Conception et adaptation de services techniques pour l'informatique ubiquitaire et nomade

Sylvain Lecomte

Habilitation à Diriger des Recherches

Présentée à
L'Université de Valenciennes et du Hainaut Cambrèsis

par

Sylvain Lecomte

Conception et adaptation de services techniques pour l'informatique ubiquitaire et nomade

Soutenue le 6 décembre 2005 devant le jury :

Président :	Prof. Arnaud Fréville, LAMIH
Rapporteurs :	Prof. Jean-Marc Geib, LIFL
	Prof. Abdelkader Hameurlain, IRIT
	Prof. Michel Riveill, I3S
Examinateurs :	Prof. Frédéric Semet, LAMIH
	Bruno Defude, Maître de conférences HDR, INT Evry

UNIVERSITÉ DE VALENCIENNES ET DU HAINAUT CAMBRESIS
LAMIH – 59310 VALENCIENNES CEDEX
Téléphone (+33) 3 27 51 12 34 – Télécopie (+33) 3 27 51 11 00

Remerciements

En premier lieu, mes remerciements iront au professeur Arnaud Freville, qui a accepté de présider mon jury, et qui a eu l'idée de créer le pôle SID à l'intérieur de l'équipe ROI, en recrutant Didier Donsez. Il a également accepté de diriger les doctorants de SID, en faisant totalement confiance aux jeunes maîtres de conférences qui les co-encadraient. Je tiens également à remercier Didier, qui a lancé l'équipe sur des bases très solides, avant de devoir partir un peu plus au sud. Une bonne partie des résultats affichés ici n'auraient sans doute pas pu être menés à bien, sans ses fortes compétences et sans son insistance à monter des projets de recherche pour faire vivre l'équipe.
J'ai également une pensée émue pour Hafid Bourzoufi, collègue des premiers jours, avec qui je partageais un bon nombre d'idées communes, et qui nous a quitté bien trop tôt.
Un travail comme celui là repose sur des échanges et des projets communs à plusieurs personnes. Je tiens donc à particulièrement remercier Thierry Delot, pour tout le travail et toutes les idées apportées depuis son recrutement en tant que Maître de conférences en 2002, ainsi que pour son soutien avant et pendant la rédaction de ce document.
Je tiens également à remercier les professeurs Patrick Millot et Fredéric Semet, pour la confiance qu'ils m'ont témoigné lorsque j'ai du remplacer Didier dans la gestion du pôle SID.
Je remercie les professeurs Jean Marc Geib, Abdelkader Hameurlain et Michel Riveill de m'avoir fait l'honneur d'accepter de rapporter mon mémoire d'habilitation, aini que pour leur disponibilité, compte tenu des courts délais dont ils disposaient.
Je remercie Bruno Defude, qui me fait le plaisir de participer à mon jury, ainsi que pour la confiance qu'il nous a témoigné à plusieurs reprises.
Ce travail est le bilan de plusieurs années d'encadrement d'étudiants. Aussi, je tiens à remercier Sergiy, Marie et Colombe pour le travail qu'ils ont fourni lors de leurs travaux de doctorat, ainsi que ceux qui viennent de commencer, Hocine et Guy, qui nous ont permis à tous d'avancer dans nos idées. J'ai également une pensée pour tous les DEA encadrés, qui n'ont pas poursuivi leurs travaux en doctorat, et notamment David, pour qui j'aurai toujours le regret de ne pas avoir eu de possibilité de poursuite en thèse.
Finalement, je tiens également à remercier l'ensemble de mes collègues du LAMIH, avec une pensée particulière pour le reste des membres de SID : Nadia, Dorian, Vincent et Marie.

Table des matières

.1 Mise en contexte des travaux présentés **1**
1.1 Avant-Propos . . . 1
1.2 Récapitulatif des enseignements donnés . . . 2
1.3 Charges Administratives . . . 3
1.3.1 Responsabilités pédagogiques . . . 3
1.3.2 Conseils divers . . . 4
1.4 Cursus Recherche . . . 4
1.4.1 Rappel du travail de thèse de doctorat . . . 5
1.4.2 Publications liées . . . 6
1.4.3 Un service transactionnel évolué . . . 7
1.4.3.1 Présentation . . . 7
1.4.3.2 Projets de recherche accompagnants . . . 7
1.4.3.3 Travaux encadrés . . . 8
1.4.3.4 Publications . . . 8
1.4.4 Nécessité d'adapter les services techniques existants aux contraintes des nouveaux terminaux . . . 9
1.4.4.1 Présentation . . . 9
1.4.4.2 Projets de recherche accompagnants . . . 9
1.4.4.3 Travaux encadrés . . . 10
1.4.4.4 Publications . . . 10
1.4.5 Créer des nouveaux services pour l'informatique ubiquitaire . . . 11
1.4.5.1 Présentation . . . 11
1.4.5.2 Projets de recherche accompagnants . . . 12
1.4.5.3 Travaux encadrés . . . 12
1.4.5.4 Publications . . . 12
1.4.6 Autres travaux . . . 13
1.4.6.1 Publications ne figurant pas dans les travaux précédents . . . 13
1.4.6.2 Membre de différents groupes de travail . . . 14
1.4.6.3 Comités de programme et de lecture . . . 14

1.4.6.4 Jurys de thèse et de DEA 15
1.5 Bilan 16
1.5.1 Encadrements 16
1.5.2 Gestion de projets 16
1.5.3 Publications et conférences 17
1.6 Plan de ce mémoire 18

.2 Nouveaux Services transactionnels pour les applications web 19
2.1 Introduction 19
2.2 Besoins des nouvelles applications 20
2.3 Transactions et nouvelles applications 22
2.3.1 Les différents modèles transactionnel 22
2.3.1.1 Les transactions Emboîtées 22
2.3.1.2 Les transactions SAGAS 22
2.3.1.3 Les transactions Emboîtées Ouvertes 23
2.3.1.4 Les transactions à base de Workflow 23
2.3.2 Les solutions dans les plate-formes existantes 24
2.3.2.1 Rappels sur les plate-formes à composants 24
2.3.2.2 Transactions et plate-formes à composants 24
2.3.3 Modèles transactionnels et nouvelles applications 26
2.4 Les ONT pour combler ces manques 28
2.4.1 ONT vs Worflow 28
2.4.2 ONT vs SAGAS 29
2.4.3 Conclusion 29
2.5 Premières spécifications des ONTs 30
2.5.1 Le mécanisme primaire 30
2.5.1.1 Une nouvelle interface pour la validation 30
2.5.1.2 Mécanisme de compensation 31
2.5.2 Les compensations cascadées 33
2.5.2.1 Détermination des transactions à compenser : dépendance entre transactions 33
2.5.2.2 Extensions souhaitables 35
2.5.3 Premier bilan 35
2.6 Politique de cohabitation avec les autres modèles 35
2.6.1 Définition de nouveaux verrous 36
2.6.2 Exemple de fonctionnement de ces nouveaux verrous 37
2.6.3 Gestion des verrous 38
2.7 Formalisation de la proposition 39
2.8 Prototypage et résultats 40
2.8.1 Objectifs 40
2.8.2 Choix du modèle à composants 40
2.8.3 Réalisation 41
2.9 Conclusions 43

.3 L'adaptation des services techniques 45
3.1 Introduction 45
3.2 Définition des services techniques 47
3.2.1 Rôle des services techniques 47
3.2.2 Les différents services techniques 48

3.2.3 Implantation dans les plateformes à composants 48
3.3 Les leçons du service transactionnel 49
3.3.1 Les limites d'une adaptation au coup par coup 49
3.3.2 L'impasse des objets notoires 50
3.3.3 Les possibilités du modèle à composants 51
3.4 L'adaptabilité statique des services techniques 52
3.4.1 Les services techniques à base de composants 53
3.4.1.1 Principe 53
3.4.1.2 Application au modèle Fractal 54
3.4.1.3 Un exemple : l'annuaire 54
3.4.2 La notion de personnalité 55
3.4.3 Adaptation au déploiement 56
3.4.4 Exemple du service transactionnel 58
3.5 Plus de flexibilité : l'adaptabilité dynamique 59
3.5.1 La boucle d'adaptation 59
3.5.2 Une architecture pour l'adaptabilité dynamique 60
3.5.2.1 Le contrat 61
3.5.2.2 Le moniteur 61
3.5.2.3 Le coordinateur 62
3.5.2.4 L'annuaire 62
3.5.3 Interactions 63
3.5.4 Réalisation 63
3.6 Conclusions 66

.4 De nouveaux services pour les applications de proximité 67
4.1 Introduction 67
4.2 Sphères de communication et applications de proximité 68
4.2.1 Contexte 68
4.2.2 Infrastructure réseau 72
4.2.2.1 Présentation des réseaux P2P 72
4.2.2.2 Une architecture pour le HAN 73
4.2.2.3 Mise en oeuvre pour applications de proximité 74
4.3 Fédération de systèmes d'information du HAN 74
4.3.1 Un index pour fédérer les informations 74
4.3.2 Déroulement d'un scénario 76
4.3.3 Recherche d'information 76
4.4 Un service de requêtes dépendantes de la localisation des utilisateurs 79
4.4.1 Déterminer la localisation d'un utilisateur mobile 80
4.4.2 Evaluation de requêtes dépendantes de la localisation 82
4.5 Travaux d'optimisation 84
4.5.1 Solutions envisagées 84
4.5.2 Stratégies d'ordonancement et résultats 85
4.6 Conclusions 88

.5 Conclusion et Perspectives 91
5.1 Conclusion 91
5.1.1 Bilan scientifique 91
5.1.2 Evolution du thème Systèmes d'Information Distribués 92

5.2 Perspectives 93

.6 Bibliographie 95

Table des figures

.2.1 Exemple : application de réservation de voyages "à la carte" 21
.2.2 voyages "à la carte" en 1 transaction 26
.2.3 voyages "à la carte" en 3 transactions 27
.2.4 voyages "à la carte" en utilisant les transactions emboîtées 27
.2.5 Les composants pour les ONTs 31
.2.6 Définition des compensateurs 32
.2.7 Recherche des transactions dépendantes 34
.2.8 Exemple de fonctionnement des verrous ONT-Lock 37
.2.9 Découpage en composants Fractal de notre moniteur transactionnel 42

.3.1 Architecture de diffusion d'information par satellite 46
.3.2 Services Techniques et plateformes à composants 49
.3.3 L'impasse des objets notoires 50
.3.4 positionnement de nos travaux dans ObjectWeb 51
.3.5 Un nouveau modèle de conception pour les services techniques 53
.3.6 Définition d'une personnalité 56
.3.7 Application dans Fractal 57
.3.8 Extrait des P1S (a) transactions plates (b) transactions ONTs 59
.3.9 Les composants de gestion pour l'adaptabilité dynamique 60
.3.10 Architecture du service de localisation de service technique 62
.3.11 création et mise à jour des contrats 64
.3.12 Implantation des composants de gestion 65

.4.1 Notion de sphère de communication 69
.4.2 Les éléments du Home Area Network 70
.4.3 Application de Commerce Electronique de proximité 71
.4.4 Réseaux P2P Hybrides 73
.4.5 Réseaux P2P Hybrides pour application de proximité 75
.4.6 Division du réseau HAN 77
.4.7 Résultats pour une recherche exhaustive 78

.4.8 Résultats pour une recherche limitée 79
.4.9 Résultats 81
.4.10 exemple de Fonctionnement de l'opérateur Inside(étage) 83
.4.11 stratégies d'évaluation 86
.4.12 stratégies d'évaluation (suite) 87

Liste des tableaux

.1.1 Répartition des heures d'enseignement 2
.1.2 Récapitulation encadrements cycles Master et Doctorat 17
.1.3 Récapitulation chiffrée de mes publications 18

.3.1 Services Techniques et Plateformes à composants 47

.4.1 Les services de découverte dynamiques 71
.4.2 Récapitulation de l'utilisation des stratégies 88

Chapitre .1

Mise en contexte des travaux présentés

1.1 Avant-Propos

Le but de cette partie est de présenter rapidement l'ensemble de mes activités, qui ont contribué à constituer ce document d'habilitation à diriger des recherches, depuis la fin de mon doctorat. Je suis actuellement maître de conférences, dans le laboratoire LAMIH [1] à l'université de Valenciennes et du Hainaut Cambrésis depuis septembre 1999. Dans cette équipe, je suis responsable du thème Systèmes d'Information Distribués, qui est une partie de l'équipe ROI. Ce thème comporte actuellement 6 maîtres de conférences. Avant cela, j'ai obtenu en novembre 1998 une thèse de doctorat, à l'université de Lille 1 au sein du laboratoire LIFL [2] dans l'équipe RD2P [3].

Je vais donc commencer par vous présenter mes activités d'enseignement et mes activités de gestion administrative au sein de l'université. En effet, le travail d'enseignant chercheur est constitué de nombreuses tâches très diverses en dehors des activités de recherche (enseignements, responsabilités de cours, responsabilités pédagogiques, charges administratives diverses, suivis de projets), tâches importantes dans la période de réforme des enseignements que nous connaissons (passage du premier, second et troisième cycle, au LMD [4], réforme pour laquelle l'université de Valenciennes a été site pilote). Dans la suite

1. LAMIH : Laboratoire d'Automatique, de Mécanique, et d'Informatique industrielles et Humaines, est une Unité Mixte de Recherche CNRS (UMR 8530)
2. LIFL : Laboratoire d'Informatique fondamentale de Lille
3. RD2P : Recherche et Développement du dossier Portable
4. LMD : Licence, Master, Doctorat

de ce chapitre, je commencerai donc par présenter très brièvement mes activités d'enseignement et administratives, depuis mon arrivée à l'université de Valenciennes, en tant que maître de conférences. Après cela, je présenterai un résumé de mes recherches avec une liste des publications qui en ont découlé. Enfin, je terminerai ce chapitre par un récapitulatif des chiffres clés (publications, encadrements, ...) qui résument mes travaux, et je présenterai le plan des chapitres suivants, qui reviendront en détail sur mes activités de recherche.

1.2 Récapitulatif des enseignements donnés

J'ai commencé à enseigner en 1996, en tant que vacataire à l'université de Lille 3 (TDs et TPs d'algorithmique en DEUG AES). J'ai ensuite été ATER en 1998/1999 au CUEEP [5] (situé sur le campus de l'université de Lille 1), où je suis intervenu en formation continue (Langage C et Système d'exploitation), ainsi que dans le montage d'un nouveau DESS [6] (création de nouveaux cours de DESS sur la sécurité, le commerce électronique, la programmation distribuée et les services transactionnels).

Cette expérience m'a permis d'être très efficace lors de mon arrivée comme maître de conférences à l'université de Valenciennes, où j'ai tout de suite pris en charge une partie du DESS Informatique [7]. A Valenciennes, j'ai pris en charge la responsabilité du cours de Langage à base de machine virtuelle pour le DESS Informatique, du cours d'informatique répartie (CORBA) et du cours de programmation par composant. Dans cette formation, Je suis également intervenu dans les modules de sécurité et de bases de données avancées. Même si les cours en DESS (puis Master 2) occupent la moitié de mon service, j'ai également tenu à effectuer une partie de mes enseignements en 1er cycle. J'ai donc pris en charge les cours de Système d'Exploitation et de programmation Java en DEUST. Malheureusement, les besoins en cycle master sont devenus de plus en plus importants (suite au départ de collègues intervenant dans cette formation), et aujourd'hui, je n'interviens plus qu'en TD de base de données dans le cycle licence.

Globalement, mes enseignements répartis par cycle, à l'université de Valenciennes et du Hainaut Cambrésis, se répartissent de la manière suivante entre 1999 et 2005 (pour les années 1999-2001, Deug / licence sont assimilés au cycle licence du LMD, et maîtrise / DESS, au cycle master, j'ai volontairement comptabilisé les enseignements de DEA à part du cycle master) :

	1999	2000	2001	2002	2003	2004
Cycle Licence	28H Cours 12H TDs 10H TPs	20H Cours 12H TDs 10H TPs	24H Cours 10H TPs	11H Cours 34H TPs	2H Cours 18H TDs	39H TDs
Cycle Master	52H Cours 44H TDs 28H TPs	71H Cours 52H TDs 20H TPs	55H Cours 60H TDs 22H TPs	58H Cours 56H TDs 20H TPs	51H Cours 51H TDs 28H TPs	46H Cours 71H TDs 15H TPs
DEA		6H Cours	10H Cours	10H Cours	13H Cours	16H Cours

TAB. .1.1 – *Répartition des heures d'enseignement*

5. CUEEP : Centre Université- Economie d' Education Permanente
6. DESS Multimédias, Internet et Commerce Electronique
7. il s'agit du DESS TNSI : Technologies Nouvelles des Systèmes d'Information

Le tableau .1.1 montre donc un investissement nettement supérieur dans le cycle master, plutôt que dans le cycle licence. Cela est dû à plusieurs éléments :

- mon recrutement à Valenciennes s'est fait sur un poste créé pour répondre aux besoins correspondant au lancement d'un DESS en compétences complémentaires en Informatique,
- dans le même temps, il y avait de forts besoins dans le DESS Informatique, créé un an plus tôt, avec la nécessité de reprendre les cours laissés vacants par notre regretté collègue Hafid Bourzoufi,
- enfin, lors de mon arrivée, j'ai tout de suite été responsable pédagogique du DESS TNSI, ce qui m'a obligé à m'investir fortement dans cette formation, notamment en reprenant une bonne partie des cours.

Cela a nécessité un investissement important dans la préparation de cours qui n'existaient pas encore, ainsi qu'une remise à jour permanente des enseignements faits dans pour des étudiants extrêmement spécialisés dans un domaine (celui des systèmes d'Information distribués).

1.3 Charges Administratives

Un certain nombre de charges collectives sont à gérer par les enseignants chercheurs. Durant mes 6 années passées en tant que maître de conférences à l'université de Valenciennes, j'ai assumé un certain nombre de ces tâches, que je vais classifier ici en 2 catégories : les tâches d'ordres pédagogiques (en lien avec les étudiants), et les tâches structurelles (qui permettent à un institut ou une université, de fonctionner normalement). Dans cette partie, je ne parlerai pas de la gestion des projets de recherche, qui sera décrite plus tard.

1.3.1 Responsabilités pédagogiques

J'ai successivement occupé trois responsabilités pédagogiques différentes au cours des six dernières années :

- J'ai tout d'abord été *responsable des stages DESS Informatique* (1999-2001) : durant cette période, je me suis occupé de l'affectation/suivi des projets individuels de DESS informatique, ainsi que du suivi des stages. La promotion de DESS informatique comportait 20 à 25 étudiants
- Puis, j'ai été *Co-responsable pédagogique DESS Informatique* (2001-2002 et 2003-2004) : durant ces 2 années, je me suis occupé avec Nadia Bennani, de la responsabilité du DESS (puis Master 2 informatique. Le travail consistait à s'occuper de l'emploi du temps, des projets individuels et de groupe, des sessions d'examens, des jurys, etc... A partir de 2001, les stages n'étaient plus gérés par les responsables de la formation, mais par 2 enseignants chercheurs qui géraient les stages pour toutes les formations de la filière informatique.
- Enfin, je suis actuellement *Co-responsable Master 1* (2004-2005) : je me suis occupé du second semestre du master 1 informatique. Cette tâche consiste tout d'abord à gérer les choix des options des étudiants, faire les emplois du temps, gérer les sessions d'examens et les jurys, faire les déclarations de service des enseignants de la formation.

Cette période a été également marquée par la réforme LMD. L'université de Valenciennes et du Hainaut Cambrésis étant une université pilote pour cette réforme. Il a alors fallu faire un certain nombre de dossiers d'habilitation pour nos diplômes, dès l'été 2002, pour les mettre en conformité avec les nouvelles directives d'harmonisation des diplômes au

niveau européen. Ces différentes tâches sont nécessaires et intéressantes par le lien qu'elles génèrent avec les étudiants, mais également par la remise en question du contenu de nos enseignements, ce qui a permis, à la fois une mise à jour, mais également une modernisation des parcours. Cette réforme globalement positive comporte cependant quelques effets induits :

- Multiplication des choix d'options : avec la réforme LMD, les étudiants peuvent choisir un certain nombre de modules, qui leur permettent de modeler leur diplôme à leur convenance. Cela permet aux étudiants de se spécialiser dans leur domaine de prédilection, et d'éviter certaines matières pourtant fondamentales, qui forment la base thèorique des enseignements.
- Multiplication des sessions d'examens : la semestrialisation (qui fait également partie de cette réforme), oblige à doubler un certain nombre de session d'examens (rattrapages à chaque semestre). Cela induit un enchaînement rapide à chaque fin de semestre de la première session d'examens, du 1er jury, de la seconde session et du second jury.
- Mise en place de cours en commun entre plusieurs formations : les modules d'options doivent être accessibles à plusieurs formations. Ce qui rend très complexe la réalisation des emplois du temps

Même si ce mémoire à pour vocation à faire ressortir les travaux d'encadrement de la recherche depuis ma nomination au poste de maître de conférences en 1999, je trouvais important de rappeler que notre métier comportait également deux autres volets : l'enseignement et les charges administratives, dont il est important que chacun en prenne une part, de manière à ce qu'elles restent acceptables pour tous.

1.3.2 Conseils divers

Il est également important pour un enseignant chercheur de participer aux instances de gestion de la vie universitaire. Cela permet de mieux en comprendre le fonctionnement. Dans ce cadre, je suis:

- Membre élu du conseil d'administration de l'ISTV[8] depuis 2003. Ce conseil se réunit trois ou quatre fois par an pour déterminer les grandes orientations de l'Institut, voter le budget, faire les choix sur les postes enseignants ou enseignants chercheurs à créer ou à redéployer.
- Membre de la CSE 27ième section de Valenciennes. Je suis vice-président maître de conférences de cette CSE depuis 2001.

1.4 Cursus Recherche

Je vais maintenant présenter les axes principaux de ma recherche. Les travaux détaillés seront présentés dans les chapitres suivants, mais dans cette section, je vais brièvement présenter comment s'articulent entre elles les différentes voies suivies en recherche. J'anime actuellement le thème SID (Systèmes d'Information Distribués) de l'équipe ROI (Recherche Opérationnelle et Informatique) du LAMIH. Ce thème a pour principaux buts l'étude de l'impact de l'arrivée de nouveaux terminaux (de type smartphones, assistants personnels, etc...) et de nouveaux usages (distribution de l'information, mobilité des utilisateurs, nouvelles applications Web) sur les systèmes d'information existants. Pour présenter cela, je

8. ISTV : Institut des Sciences et Technologies de Valenciennes

vais commencer par rappeler mes travaux de thèse de doctorat, puis je présenterai le cheminement suivi au travers des différentes thèses encadrées au cours de ces 6 années. .

1.4.1 Rappel du travail de thèse de doctorat

j'ai obtenu en novembre 1998 une thèse de doctorat, à l'université de Lille 1 au sein du laboratoire LIFL. Cette thèse, dont le titre est "COST-STIC : Cartes Orientées Services transactionnels et Systèmes Transactionnels Intégrants des cartes" à été soutenue devant le jury suivant :

- Pr Jean-Marc Geib du LIFL (président)
- Pr Michel Riveill du I3S (rapporteur)
- Pr Jean Férié du LIRMM (rapporteur)
- Didier Donsez du LAMIH (puis IMAG)
- Pr Simone Sedillot de l'INRIA Rocquencourt
- Pr Philippe Pucheral de l'INRIA / PRISM
- Pr Vincent Cordonnier du LIFL (directeur)

Elle a été financée par une bourse BDI CNRS / entreprise, avec l'entreprise Gemplus Card international.

Avant cela, j'ai obtenu un diplôme de DEA (discipline Informatique) à l'université de Lille, et un diplôme d'ingénieur, en informatique industrielle à HEI[9].

Durant mes travaux de doctorat, j'ai travaillé sur le portage des mécanismes transactionnels dans les cartes à microprocesseur. En effet, la carte à microprocesseur fait partie des terminaux qui se développent énormément, et qui peuvent dorénavant prendre part à des applications complexes, et distribuées. Ainsi, le développement des applications pour carte, dans un milieu distribué et très sujet aux pannes, se révèle très complexe. Le modèle transactionnel, de par ses propriétés (Atomicité, Cohérence, Isolation et Durabilité) représentait une bonne solution au traitement de ces problèmes.

L'implantation de ce modèle dans les cartes à microprocesseur a imposé l'adaptation de deux mécanismes (la reprise sur panne, et le contrôle de concurrence sur les données) bien connus des systèmes transactionnels classiques. Pour cela, il a notamment fallu tenir compte de la faible taille mémoire disponible et du processeur peu puissant.

L'utilisation de cartes gérant les mécanismes cités précédemment dans des systèmes distribués s'effectue grâce à une architecture, utilisant un service de validation distribuée (OTS pour CORBA). Un composant d'adaptation prend en charge les contraintes de la carte (notamment en terme de protocole de communication), de manière à ne pas modifier l'architecture des systèmes existants.

Un des points importants de ce travail, est qu'il est à l'intersection de trois domaines de la recherche en informatique :

- L'informatique embarquée : le fait de travailler sur les cartes à microprocesseur m'a appris à prendre en compte les contraintes des systèmes embarqués (faible CPU et mémoire, problèmes l'autonomie, IHM restreinte)

9. Hautes Etudes Industrielles (Lille)

- L'informatique répartie : le but étant d'intégrer un nouveau type de terminal dans des systèmes existants, il a fallu travailler sur les bus logiciels CORBA de manière à permettre aux cartes de communiquer avec les services CORBA existants
- Les bases de données : les moniteurs transactionnels concernent une partie importante de la recherche du monde des bases de données.

Ces travaux m'ont donc permis de travailler avec des équipes spécialisées dans ces domaines, et de participer à des groupes de recherche, comme par exemple ORCTrans du PRC I3.

1.4.2 Publications liées

Dans le cadre de ces recherches, j'ai publié un certain nombre d'articles dans des conférences internationales ou nationales avec comité de sélection :

- Conférences internationales avec comité de sélection
  - D. Carlier, S. Lecomte, P. Trane, " Smartcard use to manage user's mobility ", Smart Card Research and Advanced Application Conference (CARDIS) IFIP , Amsterdam, Septembre 1996
  - S. Lecomte et P. Trane, " Failure recovery using action log for smartcards transaction based system ", IEEE On Line Testing Workshop 1997, Crète, Juillet 1997
  - S. Lecomte, D. Donsez, " Intégration d'un gestionnaire de Transaction dans les cartes à microprocesseur ", Nouvelles Techniques de Répartition (NOTERE'97), Pau, France, Novembre 1997
  - D. Carlier, S. Lecomte, P. Trane, " The use of a representation Agent as a solution to Wireless Drawbacks : Application in the context of smartcards ", IEEE ICPWC 97, Bombay, Inde, Décembre 1997
  - Didier Donsez, Gilles Grimaud, Sylvain Lecomte "Recoverable Persistent Memory for Smartcard", Actes de "the 3td Smart Card Research and Advanced Application Conference (CARDIS)" IFIP, UCL Louvain-la-Neuve , Belgique, 14-16 Septembre 1998, Springer-Verlag, LNCS 1820.
  - Sylvain Lecomte, Gilles Grimaud, Didier Donsez, " Transactional Mechanisms for Open Smart Card ", Actes de GDC'99 Gemplus Developer Conference, Paris, CNIT, Juin 1999
  - Sébastien Jean, Didier Donsez, Sylvain Lecomte " Smart cards integration in Distributed Information Systems, A New Interaction Model ", Actes de la conférence IEEE ISADS 2000, Guadalajara, Mexique.
- Conférences nationales avec comité de lecture
  - Sylvain Lecomte et Didier Donsez "Gestion de Transactions pour les Cartes à Microprocesseur", NOTERE 97, Pau, France, Novembre 1997.
  - Sébastien Jean, Didier Donsez, Sylvain Lecomte "Utilisation des bases de données pour la flexibilité de services coopérants dans la carte à microprocesseur", Actes de la conférence INFORSID 2000, 9-13 Mai 2000, Lyon, France,pp 117-129, ISBN2-906855-16-2
- Conférences invité
  - P. Trane, S. Lecomte, "Integrating Smartcard into Transaction Scheme" FTS'97 IEICE, Tokyo, Japon, Juin 1997

1.4.3 Un service transactionnel évolué

Cette recherche a débuté en 1999 avec le projet Européen ITEA PEPiTA. Il s'agit des premiers travaux que j'ai mené en arrivant à l'université de Valenciennes. Le but de ce travail était de réaliser un moniteur transactionnel permettant la gestion des transactions emboîtées ouvertes **[GRA81]**. En effet, ce modèle transactionnel repond parfaitement aux besoins des nouvelles applications de commerce électronique sur le Web.

1.4.3.1 Présentation

Ces dernières années nous avons pu voir le développement rapide des applications distribuées. Cet effet est lié au développement rapide du matériel et des modèles de programmation. Par conséquent, les programmeurs peuvent se permettre de créer des applications plus complexes mettant en oeuvre plusieurs terminaux et serveurs hétérogènes. En même temps, les exigences des utilisateurs sur la qualité des services offerts ont augmenté également.

Le service transactionnel permet de créer des applications distribuées fiables malgré les pannes d'exécution possibles. Par contre, les nouvelles applications ont souvent besoin de transactions spécifiques, notamment des transactions de longue durée. Dans ce cadre, les services transactionnels existants posent de nombreux problèmes. Ainsi, les applications, possédant les bons potentiels, exécutées sur de puissants matériels peuvent être bloquées à cause de l'impuissance des services transactionnels destinés aux applications moins complexes, mais trop longues.

Les modèles transactionnels avancés sont destinés à résoudre le problème de l'utilisation des transactions longues. L'analyse montre, que le modèle des transactions emboîtées ouvertes correspond aux besoins des nouvelles applications distribuées, notamment celles offrant des échanges entre des entreprises. Ce modèle permet de valider les sous-transactions définitivement sans attendre la validation de la transaction-racine. Cependant, ce modèle nécessite un mécanisme de compensation des sous-transactions validées. Ce mécanisme doit gérer les compensations appelées par le gestionnaire transactionnel associé pour garantir la cohérence de la base de données dans le cas d'une panne d'exécution.

Dans ce travail nous avons fourni une extension du service transactionnel utilisé par les plates-formes à composants au modèle des transactions emboîtées ouvertes. Nous avons commencé par la proposition d'un mécanisme des compensations des transactions exécutées par les composants. Puis, nous avons étudié l'influence d'une compensation aux autres transactions exécutées en parallèle ou après cette transaction compensable. A la fin, nous avons proposé un mécanisme permettant la cohabitation non-conflictuelle entre les transactions classiques et les transactions avancées, et plus exactement, entre les transactions plates et les transactions emboîtées ouvertes.

1.4.3.2 Projets de recherche accompagnants

Le projet ITEA PEPiTA (septembre 1999 - Décembre 2001) auquel a participé Bull, Evidian, Alcatel, France Telecom, l'université de Grenoble, l'université de Louvain, l'université de Valenciennes, l'université Charles de Pragues, proposait d'étendre les spécifications des EJB afin de prendre en compte de nouveaux services de persistance, des clients légers et nomades, tels que les décodeurs de télévision numérique et les téléphones cellulaires,

et des modèles de transactions avancées. Ce projet a permis le financement de la thèse de doctorat de Sergiy Nemchenko et de 6 stages de DEA/DESS jusqu'à (Avril-Septembre 2001). Ce projet a été monté (pour l'université de Valenciennes) par Didier Donsez. Après sa mutation à l'université de Grenoble, j'ai assuré la co-gestion de ce projet avec Nadia Bennani. Ce projet à obtenu le prix "Achievement Award 2001" de l'ITEA.

Le projet PEPiTA s'est poursuivi par le projet IMPACT répondant à l'appel à projet RNTL 2001. Ce projet, d'une durée de 18 mois, en collaboration avec les universités de Lille, Grenoble, ainsi qu'avec les partenaires du consortium OBJECTWEB, a notamment permis de continuer le financement de la thèse débutée dans le cadre de PEPiTA, et de terminer les spécifications et les développements débutés avec le premier projet.

1.4.3.3 Travaux encadrés

Lors de ces travaux, j'ai eu l'occasion d'encadrer plusieurs étudiants. Nous pouvons notamment citer Sergiy Nemchenko, qui a été doctorant de Novembre 2000 à Septembre 2004. Son doctorat, qui portait sur "Modèle de transactions avancées et modèle à composants", s'est conclu par une mention très honorable, lors de la soutenance du 13 septembre 2004 devant le jury suivant :

- Rapporteurs : Prof. Abdelkader HAMEURLIN (président), Prof. Guy BERNARD, Bruno DEFUDE
- Examinateur : Prof. Arnaud FREVILLE (directeur), Nadia BENNANI, Didier DONSEZ, Sylvain LECOMTE

Sur ce travail, ont également été pris 2 stagiaires de niveaux bac+5 (Doriane Dusart (DESS informatique) et Daniel Herrera (Master du CINVESTAV à Guadalajara). Leurs travaux ont principalement porté sur le dévloppement des spécifications faites dans le cadre du projet européen PEPiTA (Daniel Herrera) ou du projet RNTL Impact (Doriane Dusart).

Enfin, un ingénieur de recherche (Emmanuel Adam, docteur de l'université de Valenciennes) a été embauché en CDD durant 6 mois, pour finaliser les spécifications et le développement du projet PEPiTA.

1.4.3.4 Publications

- Chapitre d'ouvrage
  -
  - HERAULT C., NEMCHENKO S., LECOMTE S. (2004). A Component-Based Transactional Service, Including Advanced Transactional Models. In Fourth IEEE International Symposium and School on Advance Distributed Systems (ISSADS 04), Mexique, 2004, Revised Selected Papers, publié dans Lecture Notes in Computer Science, Volume 3563 / 2005, ISSN: 0302-9743
- Conférences internationales avec comité de programme
  - NEMCHENKO S. (2003). Compensation management of ONT in component-based models. 7th International Forum "Radio electronics and youth in the XXI century", Kharkiv, Ukraine, avril.
  - BENNANI N., DELOT T., LECOMTE S., NEMCHENKO S., DONSEZ D. (2002). Advanced transactional model for component-based model. ISADS, International Symposium on Advanced Distributed Systems, Guadalajara, Mexique, novembre.

1.4.4 Nécessité d'adapter les services techniques existants aux contraintes des nouveaux terminaux

Un des premiers bilans de mes travaux de thèse (cf. partie 1.4.1) et des travaux sur les transactions emboîtées ouvertes (cf. partie 2.5) est que le service transactionnel, que l'on doit utiliser dans une application distribuée, dépend à la fois du type d'application cible, mais également du type de terminaux participant à cette application. Dans les deux travaux précédents, nous avons, à chaque fois, été obligés d'adapter un moniteur transactionnel, soit pour lui faire prendre en compte des cartes à microprocesseur, soit pour lui faire prendre en compte des applications Web complexes. En 2001, des nouveaux travaux ont donc commencé, sur la nécessité d'adapter des services techniques aux besoins des applications et des capacités des terminaux. Cette recherche a été portée par un projet national RNRT (COMPiTV (2001-2003)) et par un projet du contrat de plan état région (Projet TACT COLORS).

1.4.4.1 Présentation

Un des premiers buts de ces travaux a été de généraliser l'approche qui a consisté à étendre le service transactionnel pour le faire répondre aux besoins des applications et aux contraintes des terminaux. Ce travail n'a donc pas porté sur le service transactionnel, mais sur l'ensemble des services techniques mis à la disposition d'un programmeur d'applications (nommage, courtage, sécurité, transaction, persistance, etc...).

En repartant des conclusions des travaux précédents, ce travail a pour but de permettre l'adaptabilité des services techniques dans les applications distribuées basées sur le modèle à composants. En effet, le modèle à composants fournit une bonne solution, dans le cadre du développement d'applications, aux problèmes de réutilisabilité du code et de gestion des terminaux hétérogènes. Comme nous l'avons vu précédemment, cette hétérogénéité des terminaux créé également le besoin d'adapter des services techniques à leur environnement dynamique d'exécution. Il nous a donc paru intéressant d'utiliser également le modèle de programmation par composants pour la réalisation des services techniques. Pour répondre à ce besoin, nous avons donc redéfini chaque étape du cycle de vie d un service technique : conception, développement, assemblage, déploiement et exécution. Concernant la conception et le développement, nous avons également proposé de construire les services techniques eux-mêmes sous forme de composants Fractal et de donner une véritable représentation dans le système des différentes personnalités d'un service technique. Ces propositions ont été mises en oeuvre dans le service transactionnel proposé précédemment par Sergiy Nemchenko.

Le principal but de ce travail est de rendre les services techniques auto-adaptables. Aussi avons nous défini un ensemble de composants de gestion de l'adaptation : contrats, coordinateur, moniteur et annuaire de services techniques. Ces composants permettent de localiser, choisir et utiliser les nouveaux services techniques.

