3 Contraintes sur la coordination d'activités coopératives

Dans notre travail, nous nous intéressons plus particulièrement aux contraintes transactionnelles régissant la coordination d'activités. Ces considérations transactionnelles ont un impact assez important sur le travail coopératif et l'exécution d'activités coopératives. À propos des activités coopératives, il s'agit d'associer un comportement coopératif à une ou plusieurs activités d'un procédé. Il s'agit donc d'une contrainte comportementale pouvant influencer la coordination de ces activités. Nous verrons par la suite que le comportement coopératif n'induit généralement pas ou peu de contraintes sur la coordination d'activités. Néanmoins, il engendre certains problèmes de cohérence. Ces derniers sont évitables si on introduit des contraintes transactionnelles, et tout particulièrement des contraintes d'isolation.

3.3.1 Qu'est qu'une activité coopérative ?

Les activités coopératives dans les procédés sont souvent assimilées à des activités qui échangent leurs résultats intermédiaires. Du point de vue de l'exécution, une activité coopérative partage ses résultats sans attendre la fin de son exécution. Un tel partage des résultats fourni des possibilités de coopération importantes.

Nous nous sommes posé la question suivante : "*pourquoi les activités coopératives n'ont-elles pas été prises en considération dès le début des systèmes de gestion de procédés, tels que les workflow ?*". La réponse est assez simple. Dès les premiers systèmes de workflow, l'utilisation des procédés était exclusivement réservée aux procédés de production qui ne conçoivent pas la notion de résultats intermédiaires, puisque chaque activité ou tâche était atomique et isolée. Ces deux notions d'atomicité et d'isolation sont deux des quatre propriétés transactionnelles de base, connues sous l'abréviation A.C.I.D pour Atomicité, Cohérence, Isolation et Durabilité. Nous reviendrons plus longuement sur ces concepts dans la suite de ce chapitre, mais il est important de comprendre l'impact de l'isolation sur la coordination d'activités coopératives.

3.3.2 Impact de l'isolation sur la coordination d'activités coopératives

Le principe d'isolation d'une activité a_x consiste à empêcher, généralement par le biais de verrous, l'accès par d'autres activités aux données manipulées par a_x. Il s'agit de garantir que des données intermédiaires ne soient jamais dévoilées afin d'assurer une certaine cohérence des données de l'activité et des autres activités. L'isolation permet d'éviter ce qui est communément appelé "Lecture impropre". La lecture impropre correspond à la lecture d'une donnée qui n'est pas encore validée, c'est-à-dire à un résultat intermédiaire.

C'est en ce sens que l'isolation d'une activité peut être considérée comme l'inhibiteur de tout comportement coopératif puisqu'une activité isolée ne peut pas partager ses résultats avant qu'elle n'ait terminé son exécution. Ainsi, le partage des résultats intermédiaires n'est plus possible. Néanmoins, l'exécution

d'activités coopératives induit des problèmes de cohérence dus à la concurrence d'exécution. Ces problèmes sont inévitables si on n'a pas recours aux stratégies d'isolation les plus judicieuses pour autoriser le maximum de flexibilité d'isolation pour un comportement coopératif.

3.3.3 Flexibilité de l'isolation d'activités coopératives

La première et l'une des plus importantes des propriétés transactionnelle est sans aucun doute l'atomicité. Néanmoins, occulter toute autre propriété transactionnelle du champ d'action des modèles transactionnels ne ferait qu'induire les concepteurs d'application spécifiques dans l'erreur de confondre atomicité et isolation. En effet, l'isolation est souvent associée à l'atomicité. Or, cela est loin d'être toujours le cas puisque, hormis le fait que l'atomicité soit rendue plus flexible dans les modèles de transactions avancés que nous avons cités, l'isolation est demeurée peu étudiée. Dans de tels modèles de transactions, nous ne distinguons guère d'allusions à un quelconque flot de données ni à la gestion d'accès aux données manipulées. Certes, le modèle de COO-transactions évoque la gestion de l'accès aux résultats intermédiaires, mais cela ne constitue qu'une première ébauche, relativement timide, de la problématique de l'isolation dans les environnements d'exécution de procédés.

Avant de poursuivre notre recherche, il est nécessaire d'approfondir les enjeux de la gestion avancée, introduisant une plus grande flexibilité de l'isolation dans un environnement d'exécution de procédés et particulièrement celui de l'exécution d'activités coopératives.

3.4 Synthèse et conclusion

En tentant de synthétiser les apports des différents modèles de transactions avancés en termes de flexibilité des contraintes et de tolérance aux fautes, nous nous rendons vite compte du manque flagrant de traitement de l'isolation et d'un fort penchant vers la gestion de l'atomicité. Traditionnellement, l'atomicité est la première propriété transactionnelle et cela explique la focalisation sur celle-ci par de multiples chercheurs. Néanmoins, face à la limite des modèles existants en termes d'isolation, des travaux portant particulièrement sur la flexibilité de l'isolation ont permis d'aborder de face ce problème. Mais ces travaux introduisent des contraintes consistant d'une part en une organisation des activités coopératives sous forme d'arbre, excluant ainsi toute autre forme organisationnelle, et d'autre part en envisageant des politiques dans le partage des données, et donc dans l'isolation, qui se réduisent à des politiques client/serveur, écritures coopératives et rédacteur/relecteur. Cette dernière contrainte exclut toute autre forme de partage de données, ce qui aboutit au fait qu'une grande part des situations d'accès concurrentiel aux données n'est pas prise en compte dans la stratégie d'isolation.

À travers les constatations relevées dans les différents modèles de transactions avancés, nous pouvons donc conclure qu'il n'existe à ce jour aucune solution globale d'isolation capable de résoudre toute variété de situations d'accès concurrentiel à des données. Nous avons donc besoin d'apporter une nouvelle solution de flexibilité tant dans la modélisation de la propriété comportementale d'isolation que dans l'exécution de procédés exprimant une telle propriété. C'est ce que nous nous tenterons d'apporter dans le chapitre suivant.

Chapitre III

Contributions de la thèse

1. Sphères de comportement : vers une nouvelle façon de modéliser les procédés

2. Proposition d'une solution d'isolation flexible et orientée procédé

3. Apports des sphères d'isolation aux procédés coopératifs

4. Synthèse

1

Sphères de comportement

Vers une nouvelle façon de modéliser les procédés

Sommaire

1.1 Contexte de base d'exécution de procédés métiers 66
1.1.1 L'élément de base d'un procédé métier : l'activité (ou tâche) 66
1.1.2 Structure des procédés métiers 67
1.2 Notre vision des sphères de comportement 67
1.2.1 Premier principe : une séparation des préoccupations 69
1.2.2 Deuxième principe : un comportement d'ensemble 70
1.2.3 Troisième principe : une flexibilité et une expressivité accrues . . . 71
1.3 Identification de propriétés inappropriées à notre approche . . 71
1.4 Identification de propriétés favorables à notre approche 72

De nos jours, une volonté s'affirme pour l'unification des modèles de représentation et de modélisation de procédés métiers. La multiplication de standards et la volonté, à chaque fois, de prendre en charge différentes propriétés, tant organisationnelles, opérationnelles que transactionnelles sont les principales motivations des chercheurs. Néanmoins, les solutions proposées sont souvent très hétérogènes et aboutissent à des modèles de procédés également hétérogènes. Il devient alors difficile de fusionner les différentes approches pour constituer une approche commune.

Depuis quelques années, plusieurs travaux ont permis d'introduire des propriétés particulières dans les procédés métiers et cela a conduit à l'élaboration de diverses manières de les modéliser et de les exprimer. Certains adoptent l'approche qui consiste à ne pas bouleverser la structure habituelle du procédé. Il s'agit d'utiliser les éléments constituant la structure du procédé (activité, flot de contrôle, flot de données, opérateurs de base ...) afin d'exprimer de nouvelles propriétés ou de nouveaux comportements. Souvent, l'objectif est d'implémenter certains patrons de workflow [Aals 03] ou encore d'introduire des aspects transactionnels à l'exécution de workflow, ce qui a donné lieu au workflow transactionnel. D'autres travaux ont opté pour l'intégration de nouveaux éléments constituant le procédé, tels que des opérateurs exotiques ou des paramètres additionnels appliqués aux activités ou au procédé.

Notre approche, comme certaines que nous avons présentées dans l'état de l'art, adopte l'idée que spécifier une propriété particulière ne se réfère pas toujours à une activité de façon individuelle, mais plutôt à un groupe d'activités. Cela concernait d'abord des propriétés transactionnelles telles que l'atomicité ou la compensation. Ces propriétés ont une sémantique différente quand elles sont appliquées à un groupe d'activités. L'atomicité d'un groupe d'activités exprime le fait que toutes les activités du groupe doivent être réalisées, sinon aucune ne doit l'être. La compensation d'un groupe d'activités exprime le fait que toutes les activités du groupe peuvent être compensées "en groupe" par une seule activité de compensation.

Nous avons commencé à définir notre approche dès 2004 en l'appliquant, sans la formaliser, pour exprimer de façon plus expressive et plus flexible la propriété d'instanciation multiple qui régit l'exécution d'un groupe d'activités plusieurs fois, en parallèle ou en série. Cela donnait lieu à des sémantiques riches telles que les itération ou boucles dans les procédés. Ce travail a permis, à l'époque, de définir ce que nous appelions ensemble d'instanciation multiple [Guab 04] que nous avions aussi appelé "partitions multi-instances". Cette approche obéit parfaitement à la motivation de définir une propriété par rapport à un groupe d'activités et non à chaque activité individuellement.

Nous nous proposons dans ce chapitre de définir une référence commune à l'ensemble des travaux réalisés à base de sphères que nous appellerons "Sphères de comportement". Il s'agit de proposer une solution générique définissant les principes de base de la spécification des propriétés comportementales.

Nous définissons les principes fondamentaux d'utilisation de l'approche des sphères de comportement afin d'une part de garantir que l'approche en elle-même soit parfaitement assimilée et d'autre part d'éviter un usage inadéquat d'une telle approche. Plusieurs aspects comportementaux dans les systèmes de workflow peuvent être gérés à travers l'utilisation de sphères de comportement. Néanmoins, dans plusieurs cas, l'utilisation de l'approche des sphères de comportement n'a aucune valeur ajoutée et quelquesfois elle apporte beaucoup plus de complexité que de flexibilité.

L'objectif que nous nous fixons est que, pour une propriété donnée, selon la vérification des principes fondamentaux que nous allons définir, nous puissions savoir si l'utilisation de sphères de comportement est pertinente ou non.

Dans la section qui va suivre, nous allons présenter le contexte de base d'exécution de procédés métiers. Cela consiste à définir ce qu'est une activité, un procédé, ainsi que leurs instances. Nous pourrons alors nous référer à ces définitions pour introduire le concept de sphère de comportement.

1.1 Contexte de base d'exécution de procédés métiers

1.1.1 L'élément de base d'un procédé métier : l'activité (ou tâche)

L'exécution de procédés métiers tourne autour de l'ordonnancement de l'exécution d'activités. Une activité est une brique de travail permettant de traiter des informations, de prendre des décisions, etc. Elle est donc soit automatique soit humaine et beaucoup l'appellent "tâche" surtout dans les modèles de workflow. Nous commençons donc par fournir une définition de l'activité comme suit.

Définition 5 ***(Ensemble $\mathcal{A}$ des activités du système)***
Une activité est une entité de travail réalisable par un acteur et s'exécutant sur un même système d'information. $\mathcal{A}$ représente l'ensemble des activités du système. Une activité peut être automatique ou manuelle, de courte ou de longue durée d'exécution.

Une activité est généralement considérée comme la définition de ce que l'activité devra faire. Les systèmes de gestion de procédés gèrent également ce qui est appelé "instance d'activité" qui correspond à la version exécutée d'une activité. En effet, lorsque le concepteur de procédé définit des activités, leur exécution n'est pas unique et une activité peut être exécutée de multiples fois. C'est pour cela que la notion d'instance d'activité a été introduit afin d'assurer qu'il existe une instance pour chaque exécution et séparer ainsi leur devenir. Nous proposons de définir un tel concept comme suit.

Définition 6 ***(Ensemble $\widetilde{\mathcal{A}}$ des instances d'activités du système)***
Soit $a \in \mathcal{A}$, une instance $\widetilde{a}$ de l'activité a représente l'exécution de l'entité de travail a $\widetilde{\mathcal{A}}$ représente l'ensemble des instances d'activités du système.

1.1.2 Structure des procédés métiers

Les procédés métiers représentent la formalisation de l'organisation du travail dans les entreprises. Ils se structurent autour d'un ordre partiel sur des activités comme cela est défini dans la définition 7. Dans les systèmes de workflow, l'ordre partiel est représenté par une structure de flot de contrôle. Dans les organisations, les procédés sont définis pour la planification du travail d'équipes, de département, etc.

Définition 7 ***(Procédé)***
Un procédé P est un ensemble $(\mathcal{A}ct(P), \prec_P)$ où $\mathcal{A}ct(P) \subseteq \mathcal{A}$ est un sous-ensemble d'activités et $\prec_P$ est un ordre partiel minimal dans l'ensemble d'activités $\mathcal{A}ct(P)$ tel que $\forall\ a_n, a_m \in \mathcal{A}ct(P)$, $a_n \prec_P a_m$ exprime le fait que la fin d'exécution de a_n doit arriver avant le début d'exécution de a_m.
Notons $\prec_P^*$ ou encore $\prec^*$ la fermeture transitive de $\prec$.

Quand un procédé devient opérationnel, une instance de procédé est créée et gérée par le système de gestion de procédés jusqu'à sa terminaison. Les instances de procédés introduisent l'aspect exécutif de la gestion de procédés. Chaque procédé, une fois défini, peut être exécuté plusieurs fois moyennant plusieurs instances. Cette instanciation des procédés est largement adoptée par les systèmes de gestion de procédés métiers en raison de l'exploitation de procédés dont l'exécution est répétée périodiquement (exécution quotidienne de procédés de vérification, de production ...). Nous définissons de telles instances de procédés comme suit :

Définition 8 ***(Instance de procédé)***
Une instance I d'un procédé P est un tuple $(\widetilde{\mathcal{A}ct}(I), <_I)$ tel que $\widetilde{\mathcal{A}ct}(I)$ représente un ensemble d'instances des activités tel que :

$\forall\ a \in \mathcal{A}ct(P)$, $\exists \widetilde{a} \in \widetilde{\mathcal{A}ct}(I)$ une instance de a.

$\forall\ a_n, a_m \in \mathcal{A}ct(P)$, si $a_n \prec_P a_m$ alors $\widetilde{a_n} <_I \widetilde{a_m}$.

En nous basant sur les notions d'activité, de procédé et d'instances d'activités et de procédés, nous pouvons dores et déjà placer notre vision des sphères de comportement dans les procédés métiers. C'est l'objectif de la section qui suit.

1.2 Notre vision des sphères de comportement

L'idée première de l'approche des sphères de comportement est le regroupement de plusieurs activités d'un procédé autour d'une propriété particulière commune. Dès lors, les sphères de comportement peuvent être illustrées, conformément à la figure 25, comme des sous-ensembles d'activités d'un procédé, exprimant un comportement commun. L'apport d'une telle vision consiste dans la simplification du modèle de procédé de base, exprimant ainsi seulement les relations de précédence entre les activités. La

représentation des propriétés avancées s'effectue via une distribution de sphères de différentes natures sur les différents sous-ensembles d'activités du procédé de base.

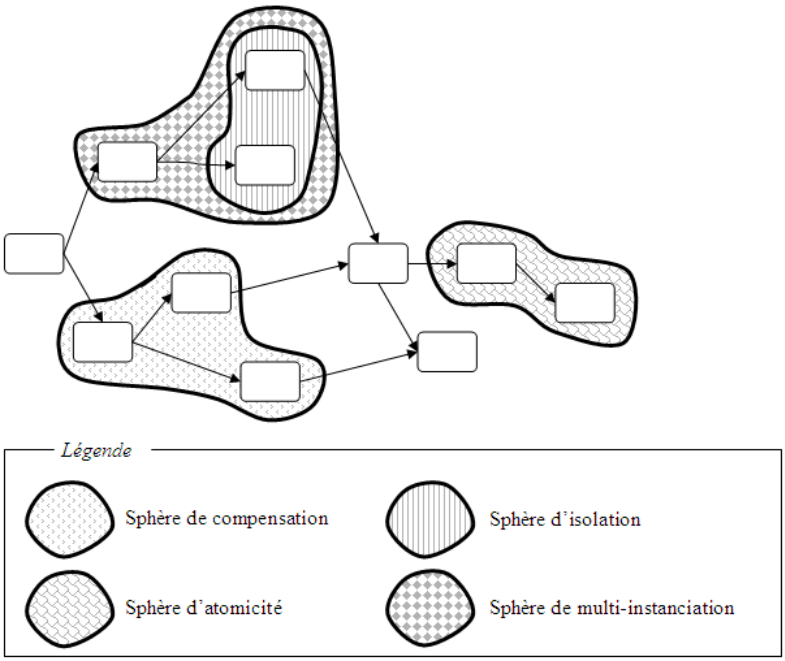


FIG. 25 – Utilisation de l'approche des sphères de comportement

Les sphères de comportement définissent certaines propriétés appliquées à l'ensemble des activités la composant et non à chacune d'elles séparément. Nous définissons la notion de sphère de comportement comme suit :

Définition 9 ***(Sphères de Comportement)***
Une sphère de comportement ω est un tuple $(\widetilde{\mathcal{A}ct}(\omega))$ représentant un ensemble d'activités ayant besoin d'assurer un comportement particulier. La flexibilité de la propriété comportementale est exprimée de façon spécifique au type de sphère à travers d'autres paramètres additionnels.

Dans la figure 25 nous illustrons plusieurs types de sphères d'ores et déjà définis dans la littérature, mais également la sphère d'instanciation multiple, que nous avons définie dans une précédente recherche, et la sphère d'isolation que nous nous proposons de définir dans le prochain chapitre de cette thèse. Les propriétés comportementales pouvant être définies à l'aide de sphères de comportement sont nombreuses mais il est important de ne pas se retrouver en train de définir des sphères pour toute propriété concernant des activités d'un procédé.

L'approche des sphères de comportement doit être utilisée afin de permettre une amélioration de la flexibilité des modèles de procédés. C'est pour cela que l'application des propriétés comportementales d'une sphère vise à être différente de leur application de façon individuelle à chaque activité. Il est donc nécessaire de définir des principes qui pourront guider l'adoption ou non de cette approche pour la réalisation de telle ou telle propriété comportementale à l'aide de sphères de comportement. Nous les avons résumés en trois principes fondamentaux :

Premier principe : une séparation des préoccupations

Séparation entre la spécification du procédé et celle des propriétés comportementales à assurer.

Deuxième principe : un comportement d'ensemble
Le comportement défini par une sphère et appliqué par les activités de la sphère en tant qu'ensemble ne produit pas les mêmes effets quand il est appliqué par chacune des activités de la sphère séparément.

Troisième principe : une flexibilité et une expressivité accrues
L'usage de sphères de comportement introduit une flexibilité et améliore l'expressivité comparativement aux autres approches.

1.2.1 Premier principe : une séparation des préoccupations

En appliquant ce premier principe, la spécification de comportements devient plus adaptée à la dimension procédé. Généralement, afin d'exprimer une propriété comportementale particulière dans un procédé, il est courant de voir cette propriété affectée à l'activité ou au procédé en entier. Ainsi, si nous reprenons l'exemple de l'atomicité (la propriété du "tout ou rien"), les modèles de procédés actuels qui la prennent en charge, l'intègrent en tant que propriété liée à l'activité. Pourtant, une telle propriété peut également concerner un groupe d'activités, sous ensemble des activités du procédé. Plusieurs autres propriétés comportementales ne peuvent être correctement exprimées à travers une telle approche que nous appellerons "traditionnelle". Ces propriétés sont particulièrement celles qui concernent un groupe d'activités et non une activité ou un procédé dans sa globalité.

L'enjeu qui en découle, en l'occurrence l'affectation de propriétés particulières à des groupes d'activités, ne peut être réalisé qu'à travers une séparation entre la définition du procédé et celle des propriétés comportementales. Cette séparation permet de considérer que la définition des sphères, pour un procédé donné, se fait indépendamment de sa structure ou de son modèle. Une seule contrainte importe, c'est le lien entre une sphère et les activités qui la composent. La figure 26 illustre bien l'architecture de prise en charge de sphères de comportement.

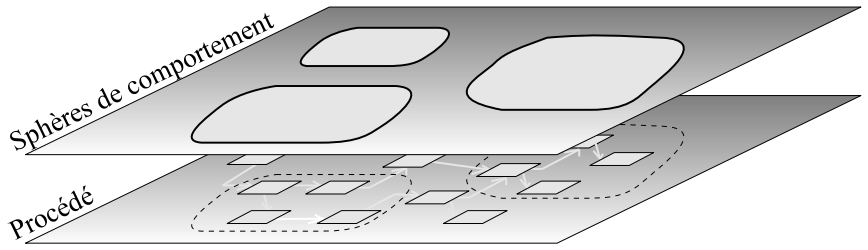


FIG. 26 – Architecture de prise en charge de sphères de comportement

Dans le cas de propriétés transactionnelles, ce premier principe fait référence à la taxonomie de workflow transactionnel dans laquelle deux classes principales de workflow transactionnel sont identifiées : celle des modèles séparés et celle du modèle intégré. Il est clair que l'approche des sphères de comportement se situe dans la classe des modèles séparés car la spécification des propriétés comportementales, en l'occurrence transactionnelles, ne se fait pas dans le modèle de procédé mais plutôt dans une couche supérieure, celle des sphères.

1.2.2 Deuxième principe : un comportement d'ensemble

Quand une propriété comportementale est définie pour un groupe d'activités, le deuxième principe assure le fait que le comportement effectif observé du groupe d'activités est différent de celui observé lors de la définition traditionnelle de la même propriété, à chacune des activités du groupe prise séparément. Afin d'exprimer plus clairement ce que ce principe suggère, nous pouvons établir la similitude avec une comparaison mathématique basique :

Soit a et b deux réels, nous avons besoin d'exprimer l'opérateur carré (puissance de 2) sur l'expression $a + b$. Syntaxiquement, cela peut s'exprimer de deux façons différentes :

$$a^2 + b^2 \qquad \text{ou} \qquad (a + b)^2$$

Ces deux notations produisent des effets différents. L'opérateur carré ^{2} peut être assimilé à une propriété que nous voulons exprimer dans un procédé. Les réels a et b peuvent être assimilés à deux activités du procédé. Nous voudrions soit définir la propriété comportementale pour le groupe d'activités composé de a et de b, soit pour a et pour b séparément.

L'objectif du deuxième principe est d'écarter les propriétés n'ayant pas un sens particulier lorsqu'elles sont appliquées à un groupe d'activité. Afin d'illustrer un tel principe, prenons l'exemple de la propriété comportementale d'instanciation multiple. Soit l'exemple de la figure 27 consistant en un procédé de remplissage de cartons. Une première activité se charge d'ouvrir un carton, une deuxième se charge de placer un jeton dans le carton et une troisième se charge de fermer le carton. Considérons maintenant que la propriété d'instanciation multiple, avec 2 instances requises, soit appliquée à chacune des activités de façon individuelle. La première activité est instanciée deux fois, ce qui revient à ouvrir deux cartons. La deuxième activité est également instanciée deux fois, ce qui revient à placer deux jetons, mais pas forcément dans deux cartons différents. Enfin, la troisième activité est instanciée deux fois, ce qui revient à fermer deux cartons.

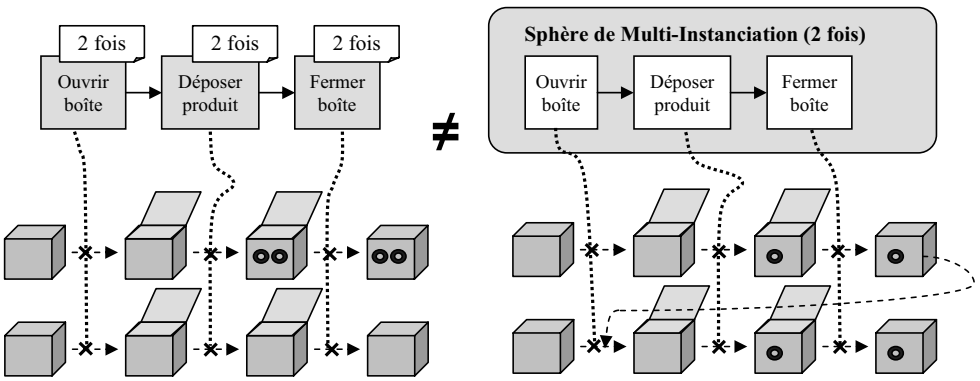


FIG. 27 – Cas de l'instanciation multiple respectant le deuxième principe des sphères de comportement

Considérons maintenant que la propriété d'instanciation multiple, avec 2 instances requises, soit appliquée à l'ensemble des trois activités en tant qu'entité unique (une sphère). La séquence d'exécution des trois activités sera alors instanciée trois fois : la cohésion de l'ensemble des trois activités sera respectée et l'ordonnancement sera cohérent. C'est ainsi que nous constatons une différence de comportement entre les deux cas de figure, ce qui place l'instanciation multiple comme une propriété comportementale de groupe respectant le deuxième principe des sphères de coopération.

1.2.3 Troisième principe : une flexibilité et une expressivité accrues

Le troisième principe permet d'assurer que l'utilisation des sphères de comportement ne s'effectue pas au détriment de la flexibilité et de l'expressivité, l'objectif principal des sphères de comportement étant d'améliorer l'expressivité des procédés métiers et d'améliorer la flexibilité de leur exécution.

Prenons l'exemple des propriétés transactionnelles telles que l'atomicité ou l'isolation. Définir, à travers une sphère, l'atomicité d'un groupe d'activités apporte une grande expressivité car, nous pouvons non seulement définir l'atomicité d'une activité ou de l'ensemble du procédé, mais aussi celle d'un sous-ensemble d'activités. La flexibilité est également au rendez-vous puisqu'une sphère d'atomicité, par exemple, peut voir sa composition modifiée au cours de l'exécution selon les besoins. Une activité faisant partie d'une sphère d'atomicité peut en être retirée si les besoins changent. Une activité peut également devenir membre d'une sphère si elle le souhaite. Cela procure une grande flexibilité.

La question qu'il faut se poser est la suivante : quelles sont les propriétés comportementales intéressantes à exprimer au travers de sphères de comportement et pourquoi l'utilisation des sphères de comportement est-elle utile ? Pourquoi apportent-t-elle une meilleure expressivité et une plus grande flexibilité dans de tels cas ?

Afin de vérifier l'intérêt d'utiliser l'approche des sphères de comportement, nous avons besoin de vérifier, pour une propriété comportementale donnée, les trois principes présentés précédemment. Nous présenterons, dans ce qui suit, quelques exemples de propriétés comportementales pour lesquelles l'approche des sphères de comportement est soit appropriée soit non appropriée, selon la vérification des principes que nous avons proposés.

1.3 Identification de propriétés inappropriées à notre approche

Nous présentons dans cette section quelques propriétés pour lesquelles au moins l'un des trois principes des sphères de comportement n'est pas vérifié. Pour ces propriétés l'utilisation de l'approche des sphères de comportement n'est donc pas conseillée du fait qu'elle n'est pas pertinente.

Distribution des rôles : la distribution des rôles dans un procédé est assez connue dans les procédés de workflows et consiste à associer un rôle à chaque activité. Plusieurs personnes jouent un même rôle et le principe est que la personne la plus disponible pourrait réaliser la tâche associée à ce rôle. Dans les systèmes de workflow, l'affectation des rôles est intégrée au modèle de procédé utilisé et s'y situe au niveau de l'activité, c'est-à-dire que le rôle constitue un paramètre de l'activité. C'est pour cette raison que nous voudrions utiliser les sphères pour séparer la spécification des distributions de rôles de celle du modèle de procédé. Néanmoins de telles sphères ne respecteraient pas tous les principes des sphères de comportement que nous avons établis. En effet, le deuxième principe n'est pas valide car il n'existe aucune différence sémantique ou différence de l'impact sur l'exécution entre affecter un rôle A à une groupe d'activités et affecter un rôle A à chacune des activités séparément : au final, toutes les activités du groupe seront exécutées par des personnes du rôle A. C'est pour cette raison que nous considérons la distribution des rôles comme une propriété non inscrite dans les perspectives de l'approche des sphères de comportement. La figure 28 illustre l'équivalence entre la distribution classique des rôles, activité par activité et celle utilisant l'approche à base de sphères.

Non exécutable : Dans une multitude de cas, nous avons besoin de désactiver une partie d'un procédé. Cela est similaire au fait de désactiver une partie d'un code source en le mettant en commentaire. Il serait intéressent d'utiliser les sphères dans un tel cas mais, en vérifiant les principes un à un, nous nous rendons compte que le deuxième principe n'est pas vérifié. En effet, il n'existe aucune différence, d'un point de vue exécutoire, entre désactiver un groupe d'activités et désactiver les activités du groupe une à une. Le résultat final demeure le même. C'est pour cette raison que nous considérons la désactivation d'activités comme une propriété non inscrite dans les perspectives de l'approche des sphères de comportement.

Plusieurs autres propriétés donnent l'impression d'être compatibles avec l'approche des sphères de

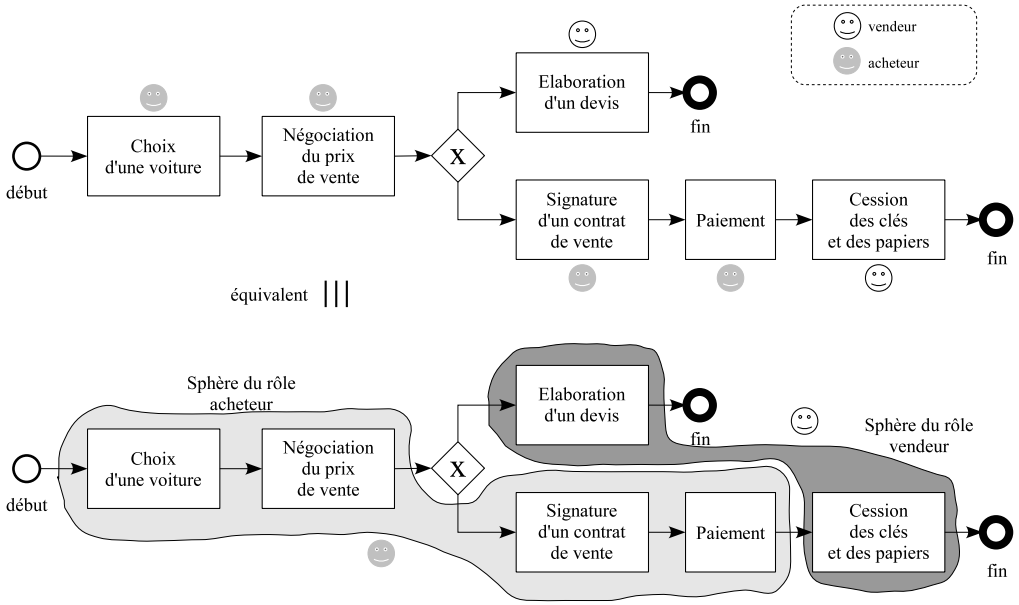


FIG. 28 – Cas de la distribution des rôles non conforme au deuxième principe des sphères de comportement

comportement mais il est essentiel d'analyser la validité des trois principes des sphères de comportement. Nous nous proposons par la suite de donner quelques exemples de propriétés prometteuses, pour lesquelles les principes des sphères de comportement sont vérifiés.

1.4 Identification de propriétés favorables à notre approche

Hormis la propriété transactionnelle d'isolation qui est la propriété dont nous étudierons, en détail, la réalisation à travers l'approche des sphères de comportement, nous avons identifié un certain nombre de propriétés comportementales diverses dont la réalisation à travers l'approche des sphères de comportement serait sans aucun doute d'un intérêt particulier.

Sphères de délai critique d'exécution : nous proposons d'étudier le problème associé à la gestion des délais critiques d'exécution (critical response time) et nous pensons qu'il serait intéressant d'introduire des sphères qui définissent le délai maximum autorisé pour la terminaison d'un groupe d'activités.

D'abord, la définition d'une telle propriété pour un groupe d'activité et non pour les activités individuelles séparément aboutit par évidence au respect du premier principe des sphères de comportement. Ensuite, si nous considérons un exemple de sphère composée de dix activités, le fait que le groupe de dix activités doive terminer son exécution en moins de dix minutes n'aura pas le même impact que le fait que chaque activité doive se terminer en moins d'une minute.

En effet, nous pouvons, dans le premier cas, admettre l'exécution d'une activité en cinq minutes et 30 secondes et celle des neuf autres activités en 30 secondes chacune. L'exécution du deuxième cas (sans sphères) n'autorisera pas une exécution pareille. C'est pour cette raison que le deuxième principe est valide.

Enfin, en ce qui concerne la contribution de la sphère à la flexibilité et l'expressivité, cela est également valide pour de telles sphères puisque nous pourrons exprimer des contraintes plus flexibles sur la durée maximale d'exécution de plusieurs activités, ce qui n'était pas possible auparavant.

Sphères d'affectation de personnes aux rôles : une autre propriété intéressante à spécifier en utilisant des sphères de comportement est celle de l'affectation des personnes aux rôles. Nous utilisons déjà, dans les systèmes de workflow, la distribution des rôles mais chaque rôle est relié à un certain nombre de personnes disponibles et, généralement, le système de gestion de workflow tente de chercher une personne qui soit disponible et corresponde à un rôle particulier.

Dans certains cas, le modélisateur du procédé peut avoir besoin d'assurer, pour un groupe d'activités requérant le même rôle, l'attribution de la même personne indépendamment de ses disponibilités, quitte à ralentir l'exécution du procédé. Nous proposons pour cela d'utiliser des sphères d'affectation de personnes aux rôles qui permettront de gérer une telle affectation en prédéfinissant ou en choisissant une fois pour toute au moment de l'exécution une personne pour l'ensemble des activités du groupe. Une telle sphère va simplement vérifier à chaque exécution d'activité de la sphère que la personne choisie est bel et bien la même que pour les autres activités de la sphère qui ont déjà été exécutées.

Sphère de séparation des fonctions En opposition au problème d'affectation des personnes aux rôles, il serait possible d'exprimer, pour un groupe d'activités ayant des rôles quelconques, l'affectation d'une personne différente pour l'exécution de chacune des activités du groupe. Ce cas est fréquent dans les procédés métiers réels, et nous pouvons l'illustrer par un exemple.

Supposons qu'une personne soit à la fois banquier et client de la même banque. La banque voudra spécifier le fait qu'un tel banquier ne pourra pas gérer son propre compte et cela pour des raisons de sécurité. Ainsi, un tel banquier pourra, sans problème, éditer une demande de prêt lui-même mais il ne pourra pas répondre favorablement à sa propre demande : un autre banquier pourra le faire. Nous pouvons ainsi concevoir, comme illustré dans la figure 29, l'idée d'une sphère encapsulant les deux activités "édition d'une demande de prêt" et "évaluation de la demande de prêt" qui ne pourront pas être opérées par un même banquier. Une telle sphère respecte de façon parfaite les trois principes des sphères de comportement.

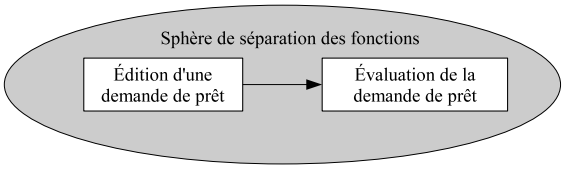


FIG. 29 – Sphère de séparation des fonctions

Après avoir présenté un certain nombre de propriétés aptes à faire l'objet d'une modélisation à l'aide de sphères, nous choisirons de nous focaliser sur une propriété particulière, celle de l'isolation, et d'étudier les caractéristiques d'une possible sphère d'isolation ainsi que son fonctionnement. C'est l'objet de la section qui suit.

2

Proposition d'une solution d'isolation flexible et orientée procédé

Sommaire

2.1 Contexte, enjeux et objectifs de l'étude de l'isolation dans les systèmes de gestion de procédés 76
2.1.1 L'isolation dans les systèmes de bases de données transactionnelles 76
2.1.2 L'isolation dans les systèmes de gestion de procédés 77
2.2 L'approche des sphères d'isolation 77
2.2.1 Présentation du formalisme utilisé et des concepts de base 77
2.2.2 Éléments de base d'une sphère d'isolation 83
2.3 La cohésion des sphères d'isolation 85
2.3.1 Le niveau de cohésion à lecture non validées (Read Uncommitted) . 85
2.3.2 Le niveau de cohésion à lectures validées (Read Committed) 86
2.3.3 Le niveau de cohésion à lecture répétables (Repeatable Read) . . . 86
2.3.4 Le niveau de cohésion sérialisable 87
2.4 La cohérence des sphères d'isolation 89
2.4.1 Le niveau de cohérence coopérative 90
2.4.2 Le niveau de cohérence d'activité 90
2.4.3 Le niveau de cohérence de sphère 91
2.5 Gestion avancée des sphères d'isolation : l'emboîtement de sphères 99
2.5.1 Enjeux de l'emboîtement de sphères d'isolation et règles de base . . 100
2.5.2 Cas de l'Intra-Sphère-Sérialisabilité et l'emboîtement de sphères . . 101
2.5.3 Cas de l'Extra-Sphère-Sérialisabilité et l'emboîtement de sphères . . 102
2.5.4 Discussion à propos des possibilités de chevauchement de sphères . 103
2.6 Synthèse 104

Nous avons clarifié, dans le chapitre précédent, notre vision des sphères de comportement. Nous nous proposons dans ce chapitre de contribuer à enrichir les approches à base de sphères en présentant la notion de sphères d'isolation. Il s'agit d'une solution flexible permettant une meilleure gestion de la concurrence d'exécution d'activités dans les systèmes de gestion de procédés et nous allons étudier l'impact d'une telle solution sur l'exécution des procédés et plus particulièrement des procédés coopératifs.

Nous avons choisi la propriété transactionnelle de l'isolation car elle a constitué l'une des réponses à la question suivante : quelles sont les propriétés transactionnelles qui concernent un groupe d'activités et qui ont une sémantique différente selon qu'elles soient appliquées au groupe entier en tant qu'entité unie ou à chaque activité individuellement ?

Il est clairement établi que l'isolation constitue une propriété transactionnelle cruciale dans tout système de gestion transactionnelle. Néanmoins, ces systèmes n'intègrent pas une gestion de l'isolation

orientée procédé, c'est à dire prenant en compte l'isolation de groupes d'activités. C'est ce que nous appelons "la dimension procédé".

Afin de prendre en compte cette dimension procédé dans la gestion de l'isolation, nous proposons d'introduire les sphères d'isolation dans les systèmes de gestion de procédés.

2.1 Contexte, enjeux et objectifs de l'étude de l'isolation dans les systèmes de gestion de procédés

Deux propriétés transactionnelles ont été étudiées selon l'approche des sphères de comportement. Il s'agit des sphères d'atomicité [Leym 00][Derk 01][Heuv 02] et des sphères de compensation [Leym 95]. Jusqu'à présent, les études portant sur les propriétés transactionnelles dans les procédés se sont focalisées sur l'atomicité. Cette focalisation sur l'atomicité que nous avons constatée dans les systèmes de gestion de procédés actuels est certainement dûe à une fausse idée souvent considérée comme évidente qui consiste à considérer que d'autres propriétés telle que l'isolation peuvent être induite de l'atomicité. Plus encore, beaucoup de systèmes de gestion de procédés considèrent l'isolation comme propriété à la charge du système de gestion de base de données utilisé.

Tout au long de nos investigations et de nos études sur la manière dont les systèmes de gestion de procédés gèrent les propriétés transactionnelles nous avons montré qu'il existe une focalisation sur l'atomicité. En nous interrogeant sur cette situation nous avons pu identifier certaines causes pouvant la justifier. D'abord, l'atomicité est considérée en premier lieu, par rapport à toute autre propriété transactionnelle, car elle est la plus simple à comprendre et à exprimer. En effet, l'atomicité représente le concept du "tout ou rien" en termes d'exécution des activités du procédé. Cette représentation binaire de l'atomicité, tout ou rien, oui ou non, 1 ou 0 ... aboutit à assimiler les autres propriétés transactionnelles à la même référence binaire de telle sorte que l'isolation est représentée souvent comme étant l'accès ou le non accès aux données. C'est en prenant en considération l'isolation comme étant un concept binaire que les concepteurs de procédés et de systèmes de gestion de procédés confondent atomicité et isolation pour aboutir en définitive à réaliser une certaine isolation à travers l'atomicité.

Si nous essayons de voir comment les systèmes de gestion de procédés actuels gèrent l'isolation, nous verrons que dans la plupart des cas, ils délèguent l'isolation aux systèmes de gestion de bases de données qu'ils utilisent. En réalité, cette délégation est l'unique solution garantissant une exécution cohérente des procédés, à défaut d'une solution d'isolation adaptée aux procédés. Les travaux de Derks et al. ont déjà évoqué l'intérêt de proposer une solution d'isolation adaptée aux procédés et non aux bases de données. Nous allons également montrer que les solutions d'isolation proposées par les systèmes de base de données ne sont pas adaptés aux systèmes de gestion de procédés. Cela est du à la rigidité dont font preuve les stratégies d'isolation existantes (celles des systèmes de bases de données).

Commençons d'abord par rappeler l'état actuel de la gestion de l'isolation dans les systèmes de bases de données et dans les systèmes de gestion de procédés métiers.

2.1.1 L'isolation dans les systèmes de bases de données transactionnelles

L'isolation d'une transaction dans une base de données est définie dans les systèmes de gestion de bases de données comme suit : "*Une transaction isolée doit paraître comme s'exécutant toute seule sur la base de données*".

Du point de vue des systèmes de gestion de bases de données, les transactions sont généralement indépendantes. Les seules transactions ayant un lien sont les transactions emboîtées qui ont une structure hiérarchique et leur dépendance dépend de leur terminaison. En effet, l'unique relation entre transactions emboîtées est la terminaison des transactions filles pour que la transaction mère puisse se terminer également. Cette vision transactionnelle ne prend en aucun cas en considération la possibilité d'avoir une dépendance entre transactions d'ordre informationnel ou sémantique. Certes, il n'est pas judicieux de dire

qu'il n'existe aucun lien entre les transactions. En effet, le lien entre transactions existe mais n'est pas exprimé au niveau du système de base de données, mais plutôt au niveau de l'application qui utilise la base de données. C'est ainsi que la gestion d'un tel lien n'est pas attribuée à un système à part entière mais reste du ressort de l'application, ce qui n'est pas toujours correctement envisagé par les concepteurs d'applications.

Un autre point important qu'il est nécessaire de mettre en avant est le fait qu'il existe certes une concurrence dans l'exécution des transactions dans une base de données, mais que, et nous pouvons l'affirmer, cette concurrence d'exécution est le fruit du hasard. En effet, de part le fait qu'il n'y ait aucune dépendance entre transactions, rien ne prédéfinit l'ordonnancement des transactions dans le système de gestion de bases de données. Le système de gestion de bases de données reçoit des requêtes, qu'il exécute parfois sous forme de transactions mais ne détient ni le sens de chacune d'entre elles ni la relation entre elles. La concurrence n'est finalement pas gérée en tant que telle.

2.1.2 L'isolation dans les systèmes de gestion de procédés

A l'opposé des transactions, les activités dans un procédé sont souvent très dépendantes les unes des autres. Dans un procédé de réservation de vols, l'activité de paiement dépend de celle du choix du vol qui dépend elle-même du choix de la compagnie, de la date, du lieu de départ, de la destination ... Les activités d'un procédé sont sémantiquement reliées les unes aux autres. C'est toute la différence avec les transactions dans les systèmes de bases de données qui n'en connaissent pas le lien.

La concurrence des activités d'un procédé, quant à elle, est définie à travers une représentation de la précédence entre activités riche de sens (flot de contrôle, opérateurs logiques ET, OU, OU EXCLUSIF ...). C'est ainsi que la concurrence de deux ou plusieurs activités peut parfois être explicitement requise et exprimée directement dans le procédé et non uniquement dûe au hasard (concurrence entre procédés). C'est en ce sens que l'isolation, vue comme un remède à la concurrence d'accès aux données, se voit redéfinie car cette concurrence d'accès tant crainte se retrouve dans les procédés métiers, et plus souvent, dans les procédés coopératifs, explicitement voulue et recherchée.

2.2 L'approche des sphères d'isolation

2.2.1 Présentation du formalisme utilisé et des concepts de base

La description d'un système d'information commence toujours par son modèle de données. Durant le développement et l'exploitation d'un système d'information, les modèles de données utilisés se doivent d'être les plus stables et les plus corrects et ainsi ont une plus grande importance que les processus qui manipulent ces données [Cook 96]. Dans cette thèse, nous supposons que nous avons affaire à un système d'information ayant un modèle de données capable d'assurer la cohérence des données dans un contexte d'exécution séquentielle d'opérations de base constituées par des lectures et des écritures.

Afin d'exprimer correctement notre vision des procédés coopératifs et d'introduire les éléments de flexibilité que nous proposons, il est important de clarifier le contexte d'exécution des activités coopératives et d'introduire des définitions préalables. Nous proposons un formalisme inspiré du formalisme défini dans ACTA [Chry 90, Chry 92, Chry 94, Rama 93] d'une part, et du formalisme défini dans [Schu 02b] d'autre part pour présenter les concepts que nous utiliserons.

Nous considérons que le système d'information contient toutes les données partagées du système. Des opérations peuvent être exécutées sur ces données. Chaque opération est opérée par une activité et aboutit à la modification de l'état de la donnée sur laquelle elle agit. L'ensemble des données gérées par le système d'information peut être défini comme suit :

Définition 10 ***(Ensemble Δ des données du système d'information)***
On définit Δ l'ensemble des données partagées du système d'information offrant des méthodes de lecture, d'écriture et de création de donnée $\delta \in \Delta$. Ces méthodes permettent de consulter ou de modifier la valeur d'une donnée telle que : $\forall\ \delta \in \Delta$,

$\delta.lire()$; renvoie la valeur de δ.
$\delta.écrire(val)$; modifie la valeur de δ pour l'assigner à la valeur val.

Définition 11 ***(État d'une donnée)***
Toute donnée $\delta \in \Delta$ est caractérisée à chaque instant t par un état $\theta(\delta, t)$ rendant compte des différentes informations concernant la donnée δ et en particulier sa valeur.

Les méthodes de lecture et d'écriture proposées par le système d'information sont utilisables via des opérations atomiques exécutées de façon séquentielle. Nous définissons l'ensemble $\mathcal{OP}$ représentant l'ensemble des opérations exécutées sur le système d'information comme suit :

Définition 12 ***(Ensemble $\mathcal{OP}$ d'opérations sur les données de Δ)***
$\mathcal{OP}$ représente l'ensemble de toutes les opérations exécutées sur le système d'information. Chaque opération $op \in \mathcal{OP}$ est caractérisée par : $op = (type, \delta, val, t)$ où $type \in \{lire, écrire\}$ qui exprime l'exécution respective, à l'instant t, de l'une des deux méthodes :

$$val \leftarrow \delta.lire() \quad \text{et} \quad booléen \leftarrow \delta.écrire(val).$$

δ est la donnée que op désire lire ou écrire.
val est la valeur de lecture ou d'écriture de δ.
t est l'instant d'exécution de l'opération op.

Chaque opération de $\mathcal{OP}$ agit sur des données du système d'information. Il est donc nécessaire de formaliser le changement d'état des données sur lesquelles une opération de $\mathcal{OP}$ est exécutée. Nous définissons les différents états particuliers reliés à l'exécution d'une opération $op \in \mathcal{OP}$ sur une donnée $\delta \in \Delta$ comme suit :

Définition 13 ***(États particuliers d'une donnée $\delta \in \Delta$ lors de l'exécution d'une opération $op \in \mathcal{OP}$)***
Nous définissons des états particuliers d'une donnée $\delta \in \Delta$ sur laquelle une opération $op \in \mathcal{OP}$ agit comme suit :
$\theta^-(\delta, op)$ représente l'état de δ **immédiatement avant** l'exécution de l'opération $op \in \mathcal{OP}$.
$\theta^+(\delta, op)$ représente l'état de δ **immédiatement après** l'exécution de l'opération $op \in \mathcal{OP}$.
$\theta^+(\delta, \{op_1, op_2, ..., op_n\})$ représente l'état de δ **immédiatement après** l'exécution de **toutes** les opérations $op_1, op_2, ..., op_n$.

L'état particulier $\theta^{+}(\delta, \{op_1, op_2, ..., op_n\})$ ne suppose pas l'exécution exclusive des opérations $op_1, op_2, ..., op_n$. D'autres opérations peuvent s'intercaler entre les $op_{i\in\{1,...,n\}}$. Ainsi $\theta^{+}(\delta, \{op_1, op_2, ..., op_n\})$ ne dépend pas uniquement des $op_{i\in\{1,...,n\}}$ mais aussi de l'histoire d'exécution des opérations telle que présentée dans la définition 16.

L'entité élémentaire dans l'organisation de travail est l'activité. Une activité a décrit une brique de travail à réaliser. Une instance d'activité $\widetilde{a}$ de a, représentant l'exécution d'une activité, se compose d'un ensemble d'opérations $\mathcal{OP}(\widetilde{a})$ qui s'exécutent sur un même système d'information. Nous définissons les opérations d'une instance d'activité $\widetilde{a}$ comme suit :

Définition 14 ***(Les opérations d'une instance d'activité $\widetilde{a}$)***
Une instance $\widetilde{a}$ d'une activité a se concrétise à travers un ensemble d'opérations $\mathcal{OP}(\widetilde{a}) \subseteq \mathcal{OP}$ s'exécutant sur un même système d'information. L'ensemble d'opérations $\mathcal{OP}(\widetilde{a})$ d'une instance d'activité $\widetilde{a} \in \widetilde{\mathcal{A}}$ comprend des opérations sur des données de Δ constituant le sous-ensemble $\Delta(\widetilde{a}) \subseteq \Delta$ des données utilisées par l'instance d'activité $\widetilde{a}$.
Également, $\mathcal{OP}(\widetilde{a})$ comprend exactement une opération de validation notée $op^{\widetilde{a}}_{commit} = (commit, \_, \_, t)$ ou une opération d'annulation notée $op^{\widetilde{a}}_{rollback} = (rollback, \_, \_, t)$. Ces deux opérations sont des opérations particulières qui définissent l'instant de terminaison d'une instance d'activité (par un succès en cas de validation et par un échec en cas d'annulation)

Sur la base du déroulement de l'exécution des opérations d'une activité, celle-ci peut acquérir plusieurs états possibles comme définit ci-dessous.

Définition 15 ***(État d'une instance d'activité)***
Soit $a \in \mathcal{A}$, une instance $\widetilde{a}$ de l'activité a a un état $\mathit{State}(\widetilde{a}) \in \mathcal{STACT}$.
$\mathcal{STACT}$ représente l'ensemble d'états possibles attribués à une instance d'activité durant son exécution

$\mathcal{STACT}$ dépend du moteur d'exécution de procédés. Dans la spécification du Wfmc (Workflow Management Coalition) [Wfmc 99], il est convenu que $\mathcal{STACT}$={Inactive, Active, Suspended, Completed}. Des états additionnels peuvent être envisagés pour introduire d'autres comportements pris en charge par le moteur d'exécution de procédés telle la compensation (les états "Compensating" et "Compensated") [Schu 02a].

Exécution des opérations (des instances d'activités) de $\mathcal{OP}$ (de $\widetilde{\mathcal{A}}$)

L'exécution d'un ensemble d'opérations sur le système d'information aboutit à la constitution d'une histoire d'exécution des opérations que nous définissons dans la définition 16. Les opérations de $\mathcal{OP}$ sont considérées de courte durée et sont traitées par le système de gestion de base de données. Nous assumons le fait qu'à l'échelle de ces opérations, l'exécution soit sérialisée. Ainsi le système ne traite qu'une seule opération à la fois et la considère comme une transaction ACID.

Définition 16 ***(Histoire d'exécution des opérations - Operations schedule)***
Nous définissons $h = (\mathcal{OP}(h), \overset{h}{\rightarrow})$ l'histoire d'exécution d'un ensemble d'opérations tel que :
$\mathcal{OP}(h)$ est un ensemble d'opérations exécutées sur le même système d'information.
$\overset{h}{\rightarrow}$ est une relation d'ordre total tel que $\forall\ op_1, op_2 \in \mathcal{OP}(h)$, tels que $op = (type_1, \delta_1, val_1, t_1)$ et $op_2 = (type_2, \delta_2, val_2, t_2)$; Si $op_1 \overset{h}{\rightarrow} op_2$ alors $t_1 < t_2$.

Sérialisation des opérations / activités : Même si l'exécution des opérations de $\mathcal{OP}$ est sérialisée, celle des activités ne l'est pas forcément et des exécutions concurrentes d'activités sont toujours possibles. La figure 30 illustre un exemple.

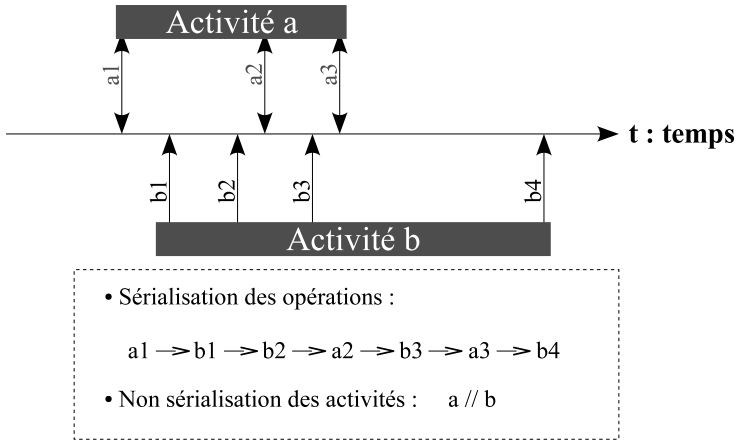


FIG. 30 – Sérialisation des opérations de $\mathcal{OP}$

L'ordonnancement d'instances d'activités $\mathcal{A}ct(I)$ d'une instance de procédé I obéit à l'ordre d'exécution $<_I$. Si nous nous référons au séquencement d'opérations d'instances d'activités, il est possible d'exprimer l'ordre d'exécution $<_I$ comme suit :

Définition 17 ***(Ordonnancement d'instances d'activités et le séquencement de leurs opérations respectives)***
$\forall\ h = (\mathcal{OP}(h), \overset{h}{\rightarrow})$ tel que $(\mathcal{OP}(\widetilde{a_n}) \subset \mathcal{OP}(h)) \wedge (\mathcal{OP}(\widetilde{a_m}) \subset \mathcal{OP}(h))$, nous avons
si $a_n \prec_P a_m$ alors $\forall\ op_i \in \mathcal{OP}(\widetilde{a_n})$ et $\forall\ op_j \in \mathcal{OP}(\widetilde{a_m})$, $op_i \overset{h}{\rightarrow} op_j$

Chaque opération ayant changé l'état du système, l'ensemble des changements opérés par les opérations d'une instance d'activité aboutissent à un changement global, au niveau de l'activité, de l'état du système d'information. L'introduction de la notion de "résultat final d'une instance d'activité" permet une abstraction des états du système d'information tel que expliqué dans [Ving 98] qui considère seulement des états globaux. Nous formalisons cette notion dans la définition 18.

Définition 18 ***(Résultat final d'une instance d'activité)***
Soit $\widetilde{a}$ une instance d'activité de $\widetilde{\mathcal{A}}$. Nous définissons le résultat final de $\widetilde{a}$ comme étant l'état des données de $\Delta(\widetilde{a})$ juste après l'exécution de l'ensemble des opérations de $\mathcal{OP}(\widetilde{a})$. Nous notons $\forall\ \delta \in \Delta(\widetilde{a})$,
$\theta^{+}(\delta, \widetilde{a}) = (\theta^{+}(\delta, \mathcal{OP}(\widetilde{a})))$: l'état de δ juste après l'exécution de toutes les opérations de $\mathcal{OP}(\widetilde{a})$.
$\theta^{-}(\delta, \widetilde{a}) = (\theta^{-}(\delta, \mathcal{OP}(\widetilde{a})))$: l'état de δ juste avant l'exécution d'une opération de $\mathcal{OP}(\widetilde{a})$.

Cette abstraction des résultats finaux permet d'identifier la notion d'activités sans effet définie comme suit :

Définition 19 ***(État d'une donnée : Résultat final / Résultat intermédiaire)***
$\forall\ \delta \in \Delta$, on dit que δ a un état "Résultat final" à l'instant t ssi $\exists\ \widetilde{a} \in \widetilde{\mathcal{A}}$ tel que $\theta(\delta, t) = \theta^{+}(\delta, \widetilde{a})$.
Si l'état à l'instant t d'une donnée n'est pas "Résultat final" alors il est "Résultat intermédiaire".
Nous appellerons un "Résultat intermédiaire" comme étant une donnée non validée et un "Résultat final" comme étant une donnée validée.

Nous décrivons l'état d'une donnée $\delta \in \Delta$ dans le diagramme d'état de la figure 31.

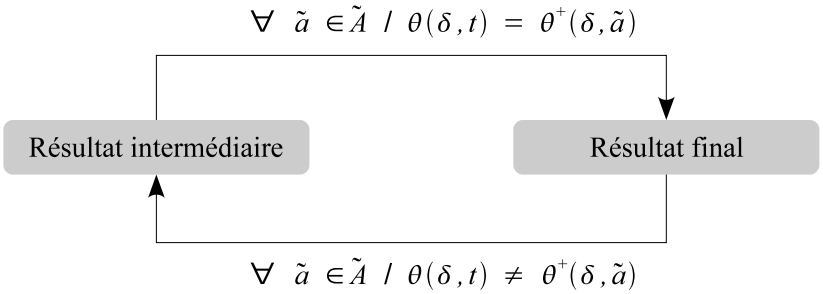


FIG. 31 – Cycle Intermédiaire/Final d'une donnée

Définition 20 ***(Activité sans effets)***
Une instance d'activité $\widetilde{a}$ est dite "sans effets" si pour toutes les séquences possibles d'instances d'activités $\beta = (\widetilde{b}_1, \widetilde{b}_2, ..., \widetilde{b}_j)$ et $\gamma = (\widetilde{c}_1, \widetilde{c}_2, ..., \widetilde{c}_k)$, le résultat final de toutes les activités de β et de γ dans la séquence concaténée $(\beta\ \{\widetilde{a}\}\ \gamma) =$ sont les mêmes que celles de la séquence $(\beta\gamma)$.

Afin de prendre en compte le recouvrement d'activités, nous définissons la notion d'activité compensatrice comme suit :

Définition 21 ***(Activité Compensatrice)***
Une d'activité $a \in \mathcal{A}$ est dite compensable s'il existe une activité, notée $a^{-1} \in \mathcal{A}$ telle que la séquence de leurs instances $(\widetilde{a}\ \widetilde{a}^{-1})$ est sans effets. L'activité a^{-1} est appelée "activité compensatrice" de a

Quand les instances d'activités partagent les mêmes ressources et y accèdent de façon concurrente, nous définissons la commutativité afin d'assurer l'équivalence entre les histoires d'exécution conformément aux travaux de Vingralek et al. [Ving 98] en se basant sur l'étude des résultats des activités. Nous assumons le fait que chaque instance d'activité aboutit à une valeur de retour qui inclut les données résultats (incluant les valeurs finales des données ainsi que l'état final de l'instance d'activité elle-même : succès, échec, compensation ...).

Définition 22 ***(Commutativité et Conflits)***
Deux instances d'activités $\widetilde{a}_i,\ \widetilde{a}_j\ \in \widetilde{\mathcal{A}}$ commutent, ce qui est noté $\widetilde{a}_i \rightleftharpoons \widetilde{a}_j$, si pour toute séquence d'instances d'activités $(\widetilde{b}_1, ...\widetilde{b}_m)$ et $(\widetilde{c}_1, ...\widetilde{c}_n)$ de $\widetilde{\mathcal{A}}$, le résultat de l'exécution de la séquence $((\widetilde{b}_1, ...\widetilde{b}_m), \widetilde{a}_i, \widetilde{a}_j, (\widetilde{c}_1, ...\widetilde{c}_n))$ est le même que celui de l'exécution de la séquence $((\widetilde{b}_1, ...\widetilde{b}_m), \widetilde{a}_j, \widetilde{a}_i(\widetilde{c}_1, ...\widetilde{c}_n))$. Deux activités qui ne commutent pas sont dites en conflit.

Dans les cas de modèles de procédés prenant en compte les mécanismes de compensation, nous définissons la commutativité parfaite adaptée à d'éventuelles interférences d'opérations de type "do" et "undo" comme suit.

Définition 23 ***(Commutativité Parfaite)***
Deux instances d'activités $\widetilde{a}_i,\ \widetilde{a}_j\ \in \widetilde{\mathcal{A}}$ commutent parfaitement si $\widetilde{a}_i \rightleftharpoons \widetilde{a}_j$, $\widetilde{a}_i \rightleftharpoons \widetilde{a}_j^{-1}$ et $\widetilde{a}_i^{-1} \rightleftharpoons \widetilde{a}_j$.

Pour le reste du document nous assumons le fait que la commutativité est parfaite.

L'exécution de procédés

Les instances de procédés n'incluent aucun formalisme de prise en compte de comportements transactionnels. Cela est compréhensible puisque de tels comportements concernent l'exécution même des activités d'un ou de plusieurs procédés de façon concurrente. Cela est du ressort de l'histoire d'exécution de toutes les activités du système.

Définition 24 ***(État d'une instance de procédé)***
Soit un procédé P, une instance I de P a un état $\mathit{State}(I) \in \mathcal{STPROC}$.
$\mathcal{STPROC}$ représente l'ensemble des états possibles attribués à une instance de procédé durant son exécution

Tout comme $\mathcal{STACT}$, $\mathcal{STPROC}$ dépend du moteur d'exécution de procédés. Dans la spécification du Wfmc (Workflow Management Coalition) [Wfmc 99], il est convenu que $\mathcal{STPROC}$={Initiated, Running, Active, Suspended, Complete, Terminated, Archived}.

L'histoire d'exécution des activités du système reflète cet aspect concurrentiel des activités des instances de procédés et inclut non seulement l'ordre d'exécution exprimé par les différentes instances de procédés mais également l'ordre observé induit. Cet ordre observé induit n'est pas totalement prévisible

car, même pour une seule instance de procédé, ces prévisions ne sont possibles que partiellement et cela est dû au fait que deux activités parallèles pourront s'exécuter dans n'importe quel ordre. De plus, il n'existe aucun moyen de prévoir l'ordre d'exécution des activités de plusieurs instances de procédés s'exécutant de façon concurrente. Cet ordre observé de l'exécution des activités dans le système donne lieu à une histoire d'exécution d'activités que nous définissons comme suit :

Définition 25 ***(Histoire d'exécution d'activités - Activities Schedule)***
Une histoire d'exécution d'activités (*Activities Schedule*) H est un tuple $(I(H), <^H)$ représentant l'exécution des activités des instances de procédés de $I(H)$ tels que :
$I(H) = \{I_1, I_2, ..., I_n\}$ est l'ensemble d'instances de procédés prises en compte dans l'histoire d'exécution.
$<^H$ est une relation d'ordre partiel reflétant l'ordre observé dans lequel les instances d'activités de toutes les instances de procédés de $I(H)$ sont exécutées telle que :
$\forall\ \widetilde{a}_n, \widetilde{a}_m \in \bigcup(\widetilde{\mathcal{A}ct}(I_i))_{I_i \in I(H)}$, si $\forall\ op_i \in \mathcal{OP}(\widetilde{a}_n)$ et $\forall\ op_j \in \mathcal{OP}(\widetilde{a}_m)$, $op_i \xrightarrow{h} op_j$ alors $\widetilde{a}_n <^H \widetilde{a}_m$.
On note également $\widetilde{\mathcal{A}ct}(H) = \bigcup(\widetilde{\mathcal{A}ct}(I_i))_{I_i \in I(H)}$ l'ensemble des instances d'activités de toutes les instances de procédés dans $I(H)$.

Soit $H{=}(I(H), <^H)$ une histoire d'exécution d'activités. Une histoire d'exécution d'opérations $h = (\mathcal{OP}(h), \xrightarrow{h})$ est dite équivalante à H si
$\mathcal{OP}(h) = \bigcup(\mathcal{OP}(\widetilde{a}_i))_{\widetilde{a}_i \in \widetilde{\mathcal{A}ct}(H)}$.

L'ordre observé inclut l'ordre requis par chaque instance de procédé dans $I(H)$. Nous pouvons également introduire $\prec^H$ comme l'ordre partiel requis des instances d'activités de $\widetilde{\mathcal{A}ct}(H)$ défini par : $\prec^H = \bigcup \prec_{ii \in I(H)}$.

Après avoir introduit les notions et définitions de base, nous allons présenter les sphères d'isolation, leurs propriétés et leur comportement dans la section suivante.

2.2.2 Éléments de base d'une sphère d'isolation

Les Sphères d'Isolation représentent une stratégie d'isolation orientée procédés et ce en opposition aux transactions plates utilisant les niveaux d'isolation SQL [ANSI SQL 92]. Les deux approches ont tout de même un objectif commun : gérer l'accès aux données dans un environnement concurrentiel.

Conformément à notre vision des sphères de comportement, nous considérons l'isolation de groupes d'activités, contrairement à l'isolation d'activités individuelles. En empruntant cette voie, nous nous confrontons à deux volets principaux de l'isolation d'un groupe d'activités. Le premier volet considère un groupe d'activités comme un sous-système et se penche sur son fonctionnement interne. Ainsi nous découvrons les besoins internes en termes de flexibilité d'isolation ce qui se traduit naturellement par le partage des données manipulées par les activités d'un groupe tout en mettant en avant leur besoin de protection contre les interventions extérieures au groupe sur ces mêmes données. Cette constatation découle du fait que, dans un groupe, toutes les activités participantes sont conscientes les unes des autres et développent donc une confiance implicite d'une part, et sont d'un nombre réduit d'autre part, donc plus simples à gérer. Cette flexibilité est d'autant plus demandée que de tels groupes se forment autour d'un objectif commun réalisé grâce à un travail coopératif.

Pour ce premier volet de l'isolation de groupes d'activités, nous choisissons le terme "**Cohésion**" afin d'exprimer la stratégie d'isolation interne au groupe. Une telle stratégie devrait permettre aux activités du groupe de partager leurs résultats même pendant leur exécution. Nous soulignons l'importance de prévoir

plusieurs niveaux d'isolation lors du partage de résultats afin de répondre aux différents besoins du groupe. Ces besoins peuvent aller de l'acceptation des accès concurrentiels même aux données intermédiaires (coopération accrue) jusqu'à l'exécution sérialisable sans partage de données intermédiaires (absence de coopération proprement dite).

Le deuxième volet de l'isolation d'un groupe d'activités que nous avons identifié le considère comme une "boite noire" interagissant avec le monde extérieur (les autres activités du système). La flexibilité d'isolation est un besoin important dans de telles situations. En effet, l'isolation concerne particulièrement la délivrance du résultat du travail de groupe. Les résultats du groupe, exploitables par les autres activités, sont-ils délivrés uniquement après l'exécution de toutes les activités du groupe ? Des résultats intermédiaires pourraient-ils être délivrés au fur et à mesure ?

Ces interrogations témoignent de l'importance d'une flexibilité de la stratégie d'isolation entre le groupe d'activités, en tant qu'entité unie, et les autres activités du système, en tant qu'environnement extérieur au groupe. Pour ce deuxième volet de l'isolation de groupes d'activités, nous choisissons le terme "**Cohérence**" afin d'exprimer la stratégie d'isolation externe au groupe. Nous proposons, tout comme dans le cas de la cohésion, une stratégie d'isolation basée sur plusieurs niveaux de cohérence allant de la mise à disposition des activités externes au groupe des moindres résultats intermédiaires (cohérence coopérative) jusqu'à la mise à disposition uniquement des résultats finaux du groupe (cohérence de sphère).

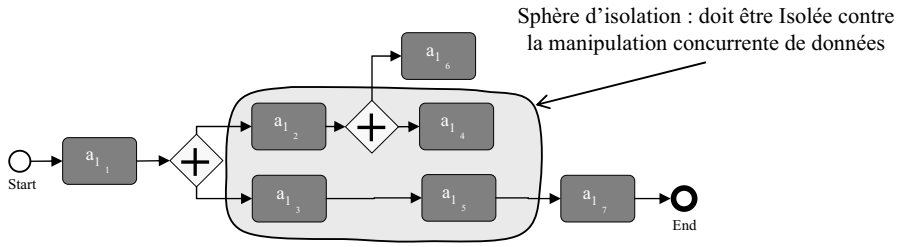


FIG. 32 – Une sphère d'isolation

La définition d'une sphère d'isolation est relative aux procédés, conformément à la figure 32. En effet, une sphère d'isolation, et généralement une sphère de comportement, est définie comme complément de la spécification d'un procédé afin d'exprimer un comportement particulier. Une fois définie pour un procédé, le comportement souhaité s'applique à chacune de ses instances. Ainsi, le concepteur de procédés définit une seule fois le comportement transactionnel des activités d'un procédé. En prenant en compte les deux volets de l'isolation d'un groupe d'activités que nous avons identifiés, nous proposons la définition d'une sphère d'isolation comme suit :

Définition 26 ***(Sphère d'isolation)***
Une sphère d'isolation ω est un tuple $(\widetilde{\mathcal{A}ct}(\omega), \mathcal{CHS}(\omega), \mathcal{CHR}(\omega))$ représentant un groupe d'instances d'activités qui ont besoin d'être isolées d'autres instances d'activités concurrentes et ce en spécifiant deux niveaux d'isolation : de Cohésion et de Cohérence.
Les éléments du tuple sont définis comme suit :
$\widetilde{\mathcal{A}ct}(\omega)$ représente l'ensemble d'activités concernées par l'isolation
$\mathcal{CHS}(\omega) \in \{\text{RU,RC,RR,S}\}$ représente le niveau de cohésion
$\mathcal{CHR}(\omega) \in \{\text{AC,SC,GC}\}$ représente le niveau de cohérence
Nous notons également
Ω l'ensemble des sphères du système.
$\Omega(\widetilde{A}) = \{\omega \in \Omega / \widetilde{\mathcal{A}ct}(\omega) \subset \widetilde{A}, \forall \widetilde{A} \subset \widetilde{\mathcal{A}}$.

La cohésion reflète le niveau de flexibilité de la concurrence d'accès aux données entre activités d'une même sphère, et la cohérence reflète le niveau de flexibilité des échanges entre activités de la sphère d'un côté et activités en dehors de celle-ci de l'autre côté.

2.3 La cohésion des sphères d'isolation

Le premier volet des sphères d'isolation, la cohésion exprime en réalité le fait que toutes les activités de la sphère s'exécutent de façon cohérente les unes par rapport aux autres, indépendamment des activités en dehors de la sphère.

Il est possible de définir une flexibilité de la cohésion en exprimant quatre niveaux de cohésion [Guab 05] qui sont les niveaux de Lecture de données non validées (Read Uncommitted), de Lecture de données validées (Read Committed), de Lecture répétée (Repeatable Read) et de niveau Sérialisable.

Les niveaux de cohésion définissent la façon dont les activités de la sphère accèdent aux données partagées dans le système d'information comme suit :

Soit $H=(I(H), <^H)$ une histoire d'exécution d'activités et $h = (\mathcal{OP}(h), \overset{h}{\rightarrow})$ l'histoire d'exécution des opérations équivalente, $\forall\ \widetilde{a}_i \in \widetilde{\mathcal{A}ct}(\omega)$. Soit $op^{begin}_{\widetilde{\mathcal{A}ct}(\omega)}(\delta)$ la première opération portant sur δ réalisée par une activité de $\widetilde{\mathcal{A}ct}(\omega)$.
$\forall\ \delta \in \Delta(a_i)$ et $\forall\ op \in OP(\widetilde{a}_i)$ alors :

Lecture non validée (Read Uncommitted) Le niveau de lecture de données non validées permet aux activités de la sphère d'utiliser des données dont l'état au moment de leur utilisation est "non validée".	$\theta^-(\delta, op) \in \theta(\delta, \widetilde{a}_j)$, $\widetilde{a}_j \in \widetilde{\mathcal{A}ct}(\omega)$ et $\theta^-(\delta, op^{begin}_{\widetilde{\mathcal{A}ct}(\omega)}(\delta))$ est quelconque
Lecture validée (Read Committed) Le niveau de lecture de données validées permet aux activités de la sphère d'utiliser uniquement des données dont l'état au moment de leur utilisation est "validée".	$\theta^-(\delta, op) \in \{\theta^+(\delta, \widetilde{a}_j)\ ,\ \widetilde{a}_j \in \widetilde{\mathcal{A}ct}(\omega), \widetilde{a}_j \neq \widetilde{a}_i\}$ et $\theta^-(\delta, op^{begin}_{\widetilde{\mathcal{A}ct}(\omega)}(\delta)) \in \{\theta^+(\delta, \widetilde{a}_j)\ ,\ \widetilde{a}_j \notin \widetilde{\mathcal{A}ct}(\omega)\}$
Lecture répétable (Repeatable Read) Le niveau de lecture répétée permet aux activités de la sphère d'utiliser uniquement des données dont l'état au moment de leur utilisation est "validée" et de plus assure que tant que l'activité utilisatrice de la donnée n'a pas fini son exécution, la donnée ne pourra changer d'état.	$\theta^-(\delta, op) \in \{\theta^+(\delta, \widetilde{a}_j)\ ,\ \widetilde{a}_j \in \widetilde{\mathcal{A}ct}(\omega), \widetilde{a}_j \neq \widetilde{a}_i\}$ et $\theta^-(\delta, op^{begin}_{\widetilde{\mathcal{A}ct}(\omega)}(\delta)) \in \{\theta^+(\delta, \widetilde{a}_j)\ ,\ \widetilde{a}_j \notin \widetilde{\mathcal{A}ct}(\omega)\}$ et $\nexists op' \in OP\ /\ op' = (ecrire, \delta, \ ,\ ) \wedge op \overset{h}{\rightarrow} op'$ et $op' \overset{h}{\rightarrow} op^{\widetilde{a}}_{commit}$
Sérialisable Le niveau serialisable simule une exécution en séquence des activités de la sphère et écarte ainsi tout inconvénient du à la concurrence d'accès.	voir le critère d'intra-sérialisabilité présenté dans la section 2.3.4.

2.3.1 Le niveau de cohésion à lecture non validées (Read Uncommitted)

Le niveau de cohésion le plus bas est celui où les activités d'une même sphère n'ont aucune contrainte quant à la lecture ou l'écriture de données partagées. Chaque activité de la sphère pourra lire n'importe quelle donnée, qu'elle soit validée ou non. C'est ainsi le mode d'écritures coopératives qui est autorisé.

C'est en quelque sorte le mode optimiste qui autorise toutes les activités de la sphère à manipuler de façon concurrente les données partagées sans se soucier des conflits que cela pourrait engendrer. En cas de conflit, aucune procédure particulière n'est appliquée. Ainsi, si une activité lit ou écrit une donnée et que, avant même sa terminaison, une autre activité écrase la donnée en question, cela ne posera pas de problème car un tel cas est supposé être toléré.

L'utilisation du niveau à lectures non validées se prête particulièrement aux activités à comportement coopératif. En effet, comme nous allons le détailler dans la section 3 de ce chapitre, le travail coopératif a besoin d'une isolation réduite au minimum pour être réellement effectif. Cela est dû au fait que travailler coopérativement revient souvent à manipuler ensemble et donc de façon concurrente des données partagées.

Malgré le fait que ce niveau procure peu de garanties quant à la gestion de la concurrence d'accès, il ne constitue tout de même pas un mode qualifiable d' "anti isolation" car pour un tel niveau, les activités externes à la sphère ne pourront pas modifier de façon concurrente à la sphère des données elles mêmes manipulées par les activités de la sphère. La figure 33 illustre le cas d'une histoire d'exécution telle que le niveau de cohésion minimum n'est pas respecté du fait qu'une activité externe à la sphère modifie des données manipulées par la sphère et ce de façon concurrente à son exécution.

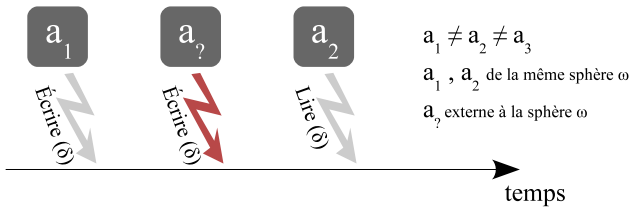


FIG. 33 – Histoire d'exécution ne respectant pas le niveau de cohésion à lectures non validées

2.3.2 Le niveau de cohésion à lectures validées (Read Committed)

Le niveau de cohésion à lectures validées (Read Committed) constitue le mode de fonctionnement d'une sphère d'isolation dans lequel les activités de la sphère ne peuvent lire que des données validées. Ainsi, si une activité tente de lire une donnée en cours de modification, elle devra attendre jusqu'à sa validation. Rappelons que la validation d'une donnée est généralement réalisée à la terminaison de toutes les activités en train de la modifier.

L'utilisation du niveau de cohésion à lectures validées rentre dans le cadre d'exécutions où la concurrence d'accès ne doit pas donner lieu à des lectures non validées. Il s'agit généralement de tâches dépendantes les unes des autres mais uniquement dans un cadre d'un protocole Rédacteur/Lecteur.

La figure 34 illustre le cas d'une histoire d'exécution telle que le niveau de cohésion à lectures validées n'est pas respecté du fait qu'une activité de la sphère lit une donnée non validée écrite par une autre activité de la sphère.

2.3.3 Le niveau de cohésion à lecture répétables (Repeatable Read)

Le niveau de cohésion à lecture répétables (Repeatable Read) correspond au besoin qu'une activité qui a lu une donnée, soit certaine que tant qu'elle n'a pas terminé son exécution, aucune autre activité ne viendra modifier la donnée en question. Ce besoin concerne principalement les exécutions critiques où nous avons besoin d'assurer une certaine sûreté de fonctionnement. Une donnée lue devra alors être protégée des modifications incontrôlées. Ce niveau de cohésion exclut évidemment les lectures non validées.

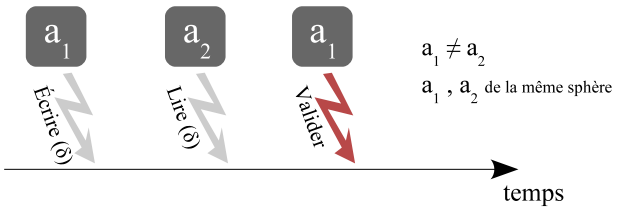


FIG. 34 – Histoire d'exécution ne respectant pas le niveau de cohésion à lectures validées

La figure 35 illustre le cas d'une histoire d'exécution telle que le niveau de cohésion à lectures répétables n'est pas respecté du fait qu'une activité de la sphère modifie une donnée préalablement lue par une autre activité de la sphère et ce avant la terminaison de celle-ci.

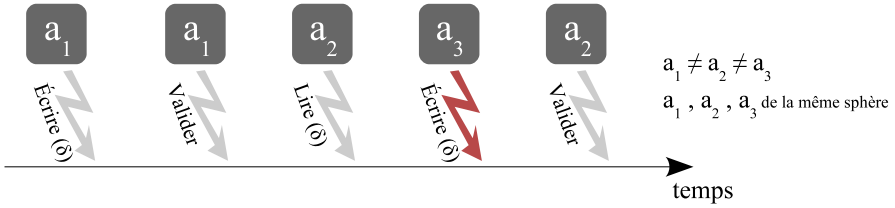


FIG. 35 – Histoire d'exécution ne respectant pas le niveau de cohésion à lectures répétables

2.3.4 Le niveau de cohésion sérialisable

Le niveau de cohésion sérialisable correspond au besoin d'exécuter les activités d'une même sphère de façon sérialisable, c'est-à-dire que leur exécution devra être équivalante à une exécution en série. La figure 36 illustre le cas d'une histoire d'exécution telle que le niveau de cohésion à lectures répétables n'est pas respecté.

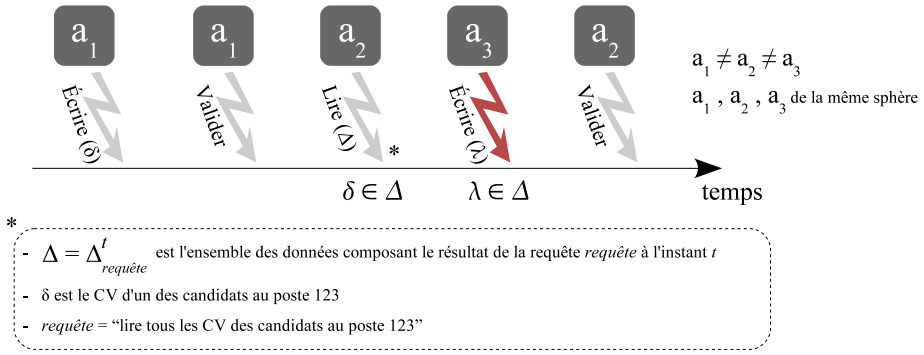


FIG. 36 – Histoire d'exécution ne respectant pas le niveau de cohésion sérialisable

Dans cet exemple, il s'agit de trois activités $\widetilde{a}_1$, $\widetilde{a}_2$ et $\widetilde{a}_3$ d'une même sphère et une donnée δ représentant le CV d'un candidat à un poste ayant pour identifiant 123. Après qu'une activité $\widetilde{a}_1$ ait modifié puis

validé le CV δ, $\widetilde{a}_2$ l'a lu au travers d'une requête portant sur tous les CV des candidats au poste 123. $\widetilde{a}_2$ a ainsi lu un ensemble $\Delta(request, t)$ contenant δ. C'est ainsi que, avant la terminaison de $\widetilde{a}_2$, une autre activité $\widetilde{a}_3$ de la même sphère, modifie un des CV appartenant à $\Delta(request, t)$ ou ajoute un nouveau CV d'un nouveau candidat susceptible de faire partie de $\Delta(request)$ si la requête est exécutée à nouveau. C'est ainsi que l'exécution pourrait ne pas être sérialisée puisque $\widetilde{a}_1 \longrightarrow \widetilde{a}_2 \longrightarrow \widetilde{a}_3$ a des effets différents de $\widetilde{a}_1 \longrightarrow \widetilde{a}_3 \longrightarrow \widetilde{a}_2$.