1.4.4.2 Projets de recherche accompagnants

Le projet COMPiTV [10] partait de la constatation selon laquelle la télévision numérique (satellitaire ou terrestre) sera un des portails privilégiés pour l'accès aux services Web dans les prochaines années. Cependant, la conception de ces services doit prendre en compte la

10. COMPosants pour la iTV, RNRT 2001, budget : 147 kEuros HT

nature du réseau sous-jacent qui est en mode diffusion pour la voie descendante, et en mode point à point pour la voie montante. Dans ce contexte, le projet COMPiTV avait pour but de proposer une plate-forme à composants ayant des services systèmes adaptés au mode diffusion. Ce projet, en partenariat avec l'université de Lille 1, l'université de Grenoble, CANAL+ Technologies et GEMPLUS, a été financé par le ministère de la recherche, dans le cadre des projets RNRT.

L'autre projet, porteur de cette recherche sur l'adaptabilité dans le cadre des plateformes à composants, est le projet du contrat de plan état région TACT COLORS[11]. Ce projet, financé à la fois par la région Nord / Pas de Calais et par les fonds FEDER, avait pour but de coordonner les recherches régionales dans les domaines des logiciels et des plateformes à composants. L'université de Lille 1, l'université du Littoral, l'école des mines de Douai et l'université de Valenciennes participaient à ce projet.

1.4.4.3 Travaux encadrés

Dans le cadre de ces travaux, Colombe Hérault a soutenu une thèse de doctorat, intitulée "Adaptabilité des services techniques dans le modèle à composants", le 23 juin 2005 avec la mention Très Honorable. Ce doctorat a débuté en octobre 2001, et a été en partie financé grâce au projet RNRT COMPiTV. Ses travaux de DEA (menés de septembre 2000 à septembre 2001) ont permis les contacts débouchant sur le projet RNRT, avec les entreprises Gemplus et Canal + Technologies.

J'ai également co-encadré Hocine Grine, qui a fait son DEA (en 2003 / 2004) sur l'étude d'un annuaire permettant de retrouver les services techniques adaptés, en fonction des besoins des applications et des caractéristiques des terminaux. Son travail se poursuit actuellement en thèse.

1.4.4.4 Publications

- Conférences internationales avec comité de programme
  - GRINE H., DELOT T, LECOMTE S. (2005) Adaptive Query Processing in Mobile Environment. In the 3rd International workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC 2005), Grenoble, France.
  - HERAULT C., LECOMTE S., DELOT T. (2004). New technical services using the component model for applications in heterogeneous environment. In Innovative Internet Community Systems (I2CS), Guadalajara, Jalisco, Mexique, juin.
  - HERAULT C., BENNANI N., DELOT T., LECOMTE S., THILLIEZ M. (2002). Adaptability of Non-Functional Services for Component Model, Application to the M-Commerce. In ISADS, International Symposium on Advanced Distributed Systems, Guadalajara, Mexique, novembre, ISBN 970-27-0358-1.
- Conférences nationales avec comité de programme
  - GRINE H., HERAULT C., LECOMTE S., DELOT T. (2005). Localisation de Services Techniques dans un Modèle à Composants. Actes des 4èmes Journées composants (JC), Le Croisic, avril.
  - HERAULT C., LECOMTE S. (2004). Gestion Dynamique des Services Techniques pour Modèle à Composants. In 1ère Conférence Francophone sur le Dé-

11. COmposants LOgiciels Rétutilisables et Sûrs

ploiement et la (Re) Configuration de Logiciels (DECOR), Grenoble, France, octobre, ISBN 2-7261-1276-5.

- HERAULT C., LECOMTE S. (2003). Adaptabilité des services techniques dans un modèle à composants. In 3ème Conférence Française sur les Systèmes d'Exploitation, CFSE, La Colle Sur Loup, France, octobre, ISBN 2-7261-1264-1.
- HERAULT C., LECOMTE S. (2003). Gestion de Personnalités de Services Non-Fonctionnels dans un Modèle à Composants. Workshop "Objets, Composants et Modèles dans l'ingénierie des Systèmes d'Information", en commun avec le congrès INFORSID 2003, Nancy, France, juin.

– En soumission
  - HERAULT C., LECOMTE S. (2005),"Gestion de Personnalités de Services Techniques dans un Modèle à Composants", soumis à la conférence LMO'2006, publiée dans la revue l'Objet.

1.4.5 Créer des nouveaux services pour l'informatique ubiquitaire

Comme nous l'avons vu précédemment, certains services techniques peuvent être adaptés pour répondre aux besoins de nouvelles applications ou aux contraintes de nouveaux terminaux. Cependant, au cours de notre recherche, nous avons également identifié de nouveaux services, qui eux sont totalement nouveaux, car ne correspondant à aucun besoin dans les applications distribuées existantes. Ce travail (débuté en 2001) n'a été porté, jusque maintenant, par aucun projet. Cependant, il est à noter que depuis quelques mois, énormément de contacts ont été pris, et un projet a enfin démarré (projet du contrat de plan état région TAC MOSAIQUES), et d'autres sont, à ce jour, en cours de montage.

1.4.5.1 Présentation

Dans la continuité des travaux portants sur l'évolution des terminaux, nous avons constaté que l'adaptation de services existants, comme présentée dans la partie précédente, ne suffirait pas. Le nomadisme grandissant des utilisateurs et les nouveaux réseaux mobiles et/ou sans fil favorisent le développement de nouveaux services et de nouvelles applications dédiées aux usagers mobiles. Parmi ces applications, nous avons identifié les applications de proximité, qui s'inscrivent dans un contexte fortement distribué. Dans ce type d'application, un utilisateur va, suivant ses déplacements, entrer et sortir de différentes zones de communication (que nous avons nommées "sphères de communication") et va pouvoir participer à différentes applications de proximité avec d'autres utilisateurs physiquement proches de lui. Il doit donc être en mesure de découvrir et/ou de localiser les données et services disponibles. Dans cet environnement mobile et dynamique, les services de localisation actuels sont très limités. Nous avons donc proposé un service complètement nouveau, adapté à la distribution de l'information et qui considère la mobilité des utilisateurs. Il est donc capable d'évaluer des requêtes dépendantes de la localisation (LDQ) telles que « quel est l'arrêt de bus le plus proche de moi? ».

De plus, afin de permettre l'évaluation des requêtes dépendantes de la localisation, ce service doit disposer de la localisation géographique du client qui émet la requête. Aujourd'hui, les techniques de localisation ne sont pas toujours disponibles : par exemple, le GPS qui est la solution la plus répandue, ne fonctionne pas à l'intérieur d'un bâtiment. Nous avons donc proposé une solution de localisation reposant sur les métadonnées de l'environnement. Notre solution permet de localiser un utilisateur de façon approximative mais

largement suffisante pour l'évaluation des requêtes dépendantes de la localisation. Cette solution a été optimisée de façon à minimiser la consommation des ressources sur les terminaux nomades et à réduire le nombre de communications entre les participants.

Ce service n'est pas une adaptation de services existants, comme nous avons pu le faire précédemment avec le service transactionnel, mais bien un service nouveau, qui n'existait pas dans les applications distribuées traditionnelles.

1.4.5.2 Projets de recherche accompagnants

Durant toute la première phase de ce travail, aucun projet n'est venu en support. Maintenant, le projet du nouveau contrat de plan état région 2004-2006 MOSAIQUES[12] vient en support aux suites de ce travail, et notamment en tant que partie du financement du doctorat de Hocine Grine, débuté fin 2004. L'objectif de ce projet MOSAIQUES est de définir un cadre de développement (méthodologie et outillage), commun à l'ensemble des équipes régionales travaillant dans le domaine, pour la définition d'applications fonctionnant dans un environnement ubiquitaire.

1.4.5.3 Travaux encadrés

Ce travail a commencé avec le DEA de Marie Thilliez sur "Commerce électronique de proximité" de septembre 2000 à Juillet 2001. Cela s'est poursuivi par son travail de thèse, qui a été soutenu avec la mention Très Honorable le 3 décembre 2004, devant le jury suivant :

- Rapporteurs : Prof. Christine COLLET, Prof. Philippe PUCHERAL
- Examinateur : Prof. Arnaud FREVILLE (directeur), Gérôme CANALS, Thierry DELOT, Christophe GRANSART, Sylvain LECOMTE

D'autres travaux de DEA ont été menés sur ce sujet, et notamment celui de David Taquet en 2001/2002, sur la Fédération de Systèmes d'Information dans le cadre du Home Area Network, qui traitait des problèmes spécifiques de la recherche d'information dans les réseaux de proximité.

1.4.5.4 Publications

- Revues nationales ou internationales
  - THILLIEZ M., DELOT T., LECOMTE S. (2005). Requêtes dépendantes de la localisation : évaluation distribuée et optimisation, numéro spécial Mobilité dans les systèmes d'information et de bases de données, revue ISI, à paraitre.
  - THILLIEZ M., DELOT T., LECOMTE S. (2005). An Original Positioning Solution to Evaluate Location-Dependent Queries in Wireless Environments. Special Issue on Distributed Data Management in Journal of Digital Information Management.
  - ANCEAUX F., BENNANI N., BRICON-SOUF N., WATBLED L., BEUSCART-ZÉPHIR M., LECOMTE S. (2003). Prise en charge de patients à domicile: un contexte nécessitant l'élaboration de systèmes d'informations coopératifs. Revue des Sciences et Technologies de l'Information, Série Ingénierie des Systèmes d'Information, 8, pp. 75-93.

12. MOdèles et infraStructures pour Applications ubIQUitairES

- Conférences internationales avec comité de programme
  - THILLIEZ M., DELOT T., LECOMTE S., BENNANI N. (2003). Hybrid Peer-To-Peer Model in Proximity Applications. In IEEE Computer Society (Ed.), Proceedings of the 17th International Conference on Advanced Information Networking and Applications, Chine, pp. 306-311, janvier, ISBN 0-7695-1906-7.
  - HERAULT C., BENNANI N., DELOT T., LECOMTE S., THILLIEZ M. (2002). Adaptability of Non-Functional Services for Component Model, Application to the M-Commerce. In ISADS, International Symposium on Advanced Distributed Systems, Guadalajara, Mexique, novembre, ISBN 970-27-0358-1.
- Conférences nationales avec comité de programme
  - THILLIEZ M., DELOT T., LECOMTE S. (2004). A Metadata-Based Positioning Solution to Evaluate Location-Dependent Queries in Wireless Environments. Actes des 20èmes journées de Bases de Données Avancées, Montpellier, octobre.

1.4.6 Autres travaux

Un certains nombres de travaux ne sont pas inclus dans les trois grands axes que j'ai défini précedemment. Il y a tout d'abord les travaux que j'ai mené avec mes anciens collègues de l'équipe RD2P du LIFL. en effet, durant ma thèse, j'ai suivi deux étudiants de DEA (Gilles Grimaud et Sébastien Jean). J'ai donc naturellement continué à travailler avec eux durant une partie de leur doctorat. Les travaux menés alors touchent à l'utilisation des cartes à microprocesseurs dans les applications distribuées, en permettant notamment à ces cartes de recevoir des requêtes asynchrones lors de leur déconnexion.

Il y a aussi d'autres travaux, plus prospectifs, qui ont été menés avec des personnes de l'équipe RAIHM[13] du LAMIH. Ces travaux portaient sur l'étude de l'utilisation des plateformes à agent dans le cadre de nos travaux sur le commerce électronique, et notamment sur le commerce électronique de proximité (qui est un cadre applicatif pour les travaux de thèse de Marie Thilliez).

1.4.6.1 Publications ne figurant pas dans les travaux précédents

- Chapitre d'ouvrage
  - BENNANI N., DONSEZ D., LECOMTE S., RAMOS F. (2002). Apport des agents mobiles dans le commerce électronique. In R. Mandiau, E. Grislin-le Strugeon, A. Péninou (Ed.), Organisation et applications des SMA, Hermès, Paris, pp. 321-331.
- Revues sur invitation
  - BENNANI N., DONSEZ D., LECOMTE S. (2003). Commerce Electronique : Des agents très mobiles. L'Informatique Professionnelle, 211, pp. 39-42.
- Conférences internationales avec comité de programme
  - JEAN S., DONSEZ D., LECOMTE S. (2001). Using some database principles to improve cooperation in multi-application smart cards. IEEE Chilean Computer Society Symposium (CCSS'01), Punta Arenas, Chili, novembre.

13. Raisonnement Automatique et Interactions Homme machine

  - DONSEZ D., JEAN S., LECOMTE S., THOMAS O. (2001). Asynchronous use of smart card services using SOAP and JMS. 3rd Gemplus Developer Conference (GDC'01), Paris, France, juin.
  - BENNANI N., DONSEZ D., LECOMTE S. (2001). Future of digital camera in the Home Entertainment Network. DPP'01, Anvers, Belgique, mai.
  - JEAN S., DONSEZ D., LECOMTE S. (2001). Utilisation des bases de données pour la flexibilité de services coopérants dans la carte à microprocesseur. INFORSID 2000, Lyon, France, mai.
  - JEAN S., DONSEZ D., LECOMTE S. (2000). Smart cards integration in Distributed Information Systems, A New Interaction Model. ISADS'00, Guadalajara, Mexique, février.
- Conférences nationales avec comité de programme
  - THILLIEZ M., DELOT T., LECOMTE S. (2004). A Metadata-Based Positioning Solution to Evaluate Location-Dependent Queries in Wireless Environments. Actes des 20èmes journées de Bases de Données Avancées, Montpellier, octobre.

1.4.6.2 Membre de différents groupes de travail

Le GDR I3 (Information-Interaction-Intelligence, créé en janvier 1998) est le résultat de la fusion des anciens GdR BD3 (Bases de Données), CHM (Communication Homme-Machine) et IA (Intelligence Artificielle). Il a été renouvelé en janvier 2002, lors des assises qui se sont déroulées à Nancy. Depuis sa création, je fais parti de plusieurs groupes de travail liés à ce GDR :

- SGBD Avancés : Il s'agissait du groupe de travail 2.1 du PRC I3; son objectif était d'explorer les problèmes de recherche posés par la conception des noyaux de SGBD avancés d'un point de vue système (transactions, réplication, parallélisme, mobilité),en prenant en compte les besoins issus des nouvelles applications : systèmes coopératifs, internet, commerce électronique. Ce groupe a plus ou moins été remplacé par le groupe GTMOB et par le groupe SPHERE.
- GTMOB : Ce groupe de travail (GT 4.2 du PRC I3, constitué en 2002) s'inscrit dans le mouvement de l'informatique mobile et ubiquitaire. Dans ce domaine en pleine effervescence, la mise en place de méthodes de conception ergonomiques et de modèles d'architecture logicielle (plate-forme logicielle) pour la conception et le développement d'applications, notamment pour l'accès à des bases de données, sur des supports mobiles et/ou à capacité réduite, est cruciale. Ce groupe de travail repose sur des personnes issues de différents GTs pré-existants, notamment le groupe Multimodalité, le groupe SGBD avancés (cité précédemment) et le groupe Collecticiels.
- SPHERE : Ce groupe de travail est très récent, car constitué en 2004. Le but de ce groupe est d'étudier l'impact du développement rapide du réseau Internet sur de nouvelles applications réparties à grande échelle. Ce groupe s'interesse aux thématiques de recherche telles que le Grid Computing, le Peer-to-Peer Computing, et le Mobile Computing.

1.4.6.3 Comités de programme et de lecture

J'ai eu l'occasion d'être relecteur dans deux conférences internationales et une revue internationale :

- MOBIS 2004 : Mobile Information Systems, IFIP TC 8 Working Conference on Mobile

Information Systems (MOBIS), 15-17 September 2004, Oslo, Norway. IFIP International Federation for Information Processing 158 2005, ISBN 0-387-22851-9

- IEEE SMC 04 : International Conference on Systems, Man and Cybernetics, October 10-13 2004 La Hague, Pays Bas.
- Revue International Journal of Computer Systems Science and Engineering IJCSSE (Special Issue on Mobile Databases), Juin 2004.

J'ai été membre du comité de programme de trois conférences internationales, une conférence nationale et d'une revue nationale :

- ISSADS 2002 : International Symposium and School on Advanced Distributed Systems, Guadalajara 2002
- ISSADS 2004 : International Symposium and School on Advanced Distributed Systems, Guadalajara 2004
- Premières Journées Francophones: Mobilité et Ubiquité 2004 : Conférence Française du PRC I3
- IEEE ICPS 2006 : International Conference on Pervasive Services
- Revue Ingénierie des Systèmes d'information : Mobilité dans les systèmes d'information et de bases de données

Et enfin, j'ai été co-président avec le professeur Joelle Coutaz de Grenoble, du comité de programme des secondes Journées Francophones: Mobilité et Ubiquité 2005. Cette conférence francophone s'est déroulée à Grenoble du 31 mai au 3 juin 2005. J'ai effectué avec Joelle Coutaz, la rédaction de l'appel à communication, la sélection des membres du comité de programme, mise en place du site web de soumission des articles, la sélection des articles, la publication des actes. Bien que très prenante, cette tâche a été pour moi, une des plus enrichissante de ces dernières années.

1.4.6.4 Jurys de thèse et de DEA

J'ai à ce jour fait partie de 5 jurys de thèse :

- Gilles Grimaud : Thèse de Doctorat d'Informatique, "CAMILLE : Un système d'exploitation ouvert pour carte à microprocesseur", LIFL, Université de Lille 1, 12 décembre 2000.
  - Président : Prof. Jean Marc GEIB
  - Rapporteurs : Prof. Bertil FOLLIOT, Prof. Daniel HAGIMONT
  - Examinateur : Prof. Vincent CORDONNIER (directeur), Pierre PARADINAS, Sylvain LECOMTE
- Sergiy Nemchenko : Thèse de Doctorat d'Informatique, "Modèle de transactions avancées et modèle à composants", LAMIH, université de Valenciennes et du Hainaut Cambrésis, 13 septembre 2004.
  - Rapporteurs : Prof. Abdelkader HAMEURLIN (président), Prof. Guy BERNARD, Bruno DEFUDE
  - Examinateur : Prof. Arnaud FREVILLE (directeur), Nadia BENNANI, Didier DONSEZ, Sylvain LECOMTE
- Emmanuel Renaux : Thèse de Doctorat d'Informatique, "Démarche à base de composants dirigée par les modèles", LIFL, Université de Lille 1, 7 décembre 2004.
  - Président : Prof. Alain DERYCKE
  - Rapporteurs : Prof. Dominique RIEU et Prof. Eric LEFEBVRE

  - Examinateur : Prof. Jean-Marc GEIB (directeur), Olivier CARON, Sylvain LECOMTE
- Marie Thilliez : Thèse de Doctorat d'Informatique, "Requêtes dépendantes de la localisation en environnements mobiles: expression, évaluation et optimisation", LAMIH, université de Valenciennes et du Hainaut Cambrésis, 3 décembre 2004.
  - Rapporteurs : Prof. Christine COLLET, Prof. Philippe PUCHERAL
  - Examinateur : Prof. Arnaud FREVILLE (directeur), Gérôme CANALS, Thierry DELOT, Christophe GRANSART, Sylvain LECOMTE
- Colombe Hérault : Thèse de Doctorat d'Informatique, "Adaptabilité des services techniques dans le modèle à composants", LAMIH, université de Valenciennes et du Hainaut Cambrésis, 23 juin 2005.
  - Président : Prof. René MANDIAU
  - Rapporteurs : Prof. Laurence DUCHIEN, Prof. Pierre PARADINAS
  - Examinateur : Prof. Arnaud FREVILLE (directeur), Nadia BENNANI, Didier DONSEZ, Sylvain LECOMTE

J'ai également été rapporteur de 5 mémoires de DEA et membre, chaque année, du jury de DEA de l'école doctorale de Valenciennes et du Hainaut Cambrésis.

1.5 Bilan

Je vais ici vous présenter un bilan chiffré de ce qui a été présenté précédemment. Cela bilan reprendra les aspects encadrement d'étudiants, gestion de projets européens, nationaux ou régionaux, et enfin, fera le bilan de mes travaux en terme de publications.

1.5.1 Encadrements

Le tableau .1.2 montre qu'au cours de ces 6 années, j'ai co-encadré 4 thèses de doctorat en Informatique, dont 3 soutenues avec succès et 1 qui a débuté en 2004. J'ai également encadré 4 DEA (ou master 2 recherche), tous obtenus avec succès. 3 de ces 4 étudiants ont poursuivi leurs travaux en doctorat dans notre équipe (dont une thèse financée par une bourse du ministère). Le 4ième (D. Taquet) travaille actuellement dans le service informatique d'une grande entreprise travaillant dans le domaine des transports (Michelin).

Parmis les thèses co-encadrées, Marie Thilliez a été recrutée comme Maître de conférences dans notre laboratoire, Sergiy Nemchenko occupe actuellement un poste important dans une entreprise informatique ukrainienne, et Colombe Hérault est actuellement en post-doctorat à l'université Joseph Fourier de Grenoble.

J'ai également encadré les projets de 34 étudiants de DESS informatique ou Master 2 science et technologie et de 20 étudiants de maitrise ou Master 1.

Je suis titulaire de la prime d'encadrement doctoral et de recherche depuis septembre 2003.

1.5.2 Gestion de projets

Les projets sont importants pour un thème émergeant comme SID dans l'équipe ROI du LAMIH. Lors de mon arrivé en 1999, Didier Donsez (alors responsable du thème) avait

Diplôme	Nom	Années	Co-encadrants
Doctorat	S. Nemchenko	2000-2004	Pr A. Freville N. Bennani
""	C. Hérault	2001-2005	Pr A. Freville N. Bennani
""	M. Thilliez	2001-2004	Pr A. Freville T. Delot
""	H. Grine	2004-	Pr F. Semet T. Delot
Master 2 Recherche (DEA)	M. Thilliez	2000-2001	D. Donsez N. Bennani
""	C. Herault	2000-2001	D. Donsez N. Bennani
""	D. Taquet	2001-2002	T. Delot
""	H. Grine	2003-2004	T. Delot
""	J. Elu	2004-2005	
Master 2 sciences et Technologies	6 étudiants	1999-2000	
""	4 étudiants	2000-2001	
""	7 étudiants	2001-2002	
""	4 étudiants	2002-2003	
""	7 étudiants	2003-2004	
""	6 étudiants	2004-2005	
Master 1	2 étudiants	1999-2000	
""	3 étudiants	2000-2001	
""	3 étudiants	2001-2002	
""	4 étudiants	2002-2003	
""	4 étudiants	2003-2004	
""	4 étudiants	2004-2005	

TAB. .1.2 – *Récapitulation encadrements cycles Master et Doctorat*

participé au montage d'un projet européen ITEA : le projet PEPiTA. (Septembre 1999 / Décembre 2001).

En 2001, a commencé le projet RNRT COMPiTV (Septembre 2001 / Décembre 2003), que j'ai intégralement géré pour le compte de l'université de Valenciennes (montage, gestion administrative et financière, rédactions des documents de fin de projet). Enfin, en Septembre 2004, a commencé le projet du contrat de plan état/région MOSAIQUES, que je gère également en intégralité pour le compte de l'université.

En outre, j'ai également participé aux projets IMPACT (RNTL 2001-2003), RNTS COQUAS (2002-2004) et du contrat de plan état/région COLORS.

1.5.3 Publications et conférences

Le tableau .1.3 récapitule l'ensemble de mes publications acceptées, ou soumises.

Chapitre de livre	2
Revues Internationales avec comité de lecture	1
Revues Nationales avec comité de lecture	2
Autre revue (sur invitation)	1
Revues en soumission	1
Conférences internationales avec comité de programme	19
Conférences nationales avec comité de programme	7
Conférences invités	1
Rapport internes et délivrables de projets	7

TAB. .1.3 – *Récapitulation chiffrée de mes publications*

1.6 Plan de ce mémoire

Dans la suite de ce mémoire, je vais revenir dans le détail sur les trois principaux axes de recherche que j'ai mené depuis ma nomination en tant que maître de conférences à l'université de Valenciennes. Je vais donc commencer par vous détailler les travaux brièvement énoncés dans la partie 1.4.3 sur la réalisation d'un moniteur transactionnel prenant en charge les contraintes des nouvelles applications.

Ces travaux ayant mis en évidence le besoin d'évolution et d'adaptation des services techniques aux nouvelles applications et aux nouveaux terminaux, je détaillerai ensuite dans le chapitre 3 les travaux menés avec Colombe Hérault, sur la définition d'un modèle de conception pour des services techniques adaptables, ainsi que la proposition d'une architecture permettant l'auto-adaptation des services techniques en fonction des besoins applicatifs et des contraintes matérielles. Enfin, le chapitre 4 sera consacré à la présentation de nouveaux services, découlants de la mobilité grandissante des utilisateurs et des terminaux. Puis, je terminerai ce mémoire en traçant les différentes perspectives qui découlent de ces travaux.

Chapitre .2

Nouveaux Services transactionnels pour les applications web

2.1 Introduction

Depuis le début des années 90, deux évolutions majeures en terme d'applications informatiques sont apparues. Tout d'abord, les applications dont le but est d'offrir des services commerciaux sur Internet sont de plus en plus nombreuses **[TSU01]**. Ce type de commerce offre une grande diversité : la vente de marchandises, la mise aux enchères de biens d'occasion, le choix parmi les offres (recherche du prix le moins cher), la consultation de comptes bancaires et l'E-administration (impots, documents à télécharger, etc ...).

Un des exemples qui pose le plus de problème actuellement dans le cadre de ces nouvelles applications, sont les applications exécutées dans le contexte du commerce entre entreprises (B2B pour "Business-to-Business" en anglais). Le principe du B2B sous-entend l'existence de services proposés à des entreprises ou des administrations, et non directement aux particuliers **[MGL02]**. Un exemple classique, que nous allons utiliser dans ce chapitre, est celui du serveur d'une agence de voyages. Une application installée sur le serveur de cette agence va permettre, par exemple, de rechercher les vols correspondants à des critères donnés. Une telle application est connecté avec d'autres serveurs, et notamment les serveurs de réservation des compagnies aériennes, des hotels, ou des loueurs de voitures.

Une autre évolution majeure est l'arrivée de terminaux de plus en plus hétérogènes et de plus en plus nomades. Trente ans seulement se sont écoulés depuis l'arrivée du premier "outil" nomade, qui était le baladeur Sony (le Walkman), et dorénavent, un utilisateur dis-

pose d'un ensemble d'outils informatiques nomades comme les assistants personnels (Palm, PocketPC, ...), les téléphones GSM toujours plus puissants, etc... Cette hétérogénéité créé de gros problèmes pour les développeurs d'applications, qui se retrouvent obligés de gérer plusieurs contextes d'exécution, et plusieurs capacité de traitement, stockage, ou connectivité réseau.

Toutes les applications reposant sur des échanges commerciaux utilisent un service permettant d'assurer un fonctionnement sûr des flux d'argents et de matière : il s'agit du service de gestion des transactions distribuées. Nous avons identifié ce service, comme étant celui qui va devoir le plus s'adapter à l'arrivée à la fois de ces nouveaux terminaux, mais également aux nouvelles applications **[BDL02]**. Une transaction peut faire participer plusieurs serveurs et plusieurs bases de données (BD) gérées par des organisations indépendantes. Ces serveurs doivent se coordonner afin de faire aboutir les transactions au bout d'une période de temps plus ou moins longue. Les transactions peuvent avoir des durées très différentes : de quelques minutes pour les opérations simples, comme l'achat du billet d'avion, à une heure et plus pour l'exemple de la préparation d'un voyage. De même, il est possible d'avoir des transactions durant quelques jours. Ainsi, ces applications posent des problèmes au niveau du traitement transactionnel, notamment car plusieurs entreprises totalement indépendantes peuvent maintenant participer à la même transaction, et ces entreprises ne peuvent pas admettre de voir certaines de leurs données affectées (ou vérouillées) de part des blocages imputables à un partenaire.

Dans la suite de ce chapitre, je vais présenter les travaux menés sur l'intégration du modèle transactionnel emboîté ouvert dans les moniteur transactionnel fournis par les plate-formes à composants. Ces travaux ont été menés dans le cadre de la thèse de Sergiy Nemchenko **[NEM04]** que j'ai dirigé avec Nadia Bennani. Je vais dans un premier temps détailler les besoins rencontrés par les nouvelles applications de commerce sur le Web, puis je reviendrai sur le choix du modèle transactionnel à implanter, en comparant tous les modèles existants. Après cela, je vous présenterai la solution retenue pour la spécification et la réalisation de ce moniteur transactionnel, ainsi que la cohabitation de ce nouveau modèle avec les solutions transactionnelles existantes.

2.2 Besoins des nouvelles applications

Comme nous l'avons vu dans l'introduction, dans le contexte des applications B2B, une transaction commerciale sur le Web peut engager plusieurs services Web gérés par différentes entreprises. Ces derniers doivent d'une part dialoguer pour négocier les biens et les services à acheter ou à vendre et, d'autre part, se coordonner pour faire aboutir la transaction au bout d'une période de temps plus ou moins longue. Prenons l'exemple d'une application de réservation de voyages "à la carte". Cette application permet de préparer les voyages des utilisateurs. Un voyage est composé d'étapes différentes, comme par exemple, les vols en avion, les séjours dans les hôtels, les locations de voitures sur place, etc. Un utilisateur prépare son voyage en réservant les billets, les chambres d'hôtel, etc. De plus, il doit avoir la possibilité de coordonner la préparation de chaque étape avec la préparation des autres étapes. Par exemple, s'il n'y a pas des billets d'avion à la date demandée, alors il faut décaler la réservation de l'hôtel si elle était déjà faite.

La figure .2.1 montre que dans notre exemple, la logique métier de l'application et une base de données de cette agence de voyages se trouvent sur le serveur où se connecte l'uti-

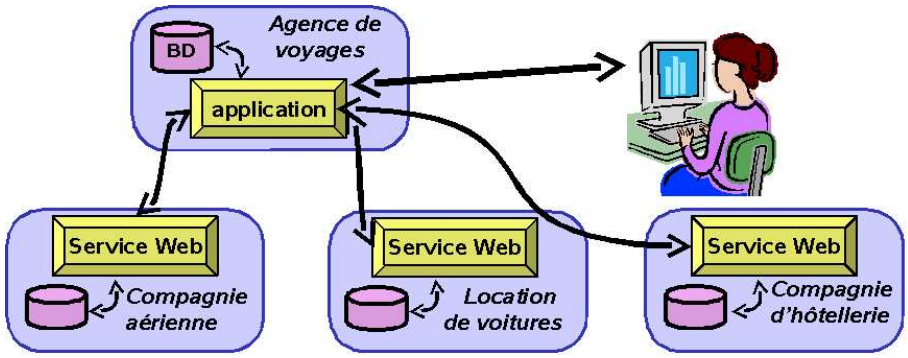

FIG. .2.1 – *Exemple : application de réservation de voyages "à la carte"*

lisateur. Il s'agit de l'applicatif déployé dans l'agence de voyage. Cette application offre l'interface d'utilisation et, en fonction de la demande de l'utilisateur, interroge les ressources situées sur les serveurs des compagnies correspondant aux besoins de l'utilisateur. Dans notre exemple trois serveurs sont montrés : les serveurs de l'hôtel, de la compagnie aérienne, et du loueur de voiture. Chaque serveur possède une (ou plusieurs) application(s) installée(s) sur ce serveur et permet de consulter les bases de données correspondantes. Ces applications sont destinées à l'utilisation par les autres applications et/ou par les utilisateurs. Dans les systèmes modernes, les applications distribuées utilisées sur Internet par les autres applications sont souvent représentées par des services-web.

Ce type d'applications pose un certain nombres de problèmes (sécurité, gestion de la distribution, répartition de charge, etc..). Le problème auquel nous nous sommes attaqués concerne la gestion de la cohérence des données sur les différents serveurs et la reprise sur panne, au travers de la gestion de transactions distribuées. Contrairement au modèle transactionnel traditionnel (OnLine Transaction Processing : OLTP) gérant et coordonnant des transactions de très courte durée et au sein d'une seule et même entreprise, les transactions commerciales au travers du Web, que nous décrivons ici au travers de cet exemple, impliquent des services Web situés sur différents sites. Ces différents sites sont reliés au travers du réseau Internet, et ont une confiance limitée les uns envers les autres. La coordination de transactions longues entre plusieurs sites distincts requiert donc de nouveaux mécanismes, car il est, par exemple, difficile d'avoir un coordinateur unique du protocole de validation à deux phases, et en outre les mécanismes courant de gestion de transactions risquent de bloquer des données sur plusieurs serveurs durant de longs moments.

Comme nous l'avons dit dans l'introduction, d'autres critères sont à prendre également en compte dans les évolutions de l'informatique à la fin des années 90. Le principal est l'arrivée de terminaux de plus en plus hétérogènes, ayant des caractéristiques très différentes, engendrant la nécessité de commercialiser plusieurs versions d'une même application. Pour cela, le modèle à composant **[HC01]** a permis de répondre à ce besoin de modularité des applications, en permettant de créer facilement de nouvelles versions des applications, en réutilisant au maximum le code existant. Il est donc important de conserver cette flexibilité dans l'écriture de ces applications, et donc d'intégrer notre solution aux nouveaux modèles de programmation existants : les plate-formes à composants. Nos travaux sur l'utilisation de modèles transactionnels avancés pour répondre aux besoins des nouvelles applications s'est donc déroulé uniquement dans le cadre d'une utilisation avec ces plate-formes.

2.3 Transactions et nouvelles applications

Depuis plus de 20 ans, le modèle transactionnel représente une bonne solution aux problèmes de gestion des pannes dans des environnements très distribués. Pour rappel, basiquement, une transaction (appelée transaction plate) peut être définie comme étant une suite d'actions commençant par un «Début_Transaction» (Begin_Transaction en anglais), et se terminant par un «Validation_Transaction» (Commit_Transaction en Anglais), si tout se passe bien, ou par un «Annulation_Transaction» (Rollback ou Abort en Anglais), s'il y a eu un problème.

Le modèle transactionnel définit un modèle d'exécution pour applications concurrentes, qui garantissent un certain nombre de propriétés d'exécution (appelées propriétés ACID [1] **[HR83]**). Dans cette partie, nous allons présenter les différents modèles transactionnels avancés existants dans la littérature (transactions emboîtées, Sagas, emboîtées ouvertes et à base de Worflow). Nous terminerons cette section en présentant les modèles intégrés actuellement dans les plate-formes à composants commerciales.

2.3.1 Les différents modèles transactionnel

2.3.1.1 Les transactions Emboîtées

Les transactions emboîtées [2] **[MOS85]** introduisent la notion de sous-transaction. Tout comme les transactions plates, elles assurent les propriétés ACID pour la transaction globale, mais seules les propriétés d'atomicité et d'isolation sont conservées pour les sous-transactions.

Il est important de noter que l'abandon d'une transaction fille n'entraîne pas obligatoirement l'abandon de la transaction mère. La transaction mère décide de la stratégie à adopter (refaire les transactions filles échouées, ignorer leur échec, ou abandonner la transaction au moindre problème). Ainsi, on peut réaliser un contrôle modulaire de l'exécution.

Ce modèle, bien qu'il permet d'augmenter le parallélisme d'exécution en relâchant un certain nombre de propriétés au niveau des sous-transactions, n'est pas suffisamment "ouvert" pour répondre aux problèmes soulevés précédemment, car il n'évite pas le verrouillage des données sur l'ensemble des serveurs pendant l'exécution de la transaction globale.

2.3.1.2 Les transactions SAGAS

Le modèle SAGAS **[GS87]**, et ses différentes extensions **[DVD01]** reprennent comme modèle d'exécution de base celui des transactions plates. La transaction globale (appelée SAGA) est divisée en un certain nombre de sous transactions, toutes au même niveau (pas d'imbrication). Ce modèle permet de se libérer de la propriété d'Isolation. Il assure donc uniquement les propriétés d'atomicité, de cohérence et de durabilité pour la transaction globale, et les propriétés d'atomicité, d'isolation et de durabilité pour les sous-transactions. La validation d'une sous-transaction n'est plus conditionnée par la validation de la transaction mère.

A l'inverse, la validation de la transaction racine est toujours conditionnée par la validation de toutes ses sous-transactions. Ainsi, si une sous-transaction est abandonnée, il faut

1. ACID : Atomicité, Cohérence, Isolation et Durabilité
2. Nested Transaction en anglais

abandonner la transaction mère, et donc défaire toutes les sous-transactions déjà validées.

Pour défaire une sous-transaction, il faut définir des *actions de compensation*[3] **[BGM95]**). Cela implique que pour chaque sous-transaction, il doit exister une transaction de compensation correspondante (ce qui n'est pas toujours possible, lorsqu par exemple, l'exécution de la transaction, et l'affichage des informations en découlant, ont provoqué d'autres décisions).