Nous connaissons tous le critère de sérialisabilité dans les systèmes transactionnels classiques. Ici, nous introduisons un nouveau critère qui se réduit aux activités d'une même sphère et non à l'ensemble des activités du procédé. Un tel critère a pour objectif d'assurer une sûreté de la concurrence d'accès pour les activités constituant une sphère d'isolation. Nous analyserons tout d'abord les enjeux et les objectifs d'un tel critère dans la section qui suit.

Enjeux et objectifs

Grâce à la cohérence des sphères et à son critère d'Extra-Sphère-Sérialisabilité, nous avions assuré une sérialisabilité de la sphère par rapport aux activités externes à celle-ci. Néanmoins, l'Extra-Sphère-Sérialisabilité ne se préoccupe pas de ce qui se passe entre les activités de la sphère, ce qui est tout à fait normal vu qu'elle répond au besoin d'assurer une cohérence de la sphère par rapport à l'extérieur.

L'intérêt de sérialiser l'exécution d'un groupe d'activités est né du besoin, pour certaines activités sensibles (tâches bancaires, financières ...) de s'exécuter de façon sérialisée les unes par rapport aux autres. Tout en déléguant la cohérence de la sphère par rapport aux activités externes à celle-ci à la cohérence et au critère d'Extra-Sphère-Sérialisabilité, nous proposons dans cette section un critère d'Intra-Sphère-Sérialisabilité qui permet d'assurer la sérialisabilité des activités de la sphère.

En considérant les activités encadrées dans un cadre gris comme faisant partie d'une même sphère d'isolation, la figure 48 illustre un exemple d'exécution Extra-Sphère-Sérialisable. L'Intra-Sphère-Sérialisabilité est illustrée dans la figure 37.

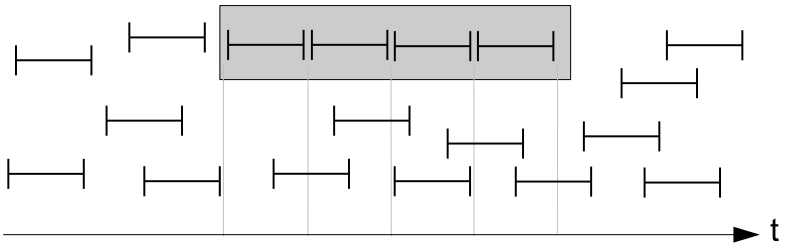


FIG. 37 – Exemple d'exécution Intra Sphère Sérialisable

Le critère d'Intra-Sphère-Sérialisabilité

Il est relativement facile de définir un tel critère. En effet, il concerne la sérialisabilité d'un sous-ensemble d'activités, indépendamment des autres. Nous utilisons alors le critère de sérialisabilité classique que nous allons restreindre aux activités d'une sphère. La sérialisabilité d'une histoire d'exécution est couramment définie comme suit :

Définition 27 ***(Sérialisabilité)***
Une histoire H d'exécution d'activités est dite sérialisable ssi elle est équivalante à une exécution en série des activités $\widetilde{Act}(H)$.

La sérialisabilité est ramenée uniquement au niveau des activités d'une sphère, ce qui nous amène à définir ce qu'on appellera histoire d'exécution d'une sphère définie comme suit :

Définition 28 ***(Histoire d'exécution $H(\omega)$ d'une sphère ω)***
L'histoire d'exécution d'une sphère $\omega \in \Omega(\mathcal{A}ct(H))$ est $H(\omega)$ tel que :
$\widetilde{\mathcal{A}ct}(H(\omega)) = \widetilde{\mathcal{A}ct}(\omega)$
$I(H(\omega)) = \{I_\omega\}$ tel que $\widetilde{\mathcal{A}ct}(I_\omega) = \widetilde{\mathcal{A}ct}(\omega)$.

En se référant à l'histoire d'exécution $H(\omega)$ d'une sphère ω, nous définissons l'intra-Sphère-Sérialisabilité comme suit :

Définition 29 ***(intra-Sphère-Sérialisabilité (iSS))***
Une histoire d'exécution d'activités $H = (I(H), \widetilde{\mathcal{A}ct}(H), <^H)$ est intra-Sphère Sérialisable et on note $iSS(\omega)$ par rapport à une sphère d'isolation $\omega \in \Omega^{A^S}$ si :
$\widetilde{\mathcal{A}ct}(\omega) \subseteq \widetilde{\mathcal{A}ct}(H)$
et que l'histoire d'exécution $H(\omega)$ est sérialisable.

La sérialisabilité d'une histoire d'exécution est généralement assurée à travers l'utilisation de verrous sur les données utilisées. Nous proposons d'utiliser les mêmes techniques de verrouillage restreintes aux activités de la sphère. Dans la section 1 du chapitre IV, nous reviendrons plus en détails sur une telle solution en proposant ce que nous appelerons verrous relatifs.

2.4 La cohérence des sphères d'isolation

La cohérence dans une sphère représente la façon dont les activités de la sphère partagent leurs résultats avec les activités externes à celle-ci. Afin de contrôler la cohérence entre les données utilisées par les activités de la sphère et celles utilisées par le reste du procédé, y compris d'autres sphères d'isolation, il est important de définir plusieurs niveaux de cohérence d'une sphère. Tandis que la cohésion permet de gérer l'échange d'informations entre les activités d'une sphère, la cohérence permet l'échange d'informations entre les activités d'une sphère et celles externes à la sphère. Ces niveaux de cohérence sont définis comme suit :

$\forall\ \widetilde{a}_{ext} \notin \widetilde{\mathcal{A}ct}(\omega)$ tel que $\exists\ op[\delta] \in \mathcal{OP}(\widetilde{a}_{ext}), Type(op) = lire$ et $\exists\ \widetilde{a}_{int} \in \widetilde{\mathcal{A}ct}(\omega)$ et
$\exists\ op'[\delta] \in \mathcal{OP}(\widetilde{a}_{int}), Type(op') = ecrire\ /\ op \xrightarrow{h} op'$ et $op' \xrightarrow{h} op^{end}_{\widetilde{\mathcal{A}ct}(\omega)}$ alors :

Cohérence coopérative : Toutes les valeurs de données écrites par des activités de la sphère sont visibles en dehors de celle-ci.	$\theta^-(\delta, op)$ est quelconque

Cohérence d'activité : Uniquement les valeurs **validées** de données écrites par des activités de la sphère sont visibles en dehors de celle-ci.	$\theta^{-}(\delta, op) \in \{\theta^{+}(\delta, \widetilde{a}) \ / \ \widetilde{a} \in \widetilde{\mathcal{A}ct}(\omega)\}$
Cohérence de sphère : Uniquement les dernières valeurs **validées** de données écrites par des activités de la sphère sont visibles en dehors de celle-ci.	Voir le critère d'intra-sérialisabilité présenté dans la section 2.4.3.

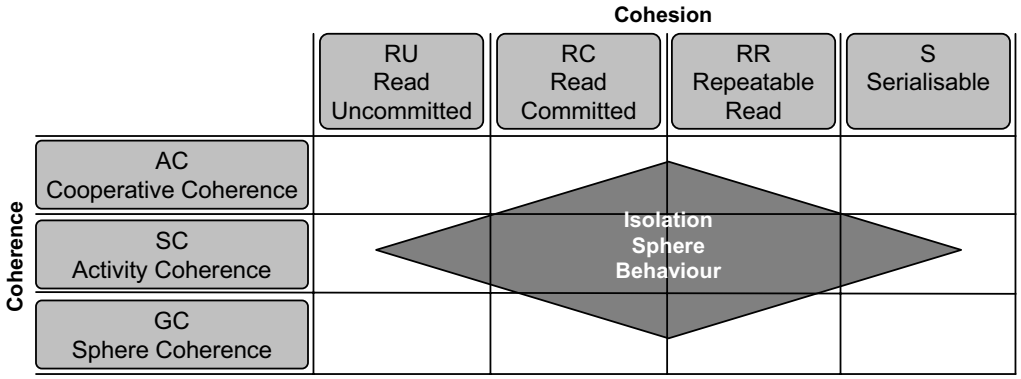


FIG. 38 – Dualité Cohésion/Cohérence

2.4.1 Le niveau de cohérence coopérative

Le niveau de cohérence coopérative correspond au cas où les données non validées écrites par des activités de la sphère sont parfaitement utilisables par des activités hors de la sphère. C'est le niveau où il n'y a pas de contrôle de l'accès concurrent aux données. Cela correspond au niveau de cohérence minimum et c'est d'ailleurs le cas dans beaucoup de systèmes de workflow actuels où seule l'atomicité est exprimée. En effet, en considérant une sphère comme un regroupement d'activités travaillant sur des données, le fait de livrer des données invalides à l'extérieur de la sphère risque de générer des erreurs incontrôlables en dehors de la sphère.

Il est clair qu'un tel niveau risque d'aboutir à une absence de cohérence, mais, généralement, le recours à un tel niveau est motivé par un besoin bien précis. Par exemple, nous avons parfois besoin d'exécuter un ensemble d'activités de traitement de données avec un niveau de cohésion élevé afin de garantir la fiabilité de leur exécution tout en ayant besoin de réaliser l'audit des modifications faites aux données. Un tel audit pourrait servir à la construction d'un historique des modifications. Dans un tel cas, il s'agit de définir une sphère d'isolation avec un niveau de cohésion maximal, permettant une exécution sérialisée entre les activités de la sphère. Le niveau de cohérence coopérative serait choisi pour permettre à des activités externes à la sphère, de faire l'audit des manipulations réalisées par les activités de la sphère. La figure 39 illustre un tel exemple.

2.4.2 Le niveau de cohérence d'activité

Le niveau de cohérence d'activité correspond quant à lui au cas où les données non validées écrites par des activités de la sphère ne sont pas utilisables par les activités en dehors de la sphère. Seules les données validées écrites par des activités de la sphère sont utilisables par les activités hors de la sphère.

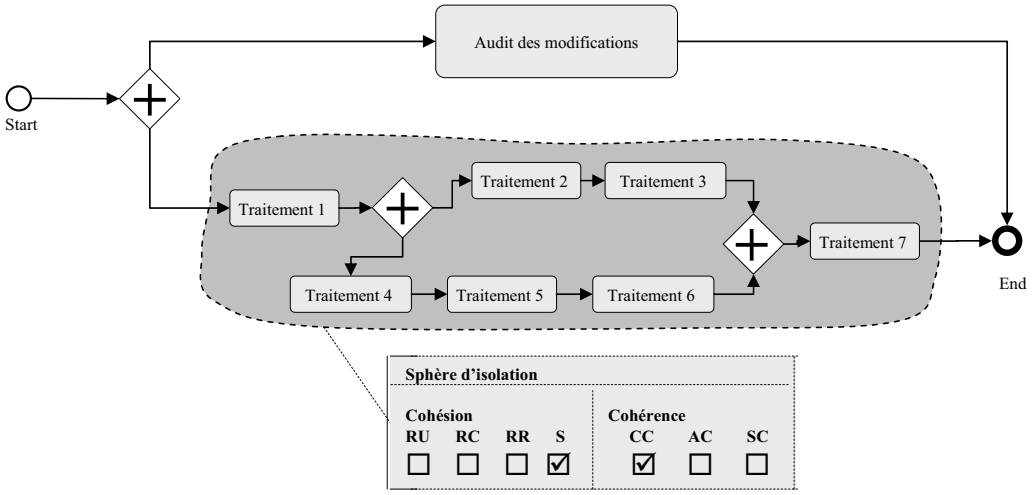


FIG. 39 – Exemple de sphère avec un niveau de cohérence coopérative

Cela permet de s'assurer que des erreurs dûes à la divulgation de données invalides ne pourront pas survenir.

Les situations motivant un tel niveau sont nombreuses. Nous pouvons citer en particulier le cas où un certain nombre d'activités traitant des données sont englobées dans une sphère d'isolation avec un niveau de cohésion minimal (lectures impropres) afin qu'elles s'exécutent sans contraintes les unes par rapport aux autres (exécution coopérative). Supposons maintenant qu'une ou plusieurs activités externes à la sphère veuille utiliser les données manipulées par la sphère et que ces activités sont critiques. Elles ont alors besoin d'avoir des données validées afin de garantir une fiabilité des actions menées à partir de ces données. Le niveau de cohérence d'activité établit par la sphère permettra de s'assurer que ces activités critiques ne pourront utiliser que des données validées et donc préserve la fiabilité de leur exécution.

L'exemple typique est celui d'une activité d'affichage d'un tableau de bord, telle qu'illustré dans la figure 40 contenant un certain nombre d'informations en provenance d'une base de données. Étant donné que l'affichage est à destination d'un "décideur" (un pilote, un manager, un boursier ...) qui pourra effectuer des actions selon les informations affichées. Si, pendant un traitement, les données sont manipulées, l'affichage ne doit pas relater toutes les valeurs temporaires écrites en cours de traitement car ces valeurs temporaires n'ont souvent aucune sémantique mais relèvent simplement du déroulement du traitement. Il est donc important que l'activité gérant le tableau de bord accède uniquement à des données validées et non temporaires (non validées).

2.4.3 Le niveau de cohérence de sphère

Le niveau maximal de cohérence correspond au cas où l'on voudrait s'assurer que l'exécution de la sphère contrôle l'utilisation des données produites par les activités de la sphère jusqu'à ce que toutes les activités de la sphère aient fini de s'exécuter.

L'exemple typique de besoin en niveau de cohérence de sphère est celui d'une composition de plusieurs activités, chacune faisant partie d'un travail global. La contrainte est que seul le résultat final du travail en sa globalité sera disponible à l'extérieur de la sphère. Il s'agit donc d'empêcher des activités externes à la sphère de lire les données produites par les activités de la sphère, qu'elles soient validées ou non, jusqu'à la fin de l'exécution de la sphère.

Afin de garantir un tel comportement, le niveau de cohérence de sphère obéira à un critère de séria-

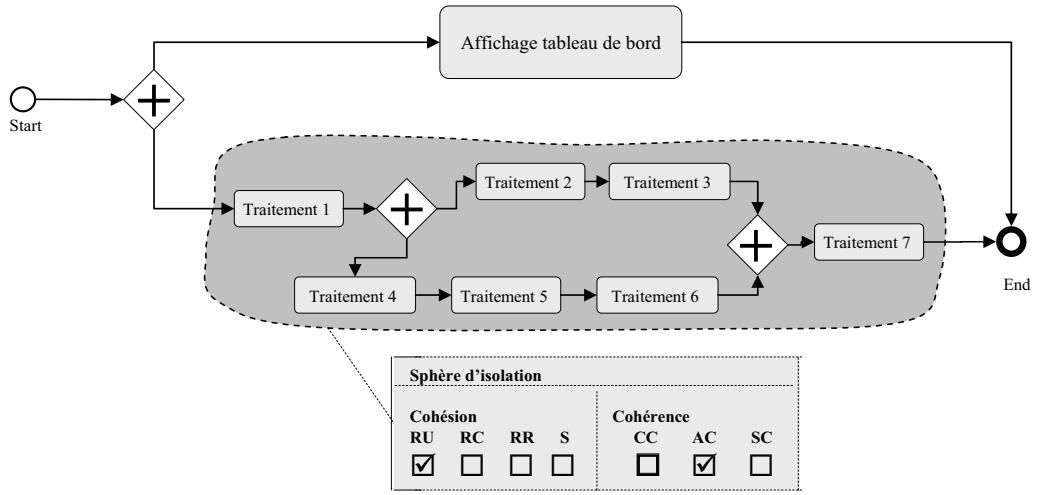


FIG. 40 – Exemple de sphère avec un niveau de cohérence d'activité

lisabilité que nous nous proposons de définir dans ce qui suit. Ce nouveau critère de sérialisabilité, que nous appelons critèrte d'extra-Sphère-Sérialisabilité est orienté cohérence de la sphère par rapport aux activités n'en faisant pas partie. Il doit permettre une exécution sérialisable de la sphère en tant qu'entité par rapport aux activités en dehors de la sphère ainsi qu'une flexibilité d'exécution des activités à l'intérieur de la sphère, qui sera gérée, quant à elle, par la cohésion de la sphère que nous avons développé précédemment à travers le critère d'intra-Sphère-Sérialisabilité en section 2.3.4 de ce chapitre.

Enjeux et objectifs

Afin d'exprimer clairement les enjeux et les objectifs d'un critère de correction de cohérence des sphères d'isolation (Extra-Sphère-Sérialisabilité), nous considérerons une exemple simple et abstrait de procédé P aboutissant à l'exécution concurrente d'activités comme illustré dans la figure 41.

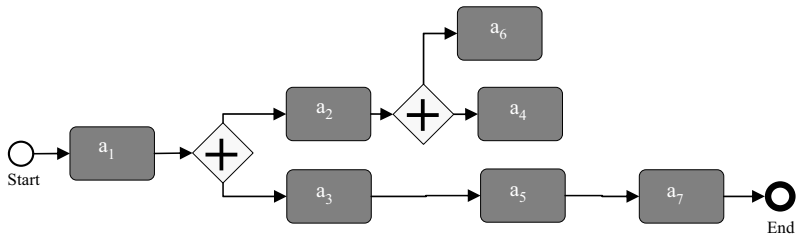


FIG. 41 – Un procédé P donnant place à la concurrence d'exécution d'activités

Le procédé P se compose de sept activités. Certaines d'entre elles expriment le besoin de coopérer et d'être isolées des autres activités.

$P = (\{a_1, a_2, a_3, a_4, a_5, a_6, a_7\}, \prec_P)$ / $a_1 \prec_P a_2$, $a_1 \prec_P a_3$, $a_2 \prec_P a_4$, $a_3 \prec_P a_5$, $a_2 \prec_P a_6$ et $a_5 \prec_P a_7$
Supposons également que certaines activités ne commutent pas : $a_3 \bowtie a_6$ et $a_4 \bowtie a_7$.

Afin de simplifier l'histoire d'exécution, nous ne considérons dans cet exemple qu'une seule instance de procédé, il n'y a donc pas d'autres procédés concurrents. Les histoires d'exécution de multiples procédés ou de multiples instances d'un même procédé présentent également le même contexte de concurrence d'exécution et peuvent être traitées en utilisant la même approche.

Soit $H_1 = (\{I_1\}, h, <^{H_1})$ l'histoire d'exécution des activités de I_1 représentant une instance de P avec $\widetilde{\mathcal{A}ct}(I_1) = \{\widetilde{a}_{1_1}, \widetilde{a}_{1_2}, \widetilde{a}_{1_3}, \widetilde{a}_{1_4}, \widetilde{a}_{1_5}, \widetilde{a}_{1_6}, \widetilde{a}_{1_7}\}$ des instances d'activités de $\mathcal{A}ct(P)$.

L'histoire d'exécution H_1 illustrée dans la figure 42 représente une exécution en série de toutes les activités du procédé P. Dans ce cas, aucune concurrence d'exécution, et donc aucune concurrence d'accès aux données n'est possible. Sans concurrence d'exécution aucun travail coopératif n'est possible et cela illustre parfaitement le fait que sérialisation et coopération ne sont pas tout à fait compatible, du moins dans l'état actuel des choses. L'objectif d'une telle sérialisation est d'assurer, en dépit de la rigidité que cela engendre, une cohérence des données accédées par les activités du procédé.

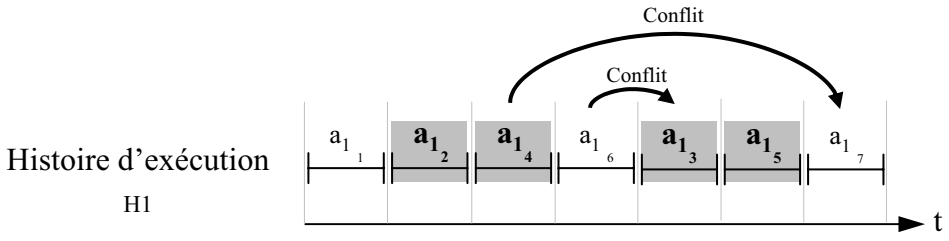


FIG. 42 – Exécution sérialisée H_1 : pas de concurrence entre activités

Supposons maintenant que les instances d'activités $\widetilde{a}_{1_2}$, $\widetilde{a}_{1_3}$, $\widetilde{a}_{1_4}$ et $\widetilde{a}_{1_5}$ expriment le besoin d'être isolées des autres instances d'activités. Cela s'exprime à travers une sphère d'isolation ω comme illustré dans la figure 43.

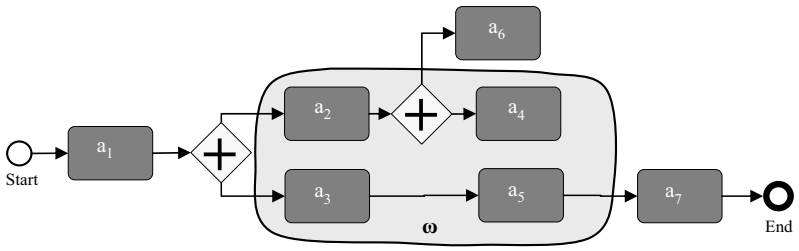


FIG. 43 – Introduction d'une sphère d'isolation dans le procédé P

Essayons d'analyser les limites de l'exécution sérialisée de la figure 42 face aux besoins en termes d'isolation de la sphère d'isolation ω. Reprenons l'exécution de l'histoire H_1 en mettant en avant dans la figure 44 l'appartenance des instances d'activités $\widetilde{a}_{1_2}$, $\widetilde{a}_{1_3}$, $\widetilde{a}_{1_4}$ et $\widetilde{a}_{1_5}$ à la sphère ω.

Nous pouvons constater que le besoin d'isolation exprimé à travers la sphère d'isolation ω n'est pas respecté lorsque le critère de sérialisabilité classique est mis en place. Ce critère assure la cohérence de chacune des activités mais pas celle d'un groupe au sens exprimé par la cohérence d'une sphère d'isolation. En effet, lorsque l'instance d'activité $\widetilde{a}_{1_6}$ s'exécute en concurrence avec la sphère, il n'est plus possible d'assurer la cohérence de la sphère. Il se peut que $\widetilde{a}_{1_6}$ écrive une donnée utilisée par $\widetilde{a}_{1_4}$, ce qui engendre le fait que $\widetilde{a}_{1_3}$ utilise une valeur différente de celle utilisée ou produite par $\widetilde{a}_{1_4}$.

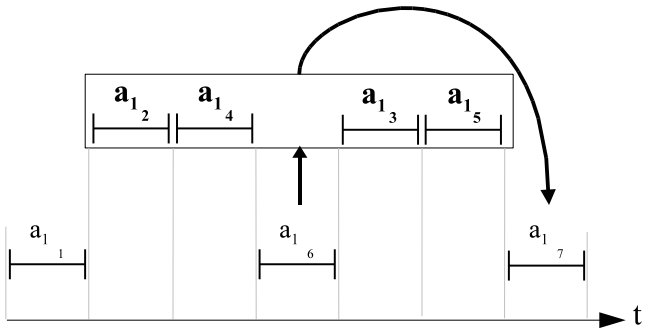


FIG. 44 – Exécution sérialisée H_1 et ses limites face aux contraintes d'isolation de la sphère ω.

Dans cet exemple, la sphère d'isolation utilise le niveau maximal de cohésion ce qui permet une forte isolation de chaque activité du groupe : Cela est loin d'être la configuration favorable à un travail coopératif. Néanmoins, nous pouvons constater que, malgré la rigidité de la cohésion, un problème de conflits peut apparaître. En effet, l'instance d'activité $\widetilde{a}_{1_6}$ est exécutée durant l'exécution de la partie coopérative du procédé. Ayant un conflit avec l'une des activités coopératives, il est permis qu'elle s'exécute si elle ne provoque pas d'incidence sur la coopération en cours. Toutefois, il est possible que $\widetilde{a}_{1_6}$ et $\widetilde{a}_{1_7}$ échangent des informations de telle sorte qu'un second conflit apparaisse entre une instance d'activité coopérative et $\widetilde{a}_{1_7}$, ce qui causerait une inconsistance. Les deux conflits révèlent un cycle de conflits entre une partie coopérative et une partie non coopérative du procédé.

L'histoire d'exécution $H2$ illustrée dans la figure 45 représente une exécution avec une concurrence d'exécution à l'intérieur d'une sphère et ce grâce à un niveau de cohésion plus souple (bas) dans la sphère ω. La concurrence d'exécution entre les instances d'activités coopératives $\widetilde{a}_{2_2}$ et $\widetilde{a}_{2_3}$ ou $\widetilde{a}_{2_4}$ et $\widetilde{a}_{2_5}$ permet une coopération effective. Mais, à cause d'un niveau de cohérence également souple (bas) (cohérence coopérative ou d'activité), les instances d'activités en dehors de la sphère, telles que $\widetilde{a}_{2_6}$ et $\widetilde{a}_{2_7}$ sont aussi exécutées de façon concurrente aux instances d'activités. Un cycle de conflit existe alors dans l'histoire d'exécution $H2$ et les niveaux d'isolation de la sphère ne sont pas suffisants pour assurer une exécution suffisamment fiable de la coopération.

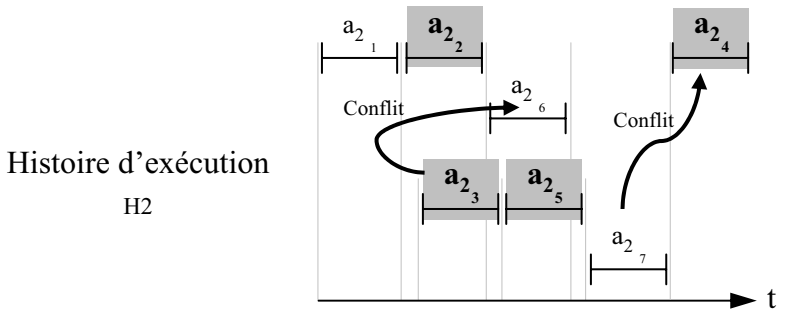


FIG. 45 – Histoire d'exécution coopérative : cycle de conflits entre des parties coopératives et non coopératives du procédé

L'exemple de l'histoire d'exécution $H3$ illustre le cas d'une coopération effective exempte d'anomalies de cohérence grâce à un niveau de cohérence élevé.

Le problème avec ce genre de contrôle de l'exécution est que toutes les instances d'activités ayant

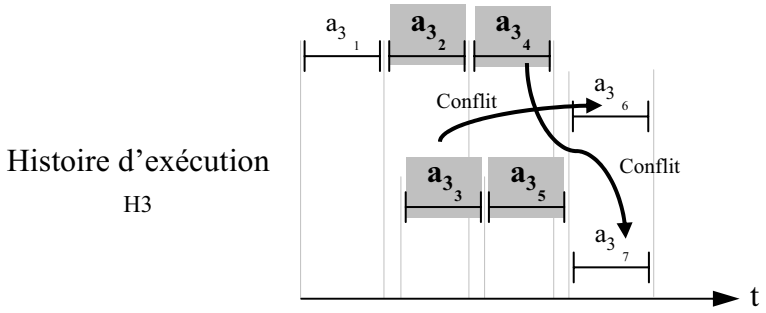


FIG. 46 – Histoire d'exécution avec une coopération effective grâce à un haut niveau de cohérence et à un bas niveau de cohésion

un conflit avec des instances d'activités de la sphère sont dans l'obligation d'attendre jusqu'à la fin d'exécution de toutes les instances d'activités de la sphère avant de finir de s'exécuter. Cela introduit un certain manque de flexibilité du à la sérialisation entre d'un côté les activités de la sphère et de l'autre les activités en dehors de la sphère. Existe-t-il une histoire d'exécution exprimant un niveau de souplesse plus important tout en gardant une protection contre risques de cycle de conflit ? Effectivement, l'histoire d'exécution $H4$ illustré dans la figure 47 représente une exécution qui procure les mêmes protections contre les cycles de conflits que celle de $H3$ tout en permettant à a_{4_6} de s'exécuter en concurrence aux activités de la sphère. Dans ce cas de figure, les conflits entre les instances d'activités de la sphère et les instances d'activités externes à la sphère sont orientées dans le même sens, ce qui ne provoque pas un cycle de conflits. $H4$ demeure équivalant en termes de conflits à $H3$.

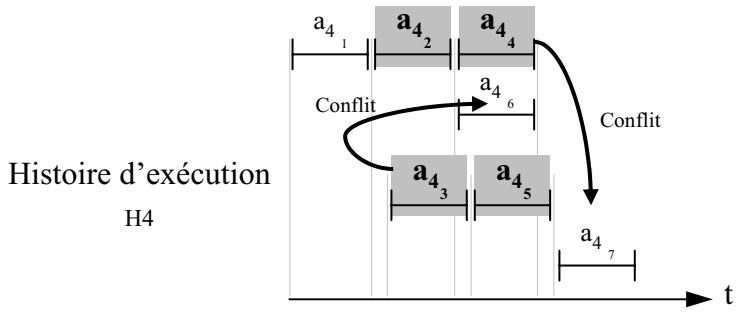


FIG. 47 – Exécution concurrente accrue et pas de cycles de conflits dans une exécution coopérative : cohérence maximale

L'exemple de l'histoire d'exécution $H4$ nous montre bien que le niveau maximal en cohérence n'induit pas forcément une sérialisabilité stricte mais permet une utilisation plus intelligente de l'ordonnancement pour profiter d'une flexibilité maximale possible. Nous avons donc besoin, en allant au delà de la simple sérialisabilité, de définir un nouveau critère capable d'assurer une telle sérialisabilité de la sphère par rapport aux activités externes, mais offrant le plus de flexibilité possible. Un tel critère devra exclure les anomalies de coopération tout en permettant une exécution la plus flexible possible. C'est ce que nous nous proposons d'introduire à travers le critère d'Extra-Sphère-Sérialisabilité que nous présentons dans la section qui suit.

Le critère d'Extra-Sphère-Sérialisabilité

En considérant les activités encadrées dans un cadre gris comme faisant partie d'une même sphère d'isolation, la figure 48 illustre un exemple d'exécution extra-sérialisable.

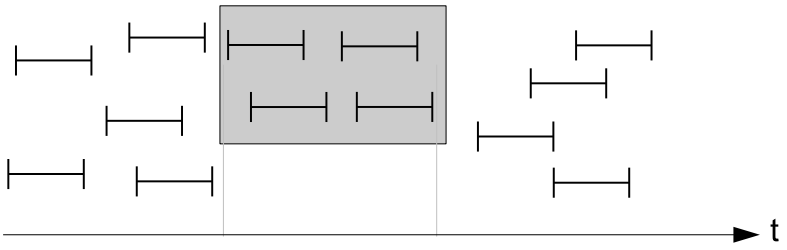


FIG. 48 – Exemple d'exécution Extra Sphère Sérialisable

Nous pouvons constater la complémentarité des deux critères de sérialisabilité, à travers la complémentarité de la cohérence et de la cohésion, comme illustré dans la figure 49.

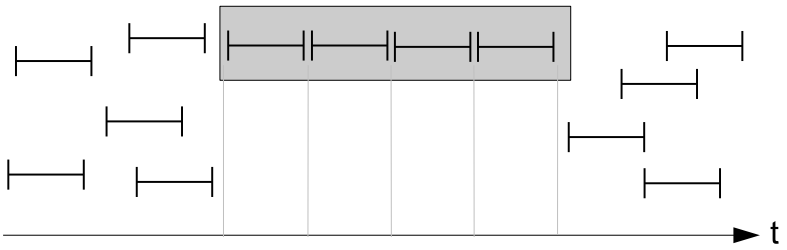


FIG. 49 – Exemple d'exécution Intra et Extra Sphère Sérialisable

Dans un environnement coopératif, les activités qui coopèrent ensemble ne prêtent généralement pas d'importance à la sérialisation de l'exécution et ce du fait de leur longue durée d'exécution. Une approche sur l'anticipation de l'exécution, décrite dans [Grig 01a] a été introduite pour réduire l'influence du flot de contrôle sur l'exécution de telles activités. Dans un tel cadre, la sérialisabilité de l'exécution devient de plus en plus laborieuse.

Nous pensons que dans de telles situations, c'est-à-dire dans le cas d'activités à longue durée d'exécution, et particulièrement d'activités coopératives, il n'est pas judicieux d'imposer une sérialisation de toutes les activités du procédé. La sérialisation de procédé, telle qu'introduite dans [Schu 02a] permet de sérialiser les procédés concurrents sans pour autant sérialiser les activités de chaque procédé. Le théorème 1, proposé et prouvé dans [Schu 02a] exprime un critère de sérialisabilité pour assurer la sérialisabilité de procédés.

Définition 30 ***(Process-Sérialisabilité (PSR))***
Une histoire d'exécution de procédés $H = (I(H), <^H)$ est Process-Sérialisable (PSR) s'il existe une équivalence de conflits entre H et une exécution en série $H_{ser} = (I(H), <^{H_{ser}})$ tel que $\forall \widetilde{a}, \widetilde{b} \in \widetilde{\mathcal{A}ct}(H_{ser})$
$\widetilde{a} <^{H_{ser}} \widetilde{b} \quad \vee \quad \widetilde{b} <^{H_{ser}} \widetilde{a}$

Définition 31 ***(Graphe de Sérialisabilité de Procédés*** $GSP(H)$***)***
Un graphe de sérialisabilité de procédés $GSP(H)$, relatif à une histoire d'exécution $H = (I(H), <^H)$ est un graphe qui contient un nœud pour chaque instance de procédé de $I(H)$ et un arc orienté de I_i à I_j pour tout $I_i, I_j \in I(H)$ ayant des instances d'activités en conflit $\widetilde{a}_i \in I_i$, $\widetilde{a}_j \in I_j$ et $\widetilde{a}_i \neq \widetilde{a}_j$ et $\widetilde{a}_i <^H \widetilde{a}_j$.

Théorème 1 ***(Graphe de Sérialisabilité de Procédés et PSR)***
Une histoire d'exécution H est PSR si et seulement si son graphe de sérialisabilité de procédés GSP(H) est acyclique.

Cette approche fournit une exécution équivalante en termes de conflits à une exécution sérialisée des instances du procédé. Nous proposons une nouvelle vision de la sérialisation qui est basée sur les sphères d'isolation. La sérialisation de sous-ensembles d'activités, similaire à la sérialisation de procédés [Schu 02a], permet d'assurer une sérialisation entre des parties du procédé sans une sérialisation entre les activités elles-mêmes. Quel est le critère de sérialisabilité qui permettra d'assurer un tel type de sérialisabilité ? Nous désignons un tel critère sous le nom d' "Extra-Sphère-Sérialisabilité" (eSS).

Les sphères d'isolation contribuent à délimiter un sous-ensemble d'activités du procédé. Parmi les activités en dehors de la sphère, nous pouvons identifier trois sous-ensembles d'activités du procédé. Le premier sous-ensemble est celui des activités externes à la sphère et qui doivent être exécutées avant ceux de la sphère et ce à cause des contraintes de type flot de contrôle. Ces activités peuvent être formellement définies comme suit :

Définition 32 ***(Activités Pré-Sphère :*** $PreSA(\omega)$***)***
Pour toute sphère d'isolation $\omega \in \Omega$, $PreSA(\omega) = \{\widetilde{a}_i \in \widetilde{\mathcal{A}} \ / \ \widetilde{a}_i$ est une instance de $a_i \in \mathcal{A}$ et $\exists \widetilde{a}_j \in \widetilde{\mathcal{Act}}(\omega)$ instance de $a_j \in \mathcal{A}$ tel que $a_i \prec^* a_j\}$.

Un deuxième sous-ensemble d'activités en dehors de la sphère correspond à celles qui ne peuvent s'exécuter qu'après la fin de l'exécution de toutes les activités de la sphère et ce à cause des contraintes de type flot de contrôle. Ces activités peuvent être formellement définies comme suit :

Définition 33 ***(Activités Post-Sphère :*** $PostSA(\omega)$***)***
Pour toute sphère d'isolation $\omega \in \Omega$, $PostSA(\omega) = \{a_i \in \mathcal{A} \ / \ \exists a_j \in \mathcal{Act}(\omega), \ a_j \prec^* a_i\}$.

Un dernier sous-ensemble d'activités en dehors de la sphère correspond à celles qui pourraient s'exécuter de façon concurrente à des activités de la sphère. Ces activités peuvent être formellement définies comme suit :

Définition 34 ***(Activités Simultanées à une Sphère :*** $SimSA(\omega)$***)***
Pour toute sphère d'isolation $\omega \in \Omega$, l'ensemble des activités dont l'exécution est susceptible d'être simultanée à celle de la sphère ω est
$SimSA(\omega) = \mathcal{A} - (PreSA(\omega) \cup PostSA(\omega)) - \mathcal{Act}(\omega)$.

Il est important de noter que les activités de $SimSA(\omega)$ pourraient (et non pas doivent) s'exécuter de façon concurrente avec les activités de la sphère ω. Certaines des activités de $SimSA(\omega)$ s'exécuteraient probablement avant même l'exécution d'activités de la sphère. D'autres le feront durant l'exécution d'activités de la sphère. Enfin certaines autres s'exécuteront probablement après l'exécution des activités de la sphère. Nous proposons de définir trois *pseudo* sous-procédés correspondant à ces trois sous-ensembles et de calculer une sous-histoire d'exécution de ces trois pseudo sous-procédés. Un quatrième pseudo sous-procédé correspondant aux activités de la sphère d'isolation est pris en considération dans le calcul de la sous-histoire d'exécution. Nous avons choisi d'appeler une telle histoire d'exécution "Histoire d'exécution de sphère" et nous la définissons formellement comme suit :

Définition 35 ***(Sous-histoire d'exécution)***
Une histoire d'exécution d'activités sH est une sous-histoire de l'histoire H si
$sH = (I(sH), \widetilde{\mathcal{A}ct}(sH), <^{sH})$ représentant l'exécution de parties des instances de procédés de H tels que
$\widetilde{\mathcal{A}ct}(sH) \subseteq \widetilde{\mathcal{A}ct}(H)$
$\forall I_i \in I(H)$, Si $(\widetilde{\mathcal{A}ct}(I_i) \cap \widetilde{\mathcal{A}ct}(sH)) \neq \oslash$ alors $\exists\ I_i^{'} \in I(sH)$ tel que $\widetilde{\mathcal{A}ct}(I_i^{'}) = (\widetilde{\mathcal{A}ct}(I_i) \cap \widetilde{\mathcal{A}ct}(sH)$.
$\forall a_i, a_j \in \widetilde{\mathcal{A}ct}(sH)$, si $a_i <^{H} a_j$ alors $a_i <^{sH} a_j$
$\forall\ a_i, a_j \in \widetilde{\mathcal{A}ct}(sH),\ a_i \bowtie_{H} a_j \Rightarrow a_i \bowtie_{sH} a_j$

Définition 36 ***(Sous-histoire d'exécution*** $SSubH(H,\omega)$ ***par rapport à une sphère)***
La sous-histoire d'exécution d'activités d'une histoire d'exécution H par rapport à une sphère d'isolation $\omega \in \Omega(\mathcal{A}ct(H))$ est $SSubH(H,\omega)$ représentant une sous-histoire d'exécution de H tel que :
$\widetilde{\mathcal{A}ct}(\omega) \subset \widetilde{\mathcal{A}ct}(SSubH(H,\omega))$
$\exists\ I_\omega \in I(SSubH(H,\omega))$ tel que $\widetilde{\mathcal{A}ct}(I_\omega) = \widetilde{\mathcal{A}ct}(\omega)$.
$\forall I_i \in I(H)$, Si $(\widetilde{\mathcal{A}ct}(I_i) \cap \widetilde{\mathcal{A}ct}(SSubH(H,\omega))) - \widetilde{\mathcal{A}ct}(\omega) \neq \oslash$ alors $\exists\ I_i^{'} \in I(SSubH(H,\omega))$ tel que $\widetilde{\mathcal{A}ct}(I_i^{'}) = (\widetilde{\mathcal{A}ct}(I_i) \cap \widetilde{\mathcal{A}ct}(SSubH(H,\omega))) - \widetilde{\mathcal{A}ct}(\omega)$.
$\forall I_i \in I(H)$, si $\exists a_i \in I_i$ telque $a_i \in \widetilde{\mathcal{A}ct}(SSubH(H,\omega))$ alors $\exists\ (\widetilde{\mathcal{A}ct}(\omega), *) \in I(SSubH(H,\omega))$ et $\forall A \in SParts(S)$, le sous procédé $(A,*) \in PI^{SSubH_\omega(S)}$.

La sous-histoire d'exécution par rapport à une sphère permet d'analyser la concurrence de la sphère en tant qu'entité par rapport au reste des activités du procédé.

Définition 37 ***(extra-Sphère-Sérialisabilité (*** eSS ***))***
Une histoire d'exécution d'activités $H = (I(H), \widetilde{\mathcal{A}ct}(H), <^{H})$ est Extra-Sphère Sérialisable et on note $eSS(\omega)$ par rapport à une sphère d'isolation $\omega \in \Omega^{A^S}$ si la sous-histoire d'exécution $SSubH(\omega, H)$ est process-sérialisable.

Théorème 2 ***(Graphe de Sérialisabilité de Procédés et*** eSS ***)***
Une histoire d'exécution H est $eSS(\omega)$ si et seulement si le graphe de sérialisabilité de procédés $SSubH(H,\omega)$ est acyclique.

Preuve du théorème 2

Soit $H = (I(H), \widetilde{\mathcal{A}ct}(H), <^H)$ une histoire d'exécution d'activités. Soit $\omega \in \Omega$. Afin de prouver le théorème 2, nous procédons à un raisonnement par contraposé. Cela consiste à prouver que $x \Longrightarrow y$ en prouvant que $\neg y \Longrightarrow \neg x$. Nous nous proposons de prouver que [H est $eSS(\omega)$] $\Longleftrightarrow$ [$GSP(SSubH(H,\omega))$ est acyclique]. Nous devons alors prouver :

(1) $\neg$[H est $eSS(\omega)$] $\Longrightarrow \neg$[$GSP(SSubH(H,\omega))$ est **acyclique**]

(2) [$GSP(SSubH(H,\omega))$ est **cylcique**] $\Longrightarrow \neg$[H est $eSS(\omega)$]

Preuve de (1) :
H n'est pas $eSS(\omega)$. En utilisant la définition 37 de l'extra-Sphère-Sérialisabilité, la sous-histoire d'exécution par rapport à la sphère d'isolation ω n'est pas process-sérialisable
(i) ($SSubH(H,\omega)$ n'est pas PSR).
Utilisons le théorème 1, on sait que pour toute histoire d'exécution d'activités X,
[GSP(X) est acyclique] $\Longleftrightarrow$ [X est PSR] ce qui peut être traduit en
(ii) $\forall$ histoire H d'exécution d'activités, $\neg$[H est PSR] $\Longrightarrow \neg$[GSP(H) est acyclique].
En utilisant (i) et (ii) nous pouvons conclure que le graphe de sérialisabilité de procédés GSP($SSubH(H,\omega)$) est **cyclique**. Ainsi nous avons prouvé (1).

Preuve de (2) :
(iii) GSP($SSubH(\omega)$) est **cyclique**.
En utilisant le théorème 1, nous savons que pour toute histoire X d'exécution d'activités, [X est PSR] $\Longleftrightarrow$ [GSP(X) est acyclique] ce qui peut être traduit en
(iv) $\forall$ histoire X d'exécution d'activités, $\neg$[GSP(X) est acyclique] $\Longrightarrow \neg$[X est PSR].
A partir de (iii) et (iv) nous pouvons conclure que la sous-histoire $SSubH(H,\omega)$) n'est pas process-sérialisable PSR. Finalement, en utilisant la définition 37, nous concluons que H n'est pas $eSS(\omega)$. Ainsi, nous avons prouvé (2) et le théorème 2.

Ici s'achève donc la définition des différents niveaux de cohésion et de cohérence des sphères d'isolation. Nous avons donc explicité ces différents niveaux pour une sphère face aux activités extérieure à celle-ci. Cependant, il est fortement envisageable que des sphères d'isolation soient définies de façon emboîtée. Cela répond à un besoin réel comme nous allons l'expliquer dans la section qui suit. Nous pourrons également montrer comment les différents niveaux de cohésion et de cohérence de sphères emboîtées peuvent coexiter et opérer de telle sorte que l'expressivité et la flexibilité se retrouve accrue.

2.5 Gestion avancée des sphères d'isolation : l'emboîtement de sphères

Les transactions plates dans le monde des bases de données a été enrichi par l'avènement des transactions emboîtées (Nested Transactions) [Moss 82]. C'est la même vision d'emboîtement de transactions qui s'impose à nos yeux aboutissant ainsi à l'emboîtement de sphères d'isolation.

Si l'approche est la même que pour les "Nested Transactions", il existe une différence entre les deux approches en ce sens que les sphères d'isolation considèrent deux dimension que sont la cohésion et la cohérence. Dans les Nested Transaction, une transaction contient des sous-transactions. Le principe de base des "Nested Transactions" est qu'une transaction ne peut commiter qu'après que toutes ses sous-transactions aient commité. Cette vision est typiquement orientée vers les solutions de flexibilité de l'atomicité. L'isolation aussi a été réalisée au travers des "Nested Transactions". Il s'agit alors d'autoriser l'accès aux données manipulées uniquement à la transaction parente (comparable à la cohérence des sphères d'isolation). Néanmoins, une telle solution ne propose pas plusieurs niveaux de cohérence. De plus, la cohésion des différentes sous-transactions n'est pas prise en compte et les sous-transactions sont exécutées indépendamment les unes des autres, et donc avec une exclusion mututelle quant à l'accès aux données partagées.

Nous proposons d'intégrer les deux dimension de cohérence et de cohésion de façon plus fine et d'expliciter la façon dont les niveaux de cohésion, et de cohérence sont pris en compte dans le cas

d'emboîtement de sphères. Nous exliquons d'abord les enjeux de l'emboîtement de sphères d'isolation et nous définirons les règles de base de gestion de la cohésion et de la cohérence. Nous étudierons ensuite, en détail, l'impact de l'emboîtement de sphères d'isolation sur les critères d'intra-Sphère-Sérialisabilité et d'extra-Sphère-Sérialisabilité.

2.5.1 Enjeux de l'emboîtement de sphères d'isolation et règles de base

L'emboîtement de sphères d'isolation apporte une sémantique très riche. En effet, une sphère d'isolation définit le comportement d'isolation d'un sous-ensemble d'activités par rapport au "reste du monde". Il serait donc intéréssant d'envisager de limiter le "reste du monde" d'une sphère à une autre sphère englobante. L'emboîtement de sphères apportera alors une sémantique de comportement d'isolation des activités d'une sphère par rapport à la sphère supérieure (englobante). La figure 50 illustre la vision globale de l'emboîtement de sphères d'isolation et pose le problème de la portée de la cohésion et de la cohérence dans de tels cas.

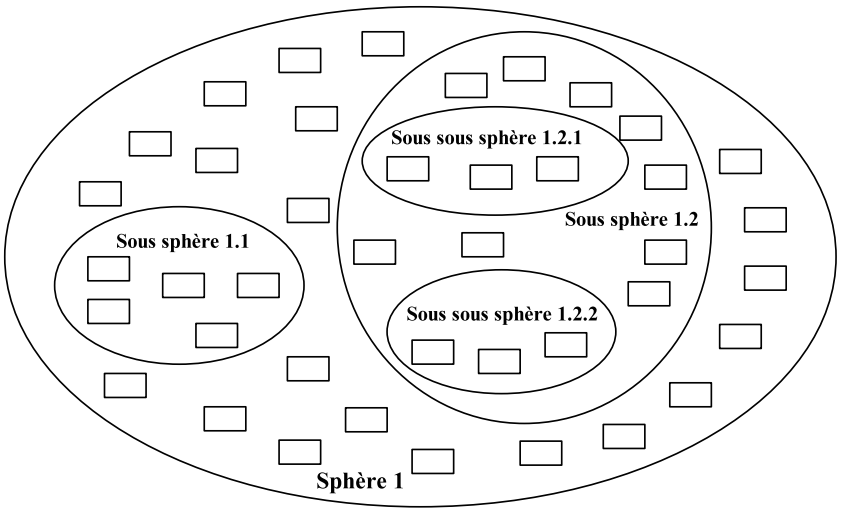


FIG. 50 – L'emboîtement de sphères d'isolation : quelle portée pour la cohésion et la cohérence d'une sphère ?

Il est nécessaire de définir clairement la portée des niveaux de cohésion et de cohérence de sphères emboîtées. Pour cela, nous nous proposons de définir les règles suivantes :

Règle 1 : Une sphère ω_1 est considérée comme emboîtée dans une autre sphère ω_2, et on note $\omega_1 \subset \omega_2$ ssi $\mathcal{A}ct(\omega_1) \subset \mathcal{A}ct(\omega_2)$

Règle 2 : Le niveau de cohésion d'une sphère définit la manière dont on gère la concurrence d'exécution des entités suivantes :
- une entité pour chaque activité de la sphère n'appartenant à aucune sous-sphère (on les appelera les activités à filiation directe de la sphère).
- une entité pour chacune des sous-sphères emboîtées (constituées d'activités à filiation indirectes de la sphère).

Règle 3 : Le niveau de cohérence d'une sphère définit la manière dont on gère les accès aux données manipulées par la sphère. Les entités concernées par la gestion d'une telle cohérence sont les activités **en dehors de la sphère**, si celle-ci n'est pas emboîtée. Dans le cas où la sphère est emboîtée, c'est-à-dire qu'elle constitue une sous-sphère, alors les entités concernées par une gestion de la cohérence sont les activités de la sphère **immédiatement parente**.

Deux sphères d'isolation ayant exactement les mêmes activités n'a aucun intérêt car, dans un tel cas, on attribut aux mêmes activités deux comportements éventuellement contradictoires. Nous supposons alors qu'il n'existe pas deux sphères ayant le même ensemble d'activités.

En reprenant l'illustration de la figure 50, nous illustrons dans la figure 51 la portée de la cohésion et de la cohérence des sphères d'isolation dans le cas de sphères emboîtées.

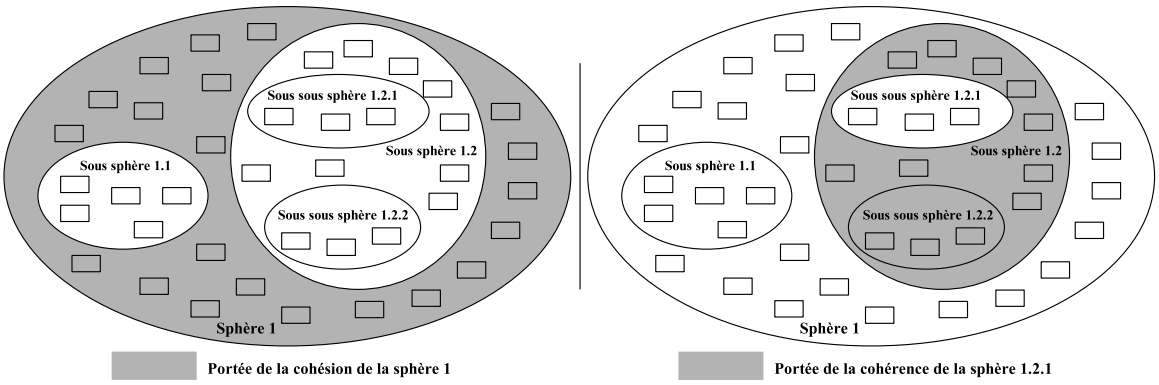


FIG. 51 – La portée de la cohésion et de la cohérence des sphères d'isolation dans le cas de sphères emboîtées

Ainsi, la portée de la cohésion se limite aux activités à filiation directe de la sphère en considérant les sous-sphères comme des entités au même titre que des activités et respectant donc le niveau de cohésion. La portée de la cohérence se limite, quant à elle, aux activités de la sphère parente, et à défaut d'une sphère parente, elle s'étend à l'ensemble des activités externes à la sphère.

Afin de l'utiliser dans la suite, et particulièrement dans l'élaboration d'un algorithme de gestion de la cohésion et de la cohérence, nous définissons $\widetilde{Act}_{\odot}(\omega)$ comme étant les activités directes d'une sphère d'isolation comme suit :

Définition 38 ***(Activités directes d'une sphère d'isolation)***
Sout ω une sphère d'isolation, nous définissons $\widetilde{Act}_{\odot}(\omega)$ comme étant les activités directes d'une sphère d'isolation tels que $\forall \widetilde{a} \in \widetilde{Act}(\omega)$:
si $\nexists \omega' \in \Omega$ telle que $\omega' \subset \omega$ et $\widetilde{a} \notin \widetilde{Act}(\omega')$
alors $\widetilde{a} \in \widetilde{Act}_{\odot}(\omega)$.

2.5.2 Cas de l'Intra-Sphère-Sérialisabilité et l'emboîtement de sphères

L'emboîtement de sphères d'isolation ayant toutes un niveau de cohésion maximal (sérialisable) ne constitue pas un intérêt particulier car leur exécution reviendrais à une seule sphère à cohésion sérialisable. Par contre, l'emboîtement de sphères d'isolation révèle un cas d'étude particulièrement intéressant lorsque les sous-sphères ont des niveaux de cohésion hétérogènes. Par exemple, supposons qu'une sphère principale de cohésion sérialisable englobe une sous-sphère de cohésion à lectures non validées. Dans un tel cas,

l'exécution globale obéit au critère d'Intra-Sphère-Sérialisabilité qui consiste à s'assurer que l'exécution des activités à filiation directe de la sphère principale, indépendamment des activités externes à celle-ci, soit sérialisable. L'exécution des activités de la sous-sphère sont alors considérées comme celles d'une seule activité en ce qui concerne la cohésion de la sphère gloable.

Le cas inverse est également intéréssant et consiste à avoir une sphère de cohésion à lectures non validées englobant une sphère de cohésion sérialisable. Les deux cas sont illustrés dans la figure 52.

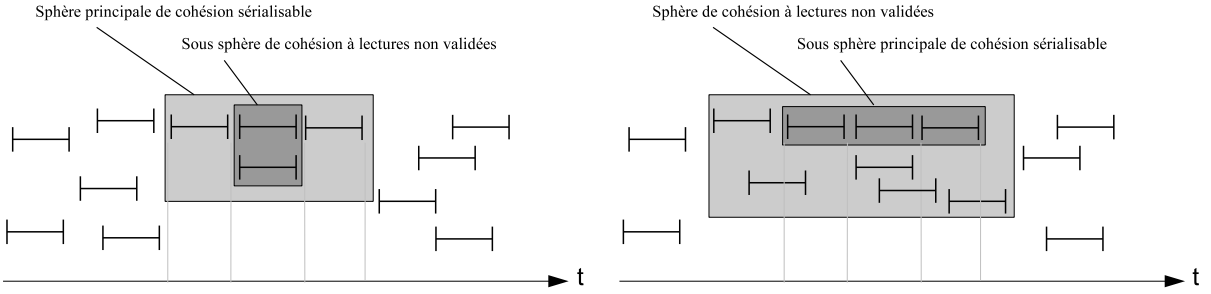


FIG. 52 – Intra-Sphère-Sérialisabilité avec des sphères emboîtées

2.5.3 Cas de l'Extra-Sphère-Sérialisabilité et l'emboîtement de sphères

Des résultats avancés de l'application du critère de Sphère-Sérialisabilité peuvent être exprimées à travers l'emboîtement de sphères d'isolation. En effet, des sphères d'isolation peuvent être emboîtées et cela enrichit la sémantique d'isolation. Il est inévitable d'avoir des besoins divergents en termes d'isolation entre les activités d'une même sphère. Un groupe d'activités peut alors contenir des sous-groupes ayant des besoins en isolation différents. Une sous-sphère d'isolation définit ses propres niveaux de cohésion et de cohérence mais gardent tout de même les bénéfices des niveaux d'isolation de la sphère mère. ' L'impact de l'emboîtement de sphères d'isolation est illustré dans l'exemple de la figure 53. Le critère d'Extra-Sphère-Sérialisabilité est également enrichi à travers l'emboîtement de sphères et un exemple d'histoire d'exécution Extra-Sphère-Sérialisable l'illustre dans la figure 54.

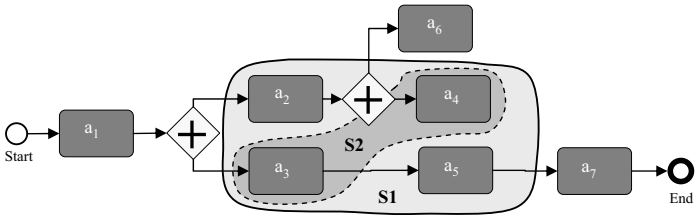


FIG. 53 – Sphères emboîtées

Dans cet exemple, la sphère d'isolation $S2$ définit ses propres niveaux d'isolation et l'Extra-Sphère-Sérialisabilité par rapport à S2 permet une exécution correcte du groupe composé de a_{4_4} et a_{4_3} avec le reste du procédé. $S1$ ajoute un autre niveau de sérialisation qui concerne les activités de $S1$ uniquement que sont a_{4_6} et a_{4_7}. Ainsi, l'Extra-Sphère-Sérialisabilité de sphères emboîtées permet d'avoir une

sérialisabilité multi-niveaux, ce qui représente un grand potentiel d'expressivité de la sérialisabilité des procédés métiers. Sur la base de la séparation entre la définition du procédé et celle du comportement transactionnel, les sphères d'isolation et l'Extra-Sphère-Sérialisabilité introduisent une nouvelle manière, pour les concepteurs de procédés métiers, de définir et d'exécuter les procédés métiers transactionnels et coopératifs.

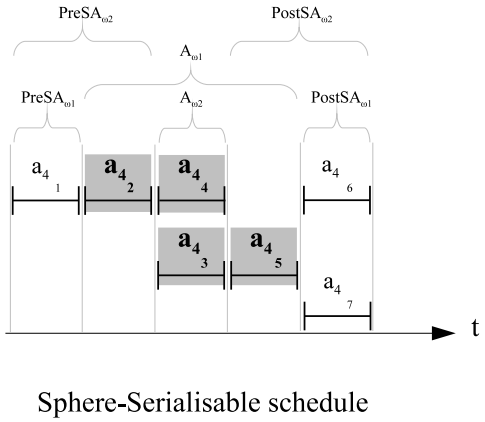


FIG. 54 – Extra-Sphère-Sérialisabilité avec des sphères emboîtées

2.5.4 Discussion à propos des possibilités de chevauchement de sphères

Le chevauchement de sphères d'isolation est une éventualité que nous devions poser à partir du moment où nous avons étudié leur emboîtement. Cela consiste à avoir deux sphères d'isolation ayant des activités communes sans que cela ne soit un emboîtement. Il existe donc un ensemble d'activités communes à deux sphères d'isolation. Le problème que nous posons est le suivant : Quel est le comportement, en termes d'isolation, que devront respecter les activités communes aux deux sphères chevauchées. Il n'existe pas une réponse catégorique à cette question. En effet, face à deux niveaux de cohésion et deux niveaux de cohérence, l'intention du concepteur du procédé n'est pas claire et il est difficile de trancher entre les deux niveaux s'ils sont différents.

Soit ω_1 et ω_2 deux sphères d'isolation telles que $\widetilde{Act}(\omega_1) \cap \widetilde{Act}(\omega_2) \neq \emptyset$ et $\widetilde{Act}(\omega_1) \not\subset \widetilde{Act}(\omega_2)$ et $\widetilde{Act}(\omega_2) \not\subset \widetilde{Act}(\omega_1)$. La première éventualité est de générer une troisième sphère d'isolation ω_3 permettant de changer la répartition des activités sur les trois sphères respectivement comme suit :

$$\begin{aligned}\widetilde{Act}(\omega_3) &= \widetilde{Act}(\omega_1) \cap \widetilde{Act}(\omega_2)\\ \widetilde{Act}(\omega_2) &= \widetilde{Act}(\omega_2) - \widetilde{Act}(\omega_3)\\ \widetilde{Act}(\omega_1) &= \widetilde{Act}(\omega_1) - \widetilde{Act}(\omega_3)\end{aligned}$$

La sphère ω_3 pourra alors définir ses propres niveaux de cohésion et de cohérence. Le problème qui perciste est le choix de ces niveaux. Trois choix s'offrent à nous. Le premier consiste à choisir le niveau le plus fort, le deuxième consiste à choisir le plus faible et enfin le dernier consiste à choisir un niveau intermédiaire. La question à se poser est la suivante : à partir du moment où l'intention du concepteur du procédé n'est pas déterministe, quel serait le choix des niveaux de cohésion et de cohérence pour la sphère ω_3

C'est pour ces raisons que nous pensons que le chevauchement de sphères d'isolation ne pourra pas aboutir à un résultat concrêt et nous préférons considérer qu'une telle éventualité restera exclue lors de la modélisation des sphères d'isolation.

2.6 Synthèse

Dans cette section, nous avons présenté l'approche des sphères d'isolation, sur la base des principes des sphères de comportement. Nous avons commencé par situer le contexte, les enjeux et les objectifs de l'étude de l'isolation dans les systèmes de gestion de procédés métiers. Nous avons ensuite posé les éléments de base d'une sphère d'isolation, en l'occurrence la cohésion et la cohérence d'une sphère.

La cohésion, tout comme la cohérence, présente plusieurs niveaux exprimant plusieurs degré de réalisation de l'isolation. La cohésion concerne alors l'isolation au sein de la sphère et la cohérence concerne l'isolation de la sphère par rapport à l'exterieur de la sphère. Nous avons également formalisé les différents niveaux de cohésion et de cohérence afin d'en expliquer, le plus clairement possible, le comportement souhaité des sphères d'isolation.

En définissant les niveaux de cohésion et de cohérence, nous avons étbalit un niveau maximal de cohésion, le niveau de sérialisable, et un niveau maximal de cohérence, le niveau de cohérence de sphère. Ces deux extrêmes nous ont amené à définir des critères de sérialisabilité assurant leur réalisation : le critère d'intra-Sphère-Sérialisabilité et le critère d'extra-Sphère-Sérialisabilité.

La définition de sphères d'isolation dans un procédé peuvent aboutir à des emboîtement de sphères. Cet emboîtement de sphères fait penser aux "Nested Transactions" qui étaient proposées dans le domaines des bases de données. Nous avons établi comment les niveaux de cohésion et de cohérence différentes peuvent cohabiter en définissant la portée de ces niveaux par rapport aux sphères parentes ou aux sphères filles. Nous avons également montré comment l'intra-Sphère-Sérialisabilité et l'extra-Sphère-Sérialisabilité peuvent être exprimées dans le cas d'emboîtement de sphères. Nous avons enfin discuter du cas du chevauchement de sphères qui s'est avéré peu fructueuse comme piste de recherche sans pour autant refermer entièrement les possibilités d'investigation dans une telle voie.

Dans la section qui suit, nous allons nous pencher plus particulièrement sur l'apport effectif des sphères d'isolation aux procédés coopératifs. Le travail coopératif a été longtemps considéré comme incompatible avec une isolation rigide et cela est justifiable. Á travers la flexibilité et la souplesse d'isolation que procurent les sphères d'isolation, nous allons analyser comment le travail coopératif sera affecté et comment les activités coopératives pourront bénéficier d'un environnement d'exécution plus fiable et adapté à leurs besoins réels.

3

Apports des sphères d'isolation aux procédés coopératifs

3.1 Les procédés coopératifs

Il n'est guère dans notre objectif de proposer une définition générale de ce qu'est un procédé coopératif [Goda 99]. Nous nous contentons de clarifier ce que nous voulons exprimer à travers un tel terme. Les procédés sont caractérisés par différentes propriétés. Certaines d'entre elles laissent apparaître leur nature coopérative.

D'abord, les procédés coopératifs sont exécutés par un groupe d'utilisateurs ayant une conscience de groupe participant à un travail coopératif. Un but commun est généralement défini et nous supposons que tous les participants sont au courant de ces détails. Ainsi chaque participant est partie prenante de la réalisation de cet objectif commun. Ensuite, la coopération dans un procédé est mise en œuvre à travers la communication et l'échange d'informations/données entre les participants. Nous supposons que le travail coopératif est organisé en plusieurs activités. Chaque participant peut réaliser plusieurs activités parmi celles constituant la coopération.

Généralement, le travail coopératif constitue un ensemble d'activités s'échangeant leurs résultats intermédiaires. Le résultat final de la coopération pourra alors changer progressivement, de façon incrémentale par exemple, durant l'exécution des activités coopératives. Nous proposons l'exemple de la figure 55 afin de mieux comprendre de tels procédés coopératifs. A travers cet exemple, nous allons mettre en avant les différents problèmes d'isolation qui peuvent survenir durant l'exécution.

L'exemple décrit un procédé coopératif de développement de logiciel utilisant la notation BPMN (Business Process Modeling Notation) [Whit 04]. Au commencement du procédé, deux activités ("Définir les modules" et "Éditer le squelette principal") permettent de définir les différents modules du projet de développement de logiciel et d'éditer son squelette (ou structure) principal. Chaque module est ensuite édité séparément.

La phase d'édition est effectuée par de multiples instances d'activités ("Éditer module" et "Tester module") durant une longue période de temps et produit de multiples versions temporaires (Drafts) de chacun des modules qui peuvent être échangés avec d'autres activités s'exécutant en parallèle. Après le test de chaque module ("Tester module"), des corrections et des mises à jour peuvent être effectuées coopérativement et cela de façon itérative jusqu'à ce que les tests soient positifs.

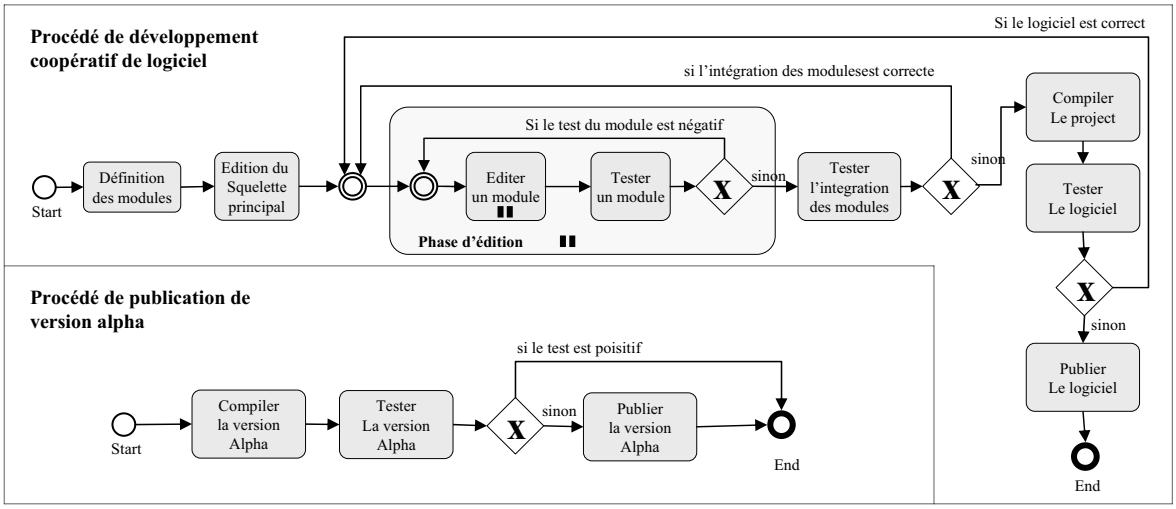


FIG. 55 – Un exemple de procédés coopératif

3.2 La controverse de la sérialisabilité dans les procédés coopératifs

La sérialisation de l'exécution d'un ensemble d'activités condamne toute possibilité de concurrence en ce sens que toute exécution est en série. Si dans un système trois activités a_1, a_2 *et* a_3 sont exécutées alors seules les exécutions suivantes sont possibles :

$$\begin{array}{ll} (a_1 \text{ puis } a_2 \text{ puis } a_3), & (a_1 \text{ puis } a_3 \text{ puis } a_2), \\ (a_2 \text{ puis } a_1 \text{ puis } a_3), & (a_2 \text{ puis } a_3 \text{ puis } a_1), \\ (a_3 \text{ puis } a_2 \text{ puis } a_1), & (a_3 \text{ puis } a_1 \text{ puis } a_2). \end{array}$$

La sérialisabilité a été introduite pour permettre une concurrence lors de l'exécution d'activités tout en garantissant la cohérence de leur exécution. C'est ainsi que parmi les exécutions suivantes seules les exécutions équivalentes à l'une des exécutions sérialisées sont possibles.

$$\begin{array}{ll} ((a_1 \text{ et } a_2) \text{ puis } a_3), & (a_1 \text{ puis } (a_2 \text{ et } a_3), \\ (a_2 \text{ puis } (a_1 \text{ et } a_3), & ((a_1 \text{ et } a_2 \text{ et } a_3). \end{array}$$

C'est là que se place la controverse avec les procédés coopératifs. En effet, les procédés coopératifs sont composés d'activités qui coopèrent entre elles à travers l'échange de données en cours d'exécution et souvent la manipulation concurrente des mêmes données. Si une exécution est équivalante à une exécution en série, aucun comportement coopératif du à l'échange de données ou à l'accès concurrent aux données ne pourra être effectif. C'est là la faille des systèmes de gestion de procédés : ils délèguent la gestion de l'isolation aux systèmes de bases de données qui ne prennent pas en considération la concurrence d'accès en tant que comportement désiré à travers la structure même du procédé. Nous appellerons cela la "dimension procédé" de la concurrence d'accès.

En ayant des exécution sérialisables, plusieurs activités pourront certes s'exécuter en concurrence mais pas en manipulant de façon concurrente les mêmes données. L'une des facettes du travail coopératif est la possibilité de manipuler librement des données de façon collective. Les anomalies d'isolation (lectures impropres, lectures non répétables, lectures fantômes) arrivent souvent dans le cadre d'un travail coopératif et cela ne pose pas forcément un problème car c'est souvent toléré. Cette tolérance s'explique par

le fait que, souvent, les participants à une coopération acceptent de faire des erreurs car ils considèrent que le travail est un brouillon à améliorer collectivement. C'est pour ces raisons que nous considérons que la sérialisabilité de l'exécution d'activités coopératives risque de limiter le comportement coopératif. Il est donc nécessaire d'apporter une certaine flexibilité de l'isolation afin de permettre aux activités coopératives de travailler dans un environnement plus souple.

3.3 Les besoins en isolation flexible dans les procédés coopératifs

A ce niveau d'analyse, nous pouvons identifier un premier besoin transactionnel. Cela consiste à relaxer les contraintes d'isolation pour les activités qui éditent un module afin qu'elles partagent leurs résultats et qu'elles éditent de façon coopérative. Il est également possible pour des activités éditant des modules différents de partager leurs versions temporaires (drafts). Cela est intéressant lorsque deux modules sont dépendants.

Après que tous les modules ont été testés séparément, une activité ("Tester intégration module") permet de tester l'intégration des modules les uns avec les autres en référence aux interfaces de chaque module qui peuvent toutefois changer durant le processus de développement. Selon le résultat obtenu, des modifications additionnelles de modules peuvent ou doivent être réalisées et cela itérativement jusqu'à l'obtention d'une solution satisfaisante.

À présent, supposons qu'un autre procédé soit exécuté en parallèle, que nous appellerons "Procédé de génération de versions Alpha" (cf. figure 55) qui consiste en un ensemble d'activités ayant pour but de fournir des versions temporaires (Drafts) du logiciel en cours de développement afin de le tester sans attendre la fin du développement. L'activité "Compiler version Alpha" réalise la compilation de tous les modules du projet afin de générer un exécutable du logiciel prêt à être testé pour l'évaluation du fonctionnement global. Ce type de procédés est fréquemment utilisé dans l'ingénierie des logiciels et permet d'avoir une vue sur le logiciel entier avant la fin de son développement. Les versions du logiciel générées par un tel procédé sont communément appelées "Versions Alpha". Une propriété transactionnelle particulière est importante à définir et à mettre en œuvre dans de tels cas. Ce second besoin consiste à assurer que, durant la compilation du projet, aucune édition de module ne peut être effectuée. L'intérêt d'une telle contrainte est principalement d'éviter, dans le cas d'une édition de fichiers, particulièrement dans le cas d'édition coopérative, que le contenu soit faussé au travers d'erreurs syntaxiques ou sémantiques dûes à des modifications interrompues en cours d'exécution.

La raison qui nous a conduit à définir ces deux contraintes est que, généralement, les procédés coopératifs sont composés de parties très flexibles, avec peu de contraintes sur l'accès aux données (isolation) et l'atomicité, et d'autres parties plus structurées et rigides, régies par des contraintes transactionnelles (par exemple la sérialisabilité). Une partie coopérative est composée d'activités qui expriment un comportement coopératif visant à accomplir un objectif commun et à partager leurs données avec les autres activités coopératives. Les parties non coopératives sont, quant à elles, composées d'activités strictement régulées (pas de partage coopératif de données).

Par exemple, dans la figure 55, les activités d'édition de chaque module ainsi que leurs activités de tests respectifs représentent une partie coopérative du procédé. Un exemple de parties non coopératives est composé des activités "Compiler le Projet", "Tester le logiciel" et "Publier le logiciel". De telles activités doivent s'exécuter en respectant une sérialisation stricte car l'activité "Publier le logiciel" doit publier la même version du logiciel qui a été compilée et testée. Mais dans l'exemple ces activités ne sont pas exposées à de possibles manipulations concurrentes des modules alors que la phase d'édition est forcément finie. Néanmoins, les activités "Compiler la version Alpha", "Tester la version Alpha" et "Publier la version Alpha" ont besoin d'avoir un même comportement transactionnel, particulièrement en raison du fait qu'elles s'exécutent pendant que des modifications concurrentes des modules sont réalisées dans la phase d'édition du "procédé de développement coopératif de logiciel".

Dans les parties coopératives, l'exécution peut être non sérialisable. Le critère de COO-Sérialisabilité [Cana 98a] a été proposé comme un critère alternatif pour les procédés coopératifs. Ce critère est une

extension du critère de sérialisabilité permettant des exécutions coopératives en autorisant un accès plus flexible aux données (opérations de lecture et d'écriture autorisées durant l'exécution des activités). Néanmoins, cette solution ne prend pas en compte la réalité des procédés coopératifs qui contiennent des parties coopératives et d'autres non coopératives.

Les modèles transactionnels actuels ne gèrent pas une flexibilité de la structure même des procédés coopératifs tels que nous venons de les présenter. Cette limitation existe car les systèmes de gestion de procédés actuels n'intègrent pas la "dimension procédé" dans leur gestion de la stratégie d'isolation. Par "dimension procédé" nous évoquons la prise en compte, non seulement du niveau de l'activité ou celui du procédé en entier comme entité, mais aussi des niveaux intermédiaires (sous-ensembles d'activités) qui rendent bien compte des interactions à l'intérieur d'un procédé pendant une gestion transactionnelle de l'exécution (une transaction dans un système de gestion transactionnelle est souvent assimilée à une activité ou à un procédé). Ainsi, nous avons pour objectif de permettre la spécification de propriétés transactionnelles pour des groupes d'activités et non seulement des activités individuelles ou des procédés entiers.

3.4 Identification des anomalies de coopération

Les systèmes de gestion de procédés métiers actuels s'assurent de la fiabilité de l'exécution en faisant recours au critère de sérialisabilité sur tous les procédés qui s'y exécutent et ce type d'approche exclut toute la flexibilité dont ont besoin les procédés coopératifs. Quelques éléments de flexibilité ont été introduits avec le modèle de transactions flexibles mais n'ont pas été adaptés à l'environnement coopératif. D'autres modèles tels que le modèle COO proposent des méthodes permettant de s'assurer d'une certaine flexibilité des exécutions coopératives d'activités. Cette dernière approche n'a pas pris en compte la co-exécution de travail coopératif et de travail non coopératif dans le même système. Dans ce cas, le critère de correction permettant les exécutions non coopératives ne satisfait pas les parties coopératives du procédé alors que le critère de correction permettant l'exécution correcte des parties coopératives ne satisfait pas les besoins des parties non coopératives.

Les sphères d'isolation introduisent une méthode permettant d'identifier formellement les parties coopératives et non coopératives des procédés. En introduisant différents niveaux de cohésion et de cohérence, les sphères d'isolation permettent aux activités coopératives de gérer leur exécution de telle sorte que les problèmes liés à la coopération soient résolus. Nous avons identifié un ensemble d'anomalies de coopération que nous tentons de gérer en utilisant notre approche des sphères de coopération. Afin d'identifier les anomalies induites par l'exécution de procédés coopératifs, nous nous focalisons particulièrement sur les deux propriétés présentés dans les sphères d'isolation, que sont la cohésion et la cohérence. Du point de vue de la cohésion, les anomalies qui s'y rattachent sont celles qui provoquent une perturbation de la coopération et la cohésion du groupe de coopérants. Cela représente la problématique de cohésion de groupes de coopérants. Nous avons identifié explicitement quatre anomalies se rattachant à la problématique de cohésion :

– **Coopération perturbée (Disrupted Cooperation) :** *deux instances d'activités* $\widetilde{a}_1$ *et* $\widetilde{a}_2$ *coopèrent à travers une sphère d'isolation en manipulant de façon concurrente une même donnée* δ*. Les deux instances d'activités utilisent une valeur de* δ *écrite durant leur exécution par une activité en dehors de la sphère.*

$$\exists\ \widetilde{a}_{ext} \notin \widetilde{\mathcal{A}ct}(\omega)\ ,\ \exists\ op[\delta] \in \mathcal{OP}(\widetilde{a}_{ext}) \text{ tel que}$$
$$\delta \in \bigcup(\Delta(\widetilde{a}_i))_{\widetilde{a}_i \in \widetilde{\mathcal{A}ct}(\omega)},\ Type(op) = \textit{écrire} \text{ et}$$
$$(op^{begin}_{\widetilde{\mathcal{A}ct}(\omega)} \xrightarrow{h} op\ \wedge\ op \xrightarrow{h} op^{end}_{\mathcal{A}ct(\omega)})$$

Les instances d'activités à l'intérieur de la sphère qui lisent des valeurs de la donnée sur laquelle elles coopèrent peuvent être induites en erreur si la donnée est modifiée, entre temps, par une

instance d'activité en dehors de la sphère.

– **Coopération à lectures impropres (Dirty Read Cooperation) :** *deux instances d'activités $\widetilde{a}_i$ et $\widetilde{a}_j$ coopèrent à travers une sphère d'isolation en manipulant de façon concurrente une même donnée δ. $\widetilde{a}_i$ écrit une valeur de δ, $\widetilde{a}_j$ lit cette valeur avant que $\widetilde{a}_i$ ne se termine et justement $\widetilde{a}_i$ fini par être abandonnée (et donc les modifications qu'elle aura réalisées sont annulées).*

$$\begin{array}{c}\exists\ \widetilde{a}_i, \widetilde{a}_j \in \widetilde{\mathcal{A}ct}(\omega)\ ,\ \exists\ op_i[\delta] \in \mathcal{OP}(\widetilde{a}_i)\ ,\ op_i^r \in \mathcal{OP}(\widetilde{a}_i) \text{ et } op_j[\delta] \in \mathcal{OP}(\widetilde{a}_j) \\ / \\ Type(op_i) = \textit{écrire}\ ,\ Type(op_j) = \text{textitlire}\ ,\ Type(op_i^r) = \text{textitrollback} \\ \text{et } (op_i \xrightarrow{h} op_j\ \wedge\ op_j \xrightarrow{h} op_i^r)\end{array}$$

Cela est similaire à l'anomalie de lecture impropre que nous rencontrons dans les systèmes de gestion de bases de données mais, dans ce cas, cela se limite à l'environnement de coopération.

– **Coopération à lectures non répétables (Fuzzy Read Cooperation) :** *deux instances d'activités $\widetilde{a}_i$ et $\widetilde{a}_j$ coopèrent à travers une sphère d'isolation en manipulant de façon concurrente une même donnée δ. $\widetilde{a}_i$ lit une valeur de δ, $\widetilde{a}_j$ écrit une nouvelle valeur de δ avant la terminaison de $\widetilde{a}_i$. Ainsi le travail de $\widetilde{a}_i$ risque d'être incorrect puisqu'elle utilise une valeur périmée de δ.*

$$\begin{array}{c}\exists\ \widetilde{a}_i, \widetilde{a}_j \in \mathcal{A}ct(\omega)\ ,\ \exists\ op_i[\delta] \in \mathcal{OP}(\widetilde{a}_i)\ ,\ op_i^c \in \mathcal{OP}(\widetilde{a}_i) \text{ et } op_j[\delta] \in \mathcal{OP}(\widetilde{a}_j) \\ / \\ Type(op_i) = \textit{lire}\ ,\ Type(op_j) = \textit{écrire}\ ,\ Type(op_i^c) = \textit{commit} \\ \text{et } (op_i \xrightarrow{h} op_j\ \wedge\ op_j \xrightarrow{h} op_i^c)\end{array}$$

Cela est analogue à l'anomalie de lecture non répétable (Fuzzy Read) que nous rencontrons dans les systèmes de gestion de bases de données mais, dans ce cas, cela se limite à l'environnement de coopération.