Ce modèle a été conçu pour répondre à la problématique des transactions longues, qui risqueraient de ne jamais être validée si on maintenait les contraintes d'isolation, compte-tenu des nombreux conflits avec les autres transactions, et les risques d'interblocage qui en découlent. Pour cela, ce modèle permet de relâcher des variables qui peuvent servir à d'autres sous-transactions concurrentes. En ce sens, il est intéressant dans notre problématique.

2.3.1.3 Les transactions Emboîtées Ouvertes

Le modèle des transactions emboîtées ouvertes (ONT pour "Open Nested Transaction" en anglais) **[GRA81]** a pour principe de relâcher la contrainte d'Isolation, comme pour les SAGAS, et de gérer les transactions emboîtées. Cette particularité du modèle des ONT permet d'augmenter fortement le parallélisme entre les transactions ; en effet, les ressources sont relâchées dès qu'une sous-transaction est validée et les transactions en attente sont ainsi bloquées moins longtemps. Cet avantage est particulièrement intéressant lorsque la durée d'une transaction est importante.

La Deuxième conséquence du relâchement de la propriété d'isolation est la simplification de la validation de la transaction-racine. Il y a moins d'opérations à valider à la fin de la transaction-racine. Cet effet est lié au fait que toutes les opérations exécutées au sein des sous-transactions ONT sont validées pendant les validations de ces sous-transactions.

Toutefois, comme pour les SAGAS, la compensation de ces transactions peut se révéler très problématique **[KLS90] [HW91], [AVA94]**, et a été, jusque maintenant, un frein à son implantation dans les moniteurs transactionnels existants.

2.3.1.4 Les transactions à base de Workflow

Ce modèle a lui aussi été conçu pour répondre aux problèmes liés aux transactions de longue durée **[BCF97]**. Il est basé sur la construction et l'exécution d'un scénario, fixant les enchaînements des transactions. Cela permet de relâcher les propriétés d'Atomicité et d'Isolation de la transaction globale, tout en maintenant la cohérence sémantique de l'application.

Le scénario est donc basé sur des dépendances entre validation et abandon des différentes transaction **[DDS97]**. Il peut être, par exemple, du type suivant :

```
Si la Transaction T1 est validée
   Alors
     Exécuter la Transaction T2
   Sinon
     Exécuter la Transaction T3
```

3. On appelle actions de compensation, des actions permettant d'annuler une transaction après sa validation

Le modèle à flots de tâches a comme objectifs de :
- Permettre l'arrêt volontaire d'une exécution, et sa reprise, dans un délai non fixé.
- Permettre une visibilité des résultats avant la fin de la transaction globale (ce qui peut nécessiter là aussi des actions de compensation).
- Définir un modèle de reprise permettant de redémarrer l'exécution après une panne (sans avoir annulé les premiers traitements).
- Contrôler et synchroniser l'exécution de la transaction.
- Spécifier les actions à exécuter en cas de conflit ou d'échec.

2.3.2 Les solutions dans les plate-formes existantes

2.3.2.1 Rappels sur les plate-formes à composants

Comme nous l'avons dit précédemment, l'hétérogénéité toujours plus grande des terminaux, et le besoin de réutilisabilité du code grandissant poussent vers l'utilisation du modèle de programmation par composant (Enterprise Java Beans, Microsoft .NET, ou CORBA Component model). Même si il ne s'agit pas ici de faire un état de l'art complet de la programmation par composants, il me semble important de résumer certains concepts.

La définition usuelle dit qu'un composant est un module logiciel autonome et intelligent qui peut fonctionner sur des réseaux différents, sous des systèmes d'exploitation différents et avec diverses palettes d'outils **[OHE99]**. Il est décrit par une (ou plusieurs) interface(s), par les propriétés configurables et les contraintes techniques. Pour être utilisé, un composant présente des interfaces qui décrivent ses fonctionnalités et il utilise les interfaces d'un ou plusieurs composants (c'est ce que l'on nomme "assemblage" de composants. Chaque "métier" définit ses propres composants : par exemple un "banquier" créera les composants liés au paiement. Cette solution permet de mieux répartir les tâches : chaque "communauté" écrit la partie logicielle qu'elle connaît parfaitement. Les applications finales sont alors obtenues par simple assemblage de composants existants.

Pour s'exécuter, un composant doit être déployé sur un serveur d'application. Il est alors pris en charge par une structure d'accueil appelée "conteneur". Le déploiement de l'application consiste en fait à paramétrer et à configurer, pour un assemblage de composants, les sources de données, la distribution des tâches sur une plate-forme matérielle et la politique de services techniques. Ce conteneur fournit au composant tous les services nécessaires aux exécutions correctes. Un client sollicite un composant via son interface. Un composant peut lui aussi utiliser un ou plusieurs autres composants via ses interfaces, situés (ou non) sur le même serveur.

Une des choses importantes en ce qui concerne nos travaux, est que le composant n'implante que le code métier autrement dit le code fonctionnel. Chaque type de composant se trouve dans un conteneur. Un conteneur prend en charge les services techniques (aussi appelés "non fonctionnels"), comme les transactions, la persistance, la sécurité, etc. Ces services sont configurés et paramétrés lors du déploiement du composant.

2.3.2.2 Transactions et plate-formes à composants

Le modèle de programmation par composants, de part la modularité qu'il propose et la réutilisabilité du code qu'il offre, répond bien aux besoins de développement des

nouvelles applications. Il est donc important de voir comment ces modèles transactionnels sont actuellement intégrés :

- Les Enterprises Java Beans: Il existe de nombreuses implémentations de la spécification EJB comme WebSphere de IBM ou JOnAS **[JON01]** de ObjectWeb. Le modèle EJB repose intégralement sur les technologies SUN (RMI/GIOP pour le transport de requête), mais emprunte également certains services au monde CORBA (le service transactionnel JTS **[SUN99]**, est en fait une adaptation au monde JAVA du service transactionnel de l'OMG dans sa version 1.2). Ces serveurs applicatifs implantent le modèle des transactions plates (par respect des standards SUN),et pour certains les transactions emboitées.
- CORBA Component Model: Les spécifications CCM pour CORBA Component Model sont très récentes et offrent le modèle de composants le plus riche. La plateforme CCM repose sur la technologie CORBA, tant pour le transport des requêtes (IIOP/GIOP), que pour les différents services (Nommage, Transaction, sécurité, etc...). Les modèles transactionnels utilisables avec cette plate-forme sont donc ceux du service transactionnel CORBA **[OMG03]**.
- .NET : La plate-forme .NET est actuellement la plate-forme qui se développe énormément dans le domaine industriel. Cette plate-forme, comme les deux précédentes, propose de nombreux services. en ce qui concerne les transactions, en mode automatique (c'est à dire avec les descripteurs de déploiement), on peut gérer des transactions plates. Cette plateforme est très orientée "Web Services" **[TL02]**, et utilise donc es toutes dernières approches en terme de transport de requêtes (avec SOAP **[BEK00]**), de localisation de services (avec UDDI/WSDL **[MER01], [CCM01]**). Il en est a peu près de même en ce qui concerne la gestion de transaction, puisque .NET supporte, en manuel (c'est à dire, en incluant le code de contrôle des transactions dans le code applicatif), la gestion des transactions emboîtées traditionnelles, voir des compensations de transaction déja validées (sans cependant supporter réellement des modèles comme les ONTs ou les Sagas).

De plus, plusieurs spécifications décrivant la gestion des transactions avancées ont été créées ces derniers temps. Trois spécifications parmi elles sont plus remarquables :

- L'OMG (Object Managment Group) a proposé des mécanismes additionnels pour la spécification OTS basés sur les Activity Services **[OMG02]**. Les mécanismes additionnels pour la spécification OTS ont pour but de permettre l'utilisation de transactions avancées dans l'environnement OTS.
- BEA, IBM et Microsoft ont créé la spécification WS-Transaction (Web Services Transaction) **[CCC02]**. La spécification WS-Transaction prévoit deux protocoles de gestion des transactions. Le premier protocole gère les transactions atomiques (les transactions classiques de courte durée). Le second protocole correspond aux transactions business. La spécification prévoit ce type de protocole pour les transactions de longue durée et supporte la relaxation de la propriété d'isolation (nous sommes donc très proche des ONT). Le mécanisme de gestion des exceptions comporte non seulement des moyens d'abandon mais aussi de compensation.
- OASIS (Organization for the Advancement of Structured Information Standards) propose un protocole BTP (Business Transaction Protocol) **[BTP02]**. Les objectifs du protocole BTP sont identiques aux transactions-business de WS-Transaction, par contre les solutions sont différentes.

2.3.3 Modèles transactionnels et nouvelles applications

Même si il est acquis que l'utilisation de services transactionnels est nécessaire pour la création d'applications distribuées fiables **[CCC02]**, pour gérer une transaction correspondant à la réservation d'un voyage à la carte (exemple de la figure .2.1), nous avons trois possibilités de gestion différentes (une transaction globale, une transaction par serveur, ou une transaction globale divisée en sous-transactions pour les serveurs). Cependant, si on tient compte du fait que la plupart des serveur applicatifs à base de composants actuels ne savent gérer que des transactions plates, il ne reste que deux possibilités.

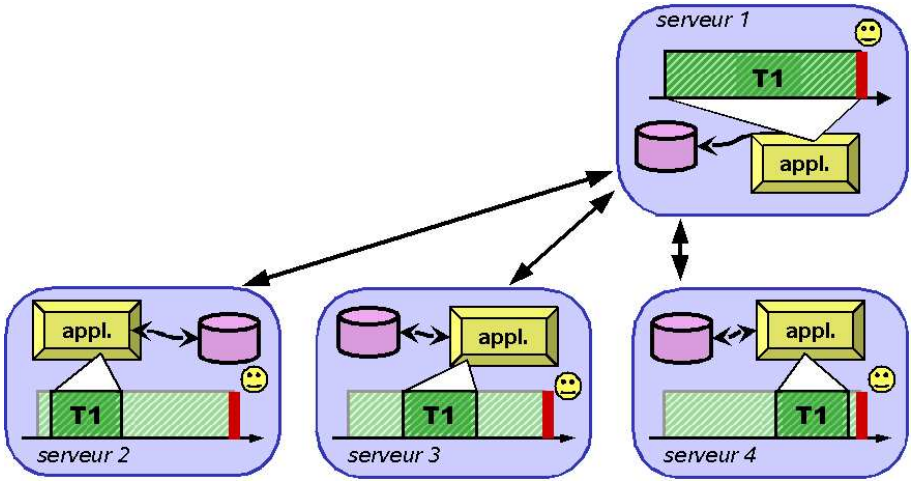

FIG. .2.2 – *voyages "à la carte" en 1 transaction*

La première possibilité est de faire une transaction globale pour les 3 sites, comme indiqué par la figure .2.2. Dans ce cas, la transaction est lancée par le serveur de l'agence de voyages. Toutes les opérations sur le serveur 1 et les opérations locales sur les serveurs des compagnies (les serveurs 2, 3 et 4) entrent dans cette transaction. Bien que fonctionnelle, cette solution a le très gros désavantage de bloquer les données de TOUS les serveurs pendant toute la durée de la transaction de réservation de voyages. De plus, l'échec de la réservation sur un des serveurs peut engendrer l'échec complet de la transaction.

La seconde solution est (toujours en utilisant uniquement des transactions plates), de lancer une transaction par serveur. Dans cette solution (figure .2.3) il n'y a pas de transaction incorporant toutes les opérations de la préparation du voyage. Par contre, il existe des transactions indépendantes (les transactions T1, T2 et T3) correspondant aux étapes de cette préparation. Dans ce cas, chaque étape est exécutée au sein d'une transaction plate indépendante. L'échec d'une étape ne provoque plus l'annulation de toutes les opérations exécutées au sein de cette préparation. Par contre, maintenant, le contrôle de l'intégrité de l'opération globale est perdue car les étapes sont absolument indépendantes et il n'y a plus de liaisons entre elles.

La figure .2.4, présente une solution basée sur l'utilisation des transactions emboîtées. Cela résout le problème d'absence de gestion des échecs partiels. En effet, dans ce cas, une étape peut être abandonnée sans avoir d'influence sur les autres étapes. En même temps, si l'utilisateur décide d'annuler tout le voyage, il suffit d'abandonner la transaction-racine.

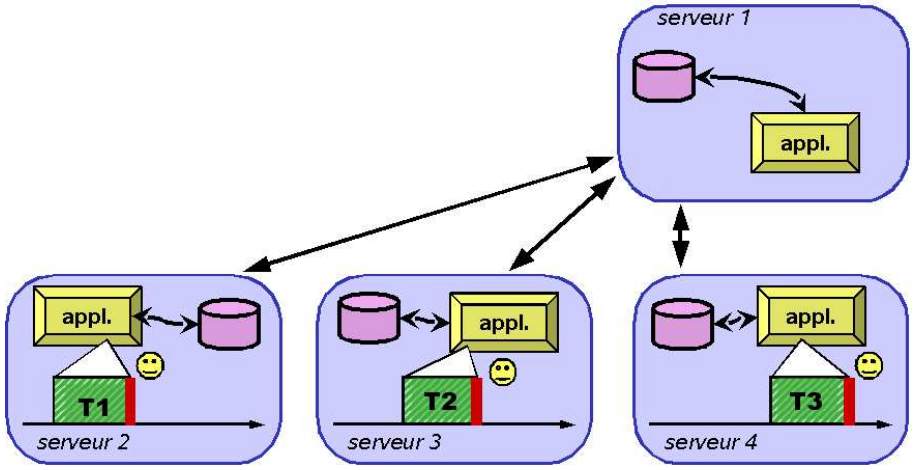

FIG. .2.3 – *voyages "à la carte" en 3 transactions*

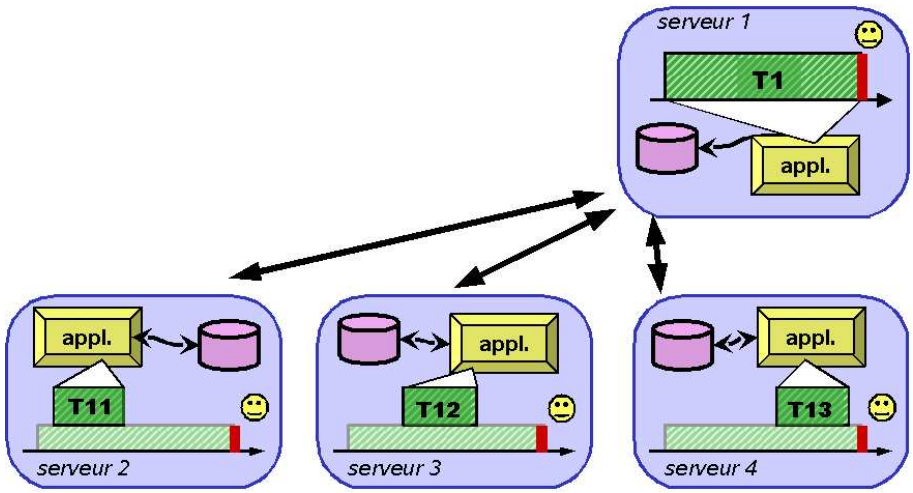

FIG. .2.4 – *voyages "à la carte" en utilisant les transactions emboîtées*

Par contre, il existe un deuxième problème qui est valable pour les transactions plates ainsi que pour les transactions emboîtées. C'est le blocage long des données. Par exemple, les billets d'avion sont réservés au tout début de la transaction, mais cette étape n'est pas validée définitivement. Par conséquent, le serveur de la compagnie aérienne verrouille les données qui correspondent à la requête de l'utilisateur. Ce processus peut durer assez longtemps. Par conséquent, les conflits et les blocages mutuels entre les transactions sont possibles.

Les modèles transactionnels "simples", implantés actuellement dans les moniteurs transactionnels utilisés dans les plate-formes à composants ne sont donc pas capables de gérer efficacement ce type d'application. Il est donc important de réfléchir à l'utilisation d'un modèle plus complexe, tel que les transactions à base de Workflow, les Sagas ou les transactions emboîtées ouvertes.

2.4 Les ONT pour combler ces manques

Comme nous l'avons vu précedemment, un des principaux problèmes lié à l'utilisation des transactions dans le cadre des nouvelles applications sur le Web, est la durée (très longue) de ces transactions. La structuration des transactions en sous-transactions atténue les effets du taux élevé d'échec des transactions. En effet cela ne remet pas en cause systématiquement toute la transaction. Le second problème évoqué dans la section 2.2, est le double problème de la monopolisation des ressources distantes par les transactions longues et le besoin dans ce cas de maintenir les connexions des serveurs concernés par ces ressources. Les mécanismes qui apportent une solution à ce double problème reposent sur la libération des ressources concernées ainsi que sur des serveurs moins sollicités grâce à la validation définitive des sous-transactions. Cette validation implique naturellement la prise en compte du mécanisme de compensation pour remettre les ressources en cohérence dans le cas de l'échec de la transaction globale. Les ONTs, les sagas et les mécanismes de workflow présentés dans la section 2.3.1 reposent tous sur ce principe. La solution que nous avons retenue pour les plate-formes à composants est le modèle ONT. Dans la suite de cette section, nous comparons les ONT successivement aux sagas et aux workflows pour justifier notre choix.

2.4.1 ONT vs Worflow

Le modèle workflow au même titre que le modèle des transactions ONT permet de subdiviser une tâche en sous-tâches pouvant ainsi être validée définitivement par anticipation pour pallier au problème du blocage long des ressources.

Le point de comparaison entre ces deux modèles repose essentiellement sur les mécanismes de prise en compte de la compensation. Il existe deux implémentations possibles de la compensation pour les modèles Workflow : la première est basée sur une description par le développeur dans des scripts des différents scenarii de compensation en fonction des différentes situations d'abandon d'étapes du workflow. La prise en compte de la compensation par scripting est plus efficace même s'il est difficile pour le développeur du workflow de prévoir toutes les situations de compensations. En effet, dans la plupart des cas, les opérations de compensation ne consistent pas à effectuer les opérations inverses des opérations réalisées dans le cadre de la tâche abandonnée.

Autre situation possible, les compensations peuvent par exemple être optimisées par une compensation simplifiée de plusieurs sous-tâches par une seule et unique opération de compensation. L'implémentation par script est plus efficace dans un contexte de workflow centralisé. Le système a ses limites dans le cas de workflows distribués sollicitant des ressources communes. En effet, les scenarii de compensations sont beaucoup plus difficiles à prévoir. Par ailleurs, les compensations dans le modèle ONT sont programmées automatiquement et peuvent dans certains cas ne pas être aussi efficaces que des compensations programmées par le développeur.

Un autre point fort du modèle ONT, dans un contexte distribué, est la journalisation effectuée par le gestionnaire de ressources qui permet de retrouver l'ordre dans lequel les compensations doivent s'effectuer. La seconde implémentation du modèle workflow propose d'établir des dépendances de tâches permettant en cas de besoin de compensation de constituer une sphère de dépendances et de déterminer les tâches compensables suite à un abandon. Le modèle workflow peut être considéré comme une solution alternative sérieuse

au modèle ONT. Néanmoins nous avons choisi d'implanter le modèle ONT plutôt que le modèle workflow dans une plate-forme à composants car il s'agit de spécifier une extension aux modèles transactionnels existants plutôt qu'une réalisation complète pour un degré équivalent de prise en compte des problèmes évoqués.

2.4.2 ONT vs SAGAS

Nous basons la comparaison entre le mécanisme des ONT et celui des SAGAS sur trois points essentiels : la flexibilité de la solution, le parallélisme et l'efficacité. Pour la flexibilité, en terme de structure, on peut considérer que les SAGAS sont des ONT à un seul niveau de sous-transactions. Cette contrainte implique la simplicité du mécanisme des SAGAS mais ne permet pas de structurer davantage les traitements en divisant en tâches plus petites des tâches longues qui pourraient monopoliser les ressources.

En ce qui concerne le parallélisme d'exécution, le mécanisme des SAGAS consiste à exécuter séquentiellement les différentes sous-tâches qui constituent la tâche globale. Il n'y a donc pas possibilité d'exploiter le parallélisme intrinsèque à la tâche principale par l'exécution parallèle de sous-tâches.

Enfin, le point essentiel concerne l'efficacité de la solution : un des points faibles des SAGAS est la nécessité d'abandonner la transaction mère en cas d'abandon d'une des sous-transactions. L'abandon d'une sous-transaction conduit donc à refaire toutes les opérations d'une transaction. Le deuxième point de comparaison sur cet aspect est la nécessité de compensation. En effet, le mécanisme des SAGAS impose que toute sous-tâche soit une ONT, donc obligatoirement compensable en cas d'échec de la transaction mère. Cela impose un coût de compensation élevé pour l'ensemble de la transaction. A l'inverse, le mécanisme des ONTs supporte la cohabitation des transactions emboîtées "classiques" et des ONT. Cela permet de limiter les sous-tâches de type ONT aux traitements distants et longs, limitant ainsi le taux de compensation.

2.4.3 Conclusion

Les traitements transactionnels basés sur le modèle ONT et sur le modèle Saga présentent de fortes similarités, et sont du même niveau de complexité de réalisation (pour les deux, la difficulté réside dans la définition de la compensation). Par contre le modèle des ONT est plus complet et plus souple. C'est pourquoi notre choix parmi ces deux modèles est le modèle des transactions ONT. En comparant le modèle de Workflow avec le modèle transactionnel ONT, nous pouvons dire que ce sont deux possibilités différentes de répondre à nos besoins. Le processus transactionnel est plus clairement et strictement structuré : les relations entre les transactions-parents et les transactions-enfants sont décrites plus strictement. Le modèle de workflow offre la possibilité de définir des scénarii plus flexibles. L'interopérabilité entre les moteurs des workflows et les passages d'activité d'une étape à l'autre sont définis par les scénarii d'exécution. Donc, ils peuvent être décrits par le programmeur de façons différentes en fonction du besoin applicatif.

Les mécanismes basés sur les transactions ONT et sur les Workflows sont des alternatives concurrentes, pouvant toutes les deux parfaitement répondre à nos besoins actuels. Le choix final d'utiliser le modèle transactionnel ONT est une conséquence des orientations globales de nos recherches. Le modèle ONT s'intègre mieux dans nos travaux sur l'adaptabilité des services techniques (débutés en 2001), car ce modèle a une grande base

commune avec les transactions emboîtées classiques, déjà présentes dans certains moniteurs transactionnels.

2.5 Premières spécifications des ONTs

Dans la partie précédente, nous avons montré l'intérêt de l'extension du mécanisme transactionnel utilisé par la plate-forme à composants aux ONT. Nous allons maintenant étudier comment implanter ce modèle transactionnel. Cela va se faire en 2 étapes : la création du mécanisme primaire du support des transactions ONT, qui va prendre en charge le fonctionnement de base de ce modèle, et l'extension de ce mécanisme, pour être en mesure de gérer les abandons en cascade d'ONT.

2.5.1 Le mécanisme primaire

La validation d'une sous-transaction de type ONT doit valider définitivement tous ses effets sur les données. Cependant, un abandon d'un des ancêtres de cette sous-transaction validée nécessite sa compensation. Ces deux idées doivent être appliquées par les gestionnaires offrant les transactions ONT : un gestionnaire supportant le modèle ONT doit donc permettre de valider définitivement les sous-transactions sans attendre la validation d'une transaction-racine. Après une telle validation, les données doivent être rendues accessibles pour les autres transactions. De plus, le service transactionnel doit comporter un mécanisme de compensation. Le gestionnaire des transactions fait appel à ce mécanisme dans le cas d'un abandon des transactions ayant des sous-transactions ONT validées. Le mécanisme de compensation exécute les opérations compensant les effets des ONT correspondantes afin de permettre de retrouver la cohérence des données.

2.5.1.1 Une nouvelle interface pour la validation

En plus d'être capable d'effectuer la validation réelle d'une sous-transaction ONT, le gestionnaire transactionnel doit être capable de distinguer les sous-transactions ONT des sous-transactions emboîtées traditionnelles, pour garder la possibilité d'exécuter les deux types de sous-transactions.

Nous avons donc décidé d'ajouter des nouvelles commandes de gestion des transactions pour marquer le type des sous-transactions ONT. Il y a une nouvelle commande pour démarrer la sous-transaction, et une autre pour valider la sous-transaction ONT. Ces deux commandes se distinguent de celles des sous-transactions emboîtées traditionnelles par la nécessité d'indiquer les mécanismes de compensation. En effet, au moment de la validation de la sous-transaction de type ONT, le gestionnaire de transaction doit connaître le compensateur de cette sous-transaction. Nous avons donc préféré étendre la liste des commandes du gestionnaire de transaction par une commande de *démarrage* d'une sous-transaction de type ONT, comportant un identificateur des opérations de compensation de cette sous-transaction. Le choix de démarquage du type des sous-transactions pendant le démarrage et non pendant la validation est nécessaire pour connaître le type d'une sous-transaction exécutée tout de suite au démarrage.

Comme le montre de manière simplifiée la figure .2.5, l'ajout de ces nouvelles méthodes se fait par la création d'une nouvelle interface sur le composant de gestion de transaction. Nous ne rentrerons pas ici dans le détail. Cependant, la figure .2.5 représente bien la manière dont cela a été implanté. Nous sommes repartis d'un moniteur transactionel défini

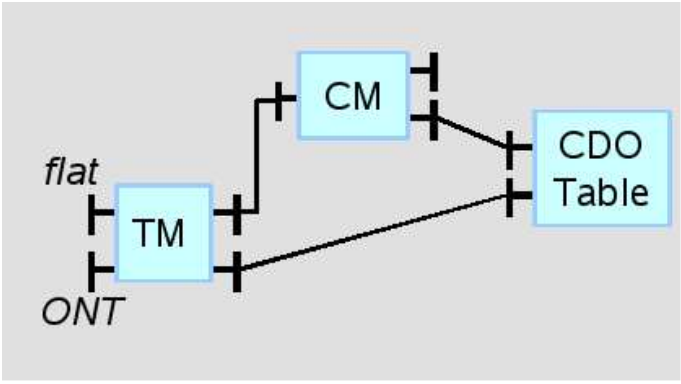

FIG. .2.5 – *Les composants pour les ONTs*

pour être utilisé dans une plate-forme à composants de type Enterprise Java Beans : il s'agit du gestionnaire transactionnel JoTM d'Objectweb que nous avons encapsulé dans un composant (appelé TM). Nous avons ensuite assemblé ce composant avec d'autres composants capables d'effectuer la gestion des ONTs et fournissant les interfaces décrites ci-dessus. Le CM (Compensation Manager, pour Gestionnaire de Compensation en Français) et le CDOtable seront décrits dans la section suivante. Ainsi, dans un premier temps, nous avons fourni un gestionnaire de transactions capable de gérer les transactions de type ONT, dans le cadre d'applications reposant sur des plate-formes à composants (car nous sommes repartis d'un moniteur transactionnel utilisé dans le cadre des Enterprises Java Beans). Après cela, mais nous avons été plus loin, en utilisant le modèle à composants proprement dit, pour étendre le gestionnaire de transactions. Cela est un premier pas, vers les travaux qui seront présentés dans le chapitre suivant, qui portent de manière plus générale, sur la réalisation des services techniques en utilisant le modèle à composants.

2.5.1.2 Mécanisme de compensation

Tout comme nous avons tiré pleinement bénéfice du modèle à composants, pour étendre les gestionnaires de transactions existants en définissant une nouvelle interface, nous allons également utiliser tout l'intérêt de ce modèle, pour définir les méthodes de compensation.

Définition des méthodes de compensation Les composants qui vont pouvoir s'exécuter dans le cadre d'une ONT, doivent fournir des méthodes de compensation de leurs effets. Jusque maintenant, c'est ce point qui a été bloquant dans l'implantation des ONTs dans les gestionnaires de transactions existants (alors que le modèles est spécifié depuis 1981). Cependant, grâce au modèle à composants, nous avons plusieurs solutions pour implanter les méthodes de compensation.

Dans ce but, nous avons défini les "compensateurs". Un compensateur est un groupe de méthodes fournissant les compensations des effets produits par les exécutions des méthodes d'un composant applicatif dont une ou plusieurs méthodes supportent les ONT **[BDL02]**. 2 possibilités se sont alors offertes à nous :

- Définir les méthodes de compensation à l'intérieur du composant applicatif (cf. figure .2.6.a). Cela à l'avantage de simplifier la gestion de la compensation, mais cela ne permet plus de définir de nouvelles solutions de compensation après déploiement du

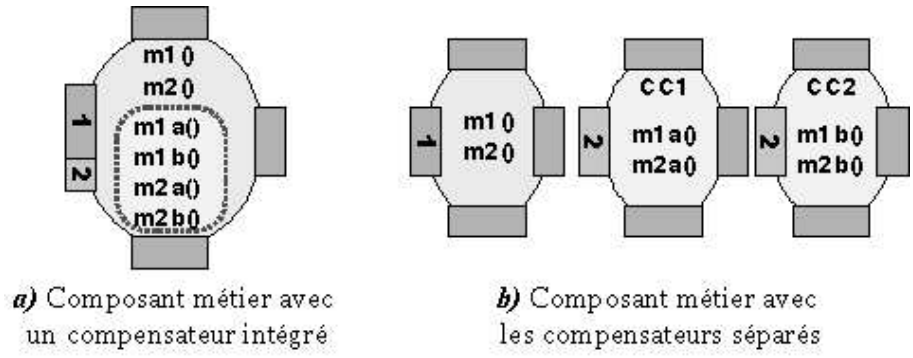

FIG. .2.6 – *Définition des compensateurs*

composant applicatif (sauf à remplacer l'intégralité de ce dernier), et cela augmente considérablement la taille de ce dernier.

– Définir le compensateur comme un composant séparé (cf. figure .2.6.b). Avec cette solution, nous définissons les méthodes de compensation à l'intérieur d'un nouveau composant (ici, deux politiques de compensation différentes du composant : CC1 et CC2). Cette technique a l'avantage de pouvoir définir plusieurs politiques de compensation, sans pour autant accroître la taille du composant applicatif. Nous avons donc choisi cette seconde solution.

.

Processus de compensation Pour pouvoir identifier les transactions à compenser, nous utilisons un identificateur des opérations de compensation. Cet identificateur est associé à chaque sous-transaction ONT pendant son démarrage. Après la validation d'une sous-transaction le gestionnaire de transaction doit conserver cet identifiant. Il pourra alors être utilisé, en cas de nécessité de compensation de ces sous-transactions, pour retrouver les objets concernés. Cet identificateur (appellé CDO pour " Compensation Data Object " en anglais) devra donc comporter (1) les informations permettant de faire une liaison entre une transaction de type ONT et une méthode de compensation de composant correspondant (2) les valeurs des données nécessaires pour l'opération de compensation. Nous proposons donc un objet CDO (cf. figure .2.5) qui comporte les informations suivantes :

– un identificateur de transaction correspondant à cet objet ;
– un identificateur de l'interface de compensation de composant ;
– une signature de méthode de compensation ;
– une liste de valeurs des paramètres de méthode de compensation.

Toutes les données contenues dans cet objet, sauf l'identificateur de transaction, peuvent être directement obtenues via le conteneur gérant le composant applicatif. L'idée de la gestion des transactions par les composants est la suivante : le conteneur du composant intercepte chaque appel de méthode à des composants applicatifs. En cas de besoin (généralement spécifié dans le fichier de déploiement du composant applicatif) une sous-transaction est démarrée par cet intercepteur avant l'appel de la méthode. Si la sous-transaction demandée est de type ONT, alors l'intercepteur appelle la méthode de démarrage de transaction sur l'interface ONT du gestionnaire de transaction. Une telle méthode récupère les attributs correspondant aux informations nécessaires pour la création d'un nouvel objet CDO. La

signature de la méthode de métier appelée et la liste de ses valeurs correspondent à celles de la méthode de compensation. L'identificateur de l'interface de compensation peut être récupéré aussi par l'intercepteur en utilisant les informations contenues dans le conteneur (les méta données ou les valeurs des attributs).

Au final, le conteneur du composant applicatif fournit donc bien toutes les informations nécessaires pour la création d'un objet CDO sauf l'identificateur de la transaction associée. Cette information manquante sera générée par le gestionnaire de transaction pendant le démarrage de la transaction. De cette manière, le gestionnaire de transaction crée un nouvel objet CDO correspondant à la sous-transaction ONT démarrée. Cet objet sera simplement utilisé par le gestionnaire de transaction, en cas d'appel à la méthode *Abort* d'annulation de transaction, pour récupérer le composant de compensation et faire appel aux méthodes de compensation nécessaires pour défaire la sous-transaction ONT.

2.5.2 Les compensations cascadées

Nous venons de définir les principes de base d'un gestionnaire de transactions à base de composants permettant de gérer les ONTs. Cependant, la compensation d'une ONT peut engendrer l'annulation d'autres ONTs. Il faut donc être en mesure de gérer cela. Dans cette partie, nous proposons un mécanisme gérant les abandons en cascade d'ONTs.

2.5.2.1 Détermination des transactions à compenser : dépendance entre transactions

Le mécanisme offrant la gestion des compensations cascadées doit garantir que les effets de toutes les transactions dépendantes de la transaction compensée sont soit compensées si ces transactions ont elles mêmes été validées, soit annulées si elles ne sont pas déja achevées. Le processus de compensation que nous avons établi dans le gestionnaire de compensation se compose de deux étapes : (1) la génération de la liste des identificateurs de transactions à compenser et (2) la recherche des compensateurs correspondants et le démarrage des compensations au niveau des compensateurs (cette étape est basée sur l'utilisation des objets CDO présentés dans la section précédente).

Le problème des compensations cascadées que nous allons traiter ici, intervient à la première étape du processus de compensation. Le gestionnaire de compensation doit identifier toutes les transactions dépendantes (directement ou non) des transactions indiquées pendant l'appel de compensation. Puis, les identificateurs des transactions trouvées doivent être, eux mêmes, ajoutés dans la liste des transactions à compenser. Le tri de cette liste termine la première étape de la compensation.

L'algorithme de recherche des transactions dépendantes d'une transaction donnée est donné figure .2.7. Il repose sur deux étapes : la recherche de toutes les ressources utilisées par les transactions compensées, ainsi que l'identification des transactions qui ont également accédées à ces ressources.

l'identifiant de la transaction qui va être compensée doit être donné en entrée de cet algorithme. A partir de cela, il faut obtenir la liste de toutes les ressources utilisées par cette transaction. Cette liste est obtenue par la méthode *findResource*$(id - t_i)$ qui travaille à partir du mécanisme de verrouillage utilisé pour assurer le contrôle de concurrence sur les données. Si la réponse est vide, cela signifie que la transaction n'a pas modifié de données

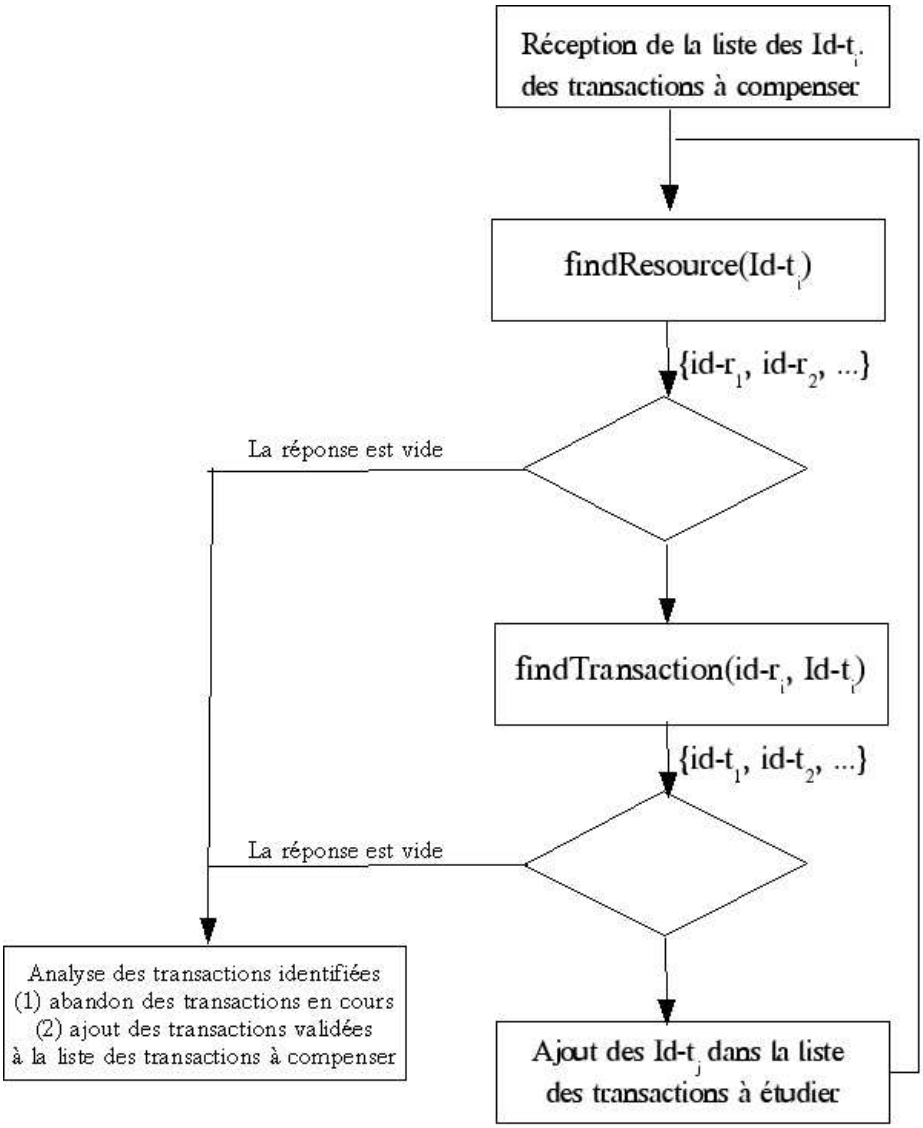

FIG. .2.7 – *Recherche des transactions dépendantes*

et cela a pour concéquence qu'il n'y aura pas d'autre transaction à compenser (car aucune transaction n'a pu lire des données "sales").

La seconde étape, plus complexe, consiste à identifier toutes les transactions qui ont elles-mêmes accédé à des données modifiées par la transaction compensée. C'est le rôle de la méthode *findtransaction*($id - r_i$,$id - t_i$), qui sera exécutée pour toutes les ressources identifiées à la première phase. A la fin, les transactions identifiées comme ayant utilisées des données "sales", sont elles mêmes ajoutées à la liste des transactions à compenser. il convient donc de réutiliser le même algorithme pour déterminer si la compensation de ces nouvelles transactions n'engendre pas de nouveaux problèmes.

2.5.2.2 Extensions souhaitables

Nous avons décrit le mécanisme qui recherche les transactions dépendantes. Les résultats trouvés sont utilisés pour démarrer les compensations cascadées. Cependant, deux extensions sont possibles pour ce mécanisme.

La première est liée aux cas des les transactions dépendantes ne doivent pas être compensées. Revenons à une application qui représente une agence de voyages (cf. figure .2.1). Supposons qu'un utilisateur commande un billet d'avion, puis cherche des places libres dans les hôtels, les locations de voitures, etc. Il est possible qu'il décide d'abandonner son voyage. Dans ce cas, une compensation de sa commande de billet d'avion est lancée. Par contre, si le gestionnaire de compensation lance les compensations cascadées, alors il doit trouver toutes les transactions dépendantes de cette sous-transaction. La compensation de la sous-transaction effectuant la commande du billet d'avion, et qui avait été validée, modifie la valeur de ressource qui présente le nombre de places libres pour le vol correspondant. Si l'on respecte les règles transactionnelles strictes, toutes les transactions qui lisent cette valeur sont dépendantes de la sous-transaction en question. Par conséquent, elles doivent être compensées ou abandonnées, car elles ont effectuées une lecture "sale". Par contre, du point de vue logique, il n'y a aucun sens à compenser les commandes de billets effectuées après la réservation annulée, car la seule dfférence est qu'il y a une place de plus disponible. Nous sommes donc dans un cas, où la décision de compensation cascadée est dépendante de l'application. La seule solution pour traiter cela est de prendre en compte la sémantique des opérations, comme le propose **[BIL95]**, mais cela n'est pas géré actuellement.

La deuxième extension du mécanisme proposé est liée à l'impossibilité de ce mécanisme de trouver les dépendances entre les transactions gérées par les différents gestionnaires transactionnels. En effet, pendant cette recherche les procédures proposées dans ce châpitre utilisent le journal géré par le gestionnaire de transaction local. En conséquence, seules les transactions exécutées par le gestionnaire de transaction courant sont prises en compte

2.5.3 Premier bilan

Dans cette partie, nous avons expliqué comment nous avions utilisé le modèle à composants pour implanter le mécanisme des ONTs. Cela s'est fait par composition d'un gestionnaire de transactions existant (gérant les transactions plates et les et les transactions emboîtées classiques), avec un ensemble de composants de gestion dédiés aux ONT. Cette solution n'est certainement pas la plus efficace : il aurait été plus performant de réaliser complètement un gestionnaire de transaction à base de composants. C'est d'ailleurs un projet en cours au niveau du consortium ObjectWeb, sous le nom de projet GoTM.

Cependant, le but de ces travaux était principalement d'étudier en quoi le modèle à composants, pouvait permettre l'implantation réaliste de mécanismes datant de 1981, mais jusque-là non utilisé, car comportant des notions "insurmontables" en terme d'implantation (la compensation). Pour cela, l'ensemble des mécanismes gérés (même si il reste un peu de travail au niveau des compensations cascadées, montre que cela est réalisable.

2.6 Politique de cohabitation avec les autres modèles

Avec les nouvelles applications distribuées, des traitements longs et complexes, monopolisant longtemps les données, sont apparus. Cependant, il est important de prendre en compte le fait que ces actions ne représentent pas la majorité des ordres échangés entre

les partenaires des applications. En effet les transactions simples ayant besoin d'accéder ponctuellement à des ressources constituent la majeure partie des transactions.

Ainsi, les transactions ONTs, choisies pour répondre au premier type d'utilisation des données, peuvent être amenées à cohabiter avec des transactions classiques. Cette cohabitation pose un problème dans le cas où une transaction classique utilise une ressource modifiée préalablement puis libérée par une sous-transaction ONT dont la transaction mère n'a pas encore été validée. Si la transaction mère est abandonnée, après la validation de la transaction classique, cela implique la nécessité de compenser les effets de la sous-transaction ONT. Cela a pour conséquence la nécessité de défaire les effets de la transaction classique et donc de remettre en cause de la propriété de durabilité du modèle des transactions plates.

Dans cette section, pour diminuer les conflits entre les transactions classiques, et les ONTs, nous proposons de différer toute transaction ayant besoin d'accéder à des ressources manipulées par des transactions ne respectant pas le principe de la durabilité. La mise en oeuvre de cette solution est réalisée grâce à une gestion des verrous à l'échelle de chaque ressource afin d'assurer la cohérence des traitements. Dans la suite de cette section, je vais donc vous présenter nos travaux de définition de nouveaux verrous pour assurer la cohabitation des ONTs avec d'autres modèles transactionnels, puis je vais détailler comment ces verrous doivent être gérés par les gestionnaires de ressources.

2.6.1 Définition de nouveaux verrous

Pour assurer les propriétés d'Isolation et de Cohérence, les gestionnaires de ressources utilisent des verrous posés sur les données manipulées par les transactions. Le mécanisme de contrôle de concurrence le plus répandu dans les SGBDs actuels, est le mécanisme de verrouillage à 2 phases. Le principe de ce mécanisme est relativement simple :

- lorsqu'une transaction accède à une donnée en lecture ou en écriture un verrou est posé (verrou d'accès partagé pour la lecture, et d'accès exclusif pour l'écriture)
- une transaction a le droit d'accéder à une donnée en lecture si cette donnée n'a pas de verrou exclusif posé, et peut accéder à une donnée en écriture, si cette dernière n'a aucun verrou. Dans le cas contraire, la transaction est mise en attente.
- Lorsqu'une transaction se termine (par validation ou par annulation), les verrous qu'elle détenait sur les données sont libérés, et les transactions bloquées par ces verrous sont relancées.

Cette politique de verrouilage pose problème dans le cadre des transactions ONTs. En effet, la relâche complète des verrous à la validation de la transaction pose problème, car on perd toutes les informations concernant l'accès à la donnée par la transaction. Or, dans le cadre d'une ONT, même après validation, la transaction peut être compensée, et donc les données peuvent changer. Il est donc nécessaire de notifier les autres transactions accédant aux données de ce fait. Nous avons donc proposé d'étendre la famille des verrous basiques (partagés et exclusifs), par une nouvelle famille de verrous nommés "Advanced-Lock" dont l'objectif est d'interdire à des transactions devant vérifier une des propriétés ACID, l'accès à des ressources qui risquent de remettre en cause cette (ou ces) même(s) propriété(s). Le verrou permettant la cohabitation des transactions ONT et des transactions plates fait partie de cette famille. Nous l'avons nommé "ONT-Lock". Une solution simple au problème de cohabitation est donc la pose du verrou "ONT-Lock" sur des ressources modifiées par une transaction de type ONT. La pose d'un tel verrou empêchera toutes les transactions simples, devant respecter la propriété de durabilité, d'utiliser ces données, puisqu'une ONT

demeure susceptible d'être compensée tant que la transaction ancêtre globale n'a pas être validée.

Cependant, l'interdiction induite par la pose d'un verrou ONT-Lock sur une ressource peut être forte. Dans ce cas, aucune utilisation de la ressource par une transaction plate n'est autorisée. Cette solution extrême peut être plus modérée. En effet, une autre solution est d'autoriser la lecture et/ou l'écriture de la ressource et d'interdire la validation de la transaction plate tant que l'ONT, qui a également accédé aux données peut être compensée (elle peut être compensée tant que sa transaction mère n'est pas validée). Il en découle différentes politiques de verrouillage qui sont plus ou moins restrictives sur le parallélisme d'exécution. L'ensemble de ces politiques sont décrites dans la thèse de Sergiy Nemchenko **[NEM04]**.

2.6.2 Exemple de fonctionnement de ces nouveaux verrous

L'exemple donné par la figure .2.8 illustre un cas de cohabitation des transactions classiques avec des ONTs qui vont produire des compensations cascadées (c'est à dire qu'une compensation d'une ONT engendre une compensation de une ou plusieurs autres ONTs). Dans cet exemple, la transaction ONT1.1 valide (t2) et pose un verrou ONT-Lock sur la ressource B. La transaction ONT2.1 lit alors la valeur de la ressource B et modifie celle de la ressource A. Enfin, ONT2.1 valide également (t5) et pose donc un verrou ONT-Lock sur la ressource A. Dès que la transaction T2 (transaction mère de ONT2.1) est validée, le verrou ONT-Lock qui a été posé sur la ressource A est levé (condition suffisante pour le modèle ONT primaire). Toutes les autres transactions peuvent donc accéder à la ressource A alors que le risque de compensation est toujours présent tant que la transaction racine T1 n'a pas été validée. En effet, si la transaction ONT1.1 doit être compensée, par effet induit, la transaction ONT2.1 devra également être compensée, car elle aura lu une donnée B non valide.

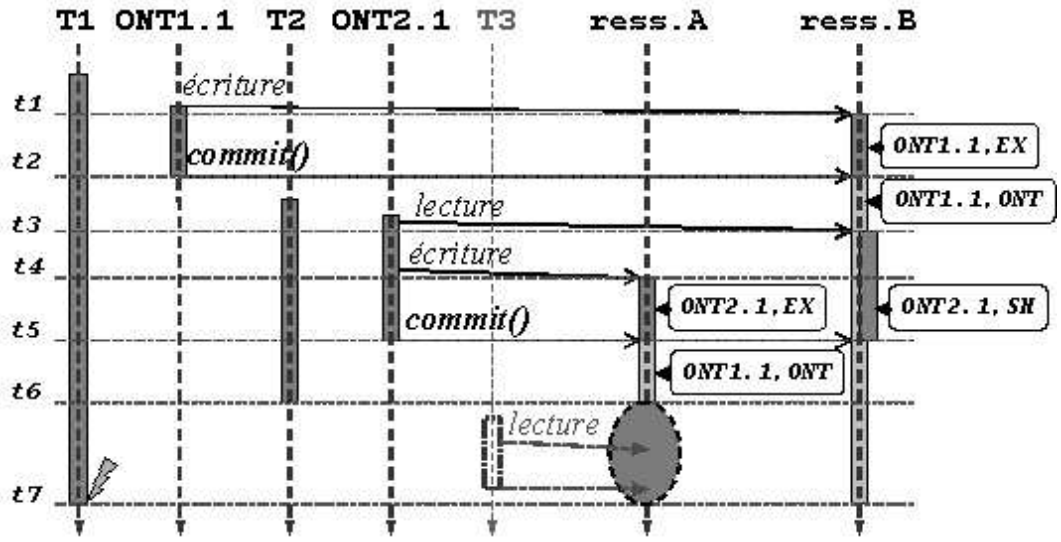

FIG. .2.8 – *Exemple de fonctionnement des verrous ONT-Lock*

Afin de formaliser ce problème, la relation de dépendance entre les verrous ONT-Lock

est spécifiée de la manière suivante : Un verrou ONT-Lock *v1* posé par une transaction *ont1* sur une ressource *r1* est *ONT-dépendant* d'un verrou ONT-Lock *v2* posé sur une ressource *r2* par une transaction *ont2* ssi :

- *v1* a utilisé *r2* en lecture ou en écriture
- A la pose de *v1, v2* est toujours posé sur *r2* (la transaction mère de *ont2* n'est toujours pas validée et *ont2* n'a pas été compensée)

Cette dépendance est alors noté $v1 \rightarrow (v2,r2)$. L'ensemble des dépendance entre ONT-lock pour un verrou v est noté $D_{ONT}(v)$.

Le problème exposé dans la figure .2.8 peut alors être résolu en appliquant la règle de relaxe des verrous ONT-Lock pour le modèle ONT avancé comme suit.

Un verrou ONT-lock v posé par une transaction ONT *ont1* peut être relâché ssi :

- la transaction racine de la transaction *ont1* valide ou *ont1* est compensée
- il n'existe plus de verrou v_i posé sur une ressource i tel que $v \rightarrow (vi,i)$

2.6.3 Gestion des verrous

En terme de gestion de verrous, la pose des verrous ONT-Lock pose de nouveaux problèmes. En effet, étant donné qu'ils sont là pour éviter les conflits avec les autres transactions après la validation de l'ONT, la particularité des verrous ONT-Lock est qu'ils sont posés à la validation de la transaction et non pas au moment d'utiliser les ressources sur lesquelles ils sont posés. Leur pose s'effectue suite à la relaxe des verrous d'écriture sur les mêmes ressources. La solution la plus simple est alors de poser les verrous ONT-Lock lors de l'appel à l'opération *unlock* du protocole de verrouillage à 2 phases traditionnel.

De même, la relaxe des verrous ONT-Lock est particulière. Dans la gestion classique des verrous, un verrou est levé à la fin de la transaction qui a demandé sa pose. Dans le cas des verrous ONT-Lock, la relaxe doit avoir lieu lors de la validation de la transaction racine. Le gestionnaire de transaction doit donc transmettre à la fin de la transaction racine, la liste des sous-transactions ONTs, de manière à ce que tous les verrous ONT-Lock correspondants soient levés.

En terme de gestion, la validation des transactions plates pose problème également. Pour préserver la propriété de durabilité, une transaction plate peut utiliser les ressources en posant des verrous classiques mais ne peut valider qu'en cas d'absence de verrous ONT-lock sur ces ressources. La demande de relaxe des verrous doit être différée dans ce cas. Le gestionnaire de ressource doit être en mesure de gérer les demandes de relaxe des verrous de manière asynchrone après libération des verrous ONT-Lock qui empêchent cette validation.

Enfin, il est important de gérer la dépendance entre les ONTs. L'opération *unlock* demandée à la validation d'une transaction racine doit s'assurer que l'ensemble des dépendances d'un verrou ONT-Lock est vide. Le gestionnaire de ressources doit donc être en mesure de gérer ces dépendances, et il doit notamment pouvoir, en cas de libération d'un verrou ONT-Lock, mettre à jour les dépendances des autres verrous ONT-Lock.

Pour répondre à ces différents besoins, des extensions du modèle de verrouillage traditionnel ont été nécessaires :

- Les opérations *Lock* et *Unlock*, du mécanisme de verrouillage à 2 phases traditionel, sont dotées d'un nouveau paramètre T_{model} indiquant le type de la transaction de-

mandant la pose ou la relaxe des verrous. Ce nouveau paramètre permettra d'adapter les effets de la méthode au type de transaction.

- L'opération *Unlock* demandée à la validation d'une transaction T est dotée d'un paramètre *Children-ONT* , représentant la liste des sous-transactions ONTs de la transaction T. Ce nouveau paramètre permet de relacher les verrous de chacune des transactions de cette liste.
- Une opération *majDependanceVerrou(v, id-r)* supprime le verrou v de la liste des dépendances D_{ONT} de chaque verrou ONT-Lock posé sur la ressource id-r.
- Une opération *majDependanceRessource (v, R)* ajoute l'ensemble des ressources R à la liste A des ressources à "notifier" en cas de libération du verrou v.
- Une opération *Unlock-ONT(T)* relâche les verrous ONT-Lock posés par T. Cette opération est lancée à la validation d'une transaction racine. Le gestionnaire de ressource relâche un verrou v si le paramètre D est vide (c.a.d tous les verrous ONT-Lock dont il est dépendant ont été relâchés). Auparavant, il applique l'opération *maj-dependance-verrou(v, id-r)* pour chaque ressource id-r de l'ensemble Rv.

La gestion des verrous se trouve donc complexifiée par l'arrivée des ONTs dans les modèles transactionnels à gérer. Cependant, si les ONTs sont utilisées correctement, l'augmentation du parallélisme d'execution des transactions distribuées contre-balancera cela.

2.7 Formalisation de la proposition

Dans la section 2.5, nous avons proposé une implantation du modèle des transactions emboitées ouvertes proposé par J. gray il y a plus de 20 ans. Cette réalisation repose sur un modèle de programmation (la programmation par composants), qui n'existait pas en 1981. Il est donc important de bien spécifier les règles que nous avons respectées pour réaliser ce service. En effet, une description des modèles et de leurs implantations en utilisant le langage courant peut être incomplète et, en même temps, compliquée pour la compréhension. L'utilisation d'un langage formel résout ces problèmes. Pour fournir cela, nous avons décidé d'utiliser le framework ACTA **[CR90]**, qui est un moyen universellement reconnu pour la description des modèles transactionnels avancés.

Il ne s'agit pas ici de décrire la spécification ACTA des ONTs. pour cela, le lecteur peut se reporter à la thèse de Sergiy Nemchenko **[NEM04]**, où cette spécification est décrite dans le détail, pour toutes les phases de notre travail (spécification des transactions emboîtées, emboîtées ouvertes, et du problème des transactions cascadées). Simplement, il est à noter que la création de ces spécifications ACTA décrivant le modèle des transactions emboîtées ouvertes nous a permis de présenter notre réalisation (1) dans une forme brève et universelle ,et en même temps, (2) de décrire toutes les situations possibles d'exécution des transactions et ainsi de détecter les situations de conflits possibles au niveau du modèle transactionnel.

La description formelle permet de présenter les idées proposées en utilisant les formalismes universels. En même temps, c'est une description complète et donc elle présente le comportement souhaité dans toutes les situations possibles. Cependant, l'inconvénient majeur de la spécification ACTA est l'absence de moyens automatiques pour vérifier les spécifications décrites. De ce fait, ce n'est que l'analyse humaine qui peut garantir la correction de la spécification. C'est en se basant sur cette spécification, que nous avons créé un prototype qui applique en pratique les axiomes décrits. La section suivante présente ce

prototype.

2.8 Prototypage et résultats

De manière à valider ces travaux, mais aussi pour répondre aux besoins des 2 projets qui ont servi de support à ces recherches (le projet ITEA PEPiTA et le projet RNTL IMPACT), nous avons réalisé un prototype implantant les transactions emboîtées ouvertes à l'intérieur d'un moniteur transactionnel, utilisable dans le cadre d'une plate-forme à composants. Dans cette partie, je vais vous présenter ce prototype, tout d'abord au travers des objectifs que nous avons tenté d'atteindre, puis en décrivant la réalisation et les résultats obtenus.

2.8.1 Objectifs

Ce prototype avait comme principal but d'apporter la preuve de la faisabilité de nos propositions, tout en fournissant une première évaluation des performances que l'on peut attendre d'un tel moniteur transactionnel. Il a été réalisé en deux phases.

La première phase a consisté à étendre le moniteur transactionnel du erveur applicatif JONAS **[JON01]**, qui repose sur le modèle des Enterprise Java Beans. Les premiers travaux ont donc porté sur l'extension du moniteur transactionnel de Jonas : JoTM. Nous avons notamment fournit une implantation des transactions emboîtées fermées pour JoTM. Cela s'est fait dans le cadre du projet PEPiTA. L'objectif était de fournir un moniteur transactionnel pour le serveur applicatif Jonas, répondant au mieux aux spécifications OTS de l'OMG.

Puis, nous avons réutilisé ce moniteur transactionnel pour fournir les ONTs. Pour atteindre ce but, nous avons créé un service transactionnel comportant les mécanismes de compensation décrits précédemment. Le principal but de ce travail était d'étudier les apports du modèle à composants dans le cadre de l'implantation des mécanismes de compensation (la réalisation des compensation étant le principal verrou technologique à l'implantation des ONTs dans les moniteurs transactionnels). Puis, pour vérifier la fonctionnalité du service transactionnel réalisé, nous avons réalisé des applications tests, modifiant les ressources au sein des transactions et des sous-transactions gérées par ce service transactionnel. Pour fournir des évaluations, durant les exécutions nous traçons les informations suivantes :

- la démarcation des sous-transactions et des compensations (démarrage et achèvement).
- les états des ressources avant la création et après l'achèvement des sous-transactions et des compensations.

Ces informations journalisées vont nous permettre d'étudier le comportement de notre moniteur transactionnel, et de garantir que son fonctionnement est conforme aux spécifications annoncées dans la section précédente.

2.8.2 Choix du modèle à composants

Pour réaliser l'implantation des ONTs, nous avons choisi d'utiliser un modèle de composants récursifs (cela signifie qu'un assemblage de composants est lui même un composant). Cette décision est basée sur les avantages offerts par ce modèle. Du point de vue de la

réalisation des applications distribuées, l'avantage principal d'utilisation des composants hiérarchiques par rapport aux composants primitifs est la granularité plus fine de la création, de l'utilisation et de la distribution des applications. Par conséquent, cela offre une amélioration de la structuration de l'application, de la possibilité de partage des tâches des programmeurs et de la réutilisabilité des composants.

En même temps, la création des services transactionnels est facilitée et optimisée aussi grâce à l'utilisation des composants récursifs. Quelque soit la granularité de découpage du service technique en composants, le service final pourra toujours être vu comme un seul composant à assembler avec l'application. Il existe d'ailleurs maintenant de nombreux travaux qui présentent les services techniques en forme de composants. Nous pouvons cité ici les projets GoTM et Jironde menés par le consortium Objectweb **[RM03],[JIR03]**.

Nous avons utilisé la spécification Fractal des composants récursifs créée par le consortium ObjectWeb **[BCS04]** pour la création de notre prototype. Dans la réalisation de ce prototype, notre choix se porte sur l'implantation Objectweb en java de FRACTAL : Julia (implémentation open-source). En effet, l'analyse de la spécification Fractal montre un point important pour la création du service transactionnel offrant des transactions emboîtées ouvertes. C'est la possibilité de contrôler la composition et les liaisons d'une composition de composants grâce au contrôleur qui encapsule un composant. En effet, ces liaisons sont explicites et accessibles. Ainsi le système a une représentation de lui-même. Ce point est très important pendant la création du mécanisme des descripteurs des compensations.

De plus, ce modèle correspondait parfaitement aux besoins d'autres travaux commencés entre temps dans l'équipe, sur une approche plus générale de l'adaptation de service non fonctionnels aux besoins des applications et aux caractéristiques des terminaux (cf. chapitre 3).

2.8.3 Réalisation

Nous n'avons pas cherché ici à réécrire un moniteur transactionnel complet à base de composants (comme c'est le cas pour GoTM), mais simplement à compléter un moniteur existant (JoTM), pour lui fournir les fonctionnalités de gestion des ONT. En effet, un service transactionnel est toujours basé sur un moniteur transactionnel. Pendant la création de notre prototype nous avons donc ré-utilisé un gestionnaire de transactions existant, que nous avons "encapsulé" dans un composant primitif **[HNL05]**. Les mesures réalisées ici sont donc à relativiser car peu représentatives du vrai coût d'utilisation d'un moniteur transactionnel réalisé avec des composants "à grains fins".

Sur le schéma suivant (cf. figure .2.9), il est représenté par le composant nommé "TM". Ce composant offre toutes les fonctionnalités nécessaires à la gestion des transactions classiques. C'est pourquoi, toute la logique applicative d'un service transactionnel offrant les transactions classiques est représenté par ce composant "TM". Dans la pratique, il s'agit du moniteur transactionnel d'ObjectWeb JoTM, que nous avions modifié dans le cadre du projet IMPACT, pour lui fournir, en plus des transactions plates, les transactions emboîtées fermées. Ce moniteur JoTM, avait l'avantage d'être intégralement écrit en Java, et donc facilement intégrable à l'intérieur d'un composant Fractal de Julia. Ce composant, gestionnaire de transactions plates et emboîtées fermées, offre les interfaces décrites par la

spécification OTS de l'OMG.

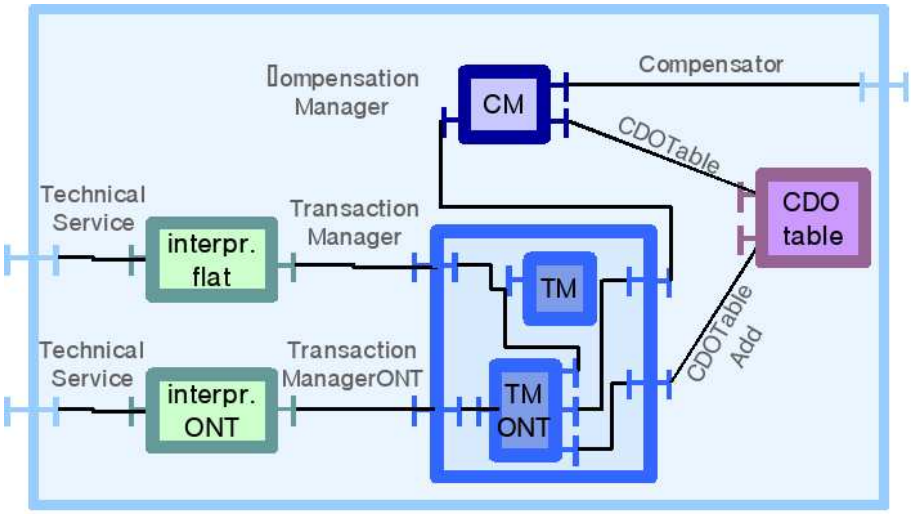

FIG. .2.9 – *Découpage en composants Fractal de notre moniteur transactionnel*

Une fois le gestionnaire de transactions intégré dans un composant Fractal, nous l'avons étendu au support des transactions emboîtées ouvertes. Le gestionnaire devait alors proposer deux modèles de gestion de transactions : transactions classiques et ONT. Pour ce faire, le composant propose deux interfaces différentes, chacune correspondant à un modèle transactionnel. Pour chacune de ces interfaces, un " interpréteur " (les composants "interpr. flat" et "Interpr. ONT") est nécessaire. Cela est dû à un détail d'implantation de part l'utilisation du modèle Fractal. L'intercepteur de la membrane du composant applicatif ne peut faire appel à un composant "technique" que par l'appel à trois méthodes : "PreMethod" (qui fait appel au composant du service technique avant l'exécution du composant applicatif), "PostMethod (pour un appel après l'exécution du composant applicatif) et "HandleException" pour gérer les erreurs **[HL03]**. Dans le cas de l'appel à "PreMethod" sur l'interface du composant "Interpr. FLAT", cette appel est converti en un appel à la méthode Begin() du moniteur transactionel (qui correspond au début d'une transaction simple), alors que le même appel sur le composant "Interpr. ONT" sera converti en un appel à la méthode Begin(CDO) du moniteur, qui correspond au lancement d'une transaction emboîtée ouverte.

Comme nous l'avons vu dans la partie 2.5, la principale difficulté des ONTs consiste à gérer les compensations. Pour cela, le gestionnaire doit être capable de :

- trouver les descripteurs des processus de compensations nécessaires ;
- appeler les compensations gérées par les compensateurs décrits par les descripteurs.

Ce gestionnaire est représenté, dans la figure .2.9 par un composant primitif nommé "CM" pour "Compensation Manager" en anglais. Ce composant offre une interface d'appel de la compensation qui est utilisée par le gestionnaire de transaction (*ICompensationManager*). Dans notre implémentation, un compensateur est représenté par un composant " call-back " de composant applicatif. Or, pour lancer la compensation, le gestionnaire de compensation doit appeler la méthode " call-back ", associée au service transactionnel du composant " call-back " de composant applicatif qui a appelé la (sous-) transaction ONT

correspondante. Cette méthode a un argument présenté par le descripteur de compensation.

De manière à conserver toutes les informations nécessaires aux compensations, nous utilisons un composant "CDO Table", qui contient les objets CDO des (sous-)transactions ONTs validées par ce service transactionnel (cf. l'explication de ce mécanisme section 2.5.1.2). Ce composant offre (1) une interface qui permet d'ajouter des nouveaux objets et de supprimer les objets non utiles, et (2) une interface de recherche d'objet demandé. La première interface est utilisée par le gestionnaire de transaction et la deuxième par le gestionnaire de compensations.

Pour la création des applications, nous avons utilisé les outils suivants :

- Les composants ont été créés en utilisant Julia 1.0.6 du consortium Objectweb (il s'agit de l'implantation en Java standard du modèle Fractal).
- Le gestionnaire des transactions que nous avons modifié et fait évolué est JOTM1.4.1, proposé également par le consortium Objectweb. Ce moniteur a également été utilisé tel quel pour les transactions classiques.
- La base de données est Mckoi SQL Database v1.00. Le langage de programmation est Java. Le choix du langage Java est basé sur le choix du gestionnaire de transactions que nous avons repris (JOTM était réalisé en Java) et de l'implémentation du modèle à composants (L'implantation standard de Fractal est en Java). Les applications sont compilées en utilisant le jdk1.4.1.
- Les tests ont été exécutés sur un ordinateur de type PC, reposant sur un P3 à 700MHz, avec 384Mo de mémoire RAM, sous Windows 2000.

L'analyse des résultats a notamment montré que l'extension du gestionnaire de transactions au modèle ONT ne dégrade pas le temps d'exécution des transactions plates. Cela est un des intérêts directs de l'utilisation du modèle à composants : quand un composant ne sert pas à l'exécution d'une application donnée, il ne ralentit pas le code de cette application. Par contre, l'utilisation du modèle à composants par lui même, ralentit la vitesse d'exécution très fortement. Cet effet a deux raisons. La première est le coût d'utilisation des composants. Le code d'un programme est entouré par un conteneur (ou " une membrane " dans les termes de Fractal) dans les applications basées sur les composants. Ainsi, la création des applications est facilitée, par contre la vitesse d'exécution est diminuée. La deuxième raison est l'utilisation de Julia (l'implémentation de la spécification Fractal), qui dans sa version 1.06 n'était pas encore optimisée.

2.9 Conclusions

Ce travail nous a permis d'évaluer les facilités qu'offrait le modèle à composants, pour réaliser des nouvelles versions de services techniques. Ainsi, nous avons une représentation des mécanismes de compensation des transactions, dans le cadre de la réalisation des ONTs. De plus, les possibilités offertes par le modèle à composants, de réutilisation du code existant, nous a permis de rapidement réaliser ce gestionnaire de transactions, à partir d'une version capable de gérer les transactions emboîtées fermées.

Au final, ces travaux ont été menés en 2 phases :

- La première phase a consisté à étudier comment faire évoluer JoTM, le moniteur transactionnel du serveur applicatif Jonas d'objectweb, pour répondre aux besoins

applicatifs posés dans le projet PEPiTA. Nous avons donc spécifié et réalisé un moniteur transactionnel capable de gérer les transactions emboîtées classiques, et compatible avec le standard des Enterprise Java Beans (par exemple, nécessité de définir de nouveaux attributs de déploiement). J'ai choisi ici de ne pas détailler ces travaux, car il s'agissait principalement de régler des problèmes propres à l'implantation de mécanismes au sein du moniteur transactionnel du serveur applicatif Jonas. Ces mécanismes étant par ailleurs parfaitement spécifiés pour d'autres plate-formes, comme par exemple dans le moniteur transactionnel OTS de l'OMG.

- La seconde étape, qui constituait la réalisation des ONTs, nous a conduit vers un autres modèle à composants : Fractal. Cela est dû aux spécificités des transactions ONT qui n'étaient pas réalisables sans le recours à un modèle à composants hiérarchique, pour pouvoir gérer correctement la compensation. Cela fut très intéressant, car cela a montré la nécessité de travailler sur une méthode de conception de services transactionnel, reposant elle même sur le modèle à composants.

Les travaux de ces deux phases ne sont, heureusement, pas incompatibles, puisque le moniteur transactionnel réalisé dans la première phase a été repris et intégré dans un composant Fractal pour la seconde. D'autre part, d'autres équipes qui travaillent maintenant sur le remplaçant de GoTM, ont également décidé d'utiliser Fractal comme modèle à composants support.

De plus, nous verrons par la suite que les idées de conception d'un service transactionnel à base de composants ont été très largement utilisées dans d'autres travaux de l'équipe. Cela à notamment posé les bases des travaux sur la conception de services techniques à base de composants, et sur l'adaptation dynamique de ces services, en fonciton de leur environnement d'exécution, et des besoisn applicatifs.

Ce travail, le premier mené à Valenciennes à mon arrivée, a également eu l'avantage d'être porté par deux très gros projets (le projet Européen ITEA PEPiTA et le projet RNTL IMPACT). Cela a permis d'équiper le pôle SID en matériel, et cela a assuré un bon fonctionnement de l'équipe.

Chapitre .3

L'adaptation des services techniques

3.1 Introduction

Comme je l'ai expliqué dans le chapitre précédent, mes travaux à l'université de Valenciennes ont commencé avec l'étude des possibilités d'extension des moniteurs transactionnels. Cependant, rapidement nous nous sommes rendus compte que cette démarche était insuffisante. En effet, avec l'arrivée de nouveaux terminaux et de nouvelles applications distribuées, de nombreux services pouvaient potentiellement demander des extensions, dans le but de s'adapter aux fonctionnements spécifiques à certains environnements (protocoles de communication, capacité de calcul) ou certaines applications .

De nouveaux travaux ont donc débuté sur l'étude de mécanismes permettant, dans un premier temps, de fournir plusieurs "versions" (que nous nommerons "personnalités" dans la suite du document) d'un même service technique (ces services seront présentés et classifiés dans la suite du document). Ainsi, en fonction des besoins applicatifs et des caractéristiques des terminaux exécutant l'application, la meilleure personnalité du service technique pourra être choisie. Puis, nous avons travaillé sur un modèle de fonctionnement permettant d'assurer l'adaptabilité des services techniques automatiquement, lors de l'exécution de l'application (après la phase de déploiement).

Cette réflexion a trouvé un écho dans le montage du projet de recherche RNRT COMPiTV. Ce projet, qui avait comme partenaires deux entreprises internationales (Canal+ Technologies et Gemplus Card International) et deux laboratoires du CNRS (l'Université de Lille 1 avec le LIFL et l'université de Valenciennes et du Hainaut Cambraisis avec le LAMIH), avait pour premier but d'étudier les apports du modèle à composants dans le cadre de la réalisation d'applications pour la télévision interactive **[CDE02]**. En effet,

comme le montre la figure .3.1, la télévision par satellite utilise une architecture de diffusion spécifique, avec une large bande passante pour émettre des données de l'opérateur vers l'utilisateur (via satellite) et une bande passante très réduite de l'utilisateur vers l'opérateur (via modem traditionnel). Cette technique est parfaitement adaptée à la diffusion de contenu multimédia, mais atteint ses limites lorsqu'il s'agit de contenu interactif, comme la consultation de compte bancaire, les paris ou le jeu. Cela est, de plus, couplé à une hétérogénéité des terminaux participants à l'application (démodulateur satellite de plus en plus hétérogènes et serveurs des prestataires). L'idée principale de ce projet était d'explorer les possibilités d'utilisation du modèle à composants, pour simplifier la tâche des programmeurs de services interactifs et pour faciliter le déploiement des applications sur les terminaux.