– **Coopération avec lectures fantômes (Phantom Read Cooperation) :** *deux instances d'activités $\widetilde{a}_i$ et $\widetilde{a}_j$ coopèrent à travers une sphère d'isolation en manipulant de façon concurrente des donnée d'une même table de base de données. $\widetilde{a}_1$ exécute une requête de sélection sur une base de donnée avec une condition de type "WHERE". Ensuite, $\widetilde{a}_2$ exécute une requête d'insertion ou de modification sur un tuple de la table avant que $\widetilde{a}_1$ ne soit terminée. Ainsi, $\widetilde{a}_1$ risque d'utiliser des données non à jour puisque l'ajout ou la modification de tuples dans la table aurait des répercutions sur le résultat de la requête de sélection initiale de $\widetilde{a}_1$.*

$$\begin{array}{c}\text{Soit } \textit{request} \text{ une requête sur les données de } \Delta. \\ \text{Soit } \Delta^-(\textit{request}, op) \text{ (respectivement } \Delta^+(\textit{request}, op)) \text{ représente le} \\ \text{sous-ensemble de données de } \Delta \text{ répondant à la requête } \textit{request} \text{ juste} \\ \text{avant (respectivement juste après) l'exécution de } op. \\ \exists\ a_i, a_j \in \mathcal{A}ct(\omega)\ ,\ \exists\ op_i[request] \in \mathcal{OP}(a_i)\ ,\ op_i^c \in \mathcal{OP}(a_i) \text{ et} \\ op_j \in \mathcal{OP}(a_j)\ / \\ Type(op_i) = \textit{lire}\ ,\ Type(op_j) = \textit{écrire}\ ,\ Type(op_i^c) = \textit{commit} \\ \text{et } (op_i \xrightarrow{h} op_j\ \wedge\ op_j \xrightarrow{h} op_i^c) \\ \text{et } \Delta^-(request, op_i) \neq \Delta^+(request, op_j)\end{array}$$

Cela est semblable à l'anomalie de lectures fantômes (Phantom Read) que nous rencontrons dans les systèmes de gestion de bases de données mais, dans ce cas, cela se limite à l'environnement de coopération.

Du point de vue de la cohérence, les anomalies que nous avons identifiées sont celles qui provoquent des perturbations sur les activités en dehors de la sphère d'isolation causées par la coopération d'activités de la sphère. Cela constitue la problématique de cohérence et représente les problèmes se répercutant sur

les activités non coopératives. L'implication des activités coopératives dans de tels problèmes consiste en un manque de vigilance quant à la mise à disposition des autres activités de données, validées ou pas, mais non encore permanentes par rapport au déroulement de la coopération. Nous avons pu identifier deux classes d'anomalies de cohérence qui sont les suivantes :

- **Lecture impropre externe (External Dirty Read) :** *une coopération à travers une sphère d'isolation induit des accès concurrents à une donnée δ et permet la lecture d'une valeur non validée de δ produite par une activité de la sphère.*

$$\begin{array}{c}\exists\ a_{int} \in \mathcal{Act}(\omega)\ ,\ \exists\ a_{ext} \notin \mathcal{Act}(\omega)\ ,\ \exists\ op_{int}[\delta] \in \mathcal{OP}(a_{int})\ ,\\ op^{r}_{int} \in \mathcal{OP}(a_{int}) \text{ et } op_{ext}[\delta] \in \mathcal{OP}(a_{ext})\ /\\ Type(op_{int}) = \acute{e}crire\ ,\ Type(op_{ext}) = lire\ ,\ Type(op^{r}_{int}) = rollback\\ \text{et } (op_{int} \xrightarrow{h} op_{ext}\ \wedge\ op_{ext} \xrightarrow{h} op^{r}_{int})\end{array}$$

  Des activités en dehors de la sphère d'isolation peuvent lire des valeurs de données non encore validées par des activités coopératives. Cela peut les induire en erreur si l'activité coopérative ayant écrit la valeur soit abandonnée et voit donc ses effets annulés. La valeur lue par les activités externes à la sphère n'est plus appropriée et c'est pour cette raison que nous appelons une telle anomalie "Lecture impropre externe".

- **Lecture externe trompeuse (External Misleading Read) :** *une coopération à travers une sphère d'isolation induit des accès concurrents à une donnée δ par les instances d'activités de la sphère et permet la lecture d'une valeur de δ validée par une instance activité de la sphère.*

$$\begin{array}{c}\nexists\ a_{ext} \notin \mathcal{Act}(\omega) \text{ tel que } \exists\ a_{int} \in \mathcal{Act}(\omega)\ ,\ \exists\ op_{int}[\delta] \in \mathcal{OP}(a_{int})\ ,\\ op^{c}_{int} \in \mathcal{OP}(a_{int}) \text{ et } op_{ext}[\delta] \in \mathcal{OP}(a_{ext})\ /\\ Type(op_{int}) = \acute{e}crire\ ,\ Type(op_{ext}) = lire\ ,\ Type(op^{c}_{int}) = commit\\ \text{et } (op^{c}_{int} \xrightarrow{h} op_{ext}\ \wedge\ op_{ext} \xrightarrow{h} op^{end}_{\mathcal{Act}(\omega)})\end{array}$$

  Les activités en dehors de la sphère qui lisent des valeurs validées par des activités de la sphère vont, généralement, considérer que ces valeurs sont des finales puisqu'elles ne sont pas forcément au courant de la coopération elle-même et que le processus de coopération n'est pas encore achevé. Le résultat final de la coopération n'étant pas encore atteint, si les activités de la sphère n'ont pas toutes fini leur exécution, la valeur validée mise à disposition des activités externe ne fait que les tromper en participant à la confusion entre valeur validée et valeur finale de la coopération. C'est pour cette raison que nous appelons une telle anomalie "Lecture externe trompeuse".

3.5 Apports des sphères d'isolation à la gestion flexible des anomalies de coopération

En appliquant les niveaux de cohésion et de cohérence aux procédés coopératifs, nous donnons aux concepteurs de procédés la possibilité de définir les niveaux de cohésion et de cohérence appropriés. Il revient aux concepteurs de tels procédés de prendre la décision à propos du niveau de cohésion ou de cohérence et ce en recourant à une approche qui autoriserait ou interdirait les anomalies de coopération que nous avons identifiées. Cela permettra aux concepteurs de procédés de choisir les niveaux adaptés à la situation étudiée.

La dualité cohésion/cohérence nous permettra de réguler la stratégie d'isolation selon un total de 12 combinaisons de niveaux (4 niveaux de cohésion **x** 3 niveaux de cohérence). Ces combinaisons illustrent toutes les possibilités de flexibilité de l'isolation en général et de la coopération en particulier comme indiqué dans la figure 56.

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. 56 – Dualité Cohésion/Cohérence : comportement coopératif à la carte

Ces possibilités introduisent une grande flexibilité et permettent d'exprimer le degré de flexibilité de la coopération par rapport au degré de cohérence des données. Les combinaisons intermédiaires prennent en compte à la fois un niveau permettant une exécution coopérative et la un échange souple de données entre les participants à la coopération ainsi qu'un certain niveau de sûreté de l'exécution. Mais cette sûreté d'exécution est basée sur la visibilité des données manipulées sur lesquelles les activités coopératives travaillent.

3.6 Application des sphères d'isolation à un exemple pratique de procédé coopératif

L'exemple que nous avons proposé dans la figure 55 au début de cette section illustrait deux procédés coopératifs : "Procédé de développement coopératif de logiciel" et "Procédé de publication de versions Alpha". Nous avons identifié différents besoins pour différentes parties de ces procédés et nous proposons dans cette section une solution à base de sphères d'isolation capable de répondre à ces besoins en isolation en accord avec les besoins en coopération. Durant la phase d'édition, de multiples activités éditent les mêmes modules et ceux là ont besoin de travailler de façon coopérative et donc d'avoir des contraintes d'isolation assez souples les unes avec les autres. Cela est possible si nous utilisons une sphère d'isolation $IS1$ telle qu'illustrée dans la figure 57.

Le choix des niveaux de cohésion et de cohérence est adapté au contexte d'édition coopérative. Nous supposons que les sphères d'isolation, lorsqu'elles contiennent une activité à instanciation multiple, incluront, à l'exécution, toutes ces instances d'activité correspondantes. D'un autre côté, la sphère $IS1$ est elle-même incluse dans une activité à instanciation multiple, l'activité "Phase d'édition". Nous supposons que pour chaque instance de "Phase d'édition", une sphère d'isolation différente est instanciée.

Pour chaque module, plusieurs instances de l'activité "Editer un module" sont exécutées et une instance de "Tester un module" est exécutée. Cela est réalisé itérativement jusqu'à ce que le test soit positif. En utilisant le niveau de cohérence tel qu'illustré dans la figure 57, nous permettons à d'autres

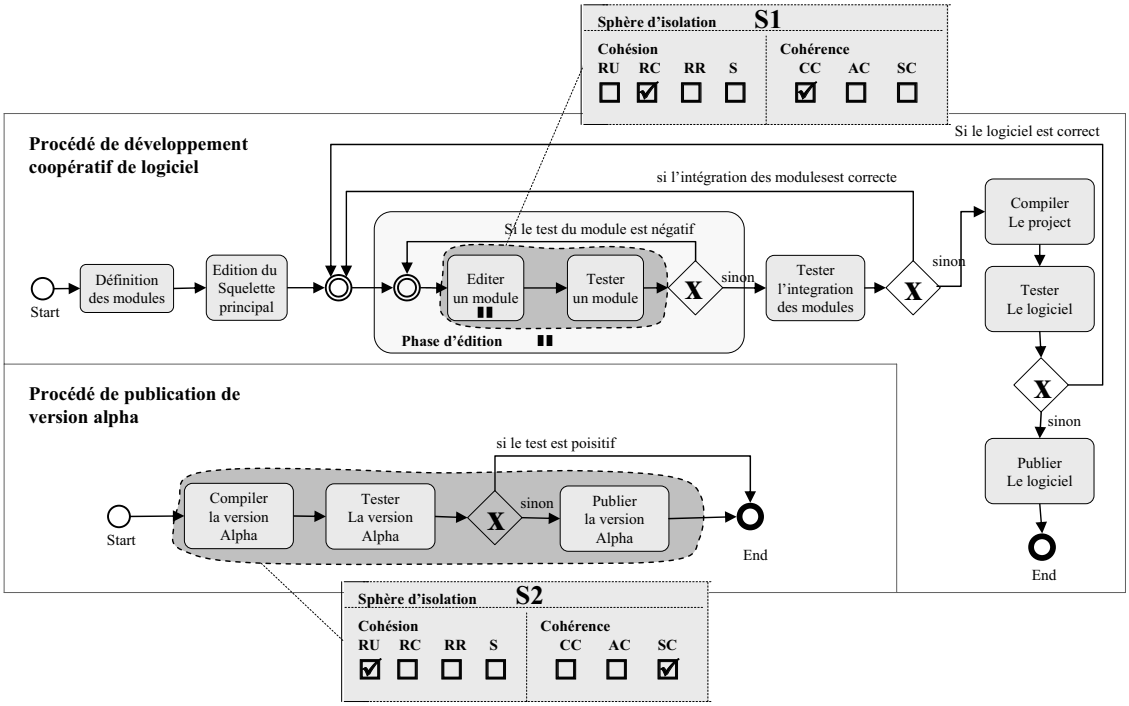


FIG. 57 – Application des sphères d'isolation à un exemple pratique de procédé coopératif

activités, en dehors de la sphère, de lire les données produites par la phase d'édition sans forcément attendre la fin de son exécution ou la validation des données. Cela procure un haut degré de flexibilité de l'exécution.

En ce qui concerne le niveau de cohésion, nous avons choisi le niveau de lectures non validées afin d'assurer le fait que les activités de la sphère travaillent avec le maximum de liberté et de flexibilité possible au sein de la même sphère. Le travail coopératif est ainsi assuré.

Étudions à présent un second besoin en isolation qui concerne le "procédé de publication de version Alpha". Il est convenu qu'une version alpha d'un logiciel est une version incomplète et dont le fonctionnement est plus ou moins chaotique. Ainsi, la sûreté, l'intégrité ou la cohérence de fonctionnement d'une version alpha d'un logiciel n'est pas du tout garantie. Néanmoins, nous avons besoin de nous assurer que la compilation d'une version alpha, son test et sa publication (les activités "Compiler une version Alpha", "Tester une version Alpha" et "Publier une version Alpha") manipulent les mêmes données (la même version de chaque module). Cela est possible à travers l'utilisation d'une sphère d'isolation $IS2$ de la figure 57. Son niveau de cohésion serait "Lecture non validées" car les modules utilisés ne sont pas obligatoirement validés lorsque "Compiler une version Alpha" est réalisée. Le niveau de cohérence serait "Cohérence de sphère" car la version Alpha publiée ne peut être utilisée par d'autres procédés/activités qu'après la fin d'exécution de toutes les activités de la sphère.

3.7 Synthèse

Dans ce chapitre, nous avons introduit la notion de sphères de comportement et nous avons tenté de fournir une référence permettant de réunir tout type de sphères. Nous avons défini trois principes fondamentaux permettant d'identifier de nouvelles propriétés dont la réalisation à travers des sphères est particulièrement intéréssente. Nous avons également cité quelques exemples de sphères possibles.

Parmi les pistes de sphères de comportement possibles, nous avons choisi de développer en profondeur le cas de la propriété d'isolation. La flexibilité et l'expressivité de l'isolation dans les procédés métiers est un problème assez connu mais souvent cité en tant que tel et jamais étudié à juste titre. Notre étude nous a amené à définir deux dimensions des sphères d'isolation : la cohésion et la cohérence. Pour chacune des deux dimensions, nous avons défini plusieurs niveaux. Ainsi, à chaque niveau de cohésion ou de cohérence correspond un comportement particulier plus ou moins flexible et plus ou moins fiable.

Nous avons montré comment la solution des sphères d'isolation pouvait apporter une valeur ajoutée particulièrement en ce qui concerne les procédés coopératifs. En nous penchant sur de tels procédés, nous avons pu identifier un certain nombre d'anomalies de coopération risquant de durvenir lors d'exécution où l'isolation n'est pas du tout contrôlée. Les sphères d'isolation permettent de gérer de façon flexible l'isolation de groupes d'activités coopérative de telle sorte que les anomalies sont autorisées ou non selon les niveaux de cohésion et de cohérence choisis. Nous avons donc montré l'intérêt, par des exemples pratiques, de disposer d'une telle solution de modélisation et d'exécution de la propriété comportementale de l'isolation dans les procédés métiers en général et particulièrement les procédés coopératifs.

Dans le chapitre qui suit, nous allons entammer l'implémentation de l'approche des sphères de comportement et des sphères d'isolation en particulier. Nous présenterons d'abord comment nous pourrons passer de la théorie à la pratique, en ce qui concerne la gestion de l'isolation et cela à travers un protocole décrit par un algorithme que nous avons appelé CCDP (Cohesion Coherence Duality Protocol). Nous présenterons ensuite l'environnement de mise en oeuvre et les différentes étapes qui nous ont conduit à la concrétisation des sphères de comportement en général et des sphères d'isolation en particulier.

Chapitre IV

Mise en œuvre des sphères d'isolation

1. **De la théorie à la pratique : gestion de la dualité Cohésion/Cohérence**
2. **Enjeux de la mise en œuvre des sphères de comportement / d'isolation**
3. **Sphères de comportement pour la coordination de services web**
4. **Mise en oeuvre des sphères d'isolation dans une plateforme de web services**

1

De la théorie à la pratique

Gestion de la dualité Cohésion/Cohérence

Après avoir présenté la notion de sphère d'isolation et formalisé ses caractéristiques ainsi que son comportement face aux accès concurrentiels aux données, il devient indispensable de proposer une solution pratique capable d'assurer un tel comportement. Nous nous proposons dans ce qui suit de présenter un nouveau protocole à base de verrous permettant d'assurer la cohésion et la cohérence des sphères d'isolation.

1.1 Le protocole CCDP (Cohesion/Coherence Duality Protocol)

Nous proposons dans cette section le protocole CCDP de gestion de la dualité Cohésion/Cohérence dans les sphères d'isolation. Ce protocole gère l'accès aux données d'un point de vue procédé par opposition aux protocoles classiques liés aux systèmes de base de données. En effet, les protocoles tels que 2PC (Two Phase Commit) contrôlent l'accès à des données verrouillées. L'approche de verrouillage ne distingue pas des sous-ensembles de transactions afin de leur attribuer plus ou moins de flexibilité dans l'accès aux données manipulées. Nous avons proposé dans la section précédente notre approche basée sur les sphères d'isolation et introduit les notions de niveau de cohésion et de cohérence. Nous proposons maintenant le protocole CCDP permettant d'assurer le respect de ces niveaux et nous nous basons sur le formalisme que nous avons introduit précédemment afin de mieux exprimer le fonctionnement d'un tel protocole.

Le protocole CCDP est utilisé dans le cadre des sphères d'isolation et a pour objectif de rendre plus ou moins optimiste la stratégie d'isolation de procédés. Par "optimiste" nous voulons parler de partage des résultats intermédiaires. C'est en ce sens que le protocole CCDP envisage la régulation des lectures et écritures effectuées par les différentes activités participant à une sphère d'isolation ainsi que celles des activités qui n'y participent pas. Nous avons donc deux volets à explorer et donc deux facettes à ce protocole (dualité).

Cette dualité est paramétrée par les deux dimensions de cohésion et de cohérence dont les niveaux sont fixés par la sphère d'isolation. Afin d'assurer une exécution conforme aux attentes exprimées par chaque niveau de cohésion et de cohérence, il est utile de définir des types de verrous correspondant à chacun des volets du protocole et d'aboutir à un algorithme unique de gestion des lectures/écritures réalisées par des activités de sphères d'isolation.

Ci-dessous nous allons présenter les types de verrous que nous définissons, régis par le protocole CCDP.

1.1.1 Les verrous du protocole CCDP

Á l'opposé des verrous traditionnels en lecture et en écriture utilisés dans les bases de données, nous définissons des verrous relatifs à un sous ensemble d'activités qui se déclinent en deux types :

- $VL(\delta, Act, e)$: verrou en lecture sur une donnée δ relatif à un sous ensemble $Act \subset \mathcal{A}$ et posé par une entité e. Une entité e peut correspondre à une instance d'activité $\widetilde{a}$ ou à une sphère d'isolation ω.
- $VE(\delta, Act, e)$: verrou en écriture sur une donnée δ relatif à un sous ensemble $Act \subset \mathcal{A}$ et posé par une entité e. Une entité e peut correspondre à une instance d'activité $\widetilde{a}$ ou à une sphère d'isolation ω.

Lorsqu'une activité $\widetilde{a}$ tente de lire une donnée δ, elle ne peut le faire qu'à condition qu'aucun verrou en écriture $VE(\delta, Act, \widetilde{a})$ ne soit posé. C'est à dire que si un verrou en écriture est posé relativement à un groupe d'activités contenant $\widetilde{a}$ alors cette dernière ne pourra lire la donnée qu'après la levée du verrou. De même, l'écriture d'une donnée n'est pas possible si un aucun verrou en écriture ne soit posé sur cette donnée relativement à l'activité désirant l'écrire.

Si le test de l'existance de verrous est facile à mettre en place, la question qu'on doit se poser est la suivante : Lorsqu'une activité est autorisée à écrire ou à lire une donnée, quels seraient les verrous à poser et quelle seraient leurs portées respectives. C'est pour répondre à cette question que nous avons développé un algorithme permettant de gérer les verrous et leurs portées à chaque lecture ou écriture d'une donnée.

1.1.2 Algorithme de gestion de la cohésion et de la cohérence des sphères lors d'une tentative de lecture ou d'écriture de données

L'algorithme permettera, suite à une demande de lecture ou d'écriture d'une donnée δ par une activité $\widetilde{a}_i$, de l'autoriser ou de la temporiser tout en prenant en compte l'emboîtement de sphères d'isolation. Cet algorithme se présente comme suit :

Algorithme : CCDPcheck
Algorithme assurant que la lecture ou l'écriture d'une donnée δ respecte la cohésion et la cohérence des sphères d'isolation.

Entrées : $\widetilde{a}_i \in \widetilde{\mathcal{A}}$
$op \in mathpcalOP$
$\delta \in \Delta$
Sortie : aucune

Début
Soit $\omega \in \Omega$ telle que $\widetilde{a}_i \in \widetilde{Act}_{\odot}(\omega)$
TantQue ($(Type(op) = lire \wedge \exists\, VE(\delta, \widetilde{a}_i, *)) \vee (Type(op) = ecrire \wedge \exists\, (VL(\delta, \widetilde{a}_i, *) \vee VE(\delta, \widetilde{a}_i, *)))$
/ L'étoile '*' veut dire 'quelconque' */*
Attendre() ou Abandonner()
FinTantQue

/ Cohésion de la sphère */*
Si ($\mathcal{CHS}(\omega) >= 1$ et $Type(op) = ecrire$)
Poser($VE(\delta, \widetilde{Act}_{\odot}(\omega), \widetilde{a}_i)$
Si ($\mathcal{CHS}(\omega) == 2$ et $Type(op) = lecture$)

Poser($VL(\delta, \widetilde{Act}_{\odot}(\omega), \widetilde{a}_i)$)
SinonSi ($\mathcal{CHS}(\omega) == 3$ et $Type(op) = lecture$)
Poser($VE(\delta, \widetilde{Act}_{\odot}(\omega), \widetilde{a}_i)$)
FinSi

/* *Cohérence de la sphère* */
Si $\exists\ \omega' \in \Omega$ la sphère immédiatement parente de ω
Si ($\mathcal{CHR}(\omega) == 0$)
Poser($VL(\delta, \widetilde{Act}_{\odot}(\omega'), \omega)$)
Pour tout $\omega_i \subset \omega'$, $\omega_i \neg \omega$
Poser($VL(\delta, \widetilde{Act}_{\odot}(\omega_i), \omega)$)
FinPour

SinonSi ($\mathcal{CHR}(\omega) == 1$)
Poser($VL(\delta, \widetilde{Act}_{\odot}(\omega'), \omega)$)
Poser($VE(\delta, \widetilde{Act}_{\odot}(\omega'), \widetilde{a}_i)$)
PourTout ($\omega_i \subset \omega'$ / $\omega_i \neg \omega$)
Poser($VE(\delta, \widetilde{Act}_{\odot}(\omega_i), \widetilde{a}_i)$)
FinPour
SinonSi ($\mathcal{CHR}(\omega) == 2$)
Poser($VE(\delta, \widetilde{Act}_{\odot}(\omega'), \omega)$)
PourTout ($\omega_i \subset \omega'$ / $\omega_i \neg \omega$)
Poser($VE(\delta, \widetilde{Act}_{\odot}(\omega_i), \omega)$)
FinPour
FinSi
Sinon
Si ($\mathcal{CHR}(\omega) == 0$)
Poser($VL(\delta, \mathcal{A} - \widetilde{Act}(\omega), \widetilde{a}_i)$)
SinonSi ($\mathcal{CHR}(\omega) == 1$)
Poser($VE(\delta, \mathcal{A} - \widetilde{Act}(\omega), \widetilde{a}_i)$)
SinonSi ($\mathcal{CHR}(\omega) == 2$)
Poser($VE(\delta, \mathcal{A} - \widetilde{Act}(\omega), \omega)$)
FinSi

/* *Cohésion de la sphère immédiatement parente* */
Si $\exists\ \omega' \in \Omega$ la sphère immédiatement parente de ω
Si ($\mathcal{CHS}(\omega') >= 1$ et $Type(op) = ecrire$)
Poser($VE(\delta, \widetilde{Act}_{\odot}(\omega'), \omega)$)
Si ($\mathcal{CHS}(\omega') == 2$ et $Type(op) = lire$)
Poser($VL(\delta, \widetilde{Act}_{\odot}(\omega'), \omega)$)
SinonSi ($\mathcal{CHS}(\omega') == 3$ et $Type(op) = lire$)
Poser($VE(\delta, \widetilde{Act}_{\odot}(\omega'), \omega)$)
FinSi
FinSi
Fin

L'algorithme CCDP gère la pose et la libération de verrous relatifs. Il est clair que poser un verrou classique (absolu) ne demande pas un traitement particulier. Dans le cas de verrous relatifs, il est nécessaire de traiter les informations disponibles afin de savoir sur qui influera le verrou à poser. C'est

ainsi que lors d'une demande de lecture ou d'écriture d'une donnée, selon l'appartenance de l'activité qui demande une telle opération à une sphère d'isolation et selon les niveaux de cohésion et de cohérence de cette sphère, des verrous relatifs en lecture ou en écriture sont posés relativement à un certain sous ensemble d'activités du système. De plus, l'algorithme présenté prend en compte l'emboîtement de sphères d'isolation et se conforme à la portée des niveaux de cohérence et de cohésion que nous avons définis dans la section 2.5.

Lorsqu'une activité $\widetilde{a}_i$ désire lire une donnée δ, elle ne peut le faire que lorsqu'aucun verrou en écriture n'est posé relativement à cette activité. Deux choix théoriques sont proposés : l'attente ou l'abandon. Dans la réalité, il s'agit souvent d'abandonner lorsque l'activité qui a posé le verrou est à longue durée d'exécution. Il s'agira d'attendre lorsque l'activité qui a posé le verrou est à courte durée d'exécution. Malheureusement, cette information n'est pas toujours disponible et le choix d'attendre ou d'abandonner demeurera un choix arbitraire de l'activité. Remarquons que le fait d'abandonner peut être suivi d'une nouvelle tentative quelque temps plus tard et n'induit pas systématiquement l'échec de l'activité $\widetilde{a}_i$. De même lorsqu'il s'agit d'une lecture de la donnée δ, celle ci n'est possible qu'en absence de verrous en écriture ou en lecture posés relativement à l'activité $\widetilde{a}_i$ sur la donnée δ.

Une fois que l'activité $\widetilde{a}_i$ se retrouve capable de lire une donnée (pas de verrous en écriture relatif à $\widetilde{a}_i$) ou d'écrire une donnée (pas de verrous en écriture ou en lecture relatifs à $\widetilde{a}_i$), l'algorithme passe alors dans une deuxième phase. Il s'agit de poser les verrous nécessaire à assurer la cohésion de la sphère, que nous noterons ω de laquelle $\widetilde{a}_i$ est une activité directe. Le choix de commencer par la sphère dont l'activité est directement participante est curcial car il permettra de respecter les règles d'emboîtement de sphères s'il y a lieu.

Si le niveau de cohésion de la sphère ω est supérieur ou égal à 1, c'est à dire que les lectures sont obligatoirement validées, alors, dans le cas d'une écriture de δ, un verrou en écriture devra être posé relativement aux activités directes de la sphère afin d'assurer le fait qu'aucune autre activité directe de la sphère ne puisse lire une données non encore validée. La terminaison de l'activité $\widetilde{a}_i$ engendrera la levée d'un tel verrou. S'il s'agit simplement d'une lecture et que le niveau de cohésion de la sphère ω est égal à 2, c'est à dire au niveau de lectures répétables, l'activitié $\widetilde{a}_i$ se trouve dans l'obligation d'interdire à d'autres activités d'écrire la donnée δ et ce jusqu'à sa terminaison. C'est pour cette raison qu'il faudra que $\widetilde{a}_i$ pose un verrou en lecture relatif à l'ensemble des activités directes de la sphère ω. Dans le cas d'une lecture avec un niveau de cohésion de la sphère ω égal à 3, c'est à dire au niveau sérialisable, l'activitié $\widetilde{a}_i$ se trouve dans l'obligation d'interdire à d'autres activités de lire ou d'écrire la donnée δ et ce jusqu'à sa terminaison afin d'assurer la sérialisabilité de l'exécution. Il s'agit alors de la solution rigide. C'est pour cette raison qu'il faudra que $\widetilde{a}_i$ pose un verrou en écriture relatif à l'ensemble des activités directes de la sphère ω.

La troisième phase de l'algorithme est celle qui permet la pose des verrous nécessaires à la satisfaction des niveaux de cohérence de la sphère ω ainsi que d'une éventuelle sphère immédiatement parente ω'. En effet, deux cas distincts existent. Le premier cas est celui de l'existence d'une sphère immédiatement parente à ω notée ω'. Dans un tel cas, si le niveau de cohérence est < 2, c'est à dire un niveau de cohérence coopératif ou d'activité, la sphère ω pose un verrou en lecture relativement aux activités directes de ω'. Un tel verrou permettra d'éviter des écritures de la part des activités directes de ω' tant que toutes les activités de la sphère ω ne soient totalement terminées. Ensuite, selon le niveau de cohérence de la sphère ω, des verrous en lecture ou en écriture sont posés par la sphère ω ou par l'activité $\widetilde{a}_i$ relativement aux activités directes de ω' et/ou des sous sphères de ω' autres que ω (voir l'algorithme). Dans le cas où aucune sphère parente n'existe, les verrous sont posés relativement au reste des activités du procédé, extérieures à la sphère ω. Si le niveau de cohérence est égal à 0, il s'agit d'un verrou en lecture posé par $\widetilde{a}_i$ relativement aux activités externes à la sphère ω. Un tel verrou permettra simplement d'éviter que des activités externes à la sphère n'écrivent de nouvelles valeurs de δ avant la fin de l'exécution de $\widetilde{a}_i$. La lecture des valeurs non validées écrites par $\widetilde{a}_i$ demeure possible. Si le niveau de cohérence est égal à 1, il s'agit d'un verrou en écriture posé par $\widetilde{a}_i$ relativement aux activités externes à la sphère ω. Face à un verrou en écriture, la lecture de valeurs non validées n'est plus possible par des activités externes à la sphère. Enfin, si le niveau de cohérence est égal à 2, il s'agit d'un verrou en écriture posé par ω relativement aux activités externes à la sphère ω. Dans un tel cas, la lecture de δ ne sera possible

qu'après la fin de l'exécution de toutes les activités de ω.

Une dernière et quatrième phase de l'algorithme concerne la gestion de l'impact du niveau de cohésion de la sphère immédiatement parente $\omega^{'}$ sur la pose de verrous par la sphère ω. Un tel impact s'explique par le fait que pour $\omega^{'}$, la sphère ω obéit à la cohésion de $\omega^{'}$ autant qu'une simple activité directe de celle-ci. D'un côté, s'il s'agit d'une écriture de δ et que le niveau de cohésion de $\omega^{'}$ est $>= 1$, un verrou en ériture est posé lecture est posé par ω sur les activités directes de $\omega^{'}$. Un tel verrou permettra d'éviter que des activités de $\omega^{'}$ lisent des données non validées de δ. D'un autre côté, s'il s'agit d'une lecture avec un niveau de cohésion de $\omega^{'}$ égal à 2 (lectures répétables), un verrou en lecture est posé par ω sur les activités directes de $\omega^{'}$. Un tel verrou permet d'éviter qu'une activité directe de $\omega^{'}$ puissent écrire le donnée δ avant que toutes les activités de ω ne finissent leurs exécutions. Dans le cas d'un niveau de cohésion égal à 3 (sérialisable), un verrou en écriture est posé par ω sur les activités directes de $\omega^{'}$. Il s'agit alors d'interdire les écritures et les lectures jusqu'à ce que toutes les activités de ω ne finissent leurs exécutions.

1.2 synthèse

Dans cette section, nous avons présenté le protocole CCDP, sous forme d'un algorithme, permettant d'assurer le respect des niveaux de cohésion et de cohérence d'une sphère d'isolation. Un tel protocole prend en considération l'emboîtement de sphères d'isolation et utilise des verrous spéciaux. Les verrous utilisés sont dits "relatifs", en ce sens qu'ils n'influencent qu'un sous ensemble d'activités. Deux types de verrous relatifs sont proposés : les verrous relatifs en lecture ou en écriture.

Selon qu'il s'agisse d'une lecture ou d'une écriture de données, l'algorithme permet de poser les bons verrous relativement aux bonnes activités et au bon moment. Un algorithme peut être posé par une activité ou par une sphère. Un verrou posé est levé dès que le poseur du verrou a fini son exécution.

En ayant définit un algorithme capable d'assurer le respect des niveaux de cohésion et de cohérence des sphères d'isolation, il nous reste de définir l'environnement de mise en oeuvre. C'est l'objectif de la section qui suit.

2

Enjeux de la mise en oeuvre des sphères de comportement / d'isolation

2.1 Les enjeux de la mise en œuvre

L'approche des sphères de comportement a, du point de vue conceptuel, introduit une nouvelle façon de modéliser des procédés métiers apportant plus de flexibilité et d'expressivité. Afin de mettre en oeuvre une telle approche, il est judicieux d'en étudier les enjeux et les objectifs. Pour cela, il est important de prendre en considération les différents volets de l'approche à base de sphères que sont le volet de notation ou graphique, le volet de spécification ou de modélisation, et enfin le volet de l'exécution ou opérationnel.

En premier lieu, l'approche des sphères d'isolation s'intéresse au volet notationnel puisqu'une sphère est définie, comme nous l'avons montré dans les exemples utilisés précédemment, à travers une représentation graphique. C'est à partir de ce constat que nous relevons un premier objectif qui est celui de la mise en oeuvre d'un outil de modélisation de procédés métiers prenant en charge, graphiquement, les sphères d'isolation, et à terme, tout type de sphères.

L'un des objectifs de la mise en oeuvre des sphères de comportement est donc de fournir un outil permettant, non seulement la modélisation de procédés métiers de manière classique, en utilisant une notation normalisée, mais aussi d'introduire la représentation des sphères de comportement et d'isolation en particulier.

En second lieu, l'approche des sphères d'isolation s'appuie sur la séparation entre la spécification du procédé métier et la spécification de la propriété d'isolation. Cette séparation des préoccupations constitue l'un des atouts de l'approche car le modèle de procédé n'est pas altéré. Ainsi, l'un des enjeux de la mise en oeuvre des sphères d'isolation réside dans la définition d'une spécification compatible avec les plateformes d'exécution de procédés métiers.

La spécification des sphères de comportement, et particulièrement celle de l'isolation, ne devra pas altérer le modèle, ou la norme de spécification du procédé utilisée. Cela est d'autant plus crucial qu'un effort important à été réalisé depuis une décennie dans le travail de normalisation et de standardisation des spécifications. C'est le cas, par exemple, des plateformes de web services et des différentes spécifications décrivant les WS-Technologies. Il est donc nécessaire que le travail de mise en oeuvre que nous nous proposons de réaliser dans cette thèse n'aille pas à l'encontre des standards existants mais, bien au contraire, les renforce en essayant d'en utiliser l'extensibilité.

En dernier lieu, l'approche des sphères d'isolation se fonde sur une structure à deux dimensions (cohésion et cohérence) qui régit l'exécution des activités de la sphère et de celles en dehors de celle-ci. C'est dans ce sens que la mise en oeuvre d'une telle approche concerne la partie opérationnelle des procédés métiers. L'exécution des procédés métiers a connu ces dernières années une standardisation à la

fois de la modélisation des procédés (BPMN) mais aussi de leur exécution avec l'avènement du langage BPEL.

Il est clair pour nous que l'objectif de la mise en oeuvre des sphères de comportement, au niveau de l'exécution des procédés, ne se situe pas dans la révolution des procédures et des routines mais bien au contraire dans l'utilisation des possibilités existantes et des extensions possibles. Nous nous proposerons dans ce sens d'utiliser les standards actuels, en l'occurrence le langage BPEL, pour exprimer les sphères de comportement au niveau de l'exécution des procédés métiers.

L'exécution des activités d'un procédé en présence de sphères d'isolation doit également être régi par un protocole d'exécution dont le déroulement dépendra des niveaux de cohésion et de cohérence requis. Cela concerne bien évidemment les activités à l'intérieur d'une sphère mais également celles en dehors puisque la cohésion régie l'exécution à l'intérieur d'une sphère alors que la cohérence régit cette même exécution par rapport à celle des activités extérieures à la sphère.

Nous nous proposons dans ce chapitre de clarifier tout d'abord l'environnement de mise en oeuvre choisi et d'en expliquer l'intérêt et les enjeux. Nous procéderons ensuite à une présentation de toutes les étapes, allant de la modélisation de sphères d'isolation dans des procédés métiers à l'exécution de ces procédés en concordance avec un protocole d'exécution permettant le respect des niveaux de cohésion et de cohérence requis par la sphère d'isolation que nous définirons.

2.2 Choix de l'environnement de mise en oeuvre

Dans un système de gestion de procédés, et plus généralement dans un environnement d'information distribué EAI, l'orchestration assure la succession des tâches, le contrôle de la bonne exécution, les reprises en cas d'incident ... Plusieurs camps s'affrontent dans la production de standards d'orchestration :

- WSCI (Web Services Choregraphy Interface) + BMPL (Business Process Modeling Language) de Sun microsystem, allié à BEA et SAP, membres du BPMI
- BPEL4WS (Business Process Execution Language for Web Services) d'IBM et Microsoft, membres du WS-I, qui fédère WSFL et WLang. BPEL4WS en sa version 2.0 est dorénavant proposé par le groupe OASIS.
- XPDL (XML Processing Description Language) du WfMC est plus destiné aux processus métiers.

Ces trois camps dévoilent en réalité deux types d'environnements d'exécution de procédés métiers. Le premier type d'environnement correspond à un environnement intégré de modélisation et d'exécution de procédés métiers. Dans un tel environnement, il s'agit généralement de modèles intégrés formant un langage unique de définition de procédés métiers qui est interprété lors de l'exécution du procédé.

Les systèmes de workflow offrent un tel environnement intégré et sont généralement basés sur des standards du WfMC ou des normes propriétaires. Un procédé métier est construit à l'aide d'éléments graphiques tels que des activités ou des opérateurs logiques et des flux de contrôle et de données. XPDL est un langage utilisé par beaucoup de systèmes de workflow. Il s'agit d'une représentation en format XML de ce que représente le "schéma" de la définition du procédé.

Le but de XPDL est de stocker et d'échanger le diagramme représentant un procédé. Il permet à un outil de conception de procédés d'écrire le diagramme, et à d'autres de le lire, de façon parfaitement interopérable et standardisée. Dès lors, la conception d'un procédé sera indépendante de la plateforme d'exécution de procédés à utiliser, sous réserve qu'elle accepte l'importation de fichiers XPDL. Cependant, XPDL permet uniquement de décrire la structure d'un procédé mais ne va pas jusqu'au niveau de l'exécution des procédés. Par conséquent, il ne garantit pas une sémantique précise de l'exécution d'un procédé. Cette sémantique de l'exécution demeure le propre de chaque système de gestion de workflow.

Après le survol de ce premier type d'environnement, celui des environnements intégrés de modélisation et d'exécution de procédés métiers, nous avons pu clairement discerner que la mise en oeuvre des sphères d'isolation n'est pas adaptée à un tel environnement. D'un côté, l'un des principes des sphères de comportement est la séparation des préoccupations, ce qui est contraire à un environnement intégré se

basant sur un modèle unique. D'un autre côté, la mise en oeuvre des sphères d'isolation suppose une spécification claire des aspects opérationnels de l'exécution indépendante de l'implémentation du système de gestion de procédés. Or les environnements à modèles intégrés, tels que les workflow ont une sémantique d'exécution qui dépend de l'implémentation du système de gestion de procédés.

Le deuxième type d'environnement correspond à celui d'une plateforme (*Framework*) regroupant une collection de spécifications, chacune résolvant une partie des problèmes autour des procédés métiers, de la modélisation à l'exécution. Il s'agit de spécifications définissant de façon plus ou moins indépendante les unes des autres, les aspects modélisation à travers une notation graphique, les aspects exécution à travers un langage d'exécution, et d'autres aspects tels que transactionnels, de coordination ou d'échanges de messages.

L'exemple le plus captivant en ce qui concerne la gestion de procédés métiers est certainement celui des plateformes de web services. En effet, depuis l'avènement de la spécification BPEL4WS (Business Process Execution Language for Web Services), nous avons assisté à un engouement des chercheurs et praticiens autour de la question d'une plateforme de modélisation et d'exécution de procédés métiers à base de web services. Cet engouement à permis à plusieurs spécifications de voir le jour, chacune d'elles ayant un objectif clair.