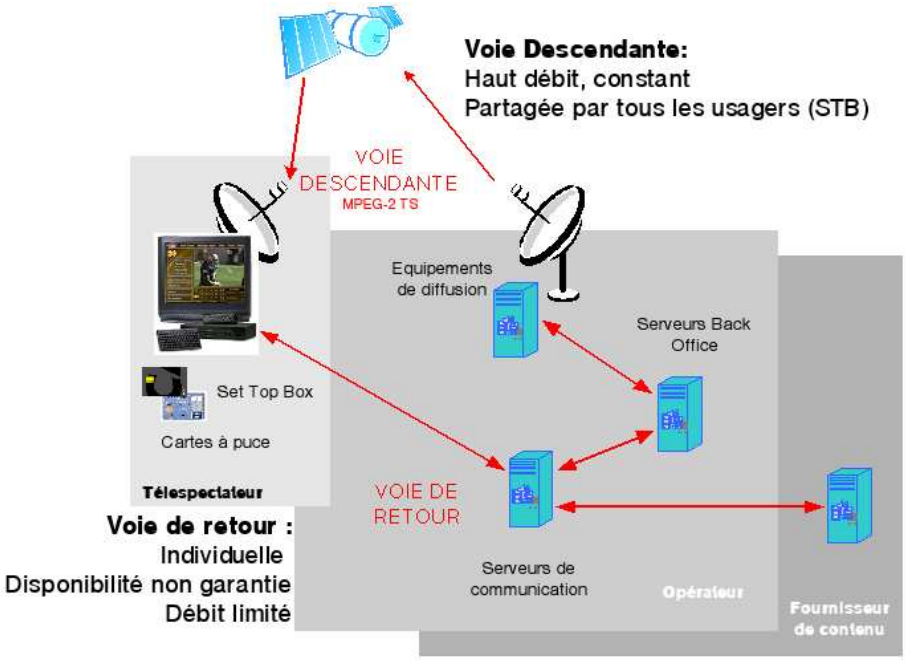

FIG. .3.1 – *Architecture de diffusion d'information par satellite*

Dans ce cadre, le travail de notre équipe a principalement constitué à réfléchir sur des solutions pour utiliser des services qui apportent une aide aux programmeurs, en le déchargeant de certaines tâches complexes, qui n'apportent pas de réelles fonctionnalités supplémentaires à l'application. Ces services sont appelés des services techniques et seront décrits dans la première partie de ce chapitre. Au final, ce projet a donc apporté un cadre applicatif aux travaux sur l'adaptabilité des services techniques, qui avaient débuté dans l'équipe, permettant de valider de nombreuses propositions, ainsi que les moyens de financer en partie la thèse de Colombe Hérault.

Après avoir rapidement présenté les services techniques, je reviendrai sur les leçons tirées des travaux menés sur le service transactionnel, pour ensuite proposer un mécanisme d'adaptation valable pour tous les services techniques. La dernière partie de ce chapitre consistera à proposer des composants de gestion, capables d'assurer l'adaptation, non pas

uniquement au déploiement de l'application, mais dès lors que le besoin s'en fait sentir. La plupart de ces recherches sont issues des travaux de thèse de Colombe Hérault, que j'ai co-encadré de 2001 à 2005 **[HER05]**. Un certain nombre de spécifications et d'implémentations de services techniques utilisants le modèle à composants sont, eux, issus de l'encadrement d'un DEA (Hocine Grine **[GRI04]**) et de projets réalisés par plusieurs étudiants de Master 2 informatique.

3.2 Définition des services techniques

3.2.1 Rôle des services techniques

Les services techniques sont des services qui n'ajoutent pas de nouvelle fonctionnalité à l'application qui les utilise : leur utilisation ne rajoute pas d'interface fonctionnelle à l'application. Ces services sont rendus directement par le bus logiciel et / ou par la structure d'accueil du composant logiciel. Ils sont instanciés lors d'invocations de méthodes sur un composant à travers son conteneur. Une telle architecture a pour but de simplifier la tâche du développeur en lui évitant de réécrire ces fonctionnalités. Il peut alors se concentrer sur la logique applicative.

Chacune des architectures à composants commerciales que nous avons décrites dans le chapitre précédent (EJB, CCM et .NET), fournit ces services techniques. Leur description est faite de façon plus ou moins standardisée, ceci étant dû aux différentes approches des consortiums et des fabricants qui les spécifient. Le tableau .3.1 présente, pour chacune de ces plateformes, des exemples de services techniques.

	Enterprise Java Bean	CORBA Component Model	.NET
Transaction	JTA / JTS	OTS	MTS
Persistance	JDBC / JDA / JDO	PSS	ADO.NET
Nommage	JNDI / Naming RMI	Nommage / Courtage	LDAP / UDDI
Sécurité	JCE	Security Service	.NET Passport

TAB. .3.1 – *Services Techniques et Plateformes à composants*

Ce tableau ne présente pas l'ensemble des services techniques, mais juste un aperçu des possibilités et de la diversité en fonction du type de plateforme utilisé. Cela permet de se rendre compte de deux choses importantes :

- Pour certains services, il existe plusieurs possibilités de gestion. Actuellement, le choix du service technique se fait soit au déploiement de l'application, soit de manière implicite, en fonction du fournisseur de plateforme. En effet, non seulement il existe plusieurs spécifications suivant que l'on utilise les EJB, CCM ou .NET. Le portage d'un service d'une plateforme à une autre peut se révéler extrêmement problématique, si les outils nécessaires ne sont pas fournis **[CER03]**. Mais de plus, pour chaque modèle, il existe des réalisations de services techniques différentes. Par exemple certaines plateformes EJB vont fournir des services transactionnels incluant les transactions emboîtées, et d'autres non.
- La diversité des services techniques : ces services agissent à différents niveaux de l'application, ce qui rend difficile (voir impossible) de les gérer tous de la même manière. Ainsi, par exemple, pour le service transactionnel, la majorité des plate-formes laissent le choix de la gestion transactionnelle au développeur de composant. Cela se fait, le plus souvent, par l'utilisation d'attributs de déploiement, insérés dans le code

du composant ou regroupés dans un fichier de déploiement. A l'inverse, d'autres services, comme le nommage, sont plus enfouis dans la plateforme, et sont alors utilisés sans spécification du programmeur des composants applicatifs.

La section suivante propose une classification des différents services techniques, en fonction des services qu'ils rendent.

3.2.2 Les différents services techniques

Les services techniques peuvent être classifiés selon plusieurs catégories, en fonction des besoins auxquels ils répondent :

- Les premiers services techniques que fournissent les plateformes, sont ceux qui gèrent la distribution des applications (transport de requêtes synchrones ou asynchrones). Généralement, chaque plateforme utilise ses propres protocoles de communication synchrones ou asynchrones (RMI, IIOP, SOAP, etc...).
- On trouve ensuite des services techniques qui sont utilisés dans la gestion du cycle de vie de l'application. On trouve ici à la fois le service de gestion de cycle de vie mais aussi les services de nommage et de courtage qui permettent de retrouver un composant applicatif. Tous ces services permettent au composant de passer d'une étape à une autre de son cycle de vie (déploiement, utilisation, mise à jour, désinstallation).
- Ensuite, nous avons des services techniques qui sont propres à la gestion des systèmes d'Information. Il s'agit de services techniques de plus haut niveau, qui effectuent des traitements plus évolués, tous en lien avec la gestion de la persistance des informations. Dans cette catégorie, nous trouvons des services comme la gestion des transactions, la persistance ou la sécurité.
- Dans la dernière catégorie, nous avons placé tous les services techniques qui ont une influence directe sur la performance d'exécution des applications. Ici, on regroupe les services de tolérance aux pannes, d'équilibrage de charge ainsi que les aspects de synchronisation. Ils permettent d'optimiser l'exécution des applications.

Cette classification n'est pas exhaustive et a simplement pour but d'expliquer au lecteur les grands ensembles de services que peut regrouper le terme "Service Technique". L'OMG [1] a par ailleurs parfaitement identifié ces services sous le terme de "CORBA Services".

3.2.3 Implantation dans les plateformes à composants

Le principe de l'utilisation de ces services techniques, dans les plateformes à composants couramment répandues, est donné par la figure .3.2. Il est globalement identique pour les trois modèles industriels existants (EJB, CCM et .NET). On peut voir qu'un composant s'exécute dans un conteneur. Ce dernier prend en charge un certain nombre de choses pour les composants applicatifs (de manière à simplifier la tâche du programmeur de composants).

Il est, par exemple, capable d'intercepter les appels aux méthodes des composants qu'il contient pour ajouter des "pré" ou des "post" traitements. C'est notamment de cette façon, que le conteneur va prendre en charge l'exécution des services techniques. Le programmeur de composants aura alors juste à indiquer de quels services son composant a besoin (soit dans le code, soit dans un fichier de déploiement), et le conteneur fera les liaisons nécessaires à l'exécution. Le tout (conteneurs, bus logiciel qui véhicule les requêtes, et services techniques) s'exécute dans une structure d'accueil, qui fournit l'environnement d'exécution.

1. OMG : Object Management Group

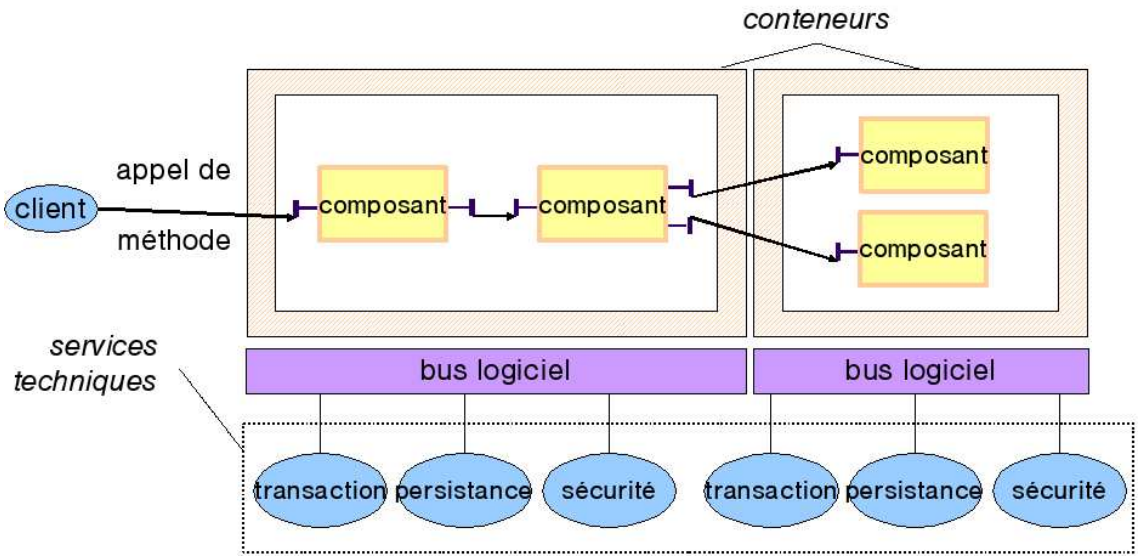

FIG. .3.2 – *Services Techniques et plateformes à composants*

3.3 Les leçons du service transactionnel

Le service technique que nous avons le plus étudié est le service transactionnel. Nous allons donc, dans cette partie, utiliser ce service pour montrer toutes les limites de la réalisation des services techniques, tel que cela est fait actuellement.

3.3.1 Les limites d'une adaptation au coup par coup

Depuis le début de mes recherches, nous avons travaillé sur plusieurs versions du service de gestion des transactions. Pour rappel, on peut notamment citer :

- Une version (basée sur OTS de CORBA) qui dispose d'une interface capable de communiquer avec une carte à microprocesseur, a été réalisée durant mon travail de thèse de doctorat **[LEC98]**,
- Une version (basée sur le moniteur transactionnel JoTM d'Objectweb) implantant les modèles des transactions emboîtées et emboîtées ouvertes **[NEM04]**, que nous avons détaillées dans le chapitre précédent.

D'autres travaux, dans d'autres équipes, ont également contribué à faire évoluer ce service. On peut notamment citer les tous premiers travaux de l'INRIA sur MAAOTS **[LS98]** (le but de ces recherches était de fournir un moniteur transactionnel basé sur un noyau auquel on pouvait adjoindre des interfaces de communication pour le rendre compatible avec différentes architectures) ou alors les travaux de proposition d'un mouveaux protocole de validation distribuée pour environnement mobile **[AP98]**, qui ont pour but d'éviter les problèmes de blocation du protocole de validation à deux phases.

Au final, en fonction de différents critères, on obtient un nombre de versions de services transactionnels assez important :

- Le type de terminal utilisé peut impliquer des modifications d'implantation liées aux capacités de ce terminal (comme pour le moniteur transactionnel utilisable avec les cartes à microprocesseur)

- La plateforme d'exécution utilisée peut impliquer des besoins spécifiques en terme de protocole de communication (comme le problème traité par MAAOTS)
- Le besoin applicatif peut impliquer la nécessité d'utiliser des modèles transactionnels spécifiques comme, par exemple, les transactions emboîtées ouvertes **[JK97]**.

Toutes ces contraintes (matériels, communication, besoin applicatif) peuvent bien entendu se combiner. Avec les méthodes de conception de services techniques actuelles, il devient alors difficile de réaliser facilement des moniteurs transactionnels devant répondre à des contraintes très différentes. De même, il n'est pas possible de réaliser un moniteur transactionnel qui implante tous les modèles, tous les protocoles de communication, et qui peut fonctionner avec tous types de terminaux. Il devient donc nécessaire de fournir des outils pour obtenir des intergiciels robustes et adaptables quelque soit leur environnemenr d'exécution et les besoins applicatifs **[DBC04]**, et plus particulièrement, dans le cas qui nous concerne, des méthodes pour réaliser à moindre coût une adaptation du service transactionnel **[KAR03], [HBD02]**.

3.3.2 L'impasse des objets notoires

Sur les serveurs applicatifs, les services techniques sont référencés sous la forme d'objets notoires. Il s'agit de références d'objets, connus "à l'avance" de la plate-forme, et qu'elle n'a pas besoin d'aller rechercher dynamiquement. Certains services CORBA fonctionnent notamment de cette façon. Ainsi, lors du déploiement de la plate-forme, les classes représentant les services techniques, ainsi que les adresses et ports de communication, sont indiqués dans des fichiers de déploiement (ou plutôt de configuration). Lors de l'exécution d'une application sur ces plate-formes, il n'est donc pas possible de supprimer ou de modifier l'implantation d'un service technique (cf.figure .3.3) **[DUC02]**.

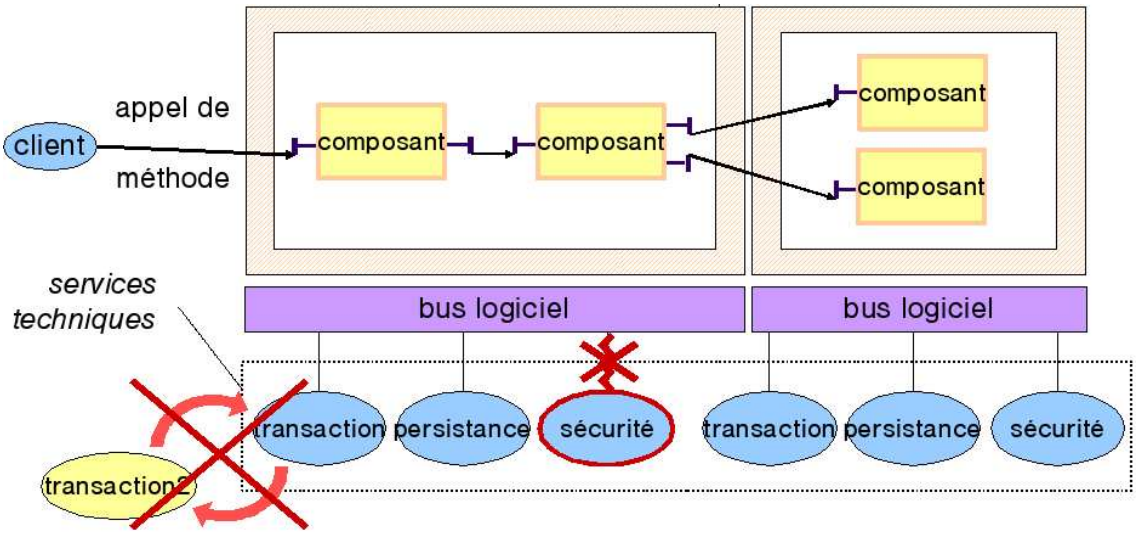

FIG. .3.3 – *L'impasse des objets notoires*

Cette technique, permettant l'initialisation de la plate-forme au démarrage de l'application, a pour principal inconvénient de rendre totalement statique la gestion de ces services techniques. Il est alors impossible d'utiliser deux versions différentes de services techniques sur une même plate-forme (par exemple, deux versions de gestionnaires de transactions).

3.3.3 Les possibilités du modèle à composants

Les pistes suivies lors du travail sur l'implantation des transactions emboîtées ouvertes nous ont déjà prouvé que le modèle à composants était intéressant pour résoudre un certain nombre de nos problèmes (cf. chapitre .2, sur la réalisation des techniques de compensation à l'aide des composants). Cela nous a permis de fournir un modèle, jusque là difficilement implémentable (les ONTs), tout en ayant une forte réutilisation du code existant (puisque nous avons procédé par extension du code du moniteur JoTM).

D'autres équipes ont travaillé dans la même voie, toujours dans le but de facilement étendre le service de gestion de transactions. On peut notamment citer les projets GoTM **[RM03]** et Jironde **[JIR03]**, qui sont maintenant deux sous-projets de JoTM. GoTM a l'avantage de proposer un modèle de conception de services transactionnels à base de composants pour facilement répondre aux besoins des nouvelles applications. Sur les mêmes bases que celles que nous avons suivies pour réaliser des transactions emboîtées ouvertes, GoTM va beaucoup plus loin dans le découpage en composants, en permettant ainsi (par ré-assemblage) de fournir facilement différentes personnalités de moniteur transactionnel. Le temps de développement se trouve ainsi considérablement raccourci. Jironde propose une solution pour intégrer les services techniques dans le modèle à composants Fractal. Cette solution est dans le même esprit que celle que nous avons utilisée dans le prototype de Sergiy Nemchenko.

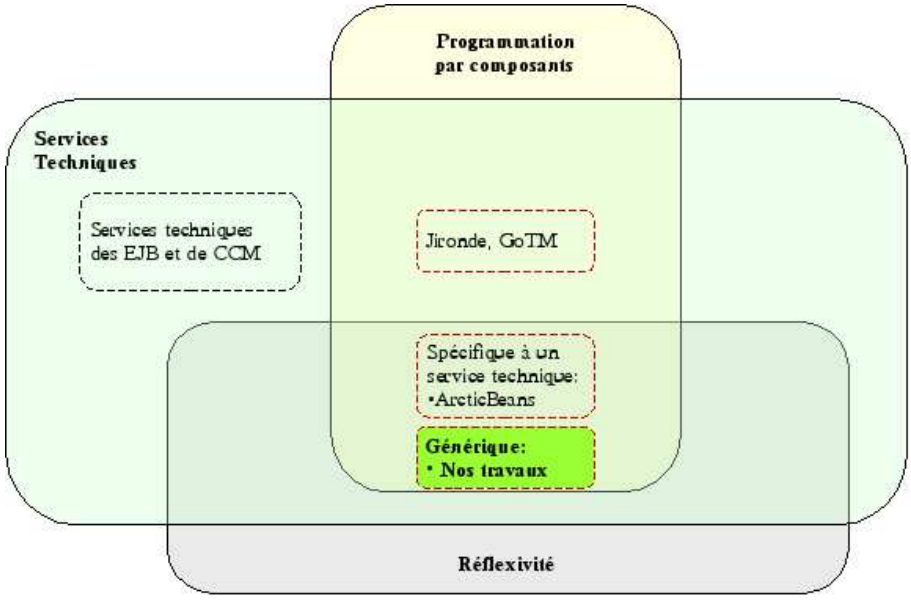

FIG. .3.4 – *positionnement de nos travaux dans ObjectWeb*

Comme le montre la figure .3.4, ces deux projets (GoTM et Jironde) prouvent que le modèle à composants va offrir la modularité nécessaire pour "facilement" réaliser des services techniques répondant aux besoins des applications. Cela répond pour une bonne part aux problèmes exposés dans la partie 3.3.1.

Enfin, comme souligné par **[DUC02], [MSK04]**, la plupart des travaux de recherche menés autour des systèmes adaptables sont à la croisée de trois concepts : la program-

mation par composant, la séparation des préoccupations (et notamment la séparation du code technique et applicatif), et la réflexivité des systèmes. L'utilisation de la propriété de réflexivité, proposée par certains modèles à composants (comme Fractal) peut offrir également des solutions au problème de la gestion des services techniques à l'aide des objets notoires (un système réflexif étant un système qui permet d'agir et de raisonner sur lui même **[MAE87] [SMI84]**). Par exemple, le projet ArticBeans **[ABG01]** est basé sur un modèle à composants récursifs. Ce travail a pour but d'adapter les services de sécurité et de transactions. Dans ce projet, ils fournissent également un langage permettant de décrire les besoins des applications en terme de sécurité. Ainsi, les services de sécurité et de transactions peuvent évoluer au fur et à mesure des besoins, et ne sont plus fixés au démarrage de la plate-forme. Cependant, dans ce projet, des solutions spécifiques sont présentées pour le service de sécurité, puis pour le service de transactions. Aucune approche globale pour l'ensemble de services techniques n'a été proposée. Dans le cadre de l'adaptabilité dynamique des services applicatifs (et non plus techniques), il est également recommandé d'avoir recours à des modèles à composants reflexifs **[CRM04]**, de manière à rendre les adaptation les plus transparentes possibles pour l'utilisateur, et qui améliorent la qualité de service qu'ils fournissent de part la conscience de leur propre environnement **[GEI01], [KCB02]**.

D'autres travaux dans ce domaine existent, notamment les travaux de Eric Bruneton **[BRU01]**, sur la définition d'un support d'exécution pour l'adaptation des services techniques. Dans ces travaux, sont définies les notions de base sur les trois niveaux d'adaptabilité des services techniques (statique, dynamique et auto-adaptabilité), ainsi ques des solutions intéressantes pour l'adaptabilité statique des services technique dans le modèle à composants (inspiré par le modèles des EJB). Tous ces travaux sont résumés dans le manuscript de thèse de Colombe Hérault **[HER05]**. Je n'ai présenté ici que les travaux les plus proches de nos réalisations.

3.4 L'adaptabilité statique des services techniques

Une fois les besoins d'adaptabilité des services techniques établis, nous avons défini différents niveaux d'abstraction de la solution envisagée (cf. figure .3.5). Cette approche, initiée en 2001/2002, a été influencée par l'approche MDA [2], maintenant largement répandue.

Dans un premier temps, sans se soucier de l'architecture finale sur laquelle nous implanterons notre solution, nous avons travaillé sur un modèle de conception pour service technique. Comme nous le verrons dans la première partie de cette section, nous nous sommes basés sur le modèle à composants, tout en restant indépendant des différentes plate-formes existantes. Puis, nous avons identifié la notion de *personnalité*, qui est présentée dans la seconde partie de cette section, et qui permet d'identifier une version donnée d'un service technique. Nous verrons également comment cette notion nous a permis de faciliter l'adaptation d'un service technique lors du déploiement d'une application.

Puis, nous avons choisi d'appliquer nos solutions au modèle composants à Fractal, et de fournir une implantation sur Julia, l'implantation standard de Fractal. Cette section va donc reprendre les différentes phases de ces travaux, en commençant par la conception des

2. MDA : Modèle Driven Architecture

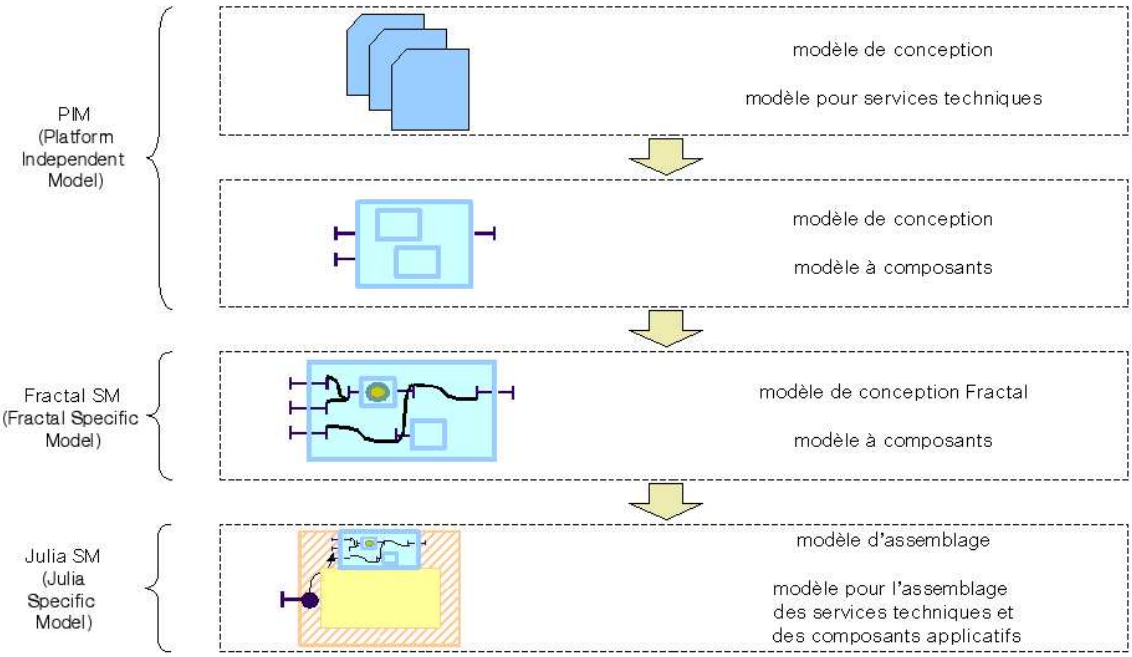

FIG. .3.5 – *Un nouveau modèle de conception pour les services techniques*

services techniques à base de composants.

3.4.1 Les services techniques à base de composants

3.4.1.1 Principe

Comme nous l'avons vu précédemment, la programmation orientée objet n'est pas la mieux adaptée pour réaliser des services techniques facilement modulables en fonction des besoins des applications et des caractérisques des terminaux qui les exécutent. Par exemple, si l'on veut reconfigurer les interactions entre les objets, une recompilation du code est nécessaire. Après ces constatations, et l'expérience du moniteur transactionnel gérant les ONTs, nous avons donc proposé de généraliser l'utilisation de la programmation par composants pour réaliser les services techniques **[HLD04]**. Les avantages sont les suivants :

- Un composant permet de modéliser la notion de tâche. Ainsi, on peut définir un service technique comme étant un assemblage de plusieurs composants, effectuant chacun des tâches élémentaires.
- La réutilisabilité du code entre deux versions d'un service est améliorée : un service technique, quelque soit les modèles qu'il implante, repose toujours sur un noyau identique. Ce noyau sera formé d'un assemblage de composants, qui pourra être partagé entre toutes les versions d'un service donné (par exemple, la structure de stockage d'un annuaire pourra être partagée entre service de nommage et un service de courtage, ou le moteur de validation distribuée d'un moniteur transactionnel (contenant les identifiants des participants aux transactions).
- La notion d'interface fournie d'un composant nous permet d'exprimer l'interface d'utilisation de chaque version du service technique.
- Les liaisons entre composants permettent de redéfinir des assemblages, qui vont permettre de reconfigurer rapidement les services techniques pour les adapter à un chan-

gement de contexte.

3.4.1.2 Application au modèle Fractal

Tout comme pour nos travaux sur la spécification et l'implantation des ONTs, le modèle à composants Fractal **[BCS04]** s'est révélé comme étant la meilleure solution pour la réalisation de notre modèle d'adaptation. En effet, en plus des concepts de la programmation par composants, dont les avantages ont été donnés dans la section 3.4.1.1, le modèle Fractal offre des spécificités qui répondent bien à nos besoins :

- Fractal utilise la notion de *composant composite*. Une composition (un assemblage) de composants, est elle même vue comme un composant à part entière Quelque soit l'assemblage qui va fournir la réalisation du service technique, au final il sera vu comme un seul et unique composant à assembler avec le composant applicatif. Cette caractéristique est définie dans les notions globales de la programmation par composants. Cependant très peu de modèles l'implantent, et elle n'est disponible sur aucun des modèles "commerciaux" (EJB, CCM et .NET).
- Fractal permet le *partage de composants*. Un composant Fractal peut être partagé entre plusieurs assemblages de composants. Ainsi, une ressource pourra facilement être partagée par plusieurs versions d'un même service. Cela facilitera la transition d'une version à une autre (lors de l'adaptation).
- Fractal repose sur la *réflexivité*. En effet, Fractal offre l'introspection et l'intercession d'un composant **[BCS04]**. Avec l'introspection, ces composants sont donc capables de décrire leurs interfaces, leurs liaisons et leur contenu (en terme de sous composants). Avec l'intercession, on peut dynamiquement modifier l'assemblage de composants au cours de l'exécution du composant. Cela nous servira dans le cadre de l'adaptabilité dynamique.

Nos travaux n'ont pas eu pour but d'étendre le modèle de programmation par composants Fractal. nous l'avons juste utlisé pour appliquer nos concepts, car à nos yeux, il s'agit du modèle qui permet, actuellement, le mieux de respecter les concepts fondamentaux qui sont à la base de la programmation à composants **[RM00]**

3.4.1.3 Un exemple : l'annuaire

Afin de mettre en pratique la méthode de conception préconisée ici, nous avons réalisé un service technique d'annuaire. Un annuaire est un service reprennant les fonctionnalités des pages blanches (recherche d'un objet ou d'un composant en fonction du nom symbolique qui lui a été donné) et de pages jaunes (la recherche s'effectue alors en fonction de propriétés dont dispose l'objet ou le composant). La réalisation de notre service, pour mettre en oeuvre les principes énoncés précédemment, est basé sur le modèle à composants Fractal, reprenant à la fois les fonctionnalités d'un service de nommage (synonyme de pages blanches) et d'un service de courtage (représentant les pages jaunes). Le cahier des charges de cette réalisation a été :

- Un maximum de composants partagés entre le service de nommage et le service de courtage.
- La possibilité d'utiliser soit le service de courtage, soit le service de nommage, soit les deux simultanément (nous verrons dans la partie suivante, que cela correspond à la définition de trois personnalités différentes du même service).
- Le partage de la structure de stockage des informations, de manière à pouvoir passer

d'une personnailité "Nommage", à une personnalité "Courtage", puis à une personnalité reprennant les deux services, rapidement, et sans perte d'information.

- Notre annuaire devra servir à retrouver des instances de composants fractal, des patrons ou des types de composants.

Ce travail est la conclusion de l'encadrement de six projets individuels de Master2, ces projets ayant été répartis sur deux années (2003/2004 et 2004/2005). La description serait trop longue à faire ici, et j'invite le lecteur à consulter les annexes de la thèse de Colombe Hérault **[HER05]**, où il retrouvera la description de tous les composants Fractal constituant cet annuaire.

Les conclusions de ce travail ont été extrêmement importantes pour la suite de nos travaux :

- Plus le découpage en composants Fractal est fin, plus le partage de composants est facilité. De plus, cela a pour avantage de diminuer la taille globale du code du service technique.
- La définition de la structure de stockage commune pour les services de courtage et de nommage (à l'aide d'un composant partagé), permet de facilement passer d'une version à l'autre de l'annuaire, car les informations concernant l'exécution en cours du service sont conservées bien que l'on change l'assemblage. La durée d'indisponibilité du service est donc très réduite. De manière générale, pour facilité l'adaptation, il conviendra de tourjours partager les données nécessaires à l'exécution du service technique, entre toutes les personnailités du service technique. Cela permettra également de conserver la cohérence des données entre les différentes personnailités d'un même service.

3.4.2 La notion de personnalité

L'objectif de la définition d'une personnalité **[HBD02]** est de permettre d'identifier, de manière unique, chacune des versions d'un service technique. Cette description (dont un exemple est donné par la figure .3.6), va donc offrir une description du service technique, qui permettra par la suite au développeur d'applications (pour l'adaptabilité statique), ou au système (pour l'adaptabilité dynamique), de facilement choisir son implantation de service technique.

Ainsi, un service technique va être doté de plusieurs personnalités (que nous nommons P1S, pour Personnalité d'un service). Par exemple, plusieurs P1S peuvent définir les deux services techniques présentés précédemment (cette liste n'est pas exhaustive, car des personnalités peuvent être ajoutées au fur et à mesure de l'évolution des services) :

- Pour le service transactionnel, on va, par exemple, avoir une personnalité représentant le service ne pouvant exécuter que des transactions plates, puis une autre offrant un service complet (transactions plates, emboîtées, et ONT)
- pour l'annuaire, on peut disposer d'une personnalité "Nommage", ou "Courtage", voire même d'une personnalité complète, comme pourrait l'être un annuaire LDAP.

La figure .3.6 présente le contenu de ces personnalités. Dans la description des personnalités, certains facteurs vont être indispensables à sa correcte identification, d'autres optionnels :

- Les facteurs indispensables sont les renseignements minimums, qui doivent être connus pour décrire les fonctionnalités du service. Cela reprend en priorité la "fonction", qui

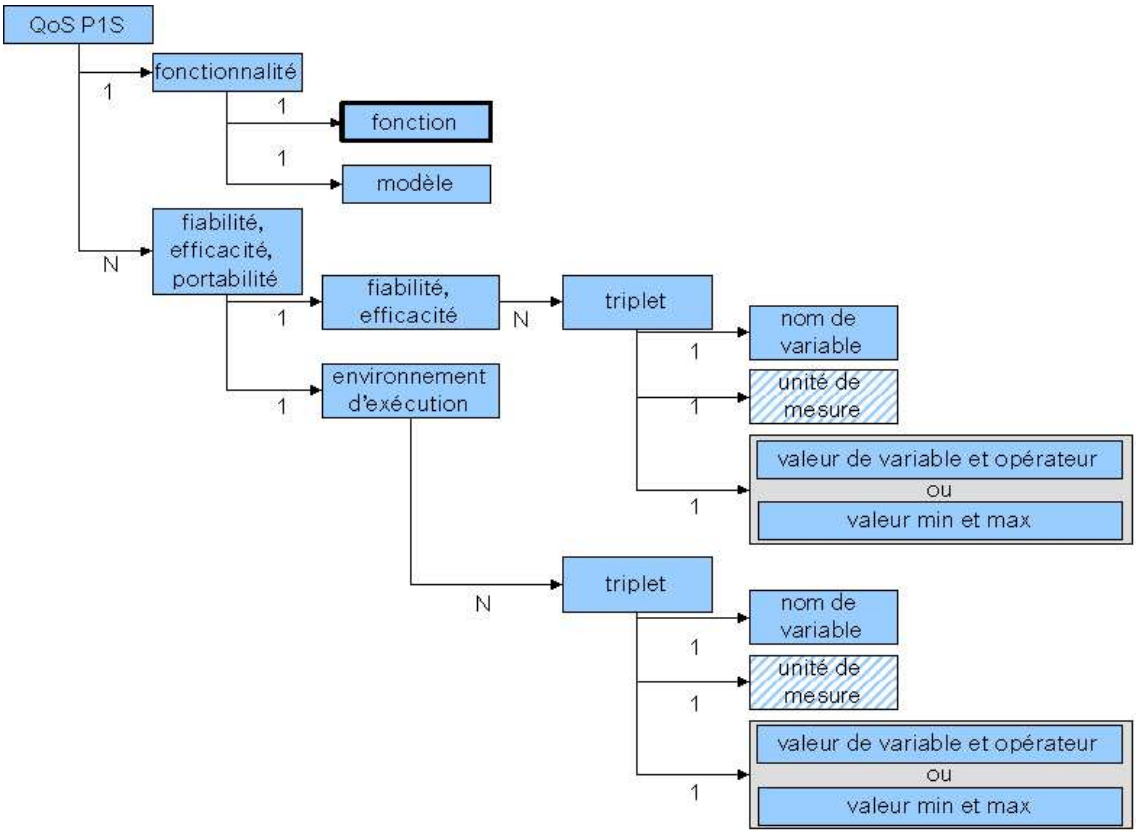

FIG. .3.6 – *Définition d'une personnalité*

est l'équivalent de l'identifiant de l'objet notoire (par exemple "NamingService" pour le service de nommage).

- Les facteurs optionnels vont permettre d'affiner le choix de la personnalité. On trouve notamment :
  - Le modèle, qui vient compléter le niveau "fonctionnalité" en indiquant le modèle implanté par la personnalité (par exemple "ONT" pour le service de gestion de transaction).
  - Les facteurs de "performances", comme "Efficacité", "Fiabilité" et "Portabilité", qui vont permettre de donner des informations de performances, comme par exemple le nombre de transactions par minute pour un service de transaction, ou la vitesse CPU nécessaire à la bonne exécution du service. Un exemple détaillé sera donné dans la partie 3.4.4.

Au final, cette P1S est renseignée par le développeur du service technique, en utilisant, dans un premier temps, les modèles standardisés par des organismes comme l'OMG, puis en utilisant des outils empiriques, comme les benchmarks, qui vont permettre d'évaluer les performances du service, en fonction de son environnement d'exécution **[HL04]**.

3.4.3 Adaptation au déploiement

Comme le montre la figure .3.7, Fractal fournit des conteneurs (ou "membranes" dans la terminologie Fractal) suffisamment ouverts et flexibles, pour ajouter les composants de services techniques, et les utiliser à l'aide d'un intercepteur.