Nous pensons qu'un tel environnement est favorable à la mise en oeuvre des sphères de comportement puisqu'il prône la séparation des préoccupations. C'est ainsi que notre choix s'est porté sur la plateforme de web services en termes d'environnement de mise en oeuvre des sphères de comportement et particulièrement des sphères d'isolation. Notre choix n'a pas été simplement celui de la technologie à la mode ou de la tendance actuelle, même si l'engouement autour des web services a certainement joué. Bien au delà, le choix des technologies des web services a été motivé par la richesse, jusque là inégalée, des spécifications et des standards développés autour des web services et de leur séparation claire, tout en gardant leur interactivité.

Pour justifier notre choix, il nous suffit d'énumérer les spécifications diverses autour de la notation (BPMN), de l'exécution (BPEL), de la sécurité (WS-Security), de l'échange de messages (WS-Messaging), de la coordination (WS-Coordination), de l'exécution transactionnelle (WS-Transaction), de la chorégraphie des services (WS-Choreography) et de bien d'autres spécifications. Ces dernières ne peuvent que rendre la plateforme d'exécution de procédés à base de web services plus intéréssante et plus propice au développement de nouvelles spécifications.

Un autre point important qui nous a interpelé dans le choix des technologies des web services comme environnement de mise en oeuvre est celui de l'existence d'un travail important dans la spécification d'un environnement d'exécution transactionnel pour les web services. En effet, la spécification WS-Transaction, elle-même basée sur la spécification WS-Coordination, se propose comme une solution de gestion de la coordination transactionnelle de services web.

Ainsi, nous allons, dans la suite de ce chapitre, porter notre attention sur l'analyse des spécifications WS-Coordination afin d'identifier la meilleure façon de mettre en oeuvre les sphères de comportement, et pluparticulièrement, nous allons analyser la spécification WS-Transaction afin d'identifier la meilleure façon de mettre en oeuvre les sphères d'isolation. Il est toutefois nécessaire de présenter la plateforme des web services et les WS-Technologies existantes.

2.3 Les technologies web services "WS-Technologies"

Beaucoup d'entreprises migrent leurs applications vers une architecture orientée services et les technologies des web services représentent une manière adaptée pour garantir une interopérabilité entre les différentes plateformes. Dans un environnement de services distribués et composés, des spécifications telles que BPEL4WS [WS BPEL] ont vu le jour afin de construire des procédés métiers dans les environnements de web services.

La gestion de procédés métiers sur les plateformes orientées web services a pris forme progressivement

principalement durant les cinq dernières années avec une prolifération de spécifications (XLANG, BPML, WSFL, BPSS, WSCL, WSCI, WS-Choreography et BPEL4WS). Ces spécifications ont été définies autour de nombreux enjeux de la gestion des procédés métiers. Nous citons comme exemples la coordination et la communication de plusieurs services, la corrélation des échanges entre plusieurs parties, l'implémentation d'exécution parallèles d'activités, la transformation de données entre partenaires, le support des transactions longues ou encore la mise en place d'un mécanisme de gestion d'exceptions.

Notre recherche se focalise sur l'orchestration/chorégraphie des Web Services mais quelques clarifications sont nécessaires au sujet de la signification des termes orchestration et chorégraphie. D'un côté, l'orchestration de Web Services comme l'organisation de l'exécution de procédés métiers sous le contrôle d'un seul site central (à l'intérieur d'une même organisation). De l'autre côté, la chorégraphie représente l'échange observable de messages, règles et contrats entre plusieurs sites de gestion de procédés métiers (inter-organisations). Contrairement à l'orchestration, la chorégraphie ne se penche pas sur la gestion de la concurrence d'accès aux ressources partagées et cela est du au fait que la manipulation concurrentielle de ressources transactionnelles est généralement limitée à un même site et donc uniquement au niveau de l'orchestration.

Comme illustré dans la figure 58, BPEL4WS et CDL4WS (Choreography Definition Language for Web Services) représentent tous deux la gestion de procédés métiers d'un côté à l'intérieur d'une organisation et de l'autre côté entre les organisations. D'autres spécifications comme WS-Reliability, WS-Security, WS-Coordination and WS-Transactions permettent d'assurer une certaine qualité de service de l'environnement de procédés métiers à base de web services.

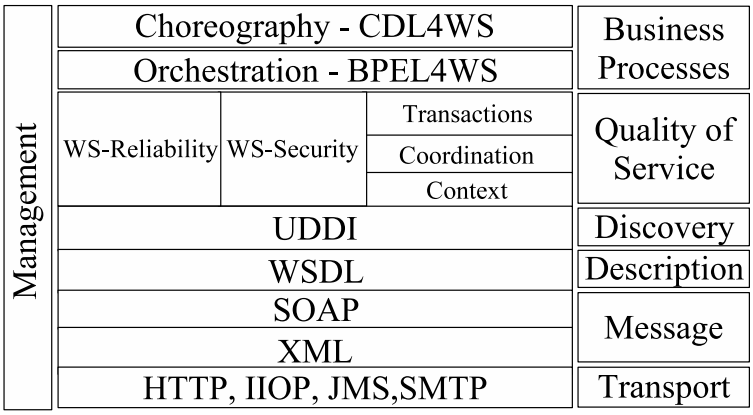


FIG. 58 – Standards utilisés avec BPEL

L'orchestration de web services a permis d'introduire des concepts de workflow, ce qui a révélé des besoins transactionnels classiques tels que l'atomicité et l'isolation afin d'assurer une exécution correcte. À partir des travaux sur la spécification WS-Coordination [WS COORD], qui se propose d'être une spécification du contexte de coordination de web services et en tenant compte des contraintes transactionnelles, la spécification WS-Transactions [WS TRANS] a été fournie dans les plateformes de web services afin de gérer de telles propriétés transactionnelles et d'y introduire une certaine flexibilité en relâchant les traditionnels verrous sur les ressources transactionnelles avant la terminaison des activités transactionnelles. Cela permet à d'autres activités d'accéder de façon concurrente aux mêmes ressources. C'est pour cette raison que nous nous penchons sur les spécifications WS-Coordination et WS-Transaction car l'objectif de notre travail est d'exprimer des contraintes d'isolation flexibles dans un environnement de procédés métiers. Nous allons détailler par la suite les travaux actuels portant sur les spécifications WS-Coordination et WS-Transaction.

2.3.1 WS-Coordination / WS-Transactions

La spécification WS-Coordination fournit un mécanisme standard pour la création de coordinateurs auxquels s'enregistrent des services utilisant des protocoles de coordination divers pour l'exécution d'opérations distribuées dans un environnement de web services.

La spécification WS-Coordination définit une plateforme de coordination prenant en charge les services suivants :

- Service d'activation pour la création d'une nouvelle activité de coordination et spécifier les protocoles de coordination disponibles. Généralement c'est l'initiateur de la coordination qui appelle ce service pour, justement, initier une coordination et définir ses propriétés.
- Service d'enregistrement pour l'enregistrement des participants et pour la sélection du protocole de coordination de chaque participant.
- Service de coordination pour la gestion de la terminaison de l'activité de coordination en recourant au protocole de coordination sélectionné.

La spécification WS-Coordination permet de fournir des types et des protocoles de coordination personnalisables et adaptés à la situation désirée. La spécification WS-Transaction représente la concrétisation de deux types particuliers de WS-Coordination que sont la WS-AtomicTransaction et la WS-BusinessActivity qui se présentent comme suit :

- WS-AtomicTransaction représente un type de coordination adapté aux activités exprimant le comportement d'atomicité de type "tout ou rien". Deux protocoles sont possibles :
  - Completion protocol : généralement, l'initiateur d'une coordination utilise ce protocole lorsqu'il est question de validation ou de recouvrement de l'ensemble des services participant à la coordination.
  - Two phase commit (2PC) - Validation à deux phases : un participant s'enregistre en optant pour ce protocole afin d'initier une validation à deux phases avec les participants ayant choisi ce même protocole. Deux sous-types de protocoles sont disponibles : "Volatile 2PC" (utilisé en cas de ressources volatiles tels que la mémoire cache) et "Durable 2PC" (utilisé en cas de ressources durables telles que les bases de données).
- WS-BusinessActivity permet de gérer les activités à longue durée d'exécution en permettant une validation partielle des participants à une coordination. Deux types de coordination sont mis en évidence
  - Type de coordination "AtomicOutcome" permet d'aboutir à ce que tous les participants terminent leur exécution ou que tous les participants soient compensés.
  - Type de coordination "MixedOutcome" permet d'aboutir à ce que certains participants terminent leur exécution alors que d'autres soient compensés.

  Pour les deux types de coordination, les participants enregistrés peuvent choisir entre deux protocoles possibles avec WS-BusinessActivity. Ces deux protocoles introduisent la notion de la décision de terminaison comme suit :
  - BusinessAgreementWithParticipantCompletion : Quand un participant s'enregistre à une WS-BusinessActivity en adoptant ce protocole, il devra notifier sa terminaison au coordinateur. Ainsi, la terminaison de la coordination n'est pas dictée par le coordinateur mais dépend de la terminaison de chacun des participants.
  - BusinessAgreementWithCoordinatorCompletion : Quand un participant s'enregistre à une WS-BusinessActivity en adoptant ce protocole, il devra attendre que le coordinateur lui notifie quand

il doit se terminer. Ainsi, la terminaison de la coordination est unilatérale pour un tel protocole et ne dépend pas de la terminaison de chacun des participants.

2.4 Synthèse

Nous avons montré la pertinence de choisir un environnemet d'exécution de procédés web services comme environnement de mise en oeuvre des sphères de comportement en général et des sphères d'isolation en particulier. Nous avons essayé d'apporter une vision des environnements éventuels pour l'implémentation de nos travaux tout en nous référant aux enjeux de la mise en oeuvre elle même. Nous avons montré comment l'architecture des plateformes d'exécution de web services est propice aux principes des sphères de comportement, notamment en ce qui concerne la séparation des préoccupations.

L'avantage des technologies web services (WS-Technologies) est que chaque spécification est dédiée à un aspect particulier de l'exécution des web services. Nous avons porté notre attention particulièrement sur les spécifications WS-Coordination et WS-Transaction. La spécification WS-Coordination apporte des fondements propices à la spécification de sphères de comportement.

Nous étudierons dans la section suivante, comment les sphères de comportement peuvent être intégrées à l'architecture des plateformes d'exécution de web services, et particulièrement à la spécification WS-Coordination. Il s'agira d'une nouvelle organisation des types de WS-Coordination permettant non seulement de prendre en charge les sphères d'isolation mais potentiellement tout type de sphère de comportement.

3

Sphères de comportement pour la coordination de services web

Nous considérons la plateforme de web services comme l'une des plateformes les plus intéressantes pour l'implémentation des sphères d'isolation. La spécification WS-Coordination est apte à offrir un contexte adaptable permettant de l'accommoder à l'approche des sphères d'isolation et des sphères de comportement en général. En effet, WS-Coordination permet de regrouper plusieurs web services qui peuvent rejoindre ou quitter un service de coordination. Cela est assez proche de ce que nous envisageons pour les sphères d'isolation : regrouper des services ensemble à travers un service de coordination et assurer des contraintes d'isolation flexibles sur leur exécution.

Les sphères d'isolation concernent les propriétés transactionnelles d'un groupe d'activités (ou de services) mais il est préférable de séparer l'atomicité de l'isolation. Nous proposons de définir la spécification de WS-Sphere parmi les types de WS-Coordination au même titre que l'actuelle spécification WS-Transaction. Nous incluerons dans la spécification WS-Sphere des sous-types, chacun correspondant à un type de sphères de comportement (Sphères d'atomicité, Sphères d'isolation, Sphères de compensation, Sphères d'instanciation multiple ...) et nous nous proposons donc de réorganiser les types existants de WS-Coordination.

Les sphères de comportement se proposent de séparer la définition du procédé de celles des propriétés comportementales. Nous avons identifié un certain nombre de propriétés qui répondent aux principes des sphères de comportement. Dans ce travail de mise en oeuvre, nous allons nous focaliser sur les sphères d'isolation, mais toute autre sphère sera implémentable en utilisant une même approche.

Nous proposons une famille de sphères non exhaustive que nous intégrons dans les sous-types de WS-Sphere. Nous donnons lieu alors aux spécifications possibles WS-AtomicitySphere, WS-IsolationSphere, WS-JointCompensationSphere et WS-MultipleInstantiationSphere. Ces quatre types correspondent à une sélection de quelques exemples de sphères.

3.1 Proposition d'une nouvelle organisation des types de WS-Coordination

Après avoir étudié l'organisation actuelle des types de WS-Coordination, nous constatons que ces derniers sont au nombre de trois que sont "WS-AtomicTransaction", "WS-BusinessActivity AtomicOutcome" et "WS-BusinessActivity MixedcOutcome". Ces trois types de coordination sont regroupés sous l'appellation WS-Transaction mais nous pensons qu'ils se rejoignent par rapport à un objectif d'atomicité. C'est pour cette raison que nous pensons que l'actuelle spécification WS-Transaction correspondrait simplement à la spécification d'une sphère d'atomicité.

La famille existante de WS-Transaction peut être considérée comme faisant partie du type "Atomicity Sphere". La figure 59 illustre l'organisation globale de l'intégration de la spécification WS-Sphere dans celle de WS-Coordination.

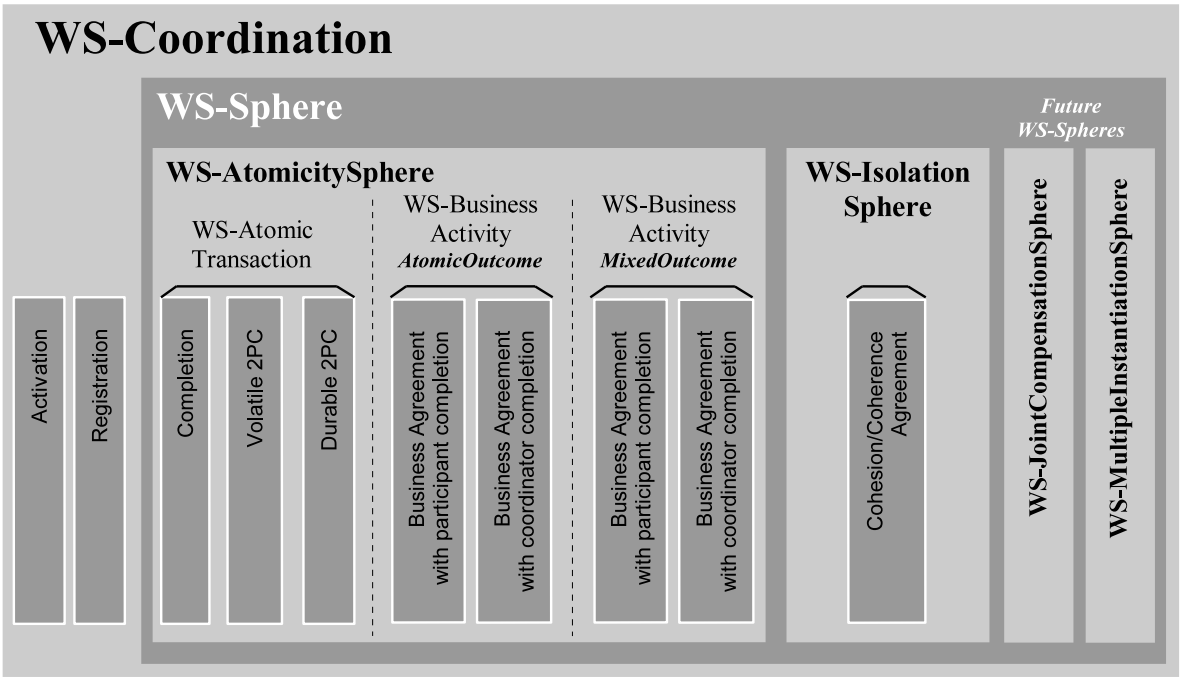


FIG. 59 – Intégration de la spécification WS-Sphere dans la spécification WS-Coordination

Notre contribution principale dans cette implémentation est de réorganiser la spécification WS-Transaction en proposant une solution plus globale autorisant la séparation entre les propriétés transactionnelles, l'atomicité d'un côté et l'isolation de l'autre.

En proposant une spécification globale WS-Sphère et des sous-types WS-AtomicitySphere, WS-IsolationSphere, ainsi de suite, nous pourrons exprimer beaucoup plus de comportements, bien au delà des aspects transactionnels.

WS-Sphere représente un nouveau type général de WS-Coordination englobant des sous-types qui sont WS-AtomicitySphere, WS-IsolationSphere, WS-JointCompensationSphere, WS-MultipleInstantitationSphere, etc comme présenté dans la figure 60

Le type WS-IsolationSphere représente un sous-type de coordination focalisé sur la gestion de l'isolation. WS-IsolationSphere hérite des propriétés de la spécification WS-Coordination (Activation et Enregistrement). Une sphère d'isolation est alors initié à travers le choix d'un niveau de cohésion et de cohérence à travers le protocole "Cohesion/Coherence Agreement" quand un participant s'enregistre à une WS-IsolationSphere. Les niveaux fixés permettent d'assurer un comportement adapté aux besoins et c'est la plateforme de coordination de web services qui devra le réaliser. Durant l'enregistrement d'un participant à une WS-IsolationSphere, le participant et le coordinateur échangent un contexte de coordination qui permet d'implémenter des éléments d'extensibilité. Nous proposons d'utiliser ces éléments d'extensibilité de la spécification WS-Coordination et particulièrement ceux du contexte de coordination qui permettra de communiquer les différents niveaux de cohésion et de cohérence du participant vers le service d'enregistrement à un coordinateur de type WS-IsolationSphere et ce en utilisant un contexte de coordination similaire au suivant.

```
<?xml version="1.0" encoding="utf-8"?>
<S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope"
    <S:Header>
        . . .
        <wscoor:CoordinationContext
            xmlns:wsme="http://www.w3.org/2002/06/msgext"
            xmlns:wscoor="http://www.w3.org/2002/06/Coordination"
            xmlns:myApp="http://www.w3.org/2002/06/myApp">
            <wsme:Identifier>http://www.loria.fr/Coor/123</wsme:Identifier>
            <wsme:Expires>2006-10-22T14:30:00.000-05:00</wsme:Expires>
            <wscoor:CoordinationType>
                http://xml-soap.org/2006/03/IsolationSphere
            </wscoor:CoordinationType>
            <wscoor:RegistrationService>
                <Address>
                    http://myservice.com/mycoordinationservice/registration
                </Address>
                <myApp:BetaMark> ... </myApp:BetaMark>
                <myApp:EBDCode> ... </myApp:EBDCode>
            </wscoor:RegistrationService>
            <myApp:CohesionLevel>
                  RepeatableRead
            </myApp:CohesionLevel>
            <myApp:CoherenceLevel>
                  ActivityCohrence
            </myApp:CoherenceLevel>
        </wscoor:CoordinationContext>
        . . .
    </S:Header>
```

FIG. 60 – Utilisation de l'extensibilité de la spécification WS-Coordination pour ajouter les propriétés des sphères d'isolation

La capacité de gérer les niveaux de cohésion et de cohérence dépend des capacités du système d'information utilisé et de l'interaction entre celui-ci et la plateforme de web services. Nous proposons une solution basée sur un middleware entre le moteur d'exécution de procédés BPEL et le système de base de données utilisé. Il est important de noter que la gestion de l'isolation est restreinte à un même point de référence (endpoint), c'est-à-dire à une même base de données ou à une même entreprise. Généralement des points de référence différents n'utilisent pas la même base de données et ne sont pas localisés sur un même moteur d'exécution de web services ce qui limite les possibilités de gestion commune des accès aux données.

3.2 Sphères emboîtées et services d'inscription / désincription du WS-Coordination

Contrairement au WS-Transaction, le service d'inscription d'une WS-IsolationSphere est capable de décider de rediriger l'inscription à une autre WS-IsolationSphere à partir du moment où un initiateur initialise un coordinateur de type WS-IsolationSphere, ce qui permet de déterminer les niveaux de cohésion et de cohérence initialement requis pas l'initiateur. Ensuite, des services participants peuvent s'inscrire mais ont également la possibilité de définir les niveaux de cohésion et de cohérence qu'ils désirent avoir. Il peut exister des différences entre les niveaux requis par l'initiateur et ceux requis par un participant. Cela est un phénomène naturel car les besoins de chacun ne peuvent pas être les mêmes. Nous proposons une

procédure exécutée à chaque inscription d'un participant à une coordination de type sphère d'isolation comme suit :

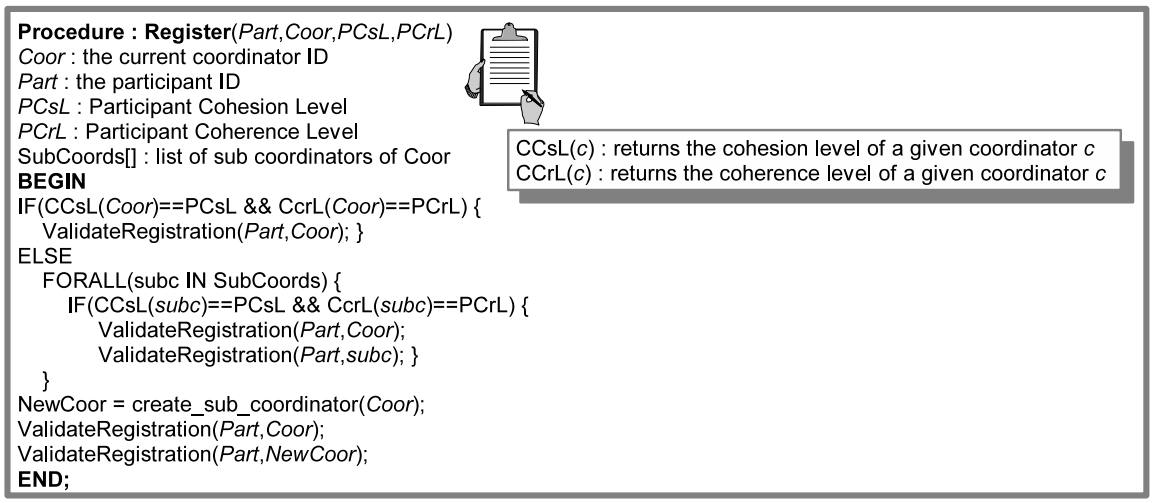


FIG. 61 – Algorithme pour l'enregistrement à une WS-Sphere

En utilisant la procédure d'inscription que nous avons définie, des sphères emboîtées sont alors possibles. L'implémentation d'une telle procédure permet d'augmenter les possibilités d'isolation flexible : les participants ne sont pas obligés d'accepter les niveaux définis par l'initiateur de la coordination mais peuvent proposer des niveaux plus hauts qui coexisteront avec les niveaux de l'initiateur à travers l'emboîtement de plusieurs sphères.

Une sphère peut alors contenir plusieurs sous-sphères qui ont différents niveaux d'isolation requis. Une sous-sphère définit ses propres niveaux de cohésion et de cohérence et bénéficie des niveaux des sphères mères.

Après avoir introduit une procédure d'inscription avancée, prenant en charge l'emboîtement de sphères, il est nécessaire de définir une procédure de désinscription qui devra gérer les sous-sphères créées. Nous présentons une telle procédure de désinscription comme suit :

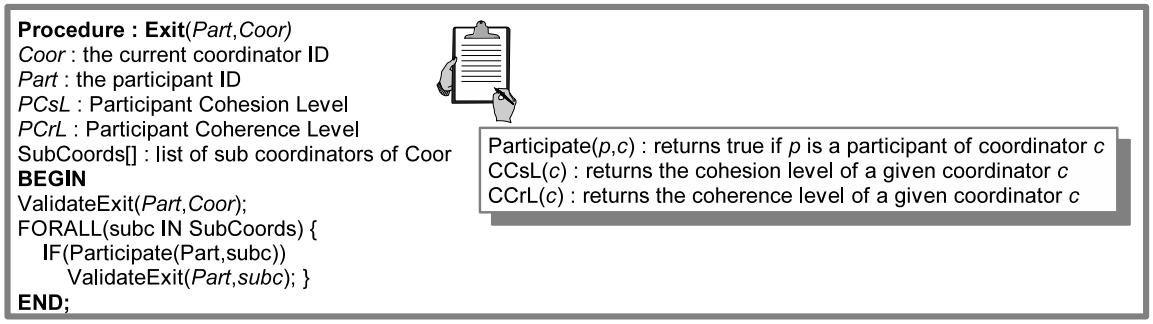


FIG. 62 – Algorithme pour la désinscription d'une WS-Sphere

4

Mise en oeuvre des sphères d'isolation dans une plateforme de web services

4.1 La démarche de mise en oeuvre

Afin de mettre en oeuvre les sphères de comportement et particulièrement les sphères d'isolation, nous venons de présenter comment de telles sphères peuvent être exprimées dans le langage d'exécution de procédés métiers à base de web services BPEL (Business Process Execution Language). Il est judicieux de se repositionner dans une vue d'ensemble qui consiste à considérer la mise en oeuvre des sphères d'isolation comme une mise en oeuvre de bout en bout. En effet, les sphères sont avant tout une notation, graphiquement parlant, permettant de distinguer un sous-ensemble d'activités d'un procédé en lui attribuant des propriétés particulières. Nous devons donc proposer un outil de modélisation graphique d'un procédé prenant en charge les sphères de comportement.

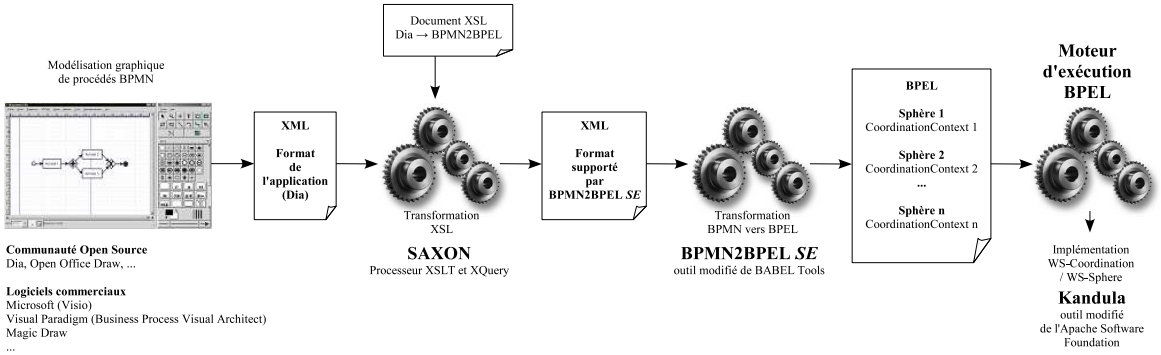


FIG. 63 – Architecture de mise en oeuvre du passage de BPMN à BPEL

Il est toutefois nécessaire de garder à l'esprit le fait que la plateforme de web services est fortement nourrie par le principe de séparation des préoccupations. La notation utilisée pour la modélisation de procédés métiers dans les plateformes de web services n'échappe pas à ce principe. En effet, la spécification BPMN a permis de définir clairement la façon dont les procédés métiers devront être modélisés graphiquement.

Nous nous proposons donc d'intégrer la notion de sphères dans la notation BPMN. En réalité, aucun changement ne va être opéré dans la spécification BPMN elle-même, il s'agit simplement de rajouter, dans

l'outil de modélisation graphique, la possibilité de définir graphiquement des sphères de comportement. Les sphères ainsi définies pourront être prises en compte lors de l'exportation du procédé en BPEL afin de générer les contexts de coordination associés à chacune des sphères d'un procédé.

L'architecture de mise en oeuvre tourne autour du passage de la notation BPMN avec prise en charge des sphères d'isolation vers un code BPEL et des contexts de coordination compatibles avec la spécification WS-Spheres que nous avons introduite précédemment. La figure 63 illustre bien les différentes composantes de cette architecture et l'enchaînement des étapes de transformation de BPMN à BPEL et aux contexte de coordination des sphères.

La mise en oeuvre des sphères d'isolation se doit d'être "de bout en bout" c'est-à-dire partir de la notation graphique et aboutir à un code BPEL exécutable par un moteur BPEL utilisant des contextes de coordination appropriés aux sphères d'isolation. Ainsi nous nous proposons de partir d'une modélisation graphique du procédé métier en utilisant l'un des outils de modélisation existants sur le marché. Nous choisissons le logiciel Dia d'une part parce qu'il est gratuit et libre et d'autre part parce qu'il est facilement extensible.

Dans la suite, nous verrons, étape après étape, comment le procédé BPMN initial pourra prendre en considération la notion de sphère d'isolation et comment il est possible de transformer un tel procédé afin d'aboutir à un code BPEL et aux contexts de coordination appropriés prenant en considération les sphères.

4.2 Le volet notation et spécification

Au lieu de développer un nouvel outil de modélisation de procédé, qui viendrait accroitre le nombre d'éditeurs existants, nous avons préféré choisir un outil de modélisation existant et de l'étendre pour la prise en charge des sphères. Notre choix a porté sur le logiciel libre Dia que nous allons présenter ici.

Dia est un logiciel de création de diagrammes développé en tant que partie du projet GNOME. Il a été originellement conçu par Alexander Larsson. Ce logiciel est conçu pour servir des buts similaires au programme Visio de Microsoft. L'avantage du logiciel Dia réside dans le fait qu'il est conçu de manière modulaire avec plusieurs paquetages de formes pour des besoins différents : diagramme de flux, diagramme de circuit électrique, diagramme UML, etc. L'ajout d'un paquetage se fait par l'écriture de fichiers XML, en utilisant un sous-ensemble du SVG (Scalable Vector Graphics) pour dessiner les formes. Cela simplifie énormément la réalisation d'un paquetage pour la notation BPMN, ce qui a été réalisé par la communauté d'utilisateurs de Dia.

Notre travail d'intégration des sphères d'isolation dans le paquetage des formes propres à BPMN a consisté à rajouter la définition graphique de ce qu'est une sphère d'isolation. Cela a été réalisé à l'aide de fichiers XML utilisant un sous-ensemble du SVG (Scalable Vector Graphics) pour dessiner les formes. SVG est une spécification du W3C permettant de décrire des graphiques vectoriels.

Nous représentons une sphère à l'aide d'un rectangle en ligne discontinue. La définition de la notation graphique d'une sphère d'isolation est alors exprimée à l'aide de la spécification SVG et le bouton de la forme correspondante est défini en utilisant une image PNG (miniature) comme illustré dans la figure 65.

Nous avons opté pour une forme rectangulaire avec une ligne discontinue. Ce choix se justifie par le besoin de calculer quelles sont les activités qui font partie de la sphère d'isolation. En effet, la forme rectangulaire permet une délimitation claire de la zone de couverture de la sphère et facilite le calcul de la position des activités par rapport à la sphère (dedans ou en dehors). Il est clair que si la sphère était dessinée à main libre, cela engendrerait des calculs additionnels sur la délimitation de la zone de couverture d'une sphère.

Une forme plus simple, rectangulaire, est certes plus contraignante visuellement, mais elle procure une plus grande facilité lors de l'identification des activités de la sphère. Il suffit alors de connaître la position de la sphère, sa largeur et sa hauteur, de connaître également la position, largeur et hauteur de chaque activité pour savoir si une activité fait partie ou non de la sphère.

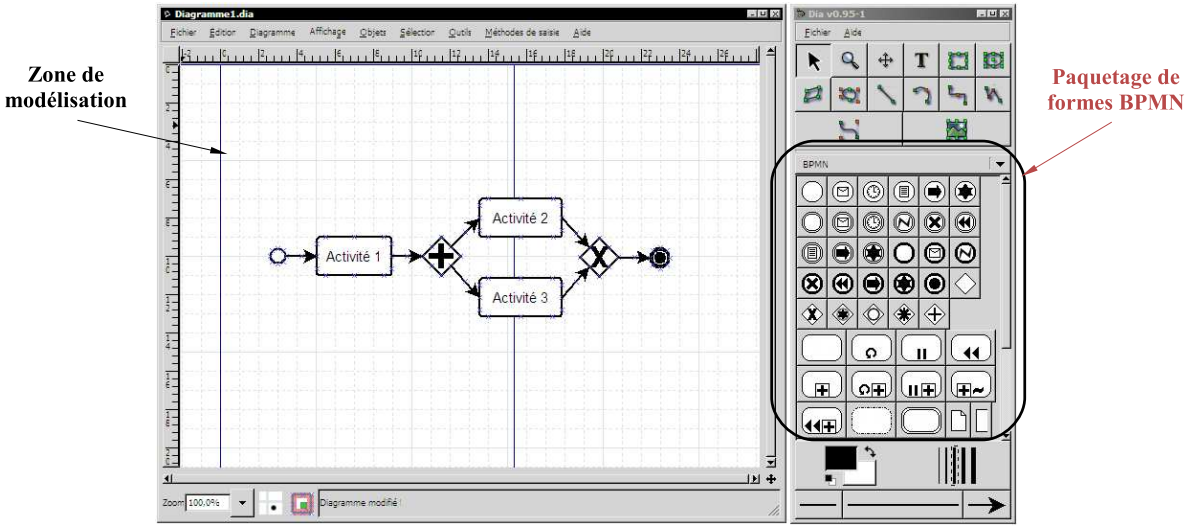


FIG. 64 – Les paquetages de formes dans Dia : le cas de BPMN

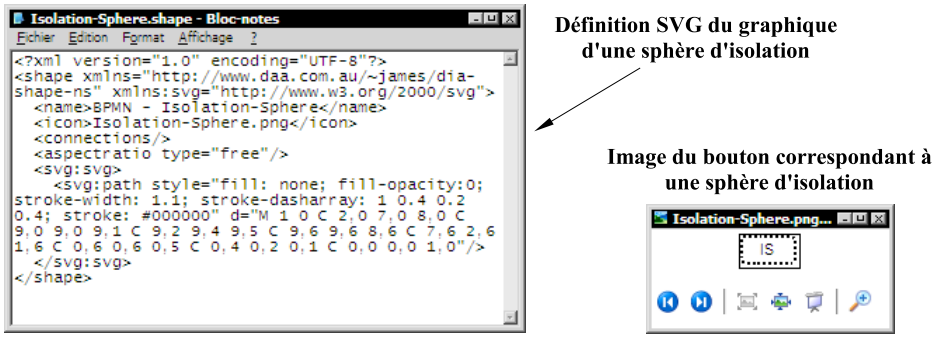


FIG. 65 – Création d'une nouvelle forme pour les sphères d'isolation dans le paquetage de formes BPMN

La modélisation des sphères d'isolation est d'une grande simplicité. En effet, il suffit de créer un procédé métier, ensuite, il s'agit d'insérer une sphère d'isolation parmi les activités du procédé puis de redimensionner la sphère et de déplacer les activités afin que la sphère englobe les activités désirées. Le résultat d'une telle manipulation est illustré par l'exemple de la figure 66

Le logiciel Dia enregistre un diagramme en utilisant le format XML et en adoptant une structure XML particulière permettant de lister les différents éléments graphiques et leur inter-relation comme illustré dans la figure 67.

4.3 Vers le volet exécution

Après avoir intégré la notion de sphère d'isolation dans le logiciel Dia et permis ainsi de modéliser des sphères d'isolation dans les procédés métiers en notation BPMN, il est nécessaire de passer, selon l'enchaînement logique des environnements web services, à la partie exécution, c'est-à-dire au langage BPEL. BPMN a été élaboré avec le souci permanent d'être entièrement compatible avec la spécification

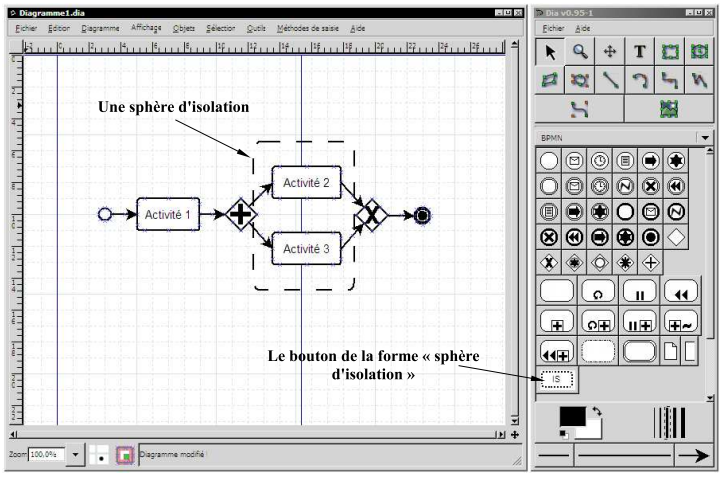


FIG. 66 – Exemple de procédé contenant une sphère d'isolation créée à l'aide de l'outil Dia et du paquetage de formes modifié

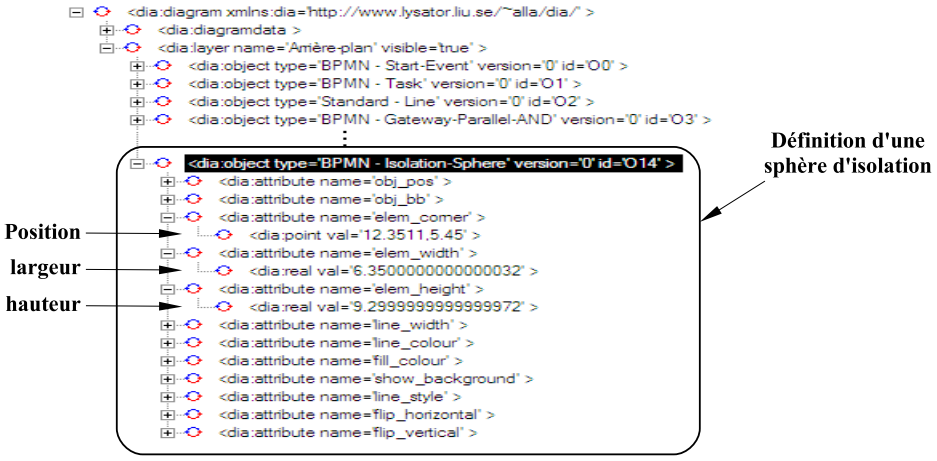


FIG. 67 – Format d'enregistrement des fichiers Dia

BPEL. Ainsi, plusieurs outils ont été développés afin d'exporter du BPEL à partir d'une notation BPMN. D'ailleurs, un outil de modélisation BPMN n'a aucun intérêt s'il ne permet pas de générer du BPEL car, sans une telle fonctionnalité, le procédé métier modélisé ne pourra pas passer à la phase d'exécution.

Après avoir fait une recherche sur les outils existants permettant la génération de code BPEL à partir d'une modélisation BPMN, beaucoup semblent être liés à l'outil de modélisation utilisé. Tout de même, nous avons pu trouver des outils indépendants de l'outil de modélisation et nous citons particulièrement la collection d'outils nommés "BABEL - Tools" développés au sein de la Queensland University of Technology, Brisbane, Australie, et qui propose un outil libre nommé BPMN2BPEL sous licence Apache 2.0. La collection d'outils "BABEL - Tools" [1] propose les outils suivants :

- BPMN2BPEL : traduit un procédé métier décrit en BPMN vers le code BPEL équivalant.
- SPM2BPEL : traduit un procédé métier décrit en SPM (Standard Process Models) vers le code

[1] http ://www.bpm.fit.qut.edu.au/projects/babel/tools/

BPEL équivalant.
- BPMN-to-Petri net transformer : traduit un procédé métier décrit en BPMN vers un petri-net équivalant écrit en PNML (Petri Net Markup Language).
- BPEL2PNML : traduit un code BPEL en un petri-net équivalant écrit en PNML.
- WofBPEL : un outil d'analyse formelle de procédé BPEL.

Nous nous intéressons plus particulièrement à l'outil BPMN2BPEL et nous nous proposons d'apporter des modifications à son code source afin de prendre en charge les sphères d'isolation en particulier et les sphères de comportement en général. Nos modifications donneront lieu à un nouvel outil que nous avons décidé d'appeler BPMN2BPEL SE (SE pour Sphere Edition) qui pourra, à partir d'un procédé BPMN créé par un outil tel que DIA prenant en charge les sphères d'isolation, générer automatiquement un code BPEL équivalant avec, pour chaque sphère d'isolation, le contexte WS-Coordination associé.

L'outil BPMN2BPEL prend en entrée un format XML défini par les auteurs de l'outil. Un exemple de ce format est illustré dans la figure 68

FIG. 68 – Format XML représentant les procédés BPMN accepté par l'outil BPMN2BPEL

Il est donc clair que nous devrons générer un tel format à partir du format XML d'enregistrement utilisé par le logiciel Dia. Partant du fait que nous voulons transformer un document XML en un autre mais en utilisant un format (balisage) différent, nous avons naturellement choisi d'utiliser une transformation sur la base du langage de transformation XSLT (eXtensible Stylesheet Language Transformation). Il s'agit d'un document XSL (eXtensible Stylesheet Language) décrivant la façon dont un document XML source donnera lieu à un autre document XML ayant un format différent.