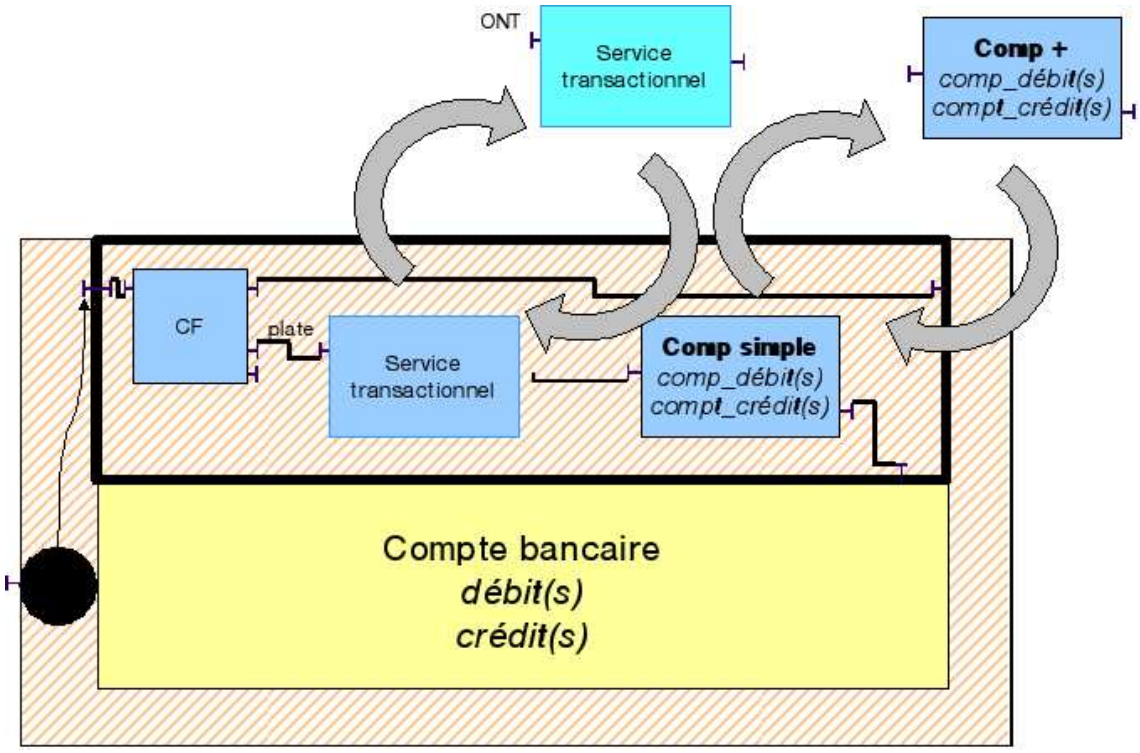

FIG. .3.7 – *Application dans Fractal*

Ainsi, le principe d'utilisation de nos composants de services techniques est le suivant :

- L'intercepteur (représenté par le cercle noir sur la figure .3.7) est placé sur l'interface métier du composant applicatif. Il a pour but d'intercepter les requêtes vers le composant applicatif pour faire effectuer des traitements par le sous-contrôleur, soit avant l'exécution de la requête (méthode "pré"), soit après (méthode "post"), par un mécanisme de réification de la requête. Par exemple, pour le service transactionnel, la méthode "pré" va correspondre au *Begin*, et la méthode "Post" au *Commit*.
- Nous définissons les sous-contrôleurs comme des composants composites, qui vont contenir le code des services techniques. Pour chaque service technique, le sous-contrôleur va se composer de différents composants **[HL03]** :
  - le composant de façade (noté CF sur la figure .3.7), est directement invoqué par l'intercepteur à l'aide des méthodes "pré" ou "post". Il sert de wrapper pour transformer la requête de l'intercepteur, en requête conforme au service technique (par exemple, pour le service transactionnel, transformer un "Pré" en "Begin" conforme aux standards de l'interface "Current" du modèle OTS de l'OMG).
  - le service technique (proposé sous l'exemple du service transactionnel sur la figure .3.7), est représenté sous la forme d'un composant exposant plusieurs interfaces, qui représentent chacune une P1S. L'une de ces P1S est liée avec le composant de façade. Il s'agit de celle qui est utilisée par le composant applicatif considéré. Le choix de la P1S s'effectue donc au déploiement du composant applicatif, lorsque les liaisons entre les composants sont effectuées. Dans l'exemple de la figure .3.7, un composant de services techniques exposant une

P1S "transaction plate", peut être remplacé par un composant qui expose une P1S "ONT". De même, nous aurions pu directement utiliser un composant qui expose les 2 P1S, et la P1S utilisée aurait alors été choisie par une liaison avec le composant CF. Enfin, ce composant de service technique peut être partagé entre plusieurs composants applicatifs Fractal (afin d'éviter une forte duplication du code technique).

- un composant de Callback est présent pour gérer le code de retour après exécution du service technique. Ce code, développé par le développeur du composant applicatif, fait le lien entre le comportement du service technique et le composant applicatif. Dans l'exemple donné par la figure .3.7, le composant de compensation des transactions de type ONTs, que nous avons défini dans le chapitre précédent, est fourni sous la forme de composants de Callback.

Avec cette solution, la séparation du code applicatif et du code technique est garantie, dans la mesure où les services techniques sont assurés par les sous-contrôleurs, qui font partie de la membrane du composant Fractal. La règle, selon laquelle, le service technique doit être géré par le conteneur du composant applicatif, est donc respectée.

D'autre part, même si cela n'est pas représenté par la figure .3.7, on peut ajouter au composant applicatif autant de services techniques qu'on le désire. Il suffit pour cela d'ajouter plusieurs sous-contrôleurs, qui seront exécutés les uns après les autres.

3.4.4 Exemple du service transactionnel

Lors des travaux détaillés dans le chapitre précédent, sur les modèles de transactions avancés pour les nouvelles applications sur le Web, le modèle à composants s'était déjà révélé comme étant la meilleure solution pour implanter la notion de compensation dans les transactions, et donc pour réaliser un moniteur transactionnel gérant les ONTs.

Afin de pouvoir utiliser le service transactionnel réalisé dans le travail de Sergiy Nemchenko, nous avons défini des personnalités P1S différentes pour le service transactionnel gérant les transactions plates et celui gérant les ONTs. Des extraits de ces P1S sont donnés par la figure .3.8.

Pour choisir entre l'une ou l'autre des personnalités, deux critères sont utilisés : l'environnement d'exécution (notamment la fréquence du CPU (donnée statique obtenue au déploiement du service technique) et l'utilisation du CPU (donnée dynamique, régulièrement rafraîchie)), et les besoins applicatifs (une application de type WebServices nécessitera un moniteur transactionnel de type ONTs).

Nous présentons ici un moniteur transactionnel avec seulement deux personnalités. Cela est dû au fait que ce moniteur transactionnel, avant toute chose, a été réalisé pour tester nos spécifications pour les ONTs. Chaque modèle de transaction présenté dans le chapitre précédent (plates, emboîtées, ONT, SAGAS, Workflows, etc...) peut donner naissance à une personnalité, sous réserve que le moniteur transactionnel qui les réalise soit modulaire et réalisé sous forme d'assemblage de composants.

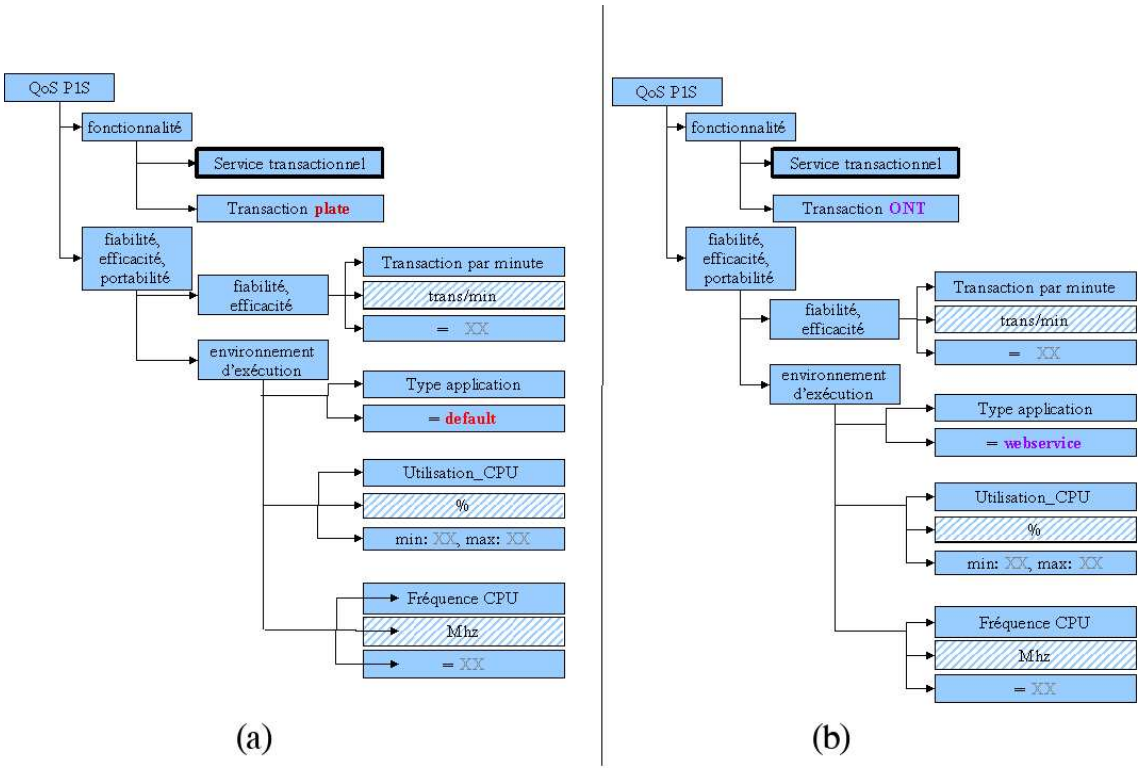

FIG. .3.8 – *Extrait des P1S (a) transactions plates (b) transactions ONTs*

3.5 Plus de flexibilité : l'adaptabilité dynamique

Dans la section précédente, nous avons proposé une méthode de conception des services techniques, permettant de fournir plusieurs "personnalités", appelées P1S, d'un service technique. Nous avons également étudié comment lier ce service technique, à un composant applicatif Fractal.

Dans cette section, je vais détailler l'architecture que nous avons fournie, pour permettre une adaptation dynamique du service technique utilisé, en fonction des besoins de l'application exécutée, et des capacités des terminaux prenant part à cette application.

Pour cela, dans un premier temps, nous étudierons les éléments déclencheurs de cette adaptation, puis, nous définirons les différents éléments de notre architecture d'adaptation dynamique, en nous penchant sur un élément particulier : l'annuaire permettant de retrouver les bonnes personnalités des services techniques.

3.5.1 La boucle d'adaptation

La boucle d'adaptation est un cycle dans lequel un service technique peut être dans deux états : soit il est dans l'état "Adapté", soit il est dans l'état "Pas adapté". Les transitions entre ces deux états sont déclenchées par deux évènements : le changement d'environnement (qui fait passer l'état de "Adapté" à "Pas adapté"), et le changement de P1S du service technique (qui fait passer l'état de "Pas adapté" à "Adapté").

Le changement d'environnement peut se produire de deux façons différentes :

- lorsque l'application est conçue pour s'exécuter dans deux environnements différents, dès sa conception. Par exemple, lorsque l'application est conçue pour s'exécuter à la fois sur un ordinateur de type PC et sur un PDA de type PocketPC. Dans ce cas, une adaptation de type statique, lors du déploiement de l'application sur le terminal considéré, suffira.
- Cependant, avec la multitude de terminaux maintenant à la disposition de l'utilisateur, il est de plus en plus difficile de prévoir quels sont les terminaux qui vont être utilisés, et quelles vont être leurs caractéristiques. Il est donc possible que l'application change de contexte au cours de son exécution. On peut prendre l'exemple d'une application fonctionnant sur un PDA en mode déconnecté, puis en mode connecté (arrivée sur un HotSpot), avec récupération de nouveaux partenaires sur le réseau. Dans ce cas, l'adaptation des services techniques doit être plus dynamique, car elle peut intervenir à n'importe quel moment.

Cette boucle d'adaptation est propre aux services techniques, et est relativement simple, car ne prend pas en compte les préférences utilisateurs (le service technique étant transparent pour ce dernier). L'étude d'un méta-modèle, permettant de décrire le comportement des applications, notamment dans les phases où le besoin d'adaptation se fait sentir, n'est pas finalisée, et est en cours de spécification dans le cadre du projet MOSAIQUES, en collaboration avec l'équipe GOAL du LIFL et l'équipe TRIGONE du CUEEP.

3.5.2 Une architecture pour l'adaptabilité dynamique

L'architecture pour l'adaptabilité dynamique repose sur un ensemble de composants de gestion **[HL04b]**qui sont représentés par la figure .3.9. Dans un premier temps, nous allons présenter les composants les uns après les autres, puis, nous présenterons les interactions entre ces composants de gestion.

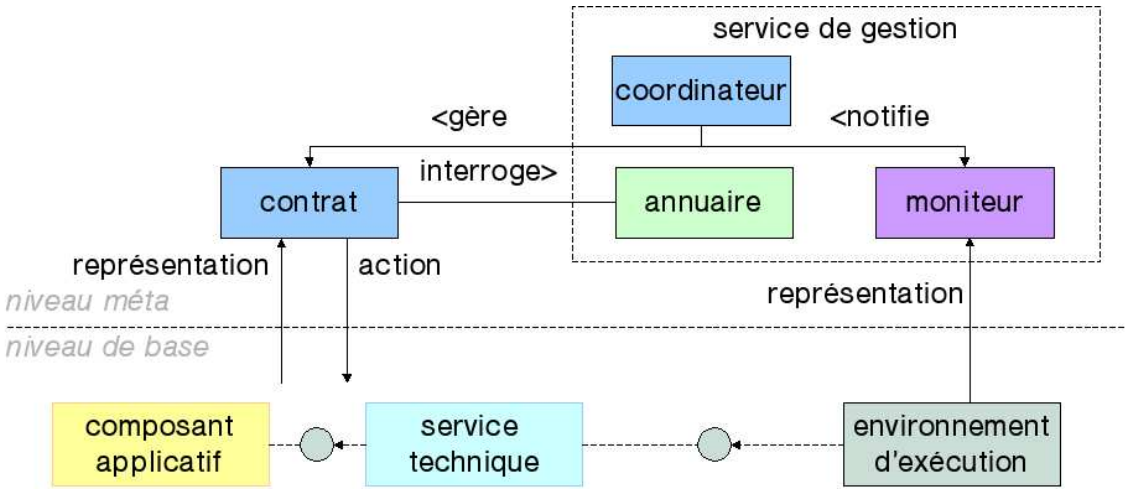

FIG. .3.9 – *Les composants de gestion pour l'adaptabilité dynamique*

L'adaptation consiste à localiser, configurer et utiliser les différentes P1S des services techniques nécessaires au bon fonctionnement de l'application. Pour effectuer ces tâches sans intervention de l'utilisateur ou du programmeur, nous avons défini un coordinateur qui gère la cohérence du système; un contrat qui va représenter la liaison entre le composant

applicatif et la P1S choisie; un annuaire qui va localiser l'ensemble des P1S disponibles pouvant convenir aux besoins applicatifs et enfin le moniteur qui remonte les informations sur l'environnement d'exécution. Nous allons maintenant détailler le fonctionnement de chacun de ces éléments.

3.5.2.1 Le contrat

Pour modéliser le lien entre un composant applicatif et le service technique, nous avons défini la notion de contrat. Il y aura en fait deux contrats. Le premier représente l'adéquation entre le service technique et l'environnement, sur lequel il va s'exécuter. Cela se traduit par l'idée que si l'environnement d'exécution correspond aux besoins du service technique, alors ce dernier garantit (par le contrat) une certaine qualité de service au composant applicatif (qui est indiqué dans sa P1S).

Un autre contrat devra représenter l'adéquation entre l'attente de l'application (c'est à dire, le besoin de l'application en terme de services techniques), et la qualité de service que s'engage à rendre le service technique. Les besoins du composant applicatif sont spécifiés par le programmeur d'application dans un fichier de déploiement (cela est déjà utilisé dans d'autres modèles à composants).

De manière à simplifier la tâche de l'annuaire (que nous étudierons dans la partie 3.5.2.4)) le composant de gestion des contrats va regrouper les informations sur l'environnement d'exécution (recupérées grâce à un moniteur présenté dans la section suivante) et les besoins applicatifs, pour que l'annuaire de courtage puisse les comparer au diffèrentes P1S dont il dispose.

En cours d'exécution, on considère que l'application ne change pas. Le seul critère d'adaptation dynamique devient alors l'environnement d'exécution. En cas de changement de l'environnement d'exécution, le contrat sera ré-évalué. Si la P1S sélectionnée ne répond plus aux conditions, un nouveau contrat sera négocié et une nouvelle recherche dans l'annuaire sera effectuée.

3.5.2.2 Le moniteur

Le moniteur est chargé de fournir les informations sur l'environnement d'exécution de l'application, pour les communiquer au coordinateur, afin qu'il évalue de façon adéquate la validité des contrats. Les informations fournies par le moniteur, sont des données statiques ou dynamiques, telles que la puissance CPU de la machine cible (donnée statique), ou la bande passante réseau disponible (donnée dynamique). Ces données matérielles sont également complétées par des données logicielles, telles que la présence d'une machine virtuelle Java, ou d'un bus CORBA. L'ensemble des informations fournies par le moniteur sont décrites dans le mémoire de thèse de Colombe Hérault **[HER05]**.

Le moniteur est en fait un programme système, qui tourne en permanence sur l'ensemble des terminaux. Pour cela, le moniteur est le seul élément de notre architecture qui n'a pas été écrit à l'aide du modèle à composants. Les différents moniteurs ont été réalisés dans des langages les plus proches des langages machines pour chaque architecture (généralement en langage C). Toutes les informations remontées par le moniteur sont stockées dans un fichier XML dont la structure est commune à toutes les versions (à l'heure actuelle, le moniteur est fonctionnel sur plateforme PocketPC, Palm et PC) (cf. partie 3.5.4.

3.5.2.3 Le coordinateur

Le rôle du coordinateur est de maintenir la cohérence du système, et donc de récupérer les informations des moniteurs, pour mettre à jour les contrats. Pour cela, la première chose que va faire le coordinateur au déploiement d'une application, est de créer les contrats correspondants. Cela lui permet de détenir les références de tous les contrats valides sur le système, ce qui induit de pouvoir les modifier facilement. Ensuite, lorsque le moniteur perçoit une modification notable de l'environnement d'exécution, il notifie le coordinateur qui va alors mettre à jour les contrats selon le modèle *Push*. Le coordinateur pourra alors sélectionner les contrats concernés par ce changement de contexte. Ce modèle de communication, basé sur un modèle à évènements, permet de ne pas surcharger les contrats et de multiplier les demandes de mise à jour.

3.5.2.4 L'annuaire

Dans les systèmes actuels, les services techniques sont "localisés" grâce à des informations entrées dans les fichiers de configuration de la plateforme, sous la forme d'objets notoires. Nous avons expliqué précédemment, que cette solution n'était pas envisageable dans un système où le service technique pouvait changer pour s'adapter aux besoins de l'application et à l'environnement. Pour gérer l'ensemble des P1S des différents services techniques, nous avons donc défini un service de localisation de service technique qui est présenté par la figure .3.10. Ce travail a été effectué dans le cadre du DEA de Hocine Grine **[GRI04]**, et a donné lieu, pendant ce DEA, à une publication aux journées composants **[GHL05]**.

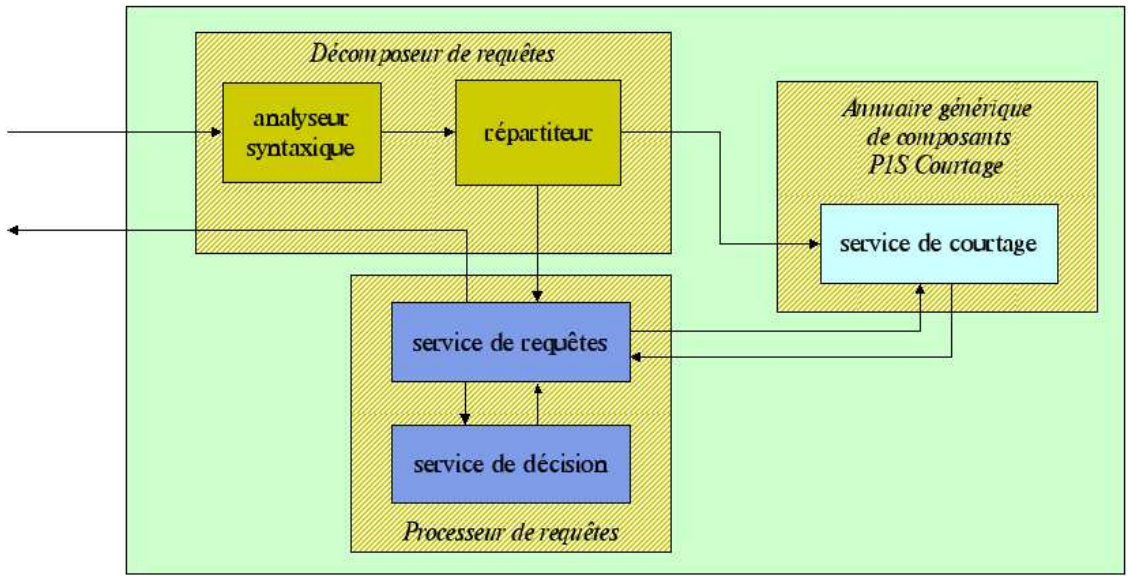

FIG. .3.10 – *Architecture du service de localisation de service technique*

Cet outil, bien qu'il ne soit pas lui même un service technique, repose, pour sa partie courtage de service technique, sur l'annuaire que nous avons défini dans la partie 3.5.2.4. Nous avons fait cela dans un but de réutilisation du code. Puis, de manière à pouvoir répondre à notre problématique, nous avons greffé à cet annuaire un mécanisme de courtage sémantique, du type de TORBA pour CORBA **[MMG01]**. Le courtage sémantique

permet en effet de fournir un service qui correspond exactement aux contraintes exprimées par l'utilisateur.

En plus d'un annuaire classique, notre service de localisation de services techniques est donc constitué de deux autres modules :

- Le gestionnaire de requêtes : il va faire une première analyse qui permet de déterminer le type de la requête ("import" ou "export"). Il s'agit simplement d'un analyseur syntaxique et d'un répartiteur de requête, qui va transférer la requête à l'annuaire traditionnel et au processeur de requête.
- Le processeur de requête traite uniquement les requêtes d'import, et va fournir une analyse sémantique pour améliorer la qualité de la réponse de l'annuaire, et fournir le service qui correspond le mieux aux besoins.

Une requête d'export (enregistrement dans l'annuaire), va juste consister à enregistrer une référence vers une instance du service technique avec sa description (l'annuaire peut également gérer les enregistrements de type ou de patron de composants Fractal, mais dans un souci de simplification, nous n'utilisons que l'enregistrement d'instance pour les services techniques).

La requête d'import, permet à un contrat de retrouver le service technique adéquat. Cette requête contient le profil requis par l'application, en terme de service technique (donné par le fichier de déploiement de l'application), et des informations sur l'environnement d'exécution (en provenance du moniteur). L'analyseur syntaxique extrait les informations des besoins applicatifs et des contraintes de l'environnement, pour générer une requête recevable par le service de courtage. Ce dernier renvoie une liste de résultats au service de décision, qui va classer les résultats en fonction de préférences que le programmeur peut indiquer dans son fichier de déploiement. Au final, la référence vers l'instance du service technique correspondant le mieux, est renvoyée au contrat, qui se charge de mettre à jour la liaison.

3.5.3 Interactions

Dans la partie précédente, nous venons de présenter les différents éléments de notre architecture pour l'adaptation dynamique. Nous allons maintenant schématiser les interactions entre ces différents éléments, qui sont représentés par la figure .3.11.

Le déploiement d'un composant applicatif se fait en transmettant, au coordinateur, le fichier de profil, représentant les besoins de l'application en terme de services techniques. D'autre part, le coordinateur est tenu d'informer de l'évolution du profil de l'environnement d'exécution en permanence. Avec ces deux informations, il va demander la création d'un nouveau contrat, représentant à la fois les besoins applicatifs, et les contraintes d'exécution. Ce contrat va utiliser l'annuaire pour rechercher le (ou les) service(s) technique(s) répondant le mieux aux besoins. A chaque évolution de l'environnement d'exécution, le contrat sera mis à jour, ce qui lancera une nouvelle requête au niveau de l'annuaire.

3.5.4 Réalisation

L'ensemble des composants de gestion a été mis en oeuvre sous forme de composants Fractal, en utilisant l'implantation standard, qui est Julia pour le langage Java. Comme nous l'avons dit précédemment, seul le moniteur, qui est un programme système, a été réalisé dans les langages les mieux adaptés à chacunes des architectures concernées (Langage

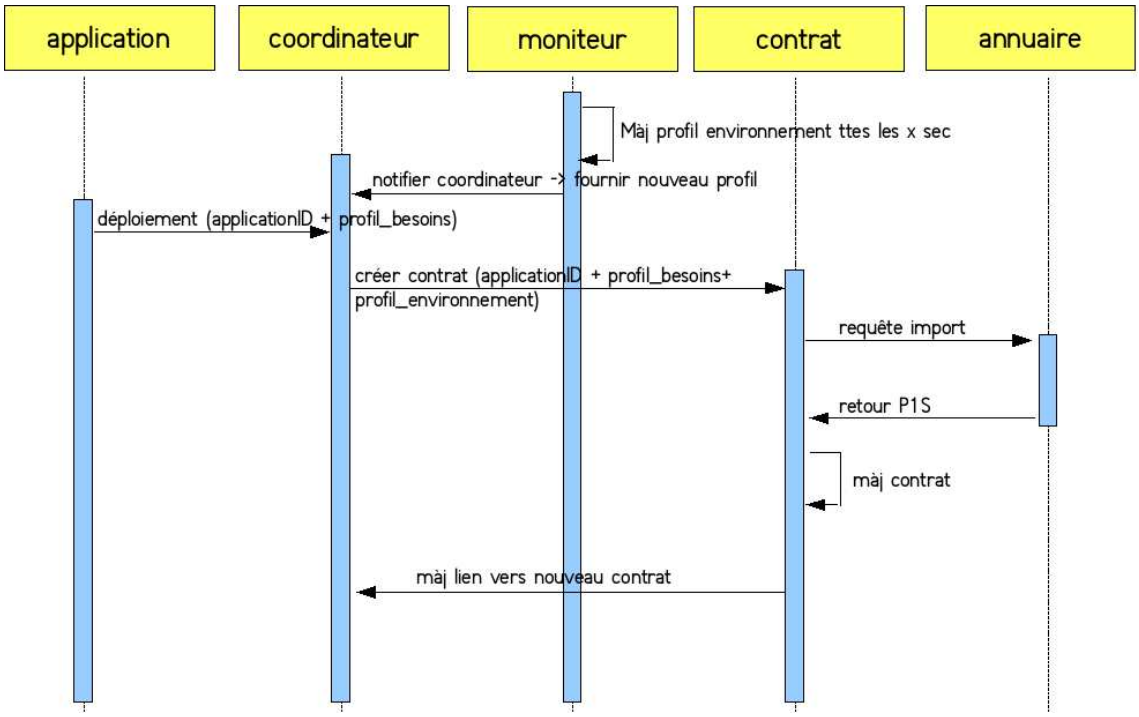

FIG. .3.11 – *création et mise à jour des contrats*

C pour la plateforme Palm, Visual C++ de Microsoft pour la plateforme PocketPC).

Le schéma d'implantation réel de ces composants de gestion est donné par la figure .3.12.

De part les contraintes d'implantation, il y a quelques modifications avec le modèle que nous avons présenté précédemment :

- On trouve tout d'abord un composant "super-contrat". Il est nécessaire, car dans Fractal, l'ajout d'un composant dans un composant composite oblige à stopper l'exécution du composant composite. Le "super-contrat" permet donc de "limiter les dégâts" : il est le seul à être stoppé (avec les composants qu'il contient : le répartiteur et les autres contrats). Sans ce composant, l'ensemble des composants de gestion devraient être arrêtés à chaque ajout d'un contrat (c'est à dire à chaque déploiement d'une nouvelle application).
- le répartiteur permet de gérer la liaison multiple entre le coordinateur et les contrats. Ainsi, le coordinateur est toujours connecté à la même interface. Seule l'interface de sortie du composant répartiteur va évoluer à l'ajout d'un nouveau contrat. Le coordinateur ne change donc jamais, et est toujours en cours d'exécution.

La liaison entre le composant applicatif (CA) et les services techniques (par exemple ST1) est aussi différente : dans le modèle que nous avons spécifié à l'origine, cette liaison devrait être faite au travers de l'incorporation du composant du service technique comme sous-contrôleur dans la membrane du composant applicatif. Cependant, actuellement, le sous-contrôleur n'est pas un composant, mais un objet. Il s'agit donc d'un lien vers un composant externe.

En terme de résultats, nous avons mesuré le temps d'indisponibilité du service tech-

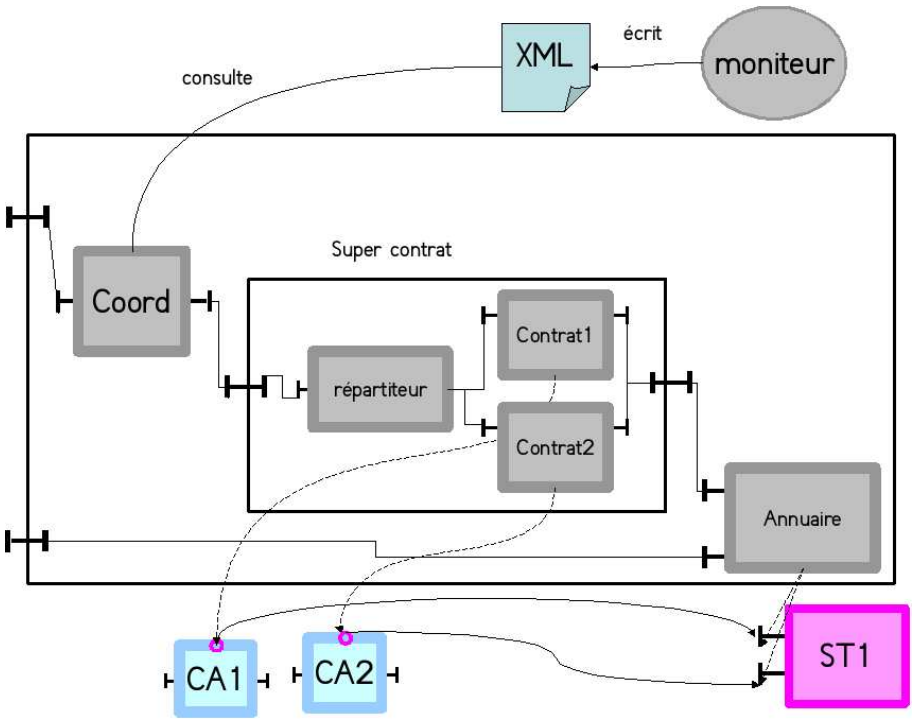

FIG. .3.12 – *Implantation des composants de gestion*

nique, lors d'une adaptation. Lors de ces tests, l'annuaire contenait une vingtaine de personnalités de services techniques. Ce temps moyen est de 96msec. Cela correspond au changement de la liaison entre le composant technique et le composant applicatif. Ce temps est relativement court. Cela s'explique par différents facteurs :

- Le modèle de conception retenu repose sur le partage des données du services technique entre toutes les personnalités de ce service. Ainsi, lors d'une adaptation du service technique, seulement deux cas sont envisageables. Soit le service technique peut continuer à fonctionner avec les mêmes données, et dans ce cas là, les applications concernées par l'adaptation peuvent continuer leur exécution normalement (sans nécessité d'être relancées). Soit le service technique ne peut plus fonctionner avec les même données internes, et dans ce cas là, suite à l'adaptation, toutes les applications générant des données incompatibles pour ce service technique devront être relancées. Prenons l'exemple du service transationnel, et d'un changement de contexte d'exécution qui oblige à passer d'une personnalité complète (transactions plates + transactions emboîtées), à une personnailité gérant uniquement des transactions plates. Si le service transactionnel, avant adaptation, n'avait en cours d'exécution que des transactions plates, alors l'adaptation pourra s'effectuer, et l'exécution pourra se poursuivre normalement. Cependant, si avant l'adaptation, le moniteur transactionnel était en cours d'exécution d'au moins une transaction emboîtée, la nouvelle personnalité du service transactionnel ne sera plus capable de gérer les identifiants de cette transaction emboîtée. Il conviendra donc de stopper cette transaction et de la relancer sous une autre forme (compatible avec la personalité du service transactionnel disponible). Ce contrôle de cohérence est une charge supplémentaire pour le développeur de service technique.
- De plus, l'aspect du déploiement du service technique n'a pas été traité ici (ce travail,

qui vient de débuter dans le cadre du projet TAC MOSAIQUES sera traité dans les perspectives de ce document). Le temps de déploiement des composants des services techniques n'est donc pas pris en compte. L'annuaire ne contiendra donc que des personnalités de services techniques disponibles sur le serveur applicatif. Cet aspect est une contrainte forte de l'actuel prototypage de notre solution, qui va évoluer dans le cadre d'un mémoire de DEA, puis d'une thèse reposant sur les échanges du projet MOSAIQUES.

3.6 Conclusions

Ce travail sur l'adaptabilité des services techniques s'est fait en deux phases. Dans un premier temps, nous avons étudié une nouvelle méthode de conception des services techniques, basée sur le modèle à composants. Pour cela, nous avons pleinement profité de l'expérience des travaux sur le moniteur transactionnel, présentés dans le chapitre .2.

L'utilisation du modèle à composants pour les services techniques n'a été envisageable qu'à partir du moment où nous disposions d'un modèle à composants offrant la récursivité (un assemblage de composants est lui même un composant) et la réflexivité. Fractal offre ces possiblités. Ainsi, nous obtenons un modèle de conception permettant de réaliser différentes instances d'un service technique, juste par réassemblage de composants de bases.

La seconde partie de ces travaux a consistè à concevoir des outils permettant l'auto-adaptation des services techniques, en fonction des besoins applicatifs, et des caracteristiques de l'environnement d'exécution. Nous avons donc fourni un ensemble de composants de gestion, capables de détecter un changement dans l'environnement d'exécution, et de retrouver, à l'aide d'un annuaire spécialisé, les services techniques capables de s'exécuter en fonction des contraintes.

L'adaptation du service technique se fait sans une longue indisponibilité du service, grâce à l'utilisation du partage de composants, possible entre deux assemblages différents. Ainsi, les données nécessaires au bon fonctionnement du service technique, vont être partagées entre les deux personnalités du service durant la phase d'adaptation.

Cependant, ce travail n'est pas encore terminé, et il demande encore de nombreuses recherches, notamment sur l'étude du déploiement des services, mais également sur la mise au point d'outils formels facilitant la tâche du programmeur de service technique, qui s'est trouvée complexifiée en deux points :

- La définition des personnailités et des besoins en terme de ressources systèmes pour exécuter les services techniques,
- Le traitement de la détection des applications à relancées suite à l'adaptation d'un service technique, car elles ne seraient plus compatibles avec la nouvelle personnalité de ce service (en prenant en compte le traitement des interblocages).

Ces deux travaux sont au programmes du projet MOSAIQUES, qui a débuté en janvier 2005, et dans lequel notre équipe est partie prennante.

Chapitre .4

De nouveaux services pour les applications de proximité

4.1 Introduction

Ces dernières années, l'évolution des terminaux nomades et des réseaux sans fil et/ou mobiles a favorisé, à la fois la croissance du nombre d'usagers d'applications distribuées, mais également la mobilité de ces terminaux**[CP03]**. Ce développement repose en particulier sur l'augmentation des capacités des nouveaux terminaux (CPU capable de traitements puissants, mémoire devenant très importante), sur leur miniaturisation mais également sur leur capacité à communiquer n'importe où et n'importe quand. En effet, dorénavant, ces terminaux reposent souvent sur l'utilisation de réseaux sans fil tels que Bluetooth ou Wifi.

Dans le précédent chapitre, nous avons vu que ces terminaux demandaient une adaptation de certains services (et notamment des serices techniques) existants, pour permettre un fonctionnement avec une qualité de service correcte. Nous allons voir ici, que les différents atouts cités précédemment, permettent aujourd'hui d'envisager des services totalement nouveaux et de nouvelles applications pour les usagers mobiles.

En effet, dès mon arrivée à l'université de Valenciennes, je me suis intéressé à un nouveau type d'applications mobiles : les applications de proximité. Cela s'est traduit par 2 soumissions (infructueuses) ACI jeune chercheur (en 2000 et en 2002), par l'encadrement de 2 DEA en 2001 et 2002 sur ce sujet **[THI01]**,**[TAQ01]** et par le co-encadrement d'une thèse de doctorat (2001-2004) **[THI04]**. Ces applications permettent, à différents participants (fixes ou mobiles), de partager des informations et de communiquer lorsqu'ils sont

physiquement proches les uns des autres. De part cette notion de proximité entre les participants, l'application repose particulièrement sur l'utilisation de réseaux sans fil et/ou mobiles à portée limitée (réseaux personnels ou locaux). Les réseaux de téléphonie mobile, qui permettent à différents usagers de communiquer quelque soit la distance qui les sépare, ne sont donc pas considérés dans les applications de proximité.

Dans ce type d'application, un utilisateur doit donc pouvoir émettre une requête sur des informations disponibles au sein du réseau de communication et retrouver l'information qui est susceptible de l'intéresser. Cet ensemble d'informations partagées est réparti sur les différents terminaux des participants présents dans le réseau et est sujet à des évolutions et des modifications constantes en fonction des déplacements des terminaux et des utilisateurs mobiles. Un exemple de requête utilisée dans les applications de proximité est "Quel est l'arrêt de bus le plus proche de moi?": cette requête permet de rechercher une information en fonction de la localisation géographique du client. C'est une requête dépendante de la localisation[1]. Un autre exemple, au niveau d'un réseau personnel dans une maison, peut être "Afficher l'ensemble des photos numériques présentes sur les terminaux de la maison". Cette requête vise alors uniquement à fédérer l'ensemble de l'information personnelle d'une famille, qui est éparpillée sur différents terminaux dans une maison.

Dans ce chapitre, je vais donc présenter précisément l'ensemble des travaux qui ont porté sur les applications de proximité, en insistant plus particulièrement sur les travaux liés à l'évaluation des LDQs dans l'environnement des applications de proximité. Dans un premier temps, je vais revenir dans le détail sur la problématique des applications de proximité, en prenant en compte à la fois les aspects de fédération d'un système d'information largement distribué dans une sphère de communication restreinte, et les aspects liés à l'évaluation des requêtes dépendantes de la localisation. Puis, je présenterai, dans le détail, les résultats obtenus au travers des travaux de thèse de Marie Thilliez.

4.2 Sphères de communication et applications de proximité

4.2.1 Contexte

Dans l'introduction, nous avons défini les applications de proximité comme des applications permettant à plusieurs utilisateurs ou terminaux *physiquement proches* les uns des autres de partager de l'information, des services et de communiquer. Ces applications reposent principalement sur l'utilisation d'un réseau sans fil, qui permet à un utilisateur de rester connecté en se déplaçant dans un périmètre géographique plus ou moins étendu, appelé zone de couverture. L'ensemble des zones de couverture des terminaux munis de capacité de communication sans fil vont se juxtaposer dès que ces zones se recoupent (si les zones de couverture sont issues de réseaux compatibles entre eux). Alors, cette association de ces différentes zones de couverture juxtaposées, sera appelée sphère de communication et permettra aux différents utilisateurs de communiquer entre eux (cf. figure .4.1).

Le premier contexte d'étude reposant sur ces sphères de communication, a été le "Home Area Network" (HAN, qui est le réseau personnel domestique). Il s'agit de la sphère de communication interne à une maison, et qui permet d'échanger des informations entre tous les terminaux informatiques présents à l'intérieur d'une maison (ordinateur, console

1. LDQ pour Location Dependent Query

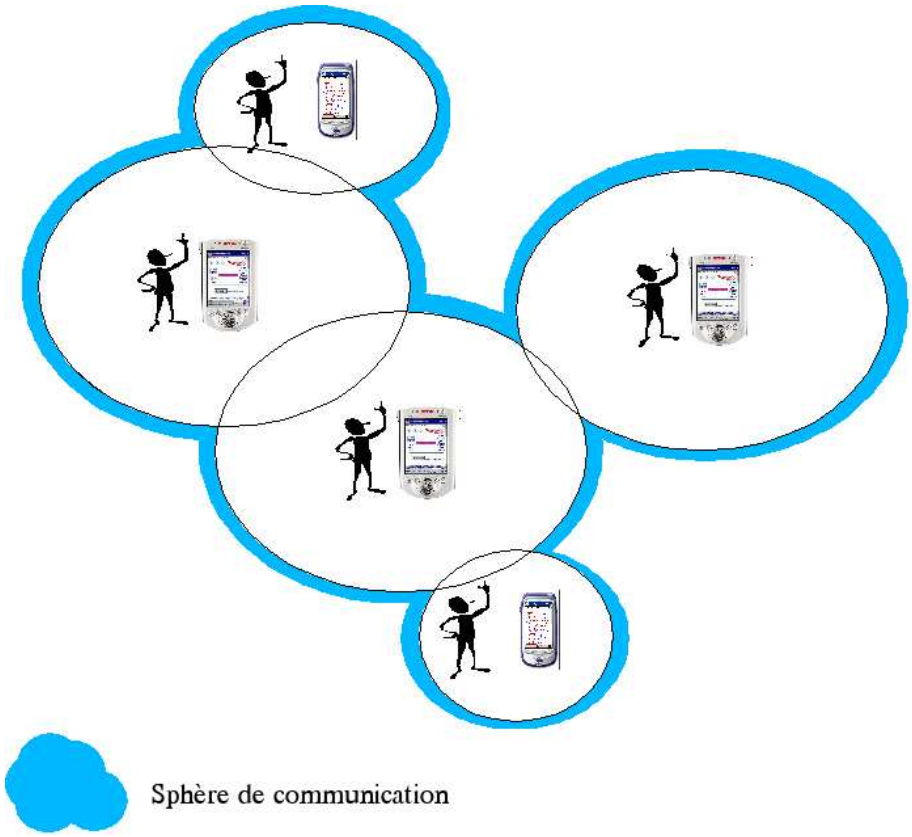

FIG. .4.1 – *Notion de sphère de communication*

de jeux, démodulateur satellite, DivX Box, etc..) (cf. figure .4.2). Tous ces outils informatiques sont dorénavant dotés de capacité de traitement, de stockage et de communication (filaires comme le câble USB ou le câble réseau, ou non filaires comme bluetooth ou Wi-Fi). Il devient donc important de fournir à l'utilisateur de ces terminaux, un moyen d'accéder, facilement, à l'ensemble des données stockées sur ces différents terminaux. Dans ce cadre, nos travaux ont consisté à fournir à l'utilsateur un moyen de retrouver facilement ces informations.

En effet, les données personnelles (agenda, messagerie, photos, films, documents textuels, . . .) d'un utilisateur du HAN, peuvent se situer sur plusieurs supports de stockage en même temps. Cette distribution des informations procure l'avantage évident de disposer de toutes ses ressources, quelque soit l'endroit où il se trouve et indépendamment de l'appareil qu'il utilise. Par contre, cela pose des problèmes liés à l'intégrité des données. En effet, des modifications sur des documents peuvent être faites sur différents appareils, ce qui peut rapidement entraîner des conflits de version. Indépendamment de ces aspects, lorsqu'un utilisateur désire consulter un de ses fichiers, il doit simplement envoyer une requête à un service présent sur le HAN, sans avoir à se soucier de ce genre de problèmes. C'est à ce niveau que se situent les travaux que nous avons mené.

L'utilisateur ne doit pas avoir à explorer tout le HAN dans le but de retrouver la dernière version de son fichier, cela deviendrait vite pénible et laborieux. Ce travail a donc eu pour but de proposer un service qui prend en charge ce genre de fonctionnalité de manière

FIG. .4.2 – *Les éléments du Home Area Network*

transparente pour l'utilisateur. Pour retrouver toutes les informations, un premier rôle du service que nous avons défini est de posséder un ensemble d'informations, qui tient à jour les données sur les différents appareils constituant le HAN et les informations possédées par chacun de ces appareils. Ce service doit également tenir compte des faibles capacités de stockage et de traitement de certains éléments du HAN. Les données que contiennent ces terminaux doivent également être prises en compte dans les moyens mis en oeuvre pour localiser et indexer toutes les informations importantes. Il faudra également tenir compte des terminaux qui ne disposent que d'une autonomie limitée et qui sont donc sujets aux déconnexions fréquentes. Enfin, certains terminaux, très mobiles, se déplacent et peuvent donc se retrouver, par moment, en dehors de la zone de couverture des réseaux sans fil. Nous devons nous doter de mécanismes capables de détecter l'arrivée, le départ et même la déconnexion momentanée d'un appareil. Le service proposé doit donc effectuer des traitements adaptés pour chaque cas de figure afin de conserver l'intégrité globale de l'information. Toutes ces caractéristiques forment le premier service spécifique sur lequel nous avons travaillé pour ce type d'applications. Des services comme UPnP [2] **[UPN00]** n'existant pas encore lorsque nous avons débuté ces travaux, nous avons basé nos réflexion sur les mécanismes de recherche d'Information dans les applications P2P telles que Napster, ou Gnutella. Ce travail sera détaillé dans la partie 4.3.

Le second domaine d'étude, portait sur les applications de proximité dans des environnements publics (et non plus personnels), comme les galeries marchandes, les gares, les aéroports, etc... Ces applications peuvent être vues comme un "passage à l'échelle" des applications du HAN. Cependant, le problème que nous avons traité ici est radicalement différent.

En effet, même si les problèmes de recherche de données et de maintien en cohérence des informations restent valables, ils semblent moins problématiques, car plus semblables à ce que l'on peut rencontrer dans un système distribué traditionnel. Cette différence vient principalement du fait que dans ces applications de proximité, l'utilisateur ne dispose que

2. Universal Plug and Play

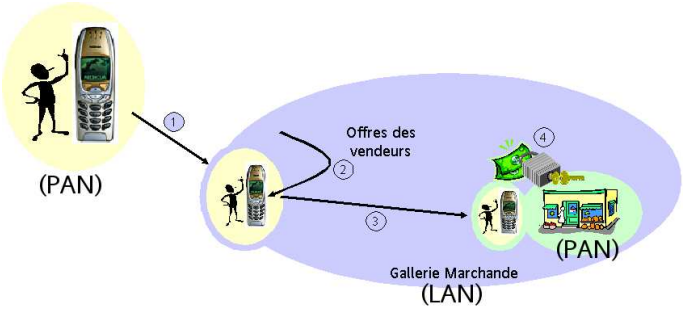

FIG. .4.3 – *Application de Commerce Electronique de proximité*

d'une seule machine qui lui appartient. Ses informations personnelles sont donc toutes regroupées sur cette machine, ce qui pose beaucoup moins de problèmes de maintient en cohérence des données. Des services de recherche dans ces environnements dynamiques existent déja. On peut notamment citer SLP[3] **[VGP97]**, JINI **[WAL00]**, SSPD[4] **[GCL99]** ou Salutation **[SAL98]**, dont les caractèristiques sont présentées dans le tableau .4.1.

	SLP	SSPD	Jini	Salutation
Fournisseur	IETF	Microsoft	Sun	Salutation
Support	TCP/IP	TCP/IP	Indépendant	Indépendant
Langage	Indépendant	Indépendant	Java	Indépendant

TAB. .4.1 – *Les services de découverte dynamiques*

Ces services, qui sont les plus connus, mais il en existe d'autres (dont vous pourrez trouver une description dans le document de thèse de Marie Thilliez **[THI04]**, sont dédiés aux nouvelles contraintes de dynamicité que nous rencontrons dans les applications de proximité, de ce fait, ils semblent adaptés à notre environnement. Cependant, de nombreuses limites subsistent, tant au niveau de l'évolution de l'architecture réseau (utilisation du modèle P2P hybride), que dans la possibilité de faire évoluer les fonctionnalités de ces services. Si l'on prend le premier exemple d'application que nous avons traité dans le cadre des applications de proximité, qui est celui du Commerce Electronique de proximité (cf. figure .4.3). Cette application représente l'arrivée d'un utilisateur dans une galerie marchande. En arrivant dans la galerie marchande, l'utilisateur va pouvoir émettre des requêtes en rapport avec des vendeurs ou d'autres utilisateurs. Nous nous sommes notamment attachés à traiter 2 types de requêtes, qui sont des requêtes liées à la proximité des informations :

- Ce sont tout d'abord des requêtes dépendantes de la localisation de l'utilisateur, comme, par exemple, "Où est la pizzeria la plus proche de l'endroit où je me trouve"
- Ce sont également des requêtes dépendantes de la localisation d'un autre utilisateur, comme par exemple "Où est Marc?"

De plus, nous avons posé un certain nombre de contraintes liées à l'environnement, comme par exemple :

- pas de mécanisme de géolocalisation (type GPS ou Galileo) disponible
- une infrastructure réseau minimale
- optimiser la gestion de l'énergie du terminal de l'utilisateur

3. Service Location Protocol
4. Simple Service Discovery Protocol

Il a donc fallu proposer un service d'évaluation de requêtes dépendantes de la localisation, pouvant se passer des infrastructures de géolocalisation (comme le GPS), et étant capable, à tout moment, d'évaluer la position, de la façon la plus précise possible, d'un terminal utilisateur (mobile) dans une sphère de communication, ce que ne proposaient pas les services existantt décrits précédemment.

4.2.2 Infrastructure réseau

Que ce soit pour les applications du HAN, ou pour les applications de proximité, l'infrastructure réseau nécessaire au fonctionnement de l'application doit être légère et peu coûteuse. La solution client/serveur a rapidement été écartée pour plusieurs raisons: elle pose des problèmes en terme de fiabilité (reprise en cas de panne du serveur), de passage à l'échelle (charge du serveur importante dans le cadre de la galerie marchande, aux heures de pointes), de coût d'exploitation et de maintenance (notamment pour un utilisateur lambda, dans le cadre du HAN). De plus, la gestion centralisée est une limite importante pour les applications de proximité: en effet, centraliser l'ensemble des informations disponibles sur les différents terminaux des clients est une opération très coûteuse et les mises à jour en fonction des entrées/sorties des clients dans la sphère de communication et le maintien de la cohérence de ces informations sont difficiles dans un environnement aussi dynamique. De plus, se posent alors de gros problèmes de confidentialité dans le stockage d'informations propres à un utilisateur, sur un serveur dans une galerie marchande. Dès lors, que ce soit pour les applications du HAN ou pour les applications de proximité, nous nous sommes tournés vers les architectures pairs à pairs (P2P) **[MHC01]**, qui offrent une meilleure exploitation des capacités des terminaux des usagers, et une meilleure gestion de la dynamicité.

4.2.2.1 Présentation des réseaux P2P

Dans une architecture P2P, chaque ordinateur peut être à la fois client et serveur **[MHC01]**. Cela signifie que chacun des ordinateurs du réseau est libre de partager ses ressources et de mettre à disposition des informations. Il peut donc, à la fois émettre des requêtes sur les autres machines disponibles mais aussi répondre aux requêtes des autres utilisateurs.

Il existe plusierus types de réseaux P2P, reposant sur des architectures différentes, et ayant chacun leurs avantages, en fonciton des besoins applicatifs recherchés: les réseaux P2P "purs" et les réseaux P2P "hybrides". Dans le réseau P2P pur, l'index (information qui sert à retrouver les données) et les données sont totalement distribués sur l'ensemble des participants. L'architecture du réseau est alors complètement dynamique. Ce modèle a le désavantage de rendre très difficile l'administration des applications dans un contexte où le nombre de noeuds est très important. Comme les requêtes peuvent potentiellement être envoyées à tous les participants, des problèmes de performances sont souvent rencontrés. Enfin, la sécurité est plus difficile à gérer dans les applications, car l'ensemble des participants, y compris ceux qui vont gérer l'index, peuvent potentiellement changer d'adresse réseau à chaque connexion/déconnexion.

Pour répondre en parti à ces problèmes, le modèle P2P hybride propose d'identifier des "super noeuds" **[YG03]** parmi les participants de l'application, qui jouent le rôle de serveur centralisé pour un sous-ensemble de noeuds (cf. figure .4.4). Certaines parties de l'application et des informations sont alors confiées à ces super-noeuds, qui sont considérés

comme des machines de confiance sur le réseau.

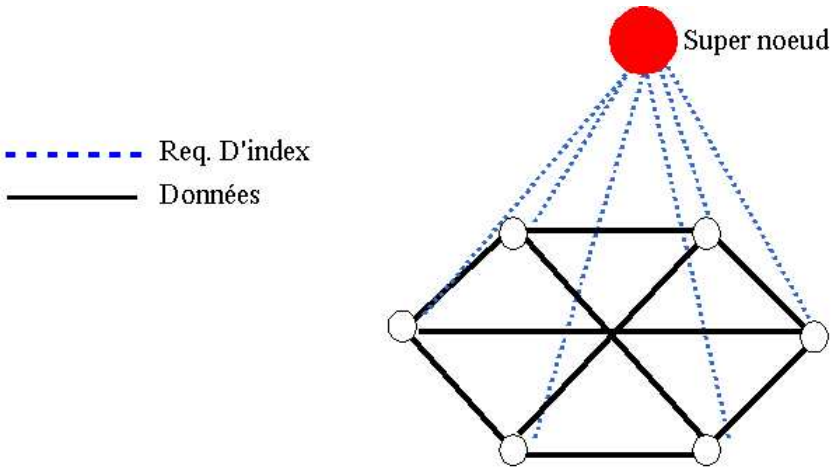

FIG. .4.4 – *Réseaux P2P Hybrides*

Que ce soit pour les applications du HAN, ou pour les applications de proximité, nous avons basé notre architecture réseau sur le P2P hybride, car :

- La notion de super-noeud est relativement facile à implanter dans ce type d'application : il y a toujours une (ou plusieurs) machine(s) particulière(s) (voire fixes) qui peuvent remplir ce rôle. Il s'agit, dans le cas du HAN, d'un ordinateur personnel, ou, par exemple, d'une console de jeu (qui sont parmis les machines les plus puissantes présentes dans les maisons), ou bien, pour les applications de proximité, les ordinateurs des commerçants.
- Cela permet de considérablement accélérer la recherche d'information (car l'index est plus facile à reconstruire) ainsi que la reprise après panne (car ces machines peuvent servir de point de synchronisation.

4.2.2.2 Une architecture pour le HAN

Dans le cadre du HAN, chaque appareil que nous allons pouvoir rencontrer (TV, PC, Assistant Personnel, Baladeur MP3, . . .) va être considéré comme un pair (un noeud) dans une architecture P2P hybride. Certains des terminaux de la maison vont donc alors devoir prendre le rôle de super noeud. Nous avons mis à profit l'hétérogénéité des différents éléments du réseau en permettant aux appareils mobiles de pouvoir se connecter sur n'importe quel appareil fixe. Ceci permet de résoudre le problème technique qu'une solution entièrement décentralisée poserait.

Nous avons donc classé les appareils en deux grandes catégories. La première catégorie, que nous nommerons la catégorie des serveurs d'index (ou super noeud), va regrouper les éléments du HAN qui sont susceptibles de rester fixes dans le HAN et qui possèdent des caractéristiques matérielles intéressantes (PC, démodulateur satellite, divX Box, etc...). La seconde catégorie, les pairs classiques, regroupera les appareils mobiles possédant des ressources matérielles et une autonomie faibles.

Il est important de bien se rappeler que les super noeuds sont des pairs comme les

autres, ils peuvent partager leurs ressources et en récupérer d'autres. Ceci représente une propriété forte du réseau : il n'y a pas de machine spécifique dans le réseau, qui va prendre en charge des tâches de gestion, sécurité ou synchronisation. Au contraire, il est important d'exploiter pleinement les ressources disponibles sur tout le réseau.

La seule contrainte de cette architecture est que, lors de la mise en place d'un réseau HAN ou de l'arrivée d'une nouvelle machine dans le réseau (c'est à dire une machine qui est connectée pour la première fois à ce réseau), il faut décider quels appareils doivent tenir le rôle de super noeud. Cette tâche ne doit pas être laissée aux utilisateurs du HAN (qui ne sont pas forcément des informaticiens), car l'importance d'une bonne répartition est stratégique pour obtenir un réseau HAN performant. La répartition se fera donc en fonction des contraintes de placement qui ont été identifiées :

- Les différents pairs serveurs d'index doivent se connaître et **être en liaison permanente les uns avec les autres**. Ceci est nécessaire pour mettre en place des mécanismes de réplication, de reprise sur panne, mais aussi pour garantir une qualité de service.
- Au moins un serveur d'index doit être accessible à chaque point d'entrée de la maison : ils jouent symboliquement le rôle de borne de connexion pour les éléments mobiles du HAN. Quand un appareil mobile arrive dans l'enceinte de la maison, il se connecte à un pair serveur d'index. De ce fait il se retrouve connecté au réseau HAN et il est considéré comme un pair classique.
- il ne doit pas y avoir de "zones d'ombres" : toute partie de la maison doit être couverte par le réseau sans fil d'au moins un serveur d'index.

4.2.2.3 Mise en oeuvre pour applications de proximité

Nous avons également choisi de développer les applications de proximité "externes" en nous basant sur le modèle d'architecture P2P hybride **[TDL03]**. Cependant, la définition de cette architecture est singulièrement différente de celle du HAN. En effet, dans le HAN, nous avons utilisé une architecture P2P hybride conforme à la littérature **[MHC01]** : les super noeuds offrent une fonctionnalité spécifique qui est le fait d'être serveur d'index, les super-noeuds doivent être interconnectés entre eux, et un pair simple doit être connecté à un super-noeud. Enfin, toutes les informations présentes sur les pairs doivent également être présentes sur les super noeuds

Ici, comme le montre la figure .4.5, certaines de ces affirmations ne peuvent plus être vraies : les pairs arrivent et se connectent par des liaisons temporaires à un super noeud, mais l'index des informations du pair n'est plus dupliqué sur le super noeud. Cela est principalement dû au fait que le super noeud ne soit pas, la plupart du temps, de confiance (alors que dans le cadre du HAN, le super noeud est souvent un ordinateur appartenant également à l'utilisateur). Le traitement des requêtes s'en trouvera complexifié.

4.3 Fédération de systèmes d'information du HAN

4.3.1 Un index pour fédérer les informations

Le HAN est composé de nombreux appareils informatiques disposant chacun d'un espace de stockage plus ou moins important. La quantité totale d'informations présentes sur tous ces appareils peut être très importante. Il est donc important de posséder un index afin

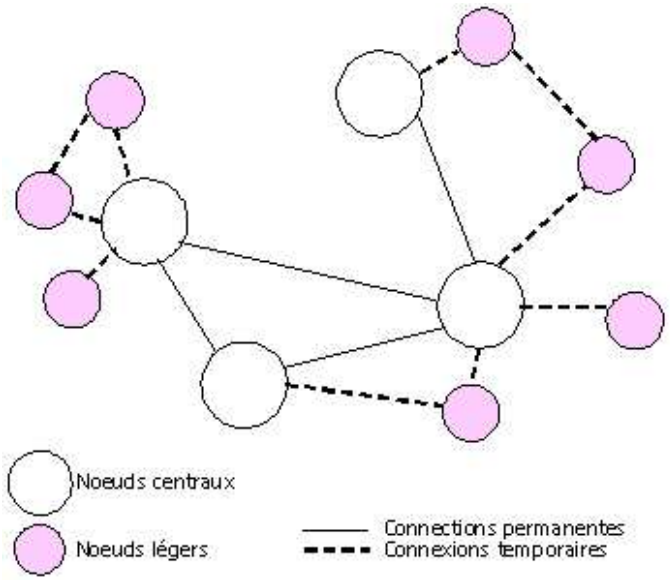

FIG. .4.5 – *Réseaux P2P Hybrides pour application de proximité*

de lister toutes ces données. Cet index doit être mis à jour fréquemment afin de garantir l'intégrité des informations qu'il contient.

Dans ce cadre, David Taquet, dans son mémoire de DEA **[TAQ01]**, a spécifié un ensemble d'informations, qu'un appareil appartenant au réseau devra fournir sur lui-même, et qui seront stockées dans l'index :

- son nom, son adresse IP et la liste des ports qu'il va écouter pour l'envoi et la réception de requêtes.
- Son type de connexion au réseau (par exemple : connexion par câble avec un débit de 100MBits). Le renseignement du type de connexion, notamment le débit, permet de savoir quels appareils sont en mesure de communiquer plus facilement que les autres.
- Ses caractéristiques matérielles générales : taille des supports de stockage, dimensions de l'écran d'affichage, autonomie de fonctionnement. Ces informations vont permettre à l'utilisateur de sélectionner une ressource cohérente avec le matériel sur lequel il désire l'exploiter (il lui sera par exemple inutile de télécharger une vidéo de 50Mo sur son appareil mobile s'il ne possède pas d'écran LCD pour le visionner).
- Ses caractéristiques logicielles : système d'exploitation, codecs, Etc.
- Ses paramètres de sécurité : les droits d'accès, les utilisateurs autorisés et les privilèges qu'ils possèdent. Une autorisation d'accès à une machine sera nécessaire à un utilisateur pour récupérer une donnée.

Le travail sur l'identification de ces informations a été très important, car bien que ces travaux aient été réalisés en 2001, ils constituent le début de nos recherches sur l'identification des données contextuelles qui ont permis également les travaux sur l'auto-adaptation des services aux contraintes des terminaux, et aux besoins des applications (cf. le chapitre précédent sur l'adaptabilité des services techniques).

En plus de ces informations contextuelles, toutes les données contenues sont indexées pour chaque pair sous la forme d'un fichier XML, qui va contenir tous les enregistrements partagés par un pair sur le réseau. Les super noeuds vont alors pouvoir regrouper les

informations des pairs légers les entourant, et ainsi former un index global, qui correspond pleinement à la définition du réseau P2P hybride (cf. section 4.3.3).

4.3.2 Déroulement d'un scénario

Pour un terminal donné, le scénario commence lors de son arrivée dans le réseau personnel. En effet, lorsqu'un nouveau pair rejoint le réseau, il est affecté à un super noeud auquel il va devoir se connecter. Si il s'agit d'une première connection, une double authorisation sera nécessaire : le super-noeud doit donner l'authorisation au terminal d'entrer sur le réseau, et le terminal doit donner l'authorisation au super-noeud d'acquérir les informations qu'il contient (cela s'effectuant de manière manuelle). Lors de sa connexion, le nouveau pair va transmettre au super noeud la totalité de son index (c'est à dire le fichier XML contenant les informations présentées dans la section précédente) afin de lui permettre de "centraliser" les informations. Dès réception de l'index, le super noeud renvoie un accusé de réception au pair pour lui indiquer qu'il a bien reçu l'index, et qu'il est bien enregistré comme un partenaire de l'application de fédération des données du HAN. Ensuite, le super noeud effectue un traitement en local, qui consiste à repérer les doublons (par exemple le même fichier audio présent sur plusieurs terminaux) et éventuellement classer ses données d'une façon particulière afin d'optimiser la gestion des requêtes. Finalement il diffuse un message à tous les autres super noeud afin de leur annoncer qu'il détient des nouvelles informations. Dès réception de cette annonce, les autres super noeuds vérifient qu'ils ne possèdent pas déjà un index du pair nouvellement référencé. Le cas échéant, ils le détruisent.

En plus de ce mécanisme d'enregistrement, dans ce travail, plusieurs autres aspects de la "vie" du réseau ont été gérés :

- La modification de l'index d'un pair : si le contenu des fichiers partagés change, l'index du pair change également. Il est donc nécessaire de mettre à jour l'index du super noeud correspondant. Le pair enverra donc un message à son super noeud pour l'avertir. Chaque ressource (fichier, personne, application, etc.) possède un identificateur unique sur le réseau. Le message du pair sera une simple liste d'enregistrements contenant l'identificateur de la ressource à modifier et les modifications à apporter.
- En cas de mobilité d'un pair, il sera amené à changer de super noeud. Il y aura donc une nouvelle phase d'enregistrement.

4.3.3 Recherche d'information

La partie de recherche de l'information au travers du réseau HAN a constitué la majeure partie de ce travail. En effet, une fois l'architecture réseau choisie (P2P hybride) et le format de l'index spécifié (contenu et technique de duplication), il a fallu mettre au point un protocole permettant de retrouver au plus vite et à moindre coût, l'information.

Le premier travail a consisté à "diviser" le réseau en fonction des index gérés par les super noeuds (cf. figure .4.6). Dans chacune des zones identifiées, seul le super noeud dispose des informations de **tous** les autres pairs de la zone. Cela nous permet d'extraire le sous ensemble de noeuds (dans la figure .4.6, les noeuds 1, 2, 3, 4, 5), qui contiennent l'ensemble de l'information présente dans le HAN. C'est sur ce sous ensemble que nous effectuons nos requêtes de recherche d'information.

La politique de recherche utilisée est celle de *l'itération sur la profondeur* **[PRI01]**. Il existe également des techniques de recherche sur la largeur d'abord. Pour la technique

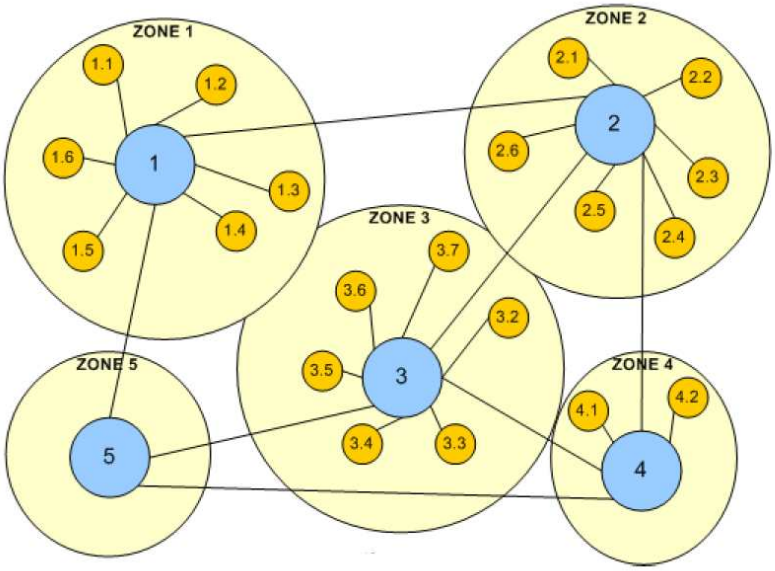

FIG. .4.6 – *Division du réseau HAN*

de recherche sur la profondeur, à chaque itération de cette technique, plusieurs recherches en largeur sont tout d'abord initialisées, avec des limites de profondeur de plus en plus grandes. Ceci jusqu'à ce que la requête soit satisfaite ou que la limite de profondeur (notée D) soit atteinte. Le principal défaut de cette technique, est que le nombre de pairs à chaque profondeur grandit de façon exponentielle. De ce fait le coût de traitement est petit losque la profondeur est faible, par rapport à celui que l'on va obtenir sur une profondeur plus grande. Les mécanismes de recherche effectuant une propagation en profondeur d'abord garantissent des réponses de qualité et des coûts réduits lorsque celles ci sont trouvées rapidement (ce qui est le cas dans un environnement où il y a relativement peu de machines, comme dans le HAN).

Dans le cadre des applications P2P traditionnelles, les systèmes vont plutôt miser sur la quantité des réponses qu'offrent les méthodes en largeur d'abord, où l'on cherche à diffuser la requête le plus rapidement possible, au travers de toutes les machines connectées. En effet, ces techniques permettent à un noeud de transmettre les requêtes le plus rapidement possible à tous les autres noeuds qu'il connaît.

Dans l'algorithme de recherche que nous avons mis au point, les notations suivantes sont utilisées :

- La politique d'itération P. Par exemple, P = a, b, c, d signifie que le nombre d'itérations est de 4 et que la première itération cherche jusqu'à une profondeur a, la seconde jusqu'à une profondeur b, la troisième jusqu'à une profondeur c et enfin la dernière jusqu'à une profondeur d.
- Une période d'attente W (en seconde) doit aussi être spécifiée, elle représente le temps d'attente entre deux étapes successives dans la politique. La réelle difficulté réside dans le choix de la politique P et du temps de latence W.

Au final, une politique est caractérisée de la façon suivante : $Pd = d, d + 1, \ldots, D$ où D correspond à la durée de vie maximale d'un message. Des mesures ont été effectuées

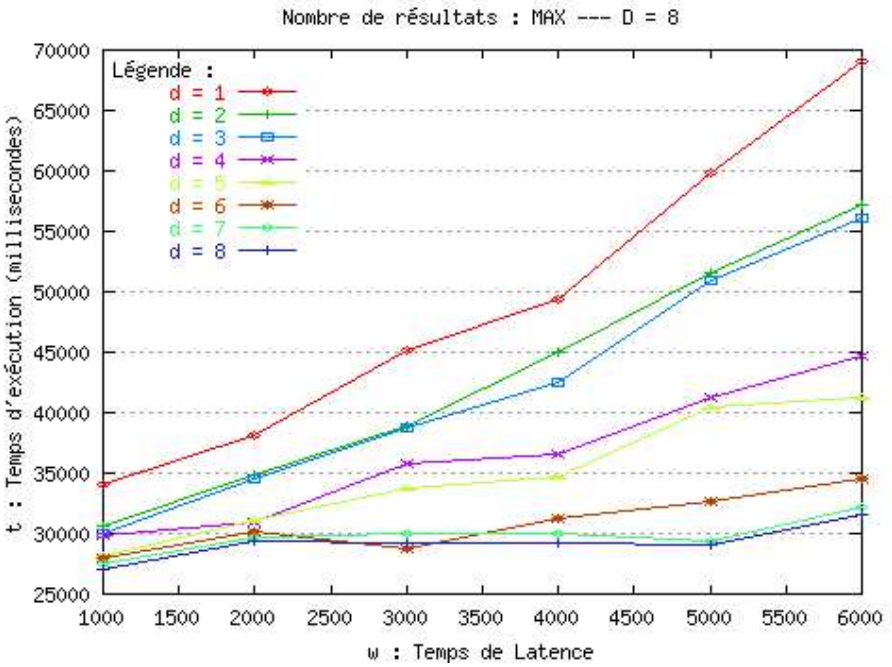

FIG. .4.7 – *Résultats pour une recherche exhaustive*

pour chacune des valeurs de d = 1, 2, 3, 4, 5, 6, 7 et pour des valeurs de W = 1s, 2s, 4s, 6s, 8s. Les résultats de ces mesures fournissent les conclusions suivantes :

- Pour une petite valeur de d associée à un W = 8s (grande latence) la consommation de la bande passante est égale à 19% de celle consommée par une politique dégénérée où d = 7 (itération sur tous les noeuds).
- Quand d augmente, la consommation de bande passante de politique Pd augmente aussi. Plus d est grand plus la politique gaspillera de la bande passante en envoyant des requêtes a de trop nombreux noeuds.
- Si W diminue, la bande passante augmente. En effet, si W est petit, la probabilité que le client détermine prématurément que la requête n'a pas été satisfaite est plus élevée, conduisant à un effet de dépassement similaire à celui que l'on rencontre pour des valeurs de d trop grandes.
- Une relation inverse entre le temps de réponse et le coût de bande passante est observée. Lorsque W augmente, le temps passé pour chaque itération augmente proportionnellement. Et par conséquent si d diminue, le nombre d'itérations nécessaires pour satisfaire une requête augmentera.

Les résultats chiffrés sont donnés par les figures .4.7 et .4.8. Ces figures représentent la variation du temps d'exécution d'une requête en fonction du temps de latence w. Il y a une courbe pour chaque valeur de d comprise entre 1 et D (où D = 8). Pour une courbe donnée, une coordonnée (w, t = f(w)) représente le temps d'exécution t de la requête pour un temps de latence égal à w où la politique de début est égale d.

La figure .4.7 représente les performances pour des requêtes recherchant le maximum de réponses à travers le réseau. Quelque soit la valeur de d, plus nous augmentons le temps de latence w plus le temps de traitement augmente. Pour une requête recherchant le plus de résultats, nous obtenons les meilleurs résultats pour un temps de latence w = 1000ms

et une politique d = 8.