En utilisant le langage XSLT, nous avons développé le moyen de transformer les fichiers *.dia en fichiers XML conformes au format requis par l'outil BPMN2BPEL. Il s'agit d'un fichier XSL ayant la structure résumée dans la figure 69

Ainsi, le document XSL contient, pour chaque élément de base de la notation BPMN, le code nécessaire à son introduction comme nœud ou arc dans le document XML résultant. Par exemple, le code nécessaire à l'insertion d'un nœud correspondant à une activité (Task) est illustré dans la figure 70 et celui nécessaire à l'insertion d'un nœud correspondant à une sphère d'isolation est illustré dans la figure 71.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:dia="http://www.lysator.liu.se/~alla/dia/"
  version="2.0">
  <xsl:output method="xml" indent="yes" encoding="utf-8"/>

  <xsl:template match="/">
    <xsl:element name="bpmn">
      <xsl:element name="process">
        <xsl:attribute name="id">MyProcess</xsl:attribute>
      <xsl:element name="code">
      </xsl:element>
      <xsl:element name="nodes">
         <xsl:apply-templates select="//dia:object[@type='BPMN - Task']"/>
         <xsl:apply-templates select="//dia:object[@type='BPMN - Start-Event']"/>
         <xsl:apply-templates select="//dia:object[@type='BPMN - End-Event-Terminate']"/>
         <xsl:apply-templates select="//dia:object[@type='BPMN - Activity-Looping']"/>


         <xsl:apply-templates select="//dia:object[@type='BPMN - Text-Annotation']"/>
         <xsl:apply-templates select="//dia:object[@type='BPMN - Transaction']"/>
         <xsl:apply-templates select="//dia:object[@type='BPMN - Isolation-Sphere']"/>
      </xsl:element>
      <xsl:element name="arcs">
         <xsl:apply-templates select="//dia:object[@type='Standard - Line']"/>
      </xsl:element>
    </xsl:element>
      </xsl:element>
  </xsl:template>
  ...
```

Prise en compte des sphères d'isolation

FIG. 69 – Structure résumée du fichier XSL permettant de transformer les fichiers Dia en fichiers au format XML requis par BPMN2BPEL

Activité (Task)

```
<xsl:template match="dia:object[@type='BPMN - Start-Event']">
  <xsl:element name="node">
     <xsl:attribute name="id">
        <xsl:value-of select="@id"/>
     </xsl:attribute>
    <xsl:attribute name="name">
      <xsl:text>start</xsl:text>
    </xsl:attribute>
    <xsl:attribute name="type">
      <xsl:text>StartEvent</xsl:text>
    </xsl:attribute>
  </xsl:element>
</xsl:template>
```

FIG. 70 – Codes dans le fichier XSL permettant de générer des nœuds de type activité (Task)

Sphère d'isolation

```
<xsl:template match="dia:object[@type='BPMN - Isolation-Sphere']">
  <xsl:element name="node">
     <xsl:attribute name="id">
        <xsl:value-of select="@id"/>
     </xsl:attribute>
    <xsl:attribute name="type">
      <xsl:text>Isolation-Sphere</xsl:text>
    </xsl:attribute>
    <xsl:for-each select="dia:attribute[@name='elem_corner']/dia:point">
      <xsl:attribute name="posx">
         <xsl:value-of select="substring-before(@val,',')"/>
      </xsl:attribute>
      <xsl:attribute name="posy">
         <xsl:value-of select="substring-after(@val, ',')"/>
      </xsl:attribute>
    </xsl:for-each>
    <xsl:for-each select="dia:attribute[@name='elem_width']/dia:real">
      <xsl:attribute name="width">
         <xsl:value-of select="@val"/>
      </xsl:attribute>
    </xsl:for-each>
    <xsl:for-each select="dia:attribute[@name='elem_height']/dia:real">
      <xsl:attribute name="height">
         <xsl:value-of select="@val"/>
      </xsl:attribute>
    </xsl:for-each>
  </xsl:element>
</xsl:template>
```

FIG. 71 – Codes dans le fichier XSL permettant de générer des nœuds de type sphère d'isolation

Comme nous l'avons illustré dans la figure 63, la transformation XSLT du format Dia au format XML accepté par l'outil BPMN2BPEL SE est réalisée à l'aide du processeur XSL nommé SAXON 8. Saxon prend en entrée le fichier Dia qui correspond au fichier XML source, le fichier XSL qui décrit la transformation à effectuer et un nom de fichier en sortie qui correspondra à un fichier XML obéissant au

format requis par l'outil BPMN2BPEL SE.

Après avoir présenté l'implémentation nécessaire en termes de modélisation (Dia) et de transformations nécessaires à l'aboutissement d'un code BPEL et des différents contextes de coordination des sphères d'isolation, il est nécessaire d'introduire comment les WS-Sphères et plus particulièrement les WS-IsolationSphere soient implémentées dans une véritable plateforme d'exécution de web services. C'est l'objet de la section qui suit.

4.4 WS-Sphère, WS-IsolationSphere et le protocole CCDP dans une implémentation de WS-Coordination

Malgré l'importance des spécifications WS-Coordination et WS-Transaction, très peu de moteurs d'exécution de procédés BPEL les prennent en charge. Cela est dû à la jeunesse de la technologie web services et WSBPEL. Toutefois, la fondation "Apache Software Foundation" a entamé le développement d'une implémentation des spécifications WS-Coordination et WS-Transaction. Cette implémentation porte le nom "Apache Kandula". Le but de ce projet est d'implémenter, en open-source, les spécification WS-Coordination, WS-AtomicTransaction et WS-BusinessActivity. Du fait que Kandula soit relativement jeune, ses développeurs n'ont pour le moment implémenté que les spécifications WS-Coordination et WS-AtomicTransaction. WS-BusinessActivity est, au moment où ce mémoire est rédigé, en cours de préparation par les développeurs du projet.

Kandula implémente dores et déjà la spécification WS-Coordination et cela suffit pour y intégrer plusieurs types de coordinations. C'est pour cette raison que nous nous sommes naturellement intéréssé à un tel projet. De plus, de part sa nature open source, il nous a été facile d'accéder au code source qui est relativement simple et une modification était parfaitement evisageable.

4.4.1 Architecture de Kandula avec prise en charge des WS-Sphere

L'architecure de Kandula se présente en deux parties. La première partie est celle du serveur qui correspond au moteur d'exécution des coordinations WS-Coordination. Nous avons opéré une modification de cette architecture afin de prendre en charge les WS-Sphere. Dans l'implémentation nous nous sommes limité aux WS-IsolationSphere. Cette architecture est présentée dans la figure 72.

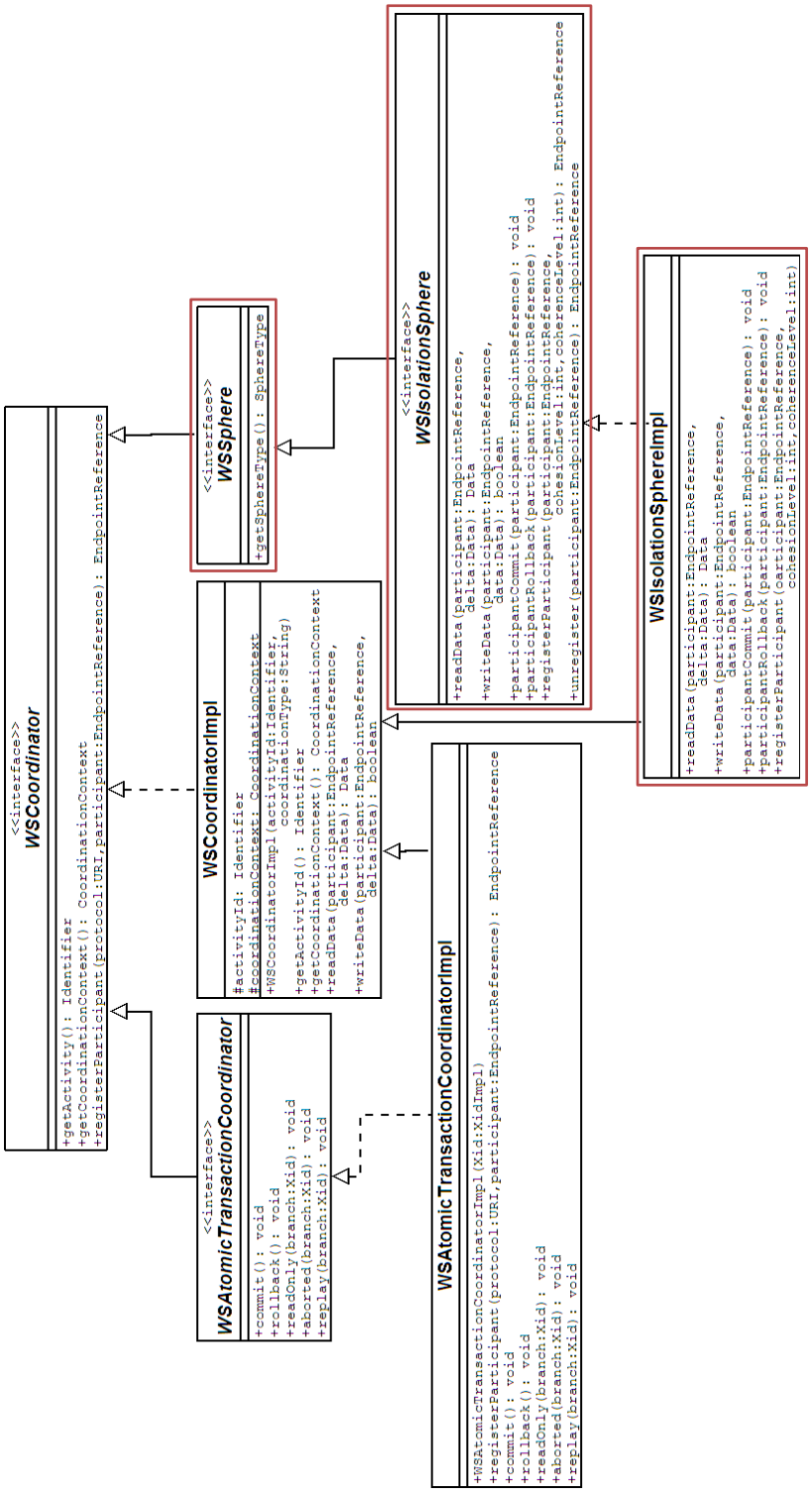


FIG. 72 – Architecture de Kandula avec prise en charge des WS-Sphere et WS-IsolationSphere

4.4.2 Implémentation du protocole CCDP

L'implémentation du protocole CCDP a été réalisée à travers la gestion des coordinateurs. En effet, dans Kandula, chaque activité est considérée comme étant elle même un coordinateur de sa propre exécution. Le type de coordinateur choisi pour une activité ne faisant pas partie de sphères d'isolation est, par défaut, le type neutre (WSCoordinatorImpl). C'est à travers ce coordinateur, à l'origine neutre, que nous avons introduit la gestion des lectures et écritures de données. Comme nous l'avons illustré dans la figure 72, nous avons intégré les méthodes readData et writeData dans l'implémentation par défaut WSCoordinationImpl. C'est ainsi, que le contrôle des lectures et écritures des activités appartenant ou non à une sphère d'isolation est assuré selon le protocole CCDP dont l'algorithme, présenté en section 1.1.2 de ce chapitre, est implémenté.

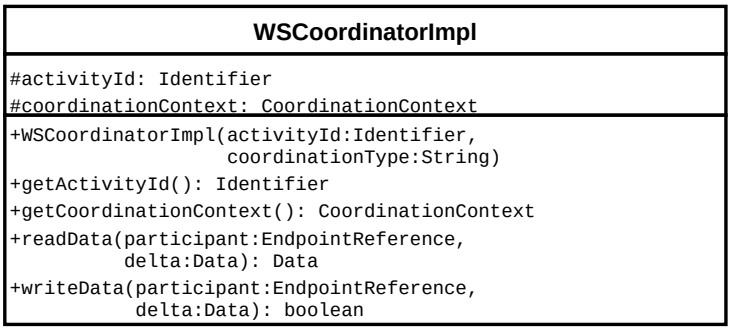


FIG. 73 – WSCoordination et la gestion des lectures/écritures

Lorsqu'un service (une activité) réalise une lecture et qu'elle fait partie d'une sphère d'isolation, la séquence d'exécution qui s'en suit est illustrée dans la figure 74. Nous y montrons d'abord comment un moteur d'exécution BPEL pourra créer les sphères nécessaires et mettre en correspondance les activités et les sphères. Ensuite, chaque lecture ou écriture opérée par un service passera automatiquement par le coordinateur WS-IsolationSphere correspondant. La sphère exécute alors l'algorithme CCDP afin d'établir si l'opération est possible ou pas, puis pose les verrous nécessaires et enfin lit ou écrit la donnée demandée et revoie le résultat. Nous illustrons l'exemple par le cas d'une sphère avec un niveau de cohésion au moins égal à 2 (Lectures répétables). Le service ws2 ne pourra alors pas écrire la même donnée que ws1 a lu tant que ws1 n'a pas fini son exécution.

Les procédures d'inscription et de désinscription d'une activité dans un coordinateur de type WS-IsolationSphere sont également implémentées conformément à l'algorithme que nous avons présenté en section 3.2 de ce chapitre. Cela a été réalisé en surchargant la méthode *register* et *unregister*.

Malgré le travail d'intégration des WS-Sphère et des WS-IsolationSphere dans une implémentation de WS-Coordination, en l'occurrence Kandular, il n'existe, de nos jours, aucune implémentation de moteur BPEL complet, prenant en charge les spécifications WS-Coordination et WS-Transaction. L'intégration de notre travail dans un tel moteur dépendra de la poursuite du projet Kandula par la fondation Apache. Les développeurs de Kandula mettent l'accent sur leur effort pour assurer l'interopérabilité de leur implémentation avec des implémentations d'autres technologies WS.

4.5 Synthèse

Le travail de mise en oeuvre a d'abord consisté à élaborer un protocole, nommé CCDP, et à développer l'algorithme équivalent. Un tel protocole permet d'assurer le respect des niveaux de cohérence et de cohésion des sphères tout en prenant en compte l'emboîtement de sphères. Nous avons ensuite choisi de

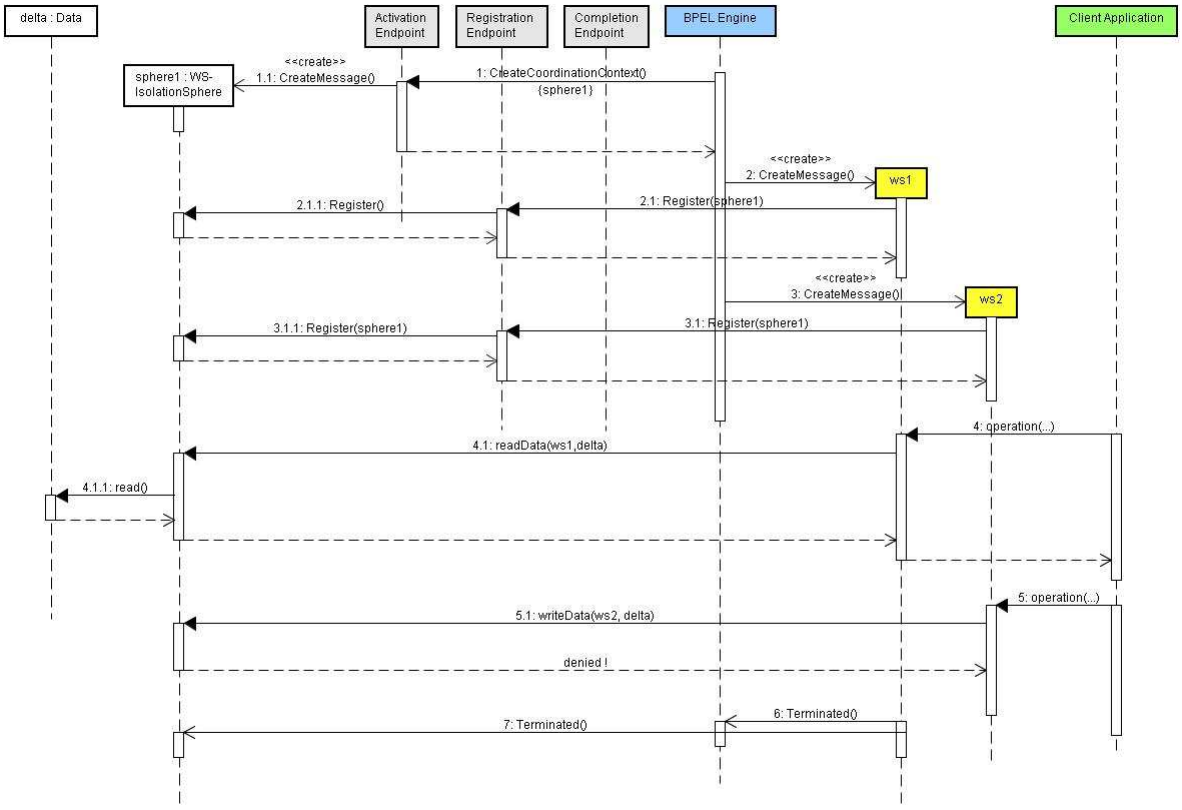


FIG. 74 – Diagramme de séquence suite à une opération de lecture ou d'écriture dans une sphère d'isolation

<<interface>>

WSIsolationSphere

```
+readData(participant:EndpointReference,
          delta:Data): Data
+writeData(participant:EndpointReference,
           data:Data): boolean
+participantCommit(participant:EndpointReference): void
+participantRollback(participant:EndpointReference): void
+registerParticipant(participant:EndpointReference,
                     cohesionLevel:int,coherenceLevel:int): EndpointReference
+unregister(participant:EndpointReference): EndpointReference
```

FIG. 75 – WSCoordination et la gestion des lectures/écritures

réaliser la mise en oeuvre de bout en bout, c'est à dire de la conception du procédé métier jusqu'à son exécution. En partant de la conception, nous avons choisi de recourir à l'éditeur DIA qui prend en charge un template BPMN, la notation Business Process standard. Nous avons étendu un tel template en y intégrant les sphères d'isolation. L'utilisateur pourra alors concevoir des diagrammes BPMN et y intégrer des sphères d'isolation.

Après avoir permis la conception graphique de procédés métiers BPMN intégrant des sphères d'iso-

lation, nous avons utilisé la fonctionnalité de DIA permettant l'exportation d'un tel procédé vers un fichier XML. La transformation de ce fichier à travers une transformation XSLT que nous avons préparé a permis de transformer le fichier XML généré par DIA en un autre fichier XML compatible avec l'outil BPMN2BPEL développé à l'Université de Melbourne. Nous avons apporté des modifications à BPMN2BPEL pour qu'il prenne en charge de nouvelles balises XML décrivant les sphères d'isolation. C'est ainsi que la transformation du procédé métier en un fichier BPEL prenant en compte le contexte WS-Coordination est possible.

Nous avons ainsi établi le lien entre la modélisation de procédés métiers, de façon graphique, et leur exécution à travers le langage BPEL. Nous avons ensuite intégré la notion de sphères dans l'implémentation de WS-Coordination connue sous le nom de Kandula et développée par la fondation Apache. Nous avons ainsi intégré les modifications faites sur l'architecture WS-Coordination dans celle de Kandula. La classe WSIsolationSphereImpl implémente la sphère d'isolation et intègre les opérations de base (readData et writeData) et permettent d'exécuter l'algorithme du protocole CCDP.

Kandula est encore en développement et la fondation Apache lui réserve une intégration dans une plateforme d'exécution de procédés BPEL. La prise en charge des sphères d'isolation n'est qu'une première étape dans notre travail et l'intégration d'autres types de sphère est une perspective qui nous tient à coeur. L'architecture proposée encourage cela, en ce sens que l'ajout de nouvelles sphères est assez facile puisqu'elles hériteront de la classe WS-Sphere.

Conclusion et perspectives

Rappel des objectifs de la recherche

Au terme de ce travail, nous avons voulu proposer une nouvelle façon de modéliser et d'exécuter des procédés métiers aboutissant à une plus grande flexibilité et une meilleure expressivité dans la gestion des propriétés comportementales. Pour cela, nous nous sommes orientés vers une solution à base de sphères, inspirée des sphères de contrôle [Davi 78]. Certes, plusieurs travaux dans la littérature ont évoqué des solutions à base de sphère comme les sphères d'atomicité et les sphères de compensation dont nous nous sommes inspirés. Toutefois, nous avons constaté qu'il était nécessaire de réunir tous ces travaux autour d'une même approche plus générale. Notre recherche vise à généraliser l'approche pour la modélisation et l'exécution de procédés métiers flexible en adoptant ce que nous appelons "sphères de comportement".

Nous avons montré qu'il existe un manque en matière de flexibilité dans la spécification de propriétés comportementales et nous avons associé cela à deux raisons majeures. En premier lieu, beaucoup de propriétés comportementales concernent non seulement l'activité à l'échelle individuelle ou le procédé en entier, mais aussi souvent des groupes d'activités. De nombreux systèmes de gestion de procédés métiers réduisent les propriétés comportementales uniquement à l'ordre individuel de l'activité. Cela diminue considérablement la flexibilité des procédés. En second lieu, les systèmes de gestion de procédés fournissent, dans la plupart des cas, un modèle de procédé intégrant la spécification de propriétés comportementales. Cela contraint les concepteurs de procédés métiers à se conformer à la structure et aux contraintes du modèle de procédé existant dans la spécification de propriétés comportementales avancées, ce qui engendre un manque de flexibilité et d'expressivité.

C'est à partir de telles constatations que nous avons opté pour la séparation des préoccupations (separation of concerns) entre la spécification du modèle de procédé et celui des propriétés comportementales. La séparation claire entre la spécification des éléments de base d'un procédé (les activités le constituant et leurs dépendances fonctionnelles, informationnelles ou temporelles) de celle des propriétés comportementales, et particulièrement celles qui concernent des groupes d'activités, est, selon nous, garant d'une flexibilité et d'une expressivité accrue des procédés métiers.

L'approche des sphères de comportement que nous avons proposée tente de constituer un modèle unifié de représentation de propriétés comportementales de groupe. Elle tire son origine de la notion de "sphère de contrôle" introduite par Davies en 1978 pour la gestion du traitement de données (Data Processing). Depuis, la notion de sphère a été adaptée et utilisée comme entité encapsulant plusieurs activités d'un procédé métier, pour exprimer de façon plus flexible et assurer de façon plus fiable des propriétés transactionnelles telles que l'atomicité et la compensation, donnant ainsi lieu aux notions de sphères d'atomicité et de sphères de compensation.

Bilan des contributions élaborées

Le travail réalisé au cours de cette thèse a aboutit à trois contributions principales répondant à l'objectif général de flexibilité, du point de vue modélisation et exécution, dans la gestion de propriétés comportementales telles que les propriétés transactionnelles, opérationnelles ou organisationnelles.

La première contribution de cette thèse a consisté à proposer une généralisation de la notion de sphère pour aboutir à la notion de sphères de comportement comme entité englobant un groupe d'activités d'un procédé. Une sphère de comportement permet de réaliser de façon fiable et flexible un comportement particulier sur un groupe d'activités. S'agissant de la séparation entre la spécification des proporiétés comportementales et celle des modèles de procédé utilisés, cela introduit une grande liberté dans la modélisation de procédés ainsi qu'une grande expressivité.

Afin de constituer une référence commune à toutes les solutions à base de sphères, nous avons définit les principes fondamentaux d'une sphère de comportement. Ces principes fondamentaux permettent de vérifier si une propriété comportementale de groupe est réalisable ou non en recourant à l'approche des sphères de comportement. En effet, ces principes fondamentaux éclaircissent suffisamment le champ d'application des sphères de comportement. Pour une propriété donnée, nous avons montré que les sphères de comportement ne sont pas toujours utiles. En vérifiant les principes appliqués à une propriété comportementale donnée, nous pouvons, dorénavant, vérifier si l'approche des sphères de comportement pourra être appliquée ou non.

La deuxième contribution de ce travail est le résultat de l'étude de cas des sphères d'isolation qui s'inscrivent dans la lignée des sphères transactionnelles (d'atomicité et de compensation) d'ores et déjà définies dans la littérature. Nous avons détaillé les caractéristiques d'une sphère d'isolation qui s'orientent vers deux dimensions, la cohésion et la cohérence. Nous avons également proposé un protocole, nommé CCDP, permettant d'assurer, à l'aide de verrous particuliers, la réalisation des contraintes requises par de telles sphères. Il a été particulièrement utile dans cette thèse de passer d'une définition des principes fondamentaux d'une sphère de comportement à l'application concrète de l'approche aux sphères d'isolation. Cela a permis de voir concrètement, comment l'approche des sphères de comportement apporte une grande flexibilité dans la modélisation et l'exécution de groupes d'activités présentant une propriété comportementale particulière, en l'occurrence l'isolation. En se référant aux théories sur la sérialisabilité des exécutions, nous avons pu proposer deux nouveaux critères de sérialisabilité particulièrement adaptés à l'exécution de sphères d'isolation : l'extra-Sphère-Sérialisabilité et l'intra-Sphère-Sérialisabilité.

Nous avons également identifié les anomalies engendrées par l'exécution d'activités coopératives dans un contexte non isolé et nous avons pu montrer l'apport, en termes de flexibilité et d'expressivité des sphères d'isolation, particulièrement à l'exécution d'un tel type d'activités. L'identification des anomalies de coopération n'a, à notre connaissance, pas été réalisée auparavant et les dimensions de cohésion et de cohérence des sphères d'isolation nous ont permis d'aller vers une réelle gestion de telles anomalies.

Enfin, la dernière et troisième contribution de la thèse a consisté à proposer une démarche pour la mise en oeuvre des sphères de comportement dans les plateformes de procédés à base de Web Services. Cela a consisté à proposer une modélisation dans la notation BPMN des sphères de comportement et leur spécification dans le langage BPEL. L'approche de mise en oeuvre est d'autant plus intéressante qu'elle est dite "de bout en bout" en ce sens que nous partons de la notation jusqu'au langage d'exécution du procédé et jusqu'à l'exécution proprement dite du procédé. Cette mise en oeuvre s'est construite autour des standards et spécifications tels que WS-Coordination et WS-Transaction. Nous avons alors proposé une nouvelle architecture WS-Coordination permettant de prendre en charge un nouveau type de coordination d'activités web services que nous avons appelé WS-Sphere. La solution technologique que nous avons proposé permet non seulement de réaliser des sphères d'isolation mais potentiellement tout type de sphères à condition d'en spécifier le fonctionnement.

Les perspectives de la recherche

Le travail réalisé dans cette thèse admet plusieurs prolongements envisageables pour de futures recherches.

D'abord, de façon générale, nous avons ouvert la voie à l'étude d'une multitude de propriétés comportementales susceptibles d'être réalisées à l'aide de sphères de comportement dans les procédés métiers. À travers l'étude de cas de l'isolation que nous avons entreprise, nous avons pu montrer l'ampleur du travail

qui reste à réaliser quant aux nombreuses autres propriétés comportementales possibles. Nous avons cité dans cette thèse quelques exemples de propriétés qui ont attiré notre attention de par leur pertinence en tant que sphères de comportement possibles. Pour chacune d'entre elles, il faudra en identifier les caractéristiques (telles que la cohésion et la cohérence pour l'isolation) ainsi que le détail de son comportement (tel que le protocole CCDP). La mise en oeuvre dans une plateforme de web services nous a permis de définir une spécification commune des sphères de comportement et de voir que les extensions en termes de WS-Sphere sont parfaitement accessibles. Cela nous encourage à aller plus loin et à proposer d'autres types de sphères de comportement.

À terme, notre objectif est de constituer une collection de sphères permettant de définir les propriétés les plus utiles à un concepteur de procédés métiers et de lui apporter un outil intégré offrant des outils de modélisation à base de sphères. Notre approche de mise en oeuvre est déjà un début puisque une extension est tout à fait possible et qu'il est parfaitement envisageable d'intégrer plusieurs types de sphères.

Néanmoins, face à la perspective de développer de plus en plus de sphères de comportement, il est important de se poser la question de l'interaction et de la co-existence de plusieurs sphères de comportement exprimant des propriétés différentes dans un même procédé. Il est donc envisageable d'étudier l'intégrité d'un nouveau type de sphères de comportement par rapport aux autres types existants. Pour aller encore plus loin, nous pourrions envisager d'étudier l'emboîtement, voire même le chevauchement de sphères hétérogènes. Cela devrait constituer une perspective certainement enrichissante.

Les principes des sphères de comportement définis dans cette thèse devront permettre d'identifier de nouvelles propriétés comportementales dont l'expression à travers des sphères sera utile. Néanmoins, il est important de souligner que nous n'avons pas proposé un modèle de sphère de comportement "générique" exprimant de façon générale tout comportement. Nous nous sommes contenté de définir la strucutre d'une sphère comme un sous ensemble d'activité mais pas les éléments décrivant son comportement proprement dit. La raison pour laquelle nous n'avons pas réalisé cela est simple. Chaque type de sphère exprime son comportement de façon totalement différente des autres et la corrélation de tous les types de comportements de sphères n'est pas évidente.

En effet, une sphère d'isolation exprime son comportement à travers deux niveaux, de cohésion et de cohérence. Une sphère d'atomicité exprime son comportement à travers le type d'atomicité désiré (atomicité stricte, atomicité flexible ...), une sphère d'instanciation multiple exprime son comportement à travers le nombre d'instances voulu ou la condition d'instancitation. Il est clair que chaque sphère exprime son comportement à sa manière. Toutefois, il serait très intéréssant de poursuivre ce travail afin de tenter de proposer un méta-modèle exprimant le comportement de toute sphère indépendamment de la propriété qu'elle modélise.

Une autre perspective intéréssante serait d'étudier les sphères de comportement dans un environnement d'exécution de procédés distribués. En effet, nous nous sommes limité dans notre travail aux procédés modélisés et exécutés au sein d'une même plateforme d'exécution centralisée. Le comportement d'une sphère ne peut pas demeurer le même dans un environnement distribué car, la sphère elle même serait composée d'activités provenant de procédés différents, pour la plupart déconnectés. Il serait intéréssant d'étudier comment un comportement tel que celui de l'isolation serait assuré dans de tels cas.

Nous avons étudié l'impact des sphères d'isolation sur les procédés coopératifs et cela nous a permis d'identifier les anomalies de coopération. Il serait également judicieux d'étudier l'impact des sphères d'isolation sur d'autres domaines d'application. Cela pourrait être celui du workflow scientifique où des quantité importante de données sont traitées, souvent de façon concurrente. Les contraintes de telles procédés sont différentes des procédés coopératifs.

Enfin, nous pensons que l'approche des sphères d'isolation devra être étendue aux objets complexes et ne pas considérer uniquement les données élémentaires. En effet, nous avons, tout au long de la thèse, assumé qu'une donnée $\delta \in \Delta$ est toujours élémentaire. Cela n'est pas toujours le cas, même si beaucoup de systèmes de bases de données font à tort, la même supposition. Si un objet complexe est manipulé par des activités d'une sphère d'isolation, quel serait le protocole d'isolation approprié ? Doit-on considérer un objet complexe comme élémentaire et appliquer les principes de verrouillage habituels ou bien, au contraire, définir des procédures plus fines pour aller à l'intérieur même d'un objet pour y poser des

verrous plus spécifiques ou pour utiliser une toute autre façon d'assurer l'isolation requise. De telles questions sont en suspend et c'est tout naturellement que cela sera le sujet de travaux futurs.

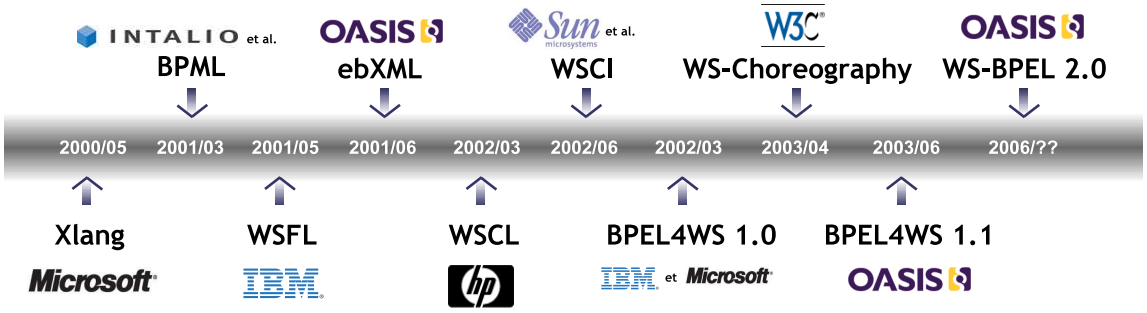
Toutes ces interrogations ne font que montrer la pertinence de l'approche des sphères de comportement et ne peuvent que nous encourager à aller plus loin pour développer l'approche et poursuivre la recherche sur la modélisation et l'exécution de procédés flexibles.

Annexe

L'évolution de la modélisation et l'exécution de procédés métiers en fonction des standards de BPM

Dans cette annexe, nous nous proposons de reprendre la figure 11 présentée dans l'état de l'art afin d'étudier l'évolution de la modélisation et de l'exécution de procédés métiers en fonction des standards qui y figurent. Nous soulignons qu'il s'agit des standards de spécification de procédés de web services. C'est ainsi que nous pouvons positionner cette annexe dans le cadre des travaux portant sur les architectures orientées services.

L'émergence des architectures orientées web services a débuté par la spécification du langage WSDL qui permet de définir le fonctionnement d'un web service. C'est à partir d'une telle spécification que plusieurs standards sont apparus. Certains standards ont permis d'étendre le langage WSDL en y apportant de nouvelles facettes. D'autres standards se sont tournés vers les aspects orchestration ou composition. Enfin, certains standards se sont orientés vers les aspects chorégraphie ou échanges de messages. Nous évoquerons dans ce qui suit, par chronologie de leur apparition, les principales spécifications définies par les acteurs du domaine depuis le début des années 2000.



Reprise de la figure 11 - Évolution des standards de la modélisation et de l'exécution de procédés métiers

6 Xlang Microsoft

Xlang est un langage proposé par Microsoft dans son logiciel serveur BizTalk pour gérer l'orchestration de web services. Ce langage est considéré comme le précurseur de la spécification BPEL4WS qui prendra la relève quelques années plus tard. Xlang représente un langage d'orchestration de web services, présenté comme une extension de WSDL (Web Services Definition Language) et élaboré particulièrement au sein du serveur BizTalk de Microsoft.

Xlang étend WSDL par l'aspect flot de messages : ce dernier est enrichi par deux nouveaux types

d'opérations que sont les "timeouts" ("deadline" et "duration") et les exceptions. Ces deux types d'opérations viennent étendre les quatre types proposés dans WSDL ("request/response", "solicit response", "one way", et "notification"). En outre, Xlang permet de définir un flot de données sous forme de données XML. Ce langage permet également une prise en charge des flots de contrôle séquentiels, parallèls ou conditionnels (<sequence>, <switch>, <all>) ainsi que les boucles itératives (<while>).

Le langage Xlang introduit aussi un contexte pour la gestion de manière robuste d'exceptions et pour l'exécution de transactions à longue durée d'exécution, et ce en définissant des actions de compensation en cas d'échec. Une autre fonctionnalité de Xlang, appelée "contract", correspond à un mapping unidirectionnel entre deux ports de services, partant d'une sortie vers une entrée. Le terme "contract" utilisé dans Xlang exprime tout simplement le fait qu'il s'agisse d'une connexion entre deux procédés de deux partenaires différents, d'où le rapprochement à une sorte de contrat. Néanmoins, les "Contracts" de Xlang ne gèrent en rien une quelconque sémantique B2B (Business to Business).

Plusieurs moteurs d'orchestration prennent encore en charge Xlang mais Microsoft a délibérément décidé de l'abandonner en faveur de BPEL4WS.

7 BPML

INTALIO et al.

BPML est un schéma XML permettant de décrire les processus métiers génériques en amont de l'enchaînement de messages WSCI. Le langage BPML sépare les interfaces publiques (description) des interfaces privées (implémentation). Ce langage a été développé par la BPMI (Business Process Management Initiative) composée d'Intalio, Sterling, Sun, CSC et d'autres organismes impliqués dans le projet.

BPML définit des activités basiques d'envoi et de réception de messages ainsi que d'invocation de services. Ces activités sont exécutées de façon séquentielle, parallèle ou conditionnelle, ce qui est classique pour un langage d'orchestration. BPML permet également une gestion transactionnelle à la fois en ce qui concerne des transactions courtes que des transactions longues. En plus, BPML adopte une gestion d'exceptions.

Dès les premiers "drafts" de BPML, le rôle d'un tel langage a été clairement défini en tant que langage d'implémentation privée d'un procédé métier, par opposition à la gestion des interactions publiques appelées chorégraphies qui seront spécifiées plus tard à l'aide de langages tels que WSCI.

8 WSFL

IBM

Le langage WSFL (Web Services Flow Language) est un langage à base d'XML pour la description de compositions de web services. Dans le langage WSFL, deux types de compositions de web services sont considérés. Le premier type de composition est orienté exécution de procédé métiers et constitue le modèle de flot (Flow Model). Dans le modèle de flot, l'unité de travail est l'activité. Les activités sont représentées comme des noeuds d'un graphe. Tout comme le workflow, les flots de contrôle ou de données correspondent aux arcs du graphe. L'objectif de WSFL est de lier chaque activité à son implémentation en tant que web service.

Le second modèle, celui du modèle global, permet une composition d'un autre niveau, celle de l'interaction inter-partenaires. En effet, il s'agit de réaliser un mapping entre les sorties d'un certain service avec les entrées d'un autre. WSFL est un peu plus avancé que Xlang par le fait que le mapping est réalisable dans les deux sens (bi-directionnel).

WSFL adopte une gestion robuste des exceptions, mais ne gère pas directement les aspects transactionnels.

9 ebXML

ebXML (Electronic Business using eXtensible Markup Language) est un langage qui a pour objectif de fournir une infrastructure globale d'échanges électroniques professionnels (Business to Business) principalement orientée vers le commerce électronique. Il s'agit d'une suite de spécifications définies autour de cinq couches :

- Processus métier (BP),
- Accords de Protocole de Collaboration (CPA),
- Composants de données (CDC),
- Transmission de messages,
- Enregistrements et dépôts.

C'est la spécification ebBP de ebXML qui nous intéresse plus particulièrement. L'ebBP se focalise sur les procédés métiers (Business Process) entre plusieurs partenaires. En décembre 2006, l'ebXML BPSS (Business Process Schema Specifications), ou l'ebBP v2.0.4 a été approuvé comme un standard d'OASIS. Un tel standard peut être soumis à la standardisation ISO dans un futur proche. L'ebBP v2 permet d'améliorer l'utilisation d'ebXML, des web services et de l'architecure orientée services (SOA). L'ebBP permet également à plusieurs partenaires d'échanger des messages SOAP et de définir la qualité des contrats de services tels que la fiabilité, la sécurité, et l'émission d'avis d'exception ou d'avis de réception. Les modèles de transaction, les expressions conditionnelles, les variables sémantiques de procédés métiers et d'autres fonctions sont prises en compte par ebBP.

10 WSCL

WSCL (Web Services Conversation Language) est une proposition supplémentaire pour orchestrer les Web Services élaborée par HP. C'est ainsi que WSCL vient rejoindre Xlang de Microsoft et WSFL de IBM. WSCL permet de définir les règles du dialogue entre Web Services à travers ce qui est appelé "conversation". L'objectif final est de décrire un procédé métier de web services.

WSCL n'a malheureusement pas eu de succès comparativement à d'autres standards proposés et ce pour une raison triviale : à quoi bon proposer une spécification alors qu'une autre permet de faire la même chose ? De plus, WSCL ne gère pas de la même manière un procédé métier, mais décrit plutôt les dialogues entre web services. Cela introduit une certaine ambiguïté entre orchestration et chorégraphie comme nous comptons l'expliquer ultérieurement.

11 WSCI

WSCI (Web Service Choreography Interface) est un standard co-développé par Sun, SAP, Intalio, et BEA pour la description du comportement global des services web. WSCI est destiné à être employé par des logiciels de chorégraphie de web services. Ce langage identifie les interactions entre web services liés au sein d'un même processus qui constitue un préalable important à la gestion de transactions complexes.

"WSCI permet ainsi d'utiliser les services web au sein de processus complexes", souligne Dario Wiser, responsable du marketing produits chez Sun Microsystems. Dario Wiser précise également que "WSCI n'est pas un concurrent des dialectes existants".

WSCI définit un langage à base d'XML pour la collaboration de web services. Le langage WSCI décrit l'échange de messages entre web services qui participent à une collaboration. En outre, WSCI gère la corrélation de messages ainsi que les exceptions, les transactions et la collaboration dynamique, et définit des règles de séquencement de messages. D'un point de vue transactionnel, WSCI décrit les transactions compensatrices.

Il est important de souligner que WSCI ne décrit que le comportement observable, au travers

d'échanges de messages, entre web services. Il ne décrit pas l'exécution de web services et n'aboutit donc pas à la description d'un procédé métier. C'est pour cette raison que WSCI a été considéré comme une couche supplémentaire à un niveau supérieur à WSDL.

12 BPEL : BPEL4WS 1.0 (2002) IBM et Microsoft
BPEL4WS 1.1 (2003) OASIS
WS-BPEL 2.0 (2006) OASIS

BPEL4WS (Business Process Execution Language for Web Services), appelé communément BPEL, est un langage basé sur XML et permettant de définir une tâche par la combinaison de services web. BPEL4WS utilise WSDL pour décrire les actions d'un processus. L'élaboration de BPEL4WS s'est nourrie, en partie, des deux standards WSFL (Web Services Flow Language) d'origine IBM, et de XLANG (XML Business Process Language) d'origine Microsoft que nous avons présenté précédemment. C'est pour cette raison que Microsoft et IBM se sont joints pour l'élaboration d'un tel standard.

BPEL4WS a pris naissance avec la version 1.0 en 2002, intitulée BPEL4WS et a permis de définir les élements de base de la spécification BPEL. L'objectif est de définir un procédé métier, lui même vu comme un web service. Des activités basiques (invocation de service) et structurées (séquentielles ou parallèles) sont prises en charge ainsi qu'une gestion des rôles.

Une gestion des transactions est également prise en charge. En effet, BPEL4WS inclut principalement deux spécifications intéressantes que sont WS-Coordination qui assure la communication entre les services web composant une tâche et WS-Transaction qui gère le déroulement transactionnel de processus de web services. De nos jours, plusieurs de compagnies clés du domaine du BPM s'orientent vers BPEL et commencent à abandonner les technologies telles que Xlang ou WSFL.

La version 1.1 a été proposé par BEA Systems, IBM, Microsoft, SAP and Siebel Systems et soumise à l'organisation de standardisation OASIS à travers un comité technique créé spécifiquement pour BPEL, l'OASIS WS-BPEL technical committee. Cette nouvelle version 1.1 a permis d'apporter plusieurs améliorations.

Enfin, le 14 septembre 2004, le comité technique OASIS WS-BPEL technical committee a voté la décision de renommer leur version 2.0 de la spécification BPEL sous l'appelation WS-BPEL 2.0. Ce changement d'appelation a été motivé par la volonté de s'aligner sur la nomenclature convenue des différents standard de web services WS-Technologies. WS-BPEL 2.0 a permis d'introduire plusieurs nouveautés. Nous pouvons citer principalement l'introduction de nouveaux types d'activités (if-then-else, repeatUntil, validate, forEach (parallèle et sequentielle), rethrow, extensionActivity). Nous citons également l'ajout d'une condition de terminaison (completion condition) pour les activités de type "forEach". D'autres améliorations en termes de gestion des variables ont été introduites, ainsi que des améliorations dans la gestion de l'échange de messages.

À l'opposé de la chorégraphie que nous présenterons ci-dessous, BPEL est orienté orchestration. La différence est particulièrement intéressante à clarifier. D'un côté, l'orchestration représentée par BPEL correspond à l'enchaînement de l'exécution de web services en vue de réaliser un procédé métier précis. Il s'agit alors de définir la séquence d'exécution, les possibles parallélismes, les boucles, etc. D'un autre côté, la chorégraphie fait intervenir plusieurs partenaires dans la réalisation de plusieurs tâches différentes généralement reliées sémantiquement. Il s'agit de décrire les échanges de messages entre participants à une chorégraphie. BPEL est alors un langage de programmation permettant de spécifier le comportement d'un participant à une chorégraphie. C'est la spécification WS-Choreography qui permet la description des échanges de messages entre participants à une chorégraphie comme présenté ci-dessous.

13 WS-Choreography

WS-CDL (WS-Choreography Definition Language) est un language de définition des échanges de messages entre participants à une chorégraphie. Il se distingue de BPEL par le fait qu'il est à l'échelle de la chorégraphie de plusieurs partenaires. Nous pouvons considérer que BPEL permet la gestion de l'exécution d'un procédé de web services (orchestration de web services) alors que WS-CDL permet de gérer l'échange de messages entre plusieurs procédés de web services (chorégraphie de procédés BPEL). Des exemples très simples sont donnés dans la littérature, mais ils ne permettent pas de saisir la différence entre BPEL et WS-CDL. Tout de même, nous pouvons lister certaines différences qui démarquent bien clairement l'objectif de chacune des spécifications, ainsi que leur complémentarité :

WS-CDL	**B**PEL
WS-CDL définit le format de données d'échange et ce pour l'ensemble des participants. L'objectif de WS-CDL est alors de gérer de façon cohérente, selon des règles définies, les échanges de messages entre les participants à une chorégraphie. Un participant implémente un ou plusieurs rôles. Un rôle étant associé à une ou plusieurs opérations pouvant être spécifiées en WSDL et donc en BPEL.	BPEL permet de définir le format de données d'échange pour chacun des participants individuellement. D'ailleurs, il n'existe par réellement de notion de participant en BPEL alors que c'est le point fort de WS-CDL. BPEL a pour but de décrire l'exécution (l'orchestration) d'une composition de web services aboutissant ainsi à un procédé BPEL. Un procédé BPEL est donc souvent un participant à une chorégraphie ce qui le place au rang de maillon de la chorégraphie.
WS-CDL définit de façon globale l'envoi de messages entre participants.	BPEL définit la façon dont les messages sont envoyés pour un web service.
WS-CDL définit des règles "réactives" qui sont utilisées par chacun des participants pour calculer quel message devra être échangé par la suite. Cela veut dire que les règles définies n'aboutissent pas à des actions proprement dites mais à des messages pourvant éventuellement, par réaction, aboutir à des actions indirectes.	BPEL définit des règles "actives" qui sont utilisées pour spécificer quelle activité devra être exécutée par la suite. Ainsi, dès que la règle est exécutée, le moteur d'orchestration déclenchera l'exécution éventuelle des activités correspondantes.

Tab. 1 – Comparaison entre WS-CDL et BPEL

En outre, WS-CDL définit des types de messages échangés entre participants permettant ainsi d'enrichir la sémantique de ces messages. WS-CDL permet également de définir des protocoles d'échanges de messages à travers des règles d'échange de messages.

Malgré l'importance de WS-CDL dans la gestion de la chorégraphie de procédés de web services, beaucoup de chercheurs et de praticiens continuent à se référer uniquement à BPEL y compris pour la chorégraphie de plusieurs procédés BPEL. Néanmoins, la chorégraphie réalisée à travers l'orchestration est assez réductrice et limite les possibilités. En effet, en utilisant BPEL, on se limite à deux types de chorégraphies que sont l'exécution en "maître/esclave" et l'exécution "unilatérale". L'exécution en "maître esclave" correspond à un procédé BPEL qui contrôlerait l'exécution d'un autre. C'est le cas classique de procédés et sous procédés. L'exécution "unilatérale" est celle où il n'y a, en définitive, aucune chorégraphie puisque chaque procédé s'exécute indépendamment de l'autre. Certes, la spécification WS-CDL apporte une couche supérieure d'échange de messages permettant une chorégraphie plus riche, sortant du clivage "maître/esclave" ou "unilatéral" et apportant une multitude de protocoles possibles.

14 Synthèse des standards et orientations actuelles

En synthèse d'après la présentation de cette annexe, nous avons clarifié l'évolution de la modélisation et de l'exécution de procédés métiers par un inventaire des différents standards proposés par les différents organismes et entreprises s'intérressant au domaine du Business Process Management à travers les web services. Cet engouement a été le fruit de l'avènement des architectures orientées services (SOA) et de la généralisation des web services. C'est la volonté de s'imposer sur le marché des normes et standards qui a poussé les différents acteurs du secteur à proposer autant de spécifications.

Nous avons pu déceler deux objectifs complémentaires visés par les standards présentés. Le premier objectif est celui de l'orchestration de web services. Orchestrer des web services revient à en décrire l'exécution au travers d'une structure de procédé à travers éventuellement un flot de contrôle (séquentiel, parallèl, conditionnel, en boucle), un flot de données, une gestion des exceptions et une gestion transactionnelle. C'est Xlang qui a posé les premières pierres, puis BPML, WSFL ainsi que d'autres spécifications, non orientées SOA, telles que XPDL qui décrit un procédé métier indépendamment de la technologie des web services.

Le deuxième objectif des spécifications et standards proposés est celui de la chorégraphie des web services. La chorégraphie, comme nous l'avons expliqué, se propose de gérer les échanges de messages selon des règles à définir entre les web services. WSCI et WS-Choreography sont les deux principaux standards pour l'orchestration de web services. La chorégraphie correspond au niveau de collaboration de web services alors que l'orchestration correspond à celui de l'exécution de procédés métiers.

Ainsi, nous pouvons conclure à travers l'identification de deux clivages. Le premier correspond à la combinaison de WSBPEL et WS-Choreography au sein de ce que nous appelons maintenant les WS-Technologies. Le second est celui de la combinaison de BPML et de WSCI. Néanmoins, nous assistons de nos jours à un engouement pour l'adoption de WSBPEL par des plateformes d'exécution de web services. D'autres standards, tels que BPMN (Business Process Modelling Notation) ont permis d'apporter un soutien à WSBPEL du fait qu'ils se proposent d'être compatibles avec une telle spécification.

Liste des symboles

- $\mathcal{A}$: ensemble des activités du système 66
- $\widetilde{\mathcal{A}}$: ensemble des instances d'activités du système 66
- $P = (\mathcal{A}ct(P), \prec_P)$: procédé composé d'un ensemble $\mathcal{A}ct(P)$ d'activités et un ordre partiel $\prec_P$ 67
- $\mathcal{A}ct(P)$: ensemble d'activités du procédé P 67
- $\prec_P$: relation d'ordre de précédence des activités du procédé P 67
- $I = (\widetilde{\mathcal{A}ct}(I), <_I)$: instance du procédé P 67
- Δ : ensemble des données du système d'information 77
- $\theta(\delta, t)$: état d'une donnée $\delta \in \Delta$ à l'instant t 78
- $\mathcal{OP}$: ensemble d'opérations sur les données de Δ 78
- $\theta^-(\delta, op)$: état d'une donnée δ immédiatement avant l'exécution de l'opération op 78
- $\theta^+(\delta, op)$: état d'une donnée δ immédiatement après l'exécution de l'opération op 78
- $\theta^+(\delta, \{op_1, ..., op_n\})$: état d'une donnée δ immédiatement après l'exécution de toutes les opérations $op_1, ..., op_n$ 78
- $\mathcal{S}tate(\widetilde{a}) \in \mathcal{STACT}$: état d'une instance d'activité $\widetilde{a} \in \widetilde{\mathcal{A}}$ 79
- $\mathcal{STACT}$: représente l'ensemble d'états possibles attribués à une instance d'activité durant son exécution 79
- $\mathcal{OP}(\widetilde{a})$: ensemble d'opération d'une instance d'activité $\widetilde{a}$ 66
- $op^{\widetilde{a}}_{commit}$: opération de validation d'une instance d'activité $\widetilde{a} \in \widetilde{\mathcal{A}}$ 66
- $op^{\widetilde{a}}_{rollback}$: opération d'annulation d'une instance d'activité $\widetilde{a} \in \widetilde{\mathcal{A}}$ 66
- $h = (\mathcal{OP}(h), \xrightarrow{h})$: histoire d'exécution des opérations de l'ensemble $\mathcal{OP}(h) \subseteq \mathcal{OP}$ 79
- $\mathcal{OP}(h)$: ensemble d'opérations intervenants dans une histoire h d'exécution d'opérations 79
- $\xrightarrow{h}$: relation d'ordre total sur les opérations exprimant l'ordre d'exécution 79
- $\theta^-(\delta, \widetilde{a})$: état d'une donnée δ immédiatement avant l'exécution d'une opération de $\mathcal{OP}(\widetilde{a})$ 80
- $\theta^+(\delta, \widetilde{a})$: résultat final de l'activité a égal à l'état d'une donnée δ immédiatement après l'exécution de toutes les opérations de $\mathcal{OP}(\widetilde{a})$ 80
- a^{-1} : activité compensatrice de a 81
- $\widetilde{a}_i \rightleftharpoons \widetilde{a}_j$: deux activités commutent 82
- $State(I) \in \mathcal{STPROC}$: représente l'état d'une instance de procédé I 82
- $\mathcal{STPROC}$: représente l'ensemble d'états possibles attribués à une instance de procédé 82
- $H = (I(H), <^H)$: représente l'histoire d'exécution des instances d'activités de l'ensemble I^H d'instances de procédés 83
- $<^H$: l'ordre partiel observé de H 83
- $\widetilde{\mathcal{A}ct}(H)$: l'ensemble d'instances d'activités de toutes les instances de procédés impliquées dans l'histoire d'exécution d'activités H 83
- $\omega = (\widetilde{\mathcal{A}ct}(\omega))$: représente une sphère de comportement 84
- $\omega = (\widetilde{\mathcal{A}ct}(\omega), \mathcal{CHS}(\omega), \mathcal{CHR}(\omega))$: représente une sphère d'isolation 84
- $\widetilde{\mathcal{A}ct}(\omega)$: ensemble d'activités participants à la sphère d'isolation ω 84

- $\mathcal{CHS}(\omega)$: niveau de cohésion de la sphère d'isolation ω 84
- $\mathcal{CHR}(\omega)$: niveau de cohérence de la sphère d'isolation ω 84
- Ω : ensemble des sphères du système 84
- $\Omega(\widetilde{A})$: ensemble des sphères impliquées dans un sous ensemble $\widetilde{A} \subset \widetilde{\mathcal{A}}$ 84
- PSR : Critère de Process-Sérialisabilité 96
- $GSP(H)$: Graphe de Sérialisabilité de Procédés relatif à une histoire d'exécution d'activités 96
- $PreSA(\omega)$: Activités Pré-Sphère 97
- $PostSA(\omega)$: Activités Post-Sphère 97
- $SimSA(\omega)$: Activités Simulanées à une Sphère 97
- $SSubH(H, \omega)$: Sous histoire d'exécution de H par rapport à une sphère ω 98
- eSS : extra-Sphère-Sérialisabilité 98
- $\widetilde{Act}_{\odot}(\omega)$: Activités directes d'une sphère d'isolation 101

Liste des définitions

1 Transaction 39
2 Sérialisabilité (contraintes sur l'ordonnancement d'opérations conflictuelles) 42
3 Sérialisabilité (l'équivalence à un ordonnancement série) 43
4 Sérialisabilité (graphe de conflits acyclique) 43
5 Ensemble $\mathcal{A}$ des activités du système 66
6 Ensemble $\widetilde{\mathcal{A}}$ des instances d'activités du système 66
7 Procédé 67
8 Instance de procédé 67
9 Sphères de Comportement 68
10 Ensemble Δ des données du système d'information 77
11 État d'une donnée 78
12 Ensemble $\mathcal{OP}$ d'opérations sur les données de Δ 78
13 États particuliers d'une donnée $\delta \in \Delta$ lors de l'exécution d'une opération $op \in \mathcal{OP}$ 78
14 Les opérations d'une instance d'activité $\widetilde{a}$ 79
15 État d'une instance d'activité 79
16 Histoire d'exécution des opérations - *Operations schedule* 79
17 Ordonnancement d'instances d'activités et le séquencement de leurs opérations respectives 80
18 Résultat final d'une instance d'activité 80
19 État d'une donnée : Résultat final / Résultat intermédiaire 81
20 Activité sans effets 81
21 Activité Compensatrice 81
22 Commutativité et Conflits 82
23 Commutativité Parfaite 82
24 État d'une instance de procédé 82
25 Histoire d'exécution d'activités - *Activities Schedule* 83
26 Sphère d'isolation 84
27 Sérialisabilité 88
28 Histoire d'exécution $H(\omega)$ d'une sphère ω 89
29 intra-Sphère-Sérialisabilité (iSS) 89
30 Process-Sérialisabilité (PSR) 96
31 Graphe de Sérialisabilité de Procédés $GSP(H)$ 96
32 Activités Pré-Sphère : $PreSA(\omega)$ 97
33 Activités Post-Sphère : $PostSA(\omega)$ 97
34 Activités Simultanées à une Sphère : $SimSA(\omega)$ 97
35 Sous-histoire d'exécution 98
36 Sous-histoire d'exécution $SSubH(H, \omega)$ par rapport à une sphère 98
37 extra-Sphère-Sérialisabilité (eSS) 98
38 Activités directes d'une sphère d'isolation 101

Bibliographie

[Aals 02] W. van der Aalst and K. van Hee. *Workflow Management : Models, Methods, and Systems*. MIT Press, Cambridge, 2002.

[Aals 03] W. M. P. V. D. Aalst, A. H. M. T. Hofstede, B. Kiepuszewski, and A. P. Barros. "Workflow Patterns". *Distributed and Parallel Databases*, Vol. 14, No. 1, pp. 5–51, July 2003.

[Aals 98] W. Aalst. "The Application of Petri Nets to Workflow Management". *The Journal of Circuits, Systems and Computers*, Vol. 8, No. 1, pp. 21–66, 1998.

[Alon 96] G. Alonso, D. Agrawal, A. E. Abbadi, M. Kamath, R. Günthör, and C. Mohan. "Advanced Transaction Models in Workflow Contexts". In : S. Y. W. Su, Ed., *Proceedings of the Twelfth International Conference on Data Engineering, February 26 - March 1, 1996, New Orleans, Louisiana*, pp. 574–581, IEEE Computer Society, 1996.

[ANSI SQL 92] *Database Language SQL ANSI X3.135-1992*. American National Standard for Information Systems, November 1992.

[Bern 80] P. A. Bernstein and N. Goodman. "Timestamp-Based Algorithms for Concurrency Control in Distributed Database Systems". In : *Sixth International Conference on Very Large Data Bases, October 1-3, 1980, Montreal, Quebec, Canada, Proceedings*, pp. 285–300, IEEE Computer Society, 1980.

[Bern 87] P. A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, 1987.

[Bhir 05] S. Bhiri, O. Perrin, and C. Godart. "Ensuring required failure atomicity of composite Web services". In : *WWW '05 : Proceedings of the 14th international conference on World Wide Web*, pp. 138–147, ACM Press, New York, NY, USA, 2005.

[Bieb 05] N. Bieberstein, S. Bose, M. Fiammante, K. Jones, and R. Shah. *Service-Oriented Architecture (SOA) Compass : Business Value, Planning, and Enterprise Roadmap*. IBM Press, 2005.

[BPML02] BPML specification 1.0 final Draft, November, 2002.

[Brei 93] Y. Breitbart, A. Deacon, H.-J. Schek, A. P. Sheth, and G. Weikum. "Merging Application-centric and Data-centric Approaches to Support Transaction-oriented Multi-system Workflows.". *SIGMOD Record*, Vol. 22, No. 3, pp. 23–30, 1993.

[Cana 98a] G. Canals, C. Godart, P. Molli, and M. Munier. "A criterion to enforce correctness of indirectly cooperating applications". *Inf. Sci.*, Vol. 110, No. 3-4, pp. 279–302, 1998.

[Cana 98b] G. Canals, C. Godart, F. Charoy, P. Molli, and H. Skaf. "COO Approach to Support Cooperation in Software Developments". *IEE Proceedings - Software*, Vol. 145, No. 2-3, pp. 79–84, Jun 1998.

[Char 06] M. V. F. Fran¸cois Charoy, Adnene Guabtni. "A Dynamic Workflow Management System for Coordination of Cooperative Activities". In : *Proceedings of the First International Workshop on Dynamic Process Management, in conjunction with the Fourth International Conference on Business Process Management*, pp. 205–216, Springer LNCS, 2006.

[Chry 90] P. Chrysanthis and K. Ramamritham. "A Framework for Specifying and Reasoning about Transaction Structure and Behavior". *Proccedings of the ACM SIGMOD International Conference on Management of Data*, pp. 194–203, 1990.

[Chry 91] P. K. Chrysanthis and K. Ramamritham. "A Formalism for Extended Transaction Model". In : G. M. Lohman, A. Sernadas, and R. Camps, Eds., *17th International Conference on Very Large Data Bases, September 3-6, 1991, Barcelona, Catalonia, Spain, Proceedings*, pp. 103–112, Morgan Kaufmann, 1991.

[Chry 92] P. Chrysanthis and K. Ramamritham. "ACTA : The SAGA Continues". *Database transaction models for advanced applications*, 1992.

[Chry 94] P. Chrysanthis and K. Ramamritham. "Synthesis of Extended Transaction Models". *ACM Transactions on Database Systems, 19(3)*, pp. 451–491, September 1994.

[Cook 96] M. Cook. *Building Enterprise Information Architectures : Reengineering Information Systems*. Englewood Cliff, nj prentice hall Ed., 1996. ISBN : 0-13-440256-1, page 78.

[Davi 78] C. T. Davies. "Data Processing Spheres of Control". *IBM Systems Journal 17(2) : 179-198*, 1978.

[Daya 90] U. Dayal, M. Hsu, and R. Ladin. "Organizing Long-Running Activities with Triggers and Transactions". In : H. Garcia-Molina and H. V. Jagadish, Eds., *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, May 23-25, 1990*, pp. 204–214, ACM Press, 1990.

[Daya 91] U. Dayal, M. Hsu, and R. Ladin. "A Transactional Model for Long-Running Activities". In : G. M. Lohman, A. Sernadas, and R. Camps, Eds., *17th International Conference on Very Large Data Bases, September 3-6, 1991, Barcelona, Catalonia, Spain, Proceedings*, pp. 113–122, Morgan Kaufmann, 1991.

[Dehn 01] J. Dehnert. "Four Systematic Steps Towards Sound Business Process Models". In : F. G. ISST, Ed., *Proceedings of the 2nd International Colloquium on Petri Net Technologies for Modelling Communication Based Systems*, pp. 55–64, Berlin, 2001.

[Derk 01] W. Derks, J. Dehnert, P. Grefen, and W. Jonker. "Customized Atomicity Specification for Transactional Workflow.". In : *Proceedings of the Third International Symposium on Cooperative Database Systems for Advanced Applications (CODAS'01)*, pp. 140–147, IEEE Computer Society, 2001.

[ebXML01] ebXML Technical Architecture Specification v1.0.4, ebXML Technical Architecture Project Team, February 16, 2001.

[Elli 80] C. A. Ellis and G. J. Nutt. "Office Information Systems and Computer Science". *ACM Comput. Surv.*, Vol. 12, No. 1, pp. 27–60, 1980.

[Elli 93] C. A. Ellis and G. J. Nutt. "Modeling and Enactment of Workflow Systems.". In : A. Marsan, M., Ed., *Lecture Notes in Computer Science ; Application and Theory of Petri Nets 1993, Proceedings 14th International Conference, Chicago, Illinois, USA*, pp. 1–16, Springer-Verlag, 1993.

[Elma 90] A. Elmagarmid, Y. Leu, W. Litwin, and M. Rusinkiewicz. "A multidatabase transaction model for InterBase". In : *Proceedings of the sixteenth international conference on Very large databases*, pp. 507–518, Morgan Kaufmann Publishers Inc., 1990.

[Elma 92] A. K. Elmagarmid, Ed. *Database Transaction Models for Advanced Applications*. Morgan Kaufmann, 1992.

[Garc 87] H. Garcia-Molina and K. Salem. "Sagas". In : *Proceedings of the 1987 ACM SIGMOD international conference on Management of data*, pp. 249–259, ACM Press, 1987.

[Geor 94] D. Georgakopoulos and M. F. Hornick. "A Framework for Enforceable Specification of Extended Transaction Models and Transaction Workflows.". *Int. J. Cooperative Inf. Syst.*, Vol. 3, No. 3, pp. 599–617, 1994.

[Geor 95a] D. Georgakopoulos, M. F. Hornick, and A. P. Sheth. "An Overview of Workflow Management : From Process Modeling to Workflow Automation Infrastructure". *Distributed and Parallel Databases*, Vol. 3, No. 2, pp. 119–153, 1995.

[Geor 95b] D. Georgakopoulos, M. F. Hornick, and A. P. Sheth. "An Overview of Workflow Management : From Process Modeling to Workflow Automation Infrastructure.". *Distributed and Parallel Databases*, Vol. 3, No. 2, pp. 119–153, 1995.

[Geor 96] D. Georgakopoulos, M. F. Hornick, and F. Manola. "Customizing Transaction Models and Mechanisms in a Programmable Environment Supporting Reliable Workflow Automation". *IEEE Trans. Knowl. Data Eng.*, Vol. 8, No. 4, pp. 630–649, 1996.

[Goda 96] C. Godart, G. Canals, F. Charoy, P. Molli, and H. Skaf. "Designing and Implementing COO : Design Process, Architectural Style, Lessons learned". In : *ICSE 18 (International Conference On Software Engineering), Berlin*, pp. 342–352, IEEE Publishing Computer Society Press, mar 1996.

[Goda 99] C. Godart, O. Perrin, and H. Skaf. "COO : a workflow operator to improve cooperation modeliong in virtual processes". *9th Int. Workshop on research Issues on Data Engineering Information technology for Virtual Entreprises (RIDEVE'99)*, 1999.

[Gref 00] P. Grefen, K. Aberer, Y. Hoffer, and H. Ludwig. "CrossFlow : Cross-Organizational Workflow Management in Dynamic Virtual Enterprises". *Computer Systems Science & Engineering*, No. 5, pp. 277–290, 2000.

[Gref 02a] P. Grefen. "A Taxonomy for Transactional Workflows". CTIT Technical Reports Vol.2, No 11, University of Twente, 2002.

[Gref 02b] P. W. P. J. Grefen. "Transactional Workflows or Workflow Transactions ?". *DEXA*, pp. 60–69, 2002.

[Gref 98] P. Grefen and R. R. de Vries. "A reference architecture for workflow management systems". *Data Knowl. Eng.*, Vol. 27, No. 1, pp. 31–57, 1998.

[Grig 01a] D. Grigori, F. Charoy, and C. Godart. "Anticipation to Enhance Flexibility of Workflow Execution.". In : *DEXA*, pp. 264–273, 2001.

[Grig 01b] D. Grigori, F. Charoy, and C. 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*, pp. 139–146, IEEE, Apr 2001.

[Grig 04] D. Grigori, F. Charoy, and C. Godart. "Coo-Flow : A Process Technology To Support Cooperative Processes.". *International Journal of Software Engineering and Knowledge Engineering*, Vol. 14, No. 1, pp. 61–78, 2004.

[Guab 04] A. Guabtni and F. Charoy. "Multiple Instantiation in a Dynamic Workflow Environment". In : A. Persson and J. Stirna, Eds., *Advanced Information Systems Engineering, 16th International Conference, CAiSE 2004, Riga, Lavtia*, pp. 175–188, Springer, Jun 2004.

[Guab 05] A. Guabtni, F. Charoy, and C. Godart. "Spheres of isolation : adaptation of isolation levels to transactional workflow". In : W. M. van der Aalst et al., Ed., *Business Process Management, 3rd International Conference, BPM 2005, Nancy, France*, pp. 458–463, Springer, September 2005.

[Haye 91] K. Hayes and K. Lavery. "Workflow management software : the business opportunity". Technical Report, Ovum Ltd, London, 1991.

[Heuv 02] W.-J. van den Heuvel and S. Artyshchev. "Developing A Three-Dimensional Transaction Model for Supporting Atomicity Spheres". *International Workshop on Web Services Research, Standardization, and Deployment*, 2002.

[Hsu 98] M. Hsu and C. Kleissner. "ObjectFlow and Recovery in Workflow Management Systems". In : V. Kumar and M. Hsu, Eds., *Recovery Mechanisms in Database Systems*, pp. 505–527, Pentice Hall, 1998.

[Jabl 96] S. Jablonski and C. Bussler. *Workflow Management : Modeling Concepts, Architecture and Implementation*. International Thomson Computer Press, 1996.

[Kiep 98] B. Kiepuszewski, R. Mühlberger, and M. E. Orlowska. "FlowBack : Providing Backward Recovery for Workflow Systems". In : L. M. Haas and A. Tiwary, Eds., *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*, pp. 555–557, ACM Press, 1998.

[Koul 95] T. M. Koulopoulos. *The Workflow Imperative : Building Real World Business Solutions.* John Wiley & Sons, Inc., New York, NY, USA, 1995.

[Kris 95] N. Krishnakumar and A. P. Sheth. "Managing Hetergeneous Multi-system Tasks to Support Enterprise-Wide Operations.". *Distributed and Parallel Databases*, Vol. 3, No. 2, pp. 155–186, 1995.

[Kung 79] H. T. Kung and J. T. Robinson. "On Optimistic Methods for Concurrency Control". In : A. L. Furtado and H. L. Morgan, Eds., *Fifth International Conference on Very Large Data Bases, October 3-5, 1979, Rio de Janeiro, Brazil, Proceedings*, p. 351, IEEE Computer Society, 1979.

[Lamp 78] L. Lamport. "Time, Clocks, and the Ordering of Events in a Distributed System.". *Commun. ACM*, Vol. 21, No. 7, pp. 558–565, 1978.

[Leym 00] F. Leymann and D. Roller. *Production Workflow*, Chap. Chapter 7 : Workflows and Transactions. Ed. Prentice Hall, Inc., Upper Saddle River, New Jersey, second edition Ed., 2000.

[Leym 95] F. Leymann. "Supporting Business Transactions Via Partial Backward Recovery In Workflow Management Systems". In : *BTW*, pp. 51–70, 1995.

[Mehr 92] S. Mehrotra, R. Rastogi, H. F. Korth, and A. Silberschatz. "A Transaction Model for Multidatabase Systems.". In : *ICDCS*, pp. 56–63, 1992.

[Mend 05] J. Mendling, G. Neumann, and M. Nüttgens. *Workflow Handbook 2005*, Chap. A Comparison of XML Interchange Formats for Business Process Modelling, pp. 185–198. Future Strategies Inc., Lighthouse Point, Florida, USA, May 2005.

[Moll 96] P. Molli. *Environnement de Développement Coopératif.* PhD thesis, Université Henri Poincaré - Nancy 1, juin 1996. Thèse de doctorat.

[Moss 82] J. E. B. Moss. "Nested Transactions and Reliable Distributed Computing". In : IEEE, Ed., *Proceedings of the 2nd Symposium in Distributed Software and Database Systems*, 1982.

[Nodi 92] M. H. Nodine, S. Ramaswamy, and S. B. Zdonik. "A Cooperative Transaction Model for Design Databases". In : *Database Transaction Models for Advanced Applications*, pp. 53–85, Morgan Kaufmann, 1992.

[Ouya 06a] C. Ouyang, W. M. van der Aalst, M. Dumas, and A. H. ter Hofstede. "From Business Process Models to Process-oriented Software Systems : The BPMN to BPEL Way". Tech. Rep., Open-access archive of QUT research literature QUTePrints, October 2006. Accessed from http ://eprints.qut.edu.au.

[Ouya 06b] C. Ouyang, W. M. van der Aalst, M. Dumas, and A. H. ter Hofstede. "Translating BPMN to BPEL". Tech. Rep., Open-access archive of QUT research literature QUTePrints, July 2006. Accessed from http ://eprints.qut.edu.au.

[Rama 93] K. Ramamritham and P. Chrysanthis. "In Search of Acceptability Criteria : Database Consistency Requirements and Transaction Correctness Properties". *Distributed Object Management*, 1993.

[Reut 89] A. Reuter. "Contracts : A Means for Extending Control Beyond Transaction Boundaries". *In Proceedings of the 2nd International Workshop on High Performance Transaction Systems, September*, 1989.

[Reut 95] A. Reuter and F. Schwenkreis. "ConTracts - A Low-Level Mechanism for Building General-Purpose Workflow Management-Systems". *IEEE Data Eng. Bull.*, Vol. 18, No. 1, pp. 4–10, 1995.

[Reut 97] A. Reuter, K. Schneider, and F. Schwenkreis. "ConTracts Revisited". *Advanced Transaction Models and Architectures*, pp. 127–151, 1997.

[Rusi 95] M. Rusinkiewicz and A. P. Sheth. "Specification and Execution of Transactional Workflows.". In : *Modern Database Systems*, pp. 592–620, 1995.

[Sadi 05] S. W. Sadiq, M. E. Orlowska, and W. Sadiq. "Specification and validation of process constraints for flexible workflows.". *Inf. Syst.*, Vol. 30, No. 5, pp. 349–378, 2005.

[Schu 02a] H. Schuldt, G. Alonso, C. Beeri, and H.-J. Schek. "Atomicity and isolation for transactional processes". *ACM Trans. Database Syst.*, Vol. 27, No. 1, pp. 63–116, 2002.

[Schu 02b] H. Schuldt, G. Alonso, C. Beeri, and H.-J. Schek. "Atomicity and isolation for transactional processes". *ACM Transactions on Database Systems (TODS)*, Vol. 27, No. 1, pp. 63–116, 2002.

[Shet 93] A. P. Sheth and M. Rusinkiewicz. "On Transactional Workflows". *IEEE Data Eng. Bull.*, Vol. 16, No. 2, pp. 37–40, 1993.

[SOAref06] "OASIS Reference Model for Service Oriented Architecture 1.0". C. Matthew MacKenzie, Ken Laskey, Francis McCabe, Peter F Brown, Rebekah Metz, Committee Specification 1, 2 August 2006.

[Tang 95] J. Tang and J. Veijalainen. "Transaction-oriented workflow concepts in inter-organizational environments". In : *CIKM '95 : Proceedings of the fourth international conference on Information and knowledge management*, pp. 250–259, ACM Press, New York, NY, USA, 1995.

[Ving 98] R. Vingralek, H. Hasse-Ye, Y. Breitbart, and H. J. Schek. "Unifying concurrency control and recovery of transactions with semantically rich operations". *Theoretical Computer Science*, pp. 363–396, 1998.

[Vonk 00] J. Vonk, W. Derks, P. W. P. J. Grefen, and M. Koetsier. "Cross-Organizational Transaction Support for Virtual Enterprises". In : O. Etzion and P. Scheuermann, Eds., *Cooperative Information Systems, 7th International Conference, CoopIS 2000, Eilat, Israel, September 6-8, 2000, Proceedings*, pp. 323–334, Springer, 2000.

[WfMC] Workflow Management Coalition, Informations sur la coalition, Modèles de référence, Glossaire. Voir la page web de la coalition http ://www.wfmc.org/.

[Wfmc 99] W. M. C. Wfmc. "The Workflow Management Coalition Specification : Terminology and Glossary - Document Number WFMC-TC-1011". February 1999. Document Status - Issue 3.0.

[Whit 04] S. White. "Business Process Modeling Notation (BPMN). Version 1.0". May 3, 2004. BPMI.org, 2004. www.bpmi.org.

[WS BPEL] *Business Process Execution Language for Web Services, version 1.1.* IBM, BEA Systems, Microsoft, SAP AG, Siebel Systems, February 01, 2005.

[WS CDL05] "Web Services Choreography Description Language Version 1.0". W3C Candidate Recommendation. By Nickolas Kavantzas, David Burdett, Gregory Ritzinger, Tony Fletcher, Yves Lafon, Charlton Barreto. November 9, 2005.

[WS COORD] *Web Services Coordination.* IBM, BEA Systems, Microsoft, Arjuna, Hitachi, IONA. August, 2005.

[WS TRANS] *Web Services Transaction.* IBM, BEA Systems, Microsoft, Arjuna, Hitachi, IONA, August 16, 2005.

[WSCI02] "Web Service Choreography Interface (WSCI) 1.0". By Assaf Arkin, Sid Askary, Scott Fordin, Wolfgang Jekeli, Kohsuke Kawaguchi, David Orchard, Stefano Pogliani, Karsten Riemer, Susan Struble, Pal Takacsi-Nagy, Ivana Trickovic, Sinisa Zimek. BEA Systems, Intalio, SAP, Sun Microsystems. August, 2002.

[WSCL01] "Web Services Conversation Language (WSCL) 1.0". By Arindam Banerji, Claudio Bartolini, Dorothea Beringer, Venkatesh Chopella, Kannan Govindarajan, Alan Karp, Harumi Kuno, Mike Lemon, Gregory Pogossiants, Shamik Sharma, and Scott Williams. Hewlett-Packard Company. May, 2001.

[WSFL01] Frank Leymann, Web Services Flow Language (WSFL 1.0), IBM, May 2001.

[XLANG01] XLANG specification : Web Services for Business Process Design, Satish Thatte, Microsoft, 2001.

[XPDL] Workflow Process Definition Interface - XML Process Definition Language, Version 0.03 (Draft), Workflow Management Coalition WFMC, May 22, 2001.

[Zhan 94] A. Zhang, M. H. Nodine, B. K. Bhargava, and O. A. Bukhres. "Ensuring Relaxed Atomicity for Flexible Transactions in Multidatabase Systems.". In : *SIGMOD Conference*, pp. 67–78, 1994.

[Zism 77] M. D. Zisman. *Representation, Specification and Automation of Office Procedures.* PhD thesis, Wharton School, University of Pennsylvania, 1977.

Résumé

L'objectif de la thèse consiste à proposer une nouvelle approche, appelée "Sphères de comportement" pour la modélisation et l'exécution de procédés flexibles. Il s'agit d'assurer une flexibilité dans la gestion de propriétés comportementales qu'elles soient transactionnelles, opérationnelles ou organisationnelles. De telles propriétés concernent non seulement des activités considérées individuellement (ce que permettent les modèles de procédés actuels), mais également des groupes d'activités d'un procédé.
La première contribution de cette thèse consiste à séparer la spécification des propriétés comportementales de celle du procédé. La solution que nous proposons permet de définir des "sphères de comportement" associant un comportement à un groupe d'activités du procédé. Cela a permis, après une définition des principes fondamentaux d'une telle solution, d'identifier quels sont les comportements dont la représentation en termes de sphères induit une plus grande flexibilité de la modélisation et de l'exécution des procédés métiers.
La seconde contribution consiste en l'étude du cas particulier de la propriété transactionnelle de l'isolation dans les procédés métiers. Cela a abouti à la définition de la notion de "sphère d'isolation" et d'étudier ses caractéristiques en termes de cohésion et de cohérence, donnant lieu à deux nouveaux critères de sérialisabilité.
La troisième contribution de ce travail consiste à déceler les anomalies engendrées par l'exécution d'activités coopératives et la contribution des sphères d'isolation à la gestion flexible de telles anomalies de coopération.
La dernière contribution de ce travail consiste à proposer une démarche permettant la mise en oeuvre des sphères de comportement dans les plateformes de procédés à base de Web Services. La mise en oeuvre a consisté à modéliser des sphères de comportement dans la notation BPMN et à transformer une telle notation en contextes de coordination WS-Coordination couplé au langage BPEL. La finalité d'une telle mise en oeuvre, dite "de bout en bout", est de connecter la modélisation des sphères de comportement (BPMN) à l'exécution du procédé métier (BPEL et WS-Coordination).

Mots-clés: workflow, sphère, comportement, isolation, transaction, travail cooperatif.

Abstract

We aim to introduce a new approach, called "Behavioural Sphere", for flexible business process modelling and execution. One principle of such approach is to provide management flexibility for group-based behavioural properties such as transactional, operational or organisational properties. These properties concerns not only single activities but also groups of activities. Current business process models, setting properties at single activities level, do not allow the expressiveness of group-based behavioural properties.
The first contribution of this thesis proposes a solution allowing us to define "Behavioural Spheres" to make a separation of concerns between process design and group-based behavioural properties definition. We contribute to define correctly the main principles of such solution and we analyse the impact of behavioural spheres on the flexibility of modelling and execution of business processes according to different behaviours.
The second contribution concerns the case study of the isolation property as a particular behaviour in business process management. We define the notion of "Isolation Sphere" and we propose details on its characteristics in terms of cohesion and coherence, which reveal two new serialisability criteria.
The third contribution of this work concerns the study of cooperation phenomena engedered by cooperative activities execution and we detailed how isolation spheres make it possible to manage such phenomena with more an more flexibility.
Finally, the last contribution of this work concerns a proposal of behavioural sphere implentation in web services platforms. This is possible through adaptations made on the modelling part of such platforms, which is represented by the BPMN notation, and an extension of the execution part of web services platforms, which represents the WS-Coordination coordination contexts coupled with BPEL execution language. The implentation proposal covers all the parts of web services platforms, from the modelling to the execution.

Keywords: workflow, sphere, behaviour, isolation, transaction, cooperative work.

Nancy-Université

Université Henri Poincaré

Monsieur **GUABTNI Adnene**

DOCTORAT DE L'UNIVERSITE HENRI POINCARE, NANCY 1

en INFORMATIQUE

VU, APPROUVÉ ET PERMIS D'IMPRIMER N°1366

Nancy, le 29 mai 2007

Le Président de l'Université

UNIVERSITÉ HENRI POINCARÉ NANCY I LE PRÉSIDENT

J.P. FINANCE

Université Henri Poincaré, Nancy I
24-30 rue Lionnois - B.P. 3069 - 54013 NANCY Cédex
Tél. : 03 83 68 20 00 - Fax : 03 83 68 21 00