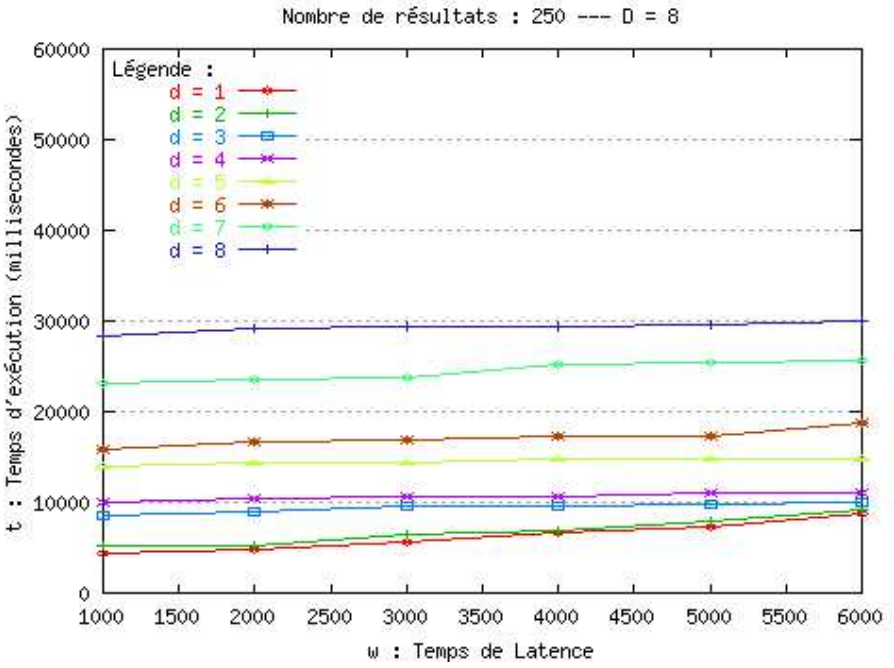

FIG. .4.8 – *Résultats pour une recherche limitée*

La figure .4.8 représente les performances pour des requêtes recherchant les 250 premières réponses à travers le réseau. Le résultat est très différent du cas où nous recherchons toutes les réponses (figure .4.7). Ici, pour un petit nombre de résultats, l'envoi de la requête sur les voisins directs du pair source suffit à trouver la quantité de résultats recherchés. Dans cette situation, la politique d = 1 est la plus efficace car elle effectue une première itération sur les voisins directs de la source et la requête s'arrête car le nombre de résultats attendus est atteint. Dans tous les autres cas de figures (d = 2, 3, 4, 5, 6, 7, 8), une seule itération sera également effectuée mais comme elle s'étends plus profondément dans le réseau, le temps de traitement s'en trouve augmenté.

Les conclusions tirées de ces deux figures sont extrêmes mais en aucun cas contradictoires. En fait, il semblerait qu'il n'y ait pas de valeurs optimales fixes pour les paramètres de la technique de recherche. Bien sûr ces paramètres dépendent de la configuration du réseau mais également de la nature de la requête. Au final, ce travail nous a surtout permis de jeter les premières bases d'une technique de mise au point d'un évaluateur de requêtes dans une architecture P2P hybride reposant sur l'utilisation de pairs et de super noeuds, ainsi que les paramètres à prendre en compte si nous voulons optimiser cet évaluateur de requêtes.

4.4 Un service de requêtes dépendantes de la localisation des utilisateurs

Après nos travaux sur la recherche d'information dans le HAN, nous avons travaillé sur la possibilité d'évaluer des requêtes dépendantes de la localisation d'un utilisateur dans le

cadre des applications de proximité. Pour cela, nous avons tout d'abord recherché un moyen d'obtenir une localisation approchée d'un utilisateur mobile sur le réseau, puis d'évaluer une requête dépendante de la localisation en fonction de ces résultats.

La configuration de cet algorithme de recherche de la localisation devra donc être sensiblement différente en fonction de la "densité" du réseau en noeuds centraux, mais également en fonction de la quantité d'informations recherchées, et de l'exhaustivité ou non, des résultats (parcours complet ou non du réseau).

4.4.1 Déterminer la localisation d'un utilisateur mobile

Actuellement de nombreuses techniques permettent de déterminer la localisation d'un utilisateur mobile. La plus répandue utilise les récepteurs GPS, qui permettent de récupérer les signaux de plusieurs satellites, pour en déterminer la position de l'utilisateur. Cependant ces solutions nécessitent souvent un équipement et une infrastructure relativement lourds et coûteux. De plus, le GPS n'est pas utilisable dans certains environnements (par exemple à l'intérieur d'un bâtiment). Enfin, la localisation fournie par des techniques comme le GPS est représentée sous la forme d'une localisation physique (latitude, longitude, altitude) qui n'est pas facile à exploiter au niveau applicatif (elle ne comporte aucune sémantique, et nécessite donc de rapporter ces coordonnées sur une carte embarquée, ce qui se révèle coûteux en place mémoire). Les travaux présentés dans cette section, se rapportant aux applications de proximité, proposent une solution de localisation, qui repose sur l'utilisation des caractéristiques des réseaux sans fil, permettant de localiser un utilisateur mobile et ne nécessitant aucune infrastructure et aucun dispositif spécifique.

L'architecture réseau utilisée est celle des applications de proximité (cf. section 4.2.2.3) (P2P hybride). Dans cette architecture, chaque pair va disposer d'un fichier de description de localisation. Un nouveau pair qui arrive dans la sphère de communication pourra déterminer sa localisation, en fonction d'un certain nombre de métadonnées contenues sur les pairs voisins (c'est à dire les terminaux avec lesquels il arrive à établir un contact direct).

Pour cela, chaque pair localisé dans le réseau doit disposer des informations suivantes :

- *locationDescription* décrit la position du pair. Cet attribut représente une métadonnée de type *localisationType*. Plusieurs descriptions peuvent être utilisées pour décrire la position d'un pair : la description symbolique (bâtiment, étage ...) et/ou la description physique (latitude, longitude et altitude). Cette position est représentée dans un fichier XML afin de préserver la sémantique des localisations et de faciliter son traitement par la suite. Cette localisation peut ne pas être définie pour un pair arrivant dans la zone de communication, ou pour un pair trop léger pour effectuer les traitements.
- *locationLastUpdate* représente la date de la dernière mise à jour du fichier référencé par l'attribut locationDescription.
- *mobilityProfile* décrit le profil de mobilité du pair. Ce profil de mobilité est représenté sous la forme d'une vitesse. Pour chaque noeud voisin, on définit sa catégorie : très mobile, assez mobile, peu mobile ou immobile.

Etant donné que nous proposons d'utiliser le signal du réseau pour affiner la position du pair, nous prenons en compte également deux attributs décrivant cette connexion :

- *connectionRange* est une donnée calculée en fonction du niveau du signal réseau.

– *connectionState* est un booléen permettant de connaître l'état de la connexion avec un pair distant.

Lorsqu'un pair entre dans la sphère de communication, ou lorsqu'il veut rafraîchir sa localisation, il récupère les *locationDescription* de tous les pairs avec qui il est capable de rentrer en contact par un lien direct (en mode Ad-Hoc). L'algorithme proposé, détaillé dans **[TDL04]**, calcule alors un degré d'approximation pour chaque fichier de localisation récupéré (i.e. l'attribut locationDescription). Le calcul de ce degré repose sur les attributs locationLastUpdate, connectionRange et mobilityProfile. Ce degré permet de classer les fichiers de localisation récupérés des pairs voisins en fonction de leur "distance" avec le pair désirant obtenir sa localisation. L'algorithme fournit donc, en sortie, une liste de couples (fichier de localisation, degré d'approximation) triés en fonction des degrés d'approximation. Nous verrons dans la section suivante comment cette localisation est utilisée dans l'évaluation des requêtes dépendantes de la localisation.

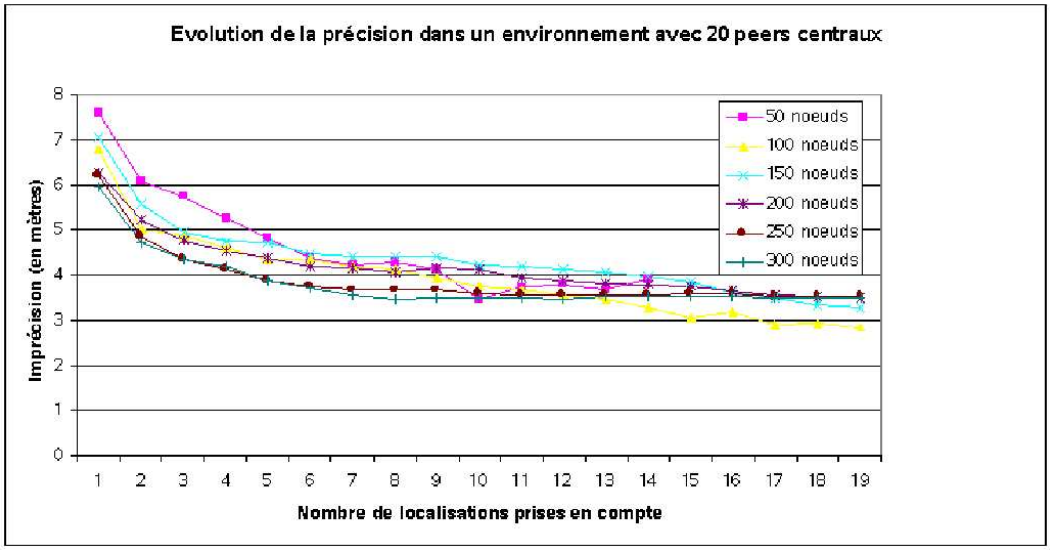

FIG. .4.9 – *Résultats*

La position obtenue par ce procédé s'est révélée précise. La solution a été évaluée en fonction de plusieurs critères. Le premier critère concerne le nombre de pairs dans la sphère de communication (50, 100, 200 ou 300). Le second critère concerne le nombre de noeuds centraux dans cette zone (5, 10, 20) (il est d'ailleurs à noter que ce critère influe très peu sur le résultat), sachant que ces noeuds disposent d'un fichier de localisation beaucoup plus fiable. Enfin, le dernier critère utilisé représente le nombre de fichiers de *locationdescription* utilisés pour déterminer la localisation approchée du pair. La figure .4.9, présente les résultats obtenus pour un réseau de 50 à 300 machines, qui dispose de 20 noeuds centraux. Sur cette figure, on peut se rendre compte que au-delà de 7 fichiers de locationdescription utilisés, le gain en précision est quasiment nul. Enfin, on peut se rendre compte que la précision obtenue est de l'ordre de 4 mètres, ce qui, pour le type d'application visée, est largement suffisant.

D'autres solutions sont proposées pour obtenir une localisation approchée d'un utilisateur pour contrer le problème de la portée des signaux GPS dans un bâtiment. On peut notamment citer les solutions basées sur l'étude de l'environnement. Dans ce cadre, on

trouve les travauxde France Telecom **[ACD05]**, qui propose de combiner l'utilisation de TAG RFID, de signaux Wi-fi, et de caméras de télésurveillance, ou le système RADAR de Microsoft **[BP00]**, qui est basé sur des données électromagnétiques pour déterminer la position et l'orientation d'un objet. Un autre solution, proposée par l'INT Evry **[SVF05]** repose sur l'utilisation de répéteurs GPS à l'intérieur du bâtiment. Enfin, d'autres travaux déterminent la localisation d'un objet si cet objet se trouve à proxiité d'une localisation connue (soit par détection à l'aide d'un contact physique, comme un capteur **[HS99], [PAB00]**; soit par utilisation d'un point d'accès Wi-Fi **[HILL99]**, ou de connections infra-rouges **[WHF92], [WSA97]**). Le problème de toutes ces solutions est qu'elles sont, la plupart du temps, dédiées à un environnement limité, et offrent des précisions très hétérogènes. Enfin, des solutions prochent de nos besoins, comme celles proposées par **[HILL99]** ou **[SVF05]** on le défaut de reposer sur une infrastructure lourde, peu compatible avec des environnements comme ceux rencontrès dans les applications de proximité (gallerie marchande, aéroport, gare, etc...).

Finalement, même si la solution que nous proposons ici peut se révéler moins précise que d'autres, elle a réalisé l'objectif, qui était de proposer un mécanisme de localisation approchée, reposant sur un architecture minimale, et donc peu couteuse à mettre en oeuvre pour ceux qui veulent proposer ce service à leurs usagers.

4.4.2 Evaluation de requêtes dépendantes de la localisation

Le concept de requêtes dépendantes de la localisation est apparu avec le développement des applications dépendantes de la localisation (LDA : Location Dependent Applications). Dans ces applications (généralement orientées Web), la localisation des utilisateurs est récupérée dans leurs profils et prise en compte pour personnaliser les recherches effectuées sur Internet. La mobilité grandissante complique grandement l'évaluation de ces requêtes. Sur la base de la localisation approchée de l'utilisateur, proposée dans la section précédente, nous avons travaillé sur un moteur de requêtes capable d'évaluer des requêtes dépendantes de la localisation **[TD04]**. Il est capable de traiter les deux types de requêtes de localisation existantes **[SDK01]**:

- les requêtes relatives à une localisation qui est différente de celle du client (LAQ, Location Aware Query). Un exemple est "quel est l'arrêt de bus le plus proche de la gare?". Dans ce cas, la localisation de référence est la localisation de la gare.
- les requêtes dépendantes de la localisation du client (LDQ, Location Dependent Query). Un exemple est "quel est l'arrêt de bus le plus proche de moi?". La localisation de référence est alors la localisation du client.

Le modèle d'évaluation proposé repose sur l'utilisation de trois opérateurs (Inside, Closest et Close) pour exprimer les requêtes dépendantes de la localisation **[TD04b]**. L'expression de telles requêtes doit permettre de prendre en compte le critère de comparaison de la localisation pour l'évaluation de la requête.

- L'opérateur *inside* est utilisé pour retrouver une information à l'intérieur d'une zone définie. Ainsi, il peut être utilisé pour retrouver l'ensemble des vendeurs du premier étage d'un bâtiment. Le type de la zone est défini comme paramètre de l'opérateur inside: dans l'exemple, le paramètre est "étage".
- L'opérateur *closest* est utilisé pour retrouver une information donnée qui soit la plus proche de l'expéditeur de la requête. Il peut être utilisé, par exemple, pour retrouver l'arrêt de bus le plus proche de moi.

- l'opérateur *close* est une extension de l'opérateur closest. Cet opérateur est utilisé pour retrouver plusieurs éléments proches de l'émeteur de la requête. Il est également possible de préciser un paramètre de distance. Ce paramètre optionnel est un entier représentant le nombre de mètres maximal séparant l'information recherchée de l'expéditeur de la requête.

Le service d'évaluation de requêtes dépendantes de la localisation s'articule donc autour de 3 étapes différentes :

- Etape A : l'évaluation de la localisation de référence (la localisation géographique du client) en fonction de la localisation des autres pairs qui l'entourent (cf. section précédente).
- Etape B : l'évaluation de la requête classique, non dépendante de la localisation (exemple : retrouver toutes les pizzérias). Cette étape peut s'exécuter simultanément à la première
- Etape C : l'évaluation de l'opérateur de localisation (inside, close ou closest). L'évaluation de l'opérateur de localisation repose sur les résultats obtenus par l'évaluation de la localisation du client et par l'évaluation de la requête classique. L'opérateur de localisation peut être évalué successivement sur chaque pair ayant évalué la requête classique et disposant de la localisation de référence. Il peut aussi être uniquement évalué sur le client lorsque celui-ci reçoit des réponses de la requête classique.

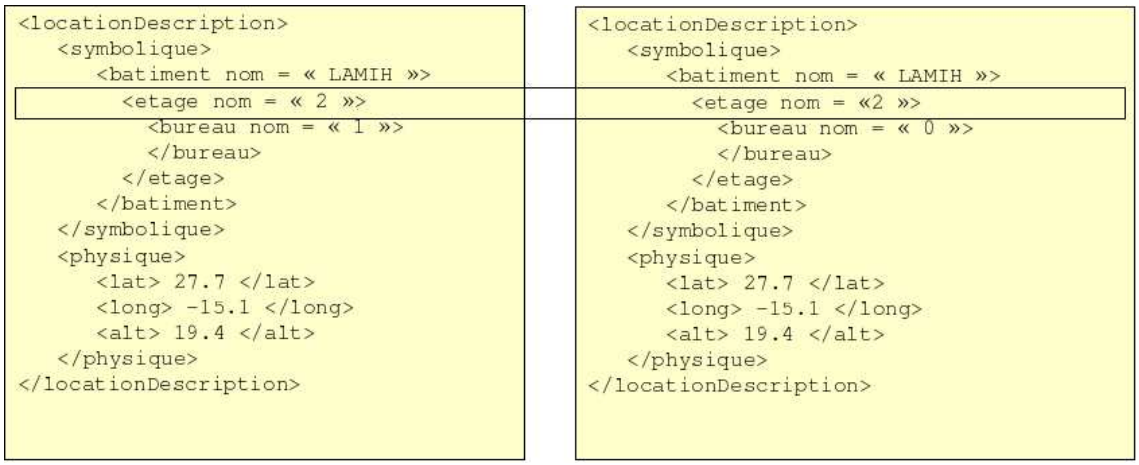

FIG. .4.10 – *exemple de Fonctionnement de l'opérateur Inside(étage)*

L'évaluation des opérateurs repose sur la comparaison entre deux fichiers de localisation (un issu de la localisation approximative du client, l'autre obtenu par les pairs répondant à la requête classique)(cf.figure .4.10 pour le fonctionnement de l'opérateur *Inside*). Cela n'est possible que si tous les pairs concernés fournissent une représentation de la localisation en données symboliques identique (DTD XML). En effet, prenons par exemple le paramètre utilisé par *inside*. Lors d'une recherche sur des éléments au même étage d'un bâtiment que le mien (exemple : "Afficher les imprimantes de mon étage"), on doit être en mesure de comparer une balise XML "étage". Chaque élément de la liste des localisations issues de la requête est comparé à chaque localisation de la liste des localisations du client (localisation de référence). Lorsqu'il y a une correspondance de la valeur de l'élément entré en paramètre, l'élément de la liste des localisations issues de la requête est sélectionné. Plus d'informations sur l'évaluation des trois opérateurs peut être trouvée dans le mémoire de doctorat de Marie Thilliez **[THI04]**.

4.5 Travaux d'optimisation

La principale limite des applications de proximité réside dans les ressources disponibles pour les utilisateurs, en particulier en terme d'énergie. De ce fait, l'évaluation des requêtes dépendantes de la localisation doit fournir une qualité de réponse satisfaisante tout en minimisant les ressources utilisées pour effectuer cette évaluation.

L'économie d'énergie ne va pas se faire en diminuant les temps d'évaluation de la requête (ce qui est souvent l'approche utilisée en base de données), mais en limitant l'utilisation de la connexion Wifi. Il est même possible, à la vue des stratégies utilisées, que le temps d'évaluationglobal augmente. En effet, nous sommes dans un réseau Ad-Hoc, et donc avec des applications et des données extrêmement distribuées. Pour obtenir une réponse satisfaisante, la requête doit donc être distribuée sur les différents noeuds de l'application ; les communications et les évaluations multiples liées à cette distribution sont particulièrement coûteuses pour les terminaux nomades.

Les solutions envisagées sont également différentes de celles proposées pour les recherches d'informations dans les systèmes P2P qui cherchent uniquement à réduire l'utilisation de la bande passante pour l'évaluation d'une requête. Ces solutions sont dédiées à des environnements fiables et filaires, elles n'intègre donc pas les contraintes de temps et de consommation d'énergie présentes dans les applications de proximité.

4.5.1 Solutions envisagées

Les solutions envisagées pour optimiser l'évaluation des requêtes dépendantes de la localisation reposent sur l'expérience acquise lors de nos travaux sur la recherche d'information dans le HAN. En effet, pour calculer des résultats satisfaisants pour une requête issue d'un pair, la requête doit, la plupart du temps, être distribuée et évaluée sur d'autres noeuds. Comme nous l'avons vu dans les travaux sur la recherche d'information dans le HAN (cf. section 4.3.3), l'espace de recherche dans lequel la requête est propagée, doit être sélectionné afin de limiter le nombre de messages échangés et le nombre d'évaluations effectuées sur des noeuds distants. Afin de garantir une qualité de réponse, il est donc important de sélectionner les pairs auxquels la requête est envoyée afin d'obtenir un résultat satisfaisant tout en minimisant les coûts associés à l'évaluation et à l'envoi de la requête. Cette sélection permet de limiter le nombre de noeuds visités et donc la charge de travail des différents noeuds présents dans l'application. De plus, sachant que les super noeuds offrent davantage de ressources que les noeuds légers, une répartition adéquate de la charge de travail doit également être mise au point.

La restriction de l'espace de recherche permet de minimiser les ressources consommées. Cependant, elle peut contribuer à une perte de la qualité de la réponse obtenue. L'objectif est donc d'obtenir le meilleur compromis entre optimisation des ressources et maintien de la qualité de la réponse. Dans une application de proximité, l'espace de recherche de la requête est, a priori, représenté par l'ensemble des noeuds présents dans la sphère de communication : à la fois les noeuds centraux et les noeuds légers. Afin de limiter le nombre de noeuds visités et les coûts associés à l'évaluation d'une requête, nous définissons deux autres sélections d'espace de recherche :

- Comme cela a été fait pour le HAN, l'espace de recherche composé uniquement des noeuds centraux,

– L'espace de recherche composé par les « bons » noeuds présents dans l'application. Ces « bons » noeuds sont sélectionnés en fonction du nombre de requêtes auxquelles ils ont réussi à répondre précédemment, en fonction de leur disponibilité et du nombre de leurs voisins directs.

Ces deux sélections d'espace de recherche permettent de minimiser la charge de travail des noeuds légers présents dans la sphère de communication puisqu'ils ne sont pas visités par la requête. En effet, ces noeuds légers sont généralement des PDA de personnes peu présentes dans la sphère de communication, et qui peuvent donc faiblement venir en aide aux autres (ex : un visiteur venant pour la première fois dans une ville, ne peut que faiblement guider les autres). L'espace de recherche peut, bien sûr, également toujours être réduit en minimisant la profondeur maximale (cf. section 4.3.3).

Afin d'économiser les ressources des clients légers, nous avons également considéré l'utilisation d'un noeud de service. Ce noeud de service représente un noeud connecté au noeud client qui est capable de servir de relais entre le client et les autres noeuds de l'application. Il peut être vu comme un représentant externe sur le réseau, plus puissant, et pouvant se suppléer en partie au travail du client. Si un tel noeud est disponible à proximité du client, son utilisation permet, en particulier, de réduire la charge de travail du noeud client liée à la réception de multiples messages. En effet, le noeud de service permet de réceptionner l'ensemble des réponses reçues, de les traiter et de ne renvoyer que les réponses sélectionnées au noeud client. Le noeud de service fournit également en partie un support aux déconnexions du noeud client. L'utilisation d'un tel noeud de service peut se rapprocher des travaux sur l'optimisation de requêtes dans les environnements mobiles, reposant sur l'utilisation d'agents mobiles, venant suppléer les terminaux légers **[OMH05]**.

4.5.2 Stratégies d'ordonancement et résultats

En ce qui concerne les requêtes de localisation, l'ordonnancement des différentes étapes à effectuer pour l'évaluation de ces requêtes participe également à l'optimisation des coûts. En effet, les différentes étapes (évaluation de la localisation de référence, évaluation de la requête classique et évaluation de l'opérateur de localisation) peuvent être ordonnancées différemment. Certaines étapes peuvent être effectuées en parallèle. Dans la section suivante, nous allons étudier les différentes stratégies d'optimisation proposées en fonction de l'ordonnancement des étapes et de l'utilisation d'un noeud de service.

Les différentes stratégies sont présentées par les figures .4.11 et .4.12. Les stratégies (a) et (b) reposent sur un ordonnancement logique des opérations tandis que les stratégies (c) et (d) exploitent la parallélisation possible des deux premières étapes du modèle d'évaluation : évaluation de la localisation géographique approchée du client et évaluation de la requête classique. Les stratégies (b) et (d) reposent sur l'utilisation d'un noeud de service proposée dans la section précédente. Pour rappel, l'étape A correspond à l'évaluation de la localisation du client (cf. section 4.4.1), l'étape B correspond à l'évaluation de la requête "classique", et l'étape C correspond à l'évaluation de l'opérateur de localisation.

La stratégie (a) peut être considérée comme naïve, car elle est la plus "séquencielle" des quatres, mais elle a l'avantage de préserver les ressources des noeuds légers. Le module de positionnement calcule tout d'abord la localisation du client sur le noeud client. Ensuite, la requête de localisation (avec la localisation du client en paramètre) est envoyée aux noeuds de l'espace de recherche. Quand un noeud reçoit et accepte la requête, elle est évaluée par

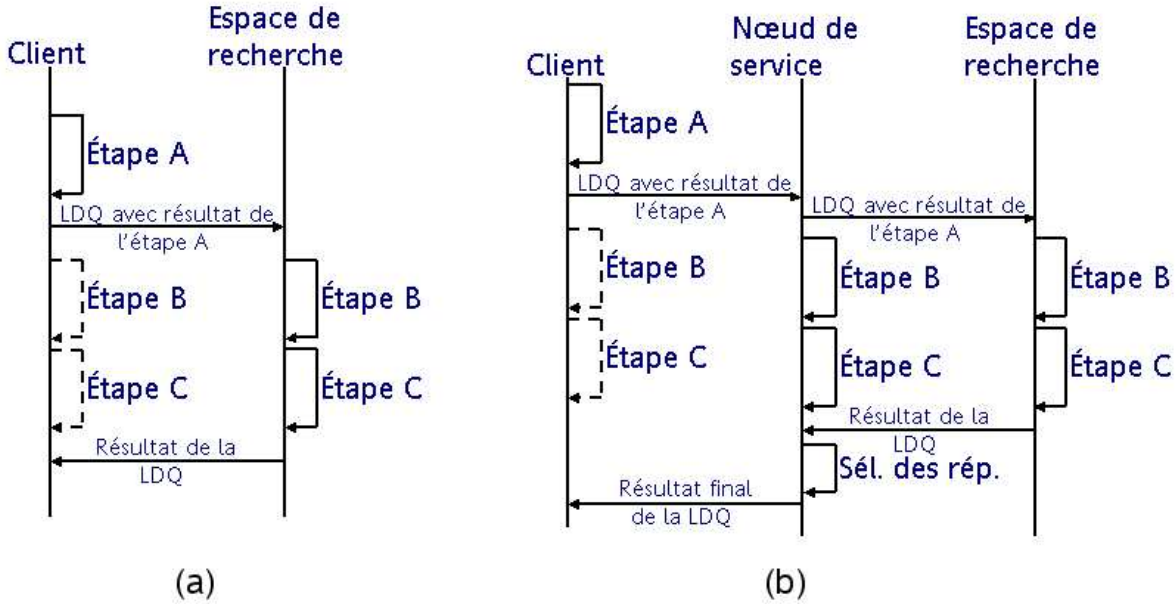

FIG. .4.11 – *stratégies d'évaluation*

son service de localisation en utilisant les données stockées localement : chaque noeud évalue à la fois la requête classique et l'opérateur de localisation en fonction de la localisation définie en paramètre et des résultats de la requête classique. Si une réponse existe, celle-ci est envoyée au client en utilisant le chemin inverse de la requête. Finalement, le noeud client construit le résultat final avec les réponses reçues en fonction des paramètres désirés par le client tels que le nombre de réponses souhaitées.

La seconde stratégie repose sur l'utilisation d'un noeud de service. Comme dans la première stratégie, le module de positionnement du client calcule sa position. Ensuite, la requête (avec le paramètre de localisation renseigné) est envoyée au noeud de service. Le noeud de service envoie ensuite la requête aux noeuds de l'espace de recherche. Ces noeuds effectuent, s'ils le souhaitent, l'évaluation globale de la requête de localisation : évaluation de la requête classique et évaluation de l'opérateur de localisation. Les réponses sont récupérées par le noeud de service qui construit la réponse finale et la renvoie au noeud client.

La stratégie (c) diffère des deux premières stratégies par l'ordonnancement des étapes de l'évaluation. Pendant que la requête classique (sans le paramètre de localisation) est envoyée aux noeuds de l'espace de recherche, le noeud client évalue sa localisation. Ensuite, lorsque que le noeud client reçoit les différentes réponses, l'évaluateur de requêtes du service de localisation évalue l'opérateur de localisation en fonction de la localisation calculée et des réponses reçues sur le noeud client. Enfin la dernière stratégie reprend la précédente en rajoutant l'utilisation d'un noeud de service.

L'évaluation de ce travail s'est faite en deux temps. Tout d'abord, un prototype reprenant le principe de Commerce electronique de Proximité, a été réalisé par un étudiant de Master 2 sous la responsabilité de Marie Thilliez et Thierry Delot **[TCD03]** dans le cadre de son stage de fin d'année. Puis, ce travail a été repris dans le cadre de plusieurs projets individuels de master2. Le résultat a abouti à un prototype très fidèle à l'application dé-

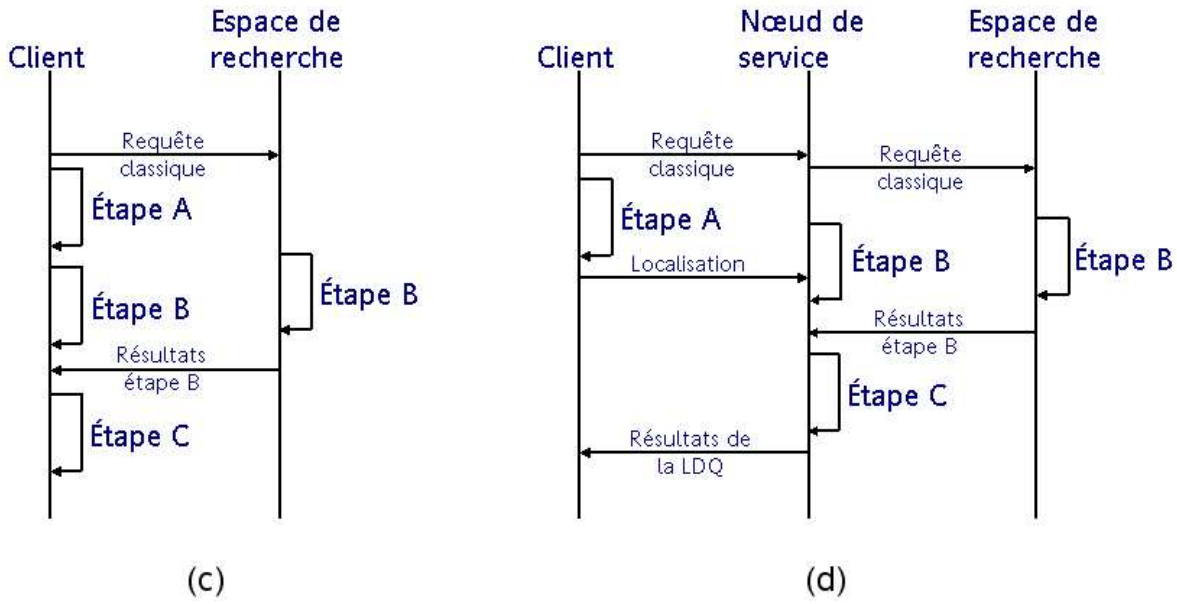

FIG. .4.12 – *stratégies d'évaluation (suite)*

crite, et qui a été plusieurs fois présenté lors de réunion des groupes de recherche, ou lors de démonstration en conférence.

Afin d'évaluer les différentes stratégies d'optimisation, Marie Thilliez a effectué des simulations, à partir des données extraites du prototype. Les critères d'évaluation pour les différentes stratégies ont été la consommation d'énergie et le temps de réponse. Pour chaque stratégie, les différentes étapes nécessaires à l'évaluation des requêtes dépendantes de la localisation ont été simulées ainsi que le parcours des requêtes et des réponses afin de définir le nombre d'opérations nécessaires. Suite à l'expérience tirée des travaux sur le HAN, le simulateur permet de paramétrer pour chaque stratégie la profondeur choisie et l'espace de recherche sélectionné (tous les noeuds, les noeuds centraux ou les « bons » noeuds).

Le simulateur permet de définir également l'environnement de l'application de proximité. Le but ici était de simuler un contexte bien connecté afin d'analyser au mieux les performances des différentes stratégies (surface de 100m x 100m avec 50 noeuds et la portée du réseau est fixée à 40m). Ensuite, différents tests ont été effectués en fonction de différentes densités de noeuds centraux (5 à 90% des participants) et la densité des « bons » noeuds varie (de 0 à 100% des participants). Deux environnements représentatifs des applications de proximité ont été définis :

- Un environnement de référence composé de 50% de noeuds centraux et de 50% de noeuds légers. Les « bons » noeuds représentent 50% des noeuds centraux. Cet environnement de laboratoire sert ensuite de référence.
- Un environnement « réel », où la densité de noeuds centraux est de 10% et la densité de noeuds légers de 90%. La densité de « bons » noeuds parmi les noeuds centraux est toujours paramétrée à 50%. Cet environnement reflète relativement bien les proportions réelles que l'on pourrait rencontrer dans les applications de proximité.

Pour un espace de recherche non limité, les variations des temps de réponses ne sont

pas suffisamment importantes comparées aux coûts en terme de consommation d'énergie : la stratégie (b) reste plus intéressante si un noeud de service est disponible, la stratégie (a) sinon. En revanche, pour un espace de recherche limité (noeuds centraux ou bons noeuds), la stratégie (d) devient alors la plus intéressante car elle permet de minimiser considérablement le temps d'évaluation tout en n'ajoutant pas de consommation d'énergie pour les noeuds légers.

	Noeud de service	Pas de noeud de service
Espace restreint	stratégie (b)	stratégie (a)
Recherche exhaustive	stratégie (d)	stratégie (c)

TAB. .4.2 – *Récapitulation de l'utilisation des stratégies*

Bien que les différentes stratégies d'optimisation proposées soient indépendantes du choix de l'espace de recherche, le choix de cet espace de recherche a un impact important sur les coûts liés à l'évaluation et donc sur la sélection de la meilleure stratégie d'évaluation à utiliser. Si l'espace de recherche choisi est limité aux noeuds centraux (comme dans le cas du HAN), les coûts associés à l'évaluation sont considérablement diminués. Cependant, aujourd'hui, il nous est difficile de définir de façon précise quel est l'espace de recherche à choisir pour l'évaluation d'une LDQ dans un environnement donné. En effet, dans un environnement connu comme le HAN, où tous les terminaux sont de confiance, utiliser massivement les super-noeuds ne pose pas de problème. Dans le cas d'une application de proximité tel qu'elle est définie ici, cela est plus aléatoire. Le tableau .4.2 montre bien qu'il n'existe pas alors de stratégie idéale. Il devient presque obligatoire de définir la stratégie à utiliser, dynamiquement, en fonction du contexte de l'application.

4.6 Conclusions

Comme nous pouvons nous en rendre compte à la lecture des différents résultats (aussi bien pour le HAN que pour le service de requêtes dépendantes de la localisation), il n'existe pas une solution idéale permettant de répondre aux différents besoins applicatifs et d'optimsation que 'on peut rencontrer. Pourtant, l'architecture réseau est toujours sensiblement la même (architecture P2P hybride, utilisant des super-noeuds). Mais, le contexte de l'application (confiance dans les partenaires, nombre de pairs dans le réseau, etc...) change les stratégies idéales à choisir.

La concéquence de cela, alors que l'approche suivie était radicalement différente par rapport aux travaux sur l'adaptabilité des services techniques (nous sommes ici partis d'un service totalement nouveau et non pasa un service que nous cherchions à adapter à l'informatique ubiquitaire et nomade), nous nous retrouvons une nouvelle fois avec des besoins d'adaptabilité.

Nous pouvons donc nous rendre compte qu'il est important, ici aussi, de fournir plusieurs versions (ou personnalités) de notre service de requêtes, de manière à tenir compte de données environnementales et applicatives. Il est donc dorénavent important de travailler sur un service de requêtes qui disposera de caractéristiques adaptables, et qui pourra être utilisé par les mécanismes définis dans le chapitre précédent.

Ce travail fait l'objet d'une thèse débutée en septembre 2004 par Hocine Grine **[GRI04], [GDL05]**, et co-encadrée par Thierry Delot et moi même. Dans le cadre de ce doctorat, Hocine Grine devra se servir des travaux de Colombe Hérault et de ceux de Marie Thilliez, pour fournir un évaluateur de requètes, capable d'évoluer, à la fois en fonction des besoins applicatifs (besoin de requètes dépendantes de la localisation, d'une évaluation de requètes continues, etc...) et des contraintes d'environnement (présence ou non d'un signal GPS, d'un réseau Wifi, environnement fortement distribué, etc...).

Chapitre .5

Conclusion et Perspectives

5.1 Conclusion

5.1.1 Bilan scientifique

Dans un premier temps, nos travaux ont été consacrés à fournir des services techniques pour faciliter la tâche des développeurs de nouvelles applications nomades. Nous avons donc travaillé sur une méthode de conception de ces services, offrant une réutilisabilité maximale, et une adéquation avec les plateformes d'exécution actuelles des applications (à savoir les plateformes à composants).

Cela s'est fait en deux étapes : dans un premier temps, nous avons disséqué un service technique, qui fait partie de ceux qui sont le plus utilisés : le service de gestion de transactions distribuées. Ce service est représentatif des besoins en adaptation que l'on peut rencontrer, puisque (1) il peut implanter des modèles différents de transactions, correspondants à des applications plus ou moins complexes, (2) les protocoles de communication et de validation doivent être adaptés aux types de participants à l'application. Puis, nous avons fait évoluer ce service, de manière à ce qu'il soit en mesure de répondre au développement des applications commerciales inter-entreprises sur le web. Cela s'est fait en ajoutant, au gestionnaire de transactions de départ, les modèles des transactions emboîtées et des transactions emboîtées ouvertes.

Pour réaliser cela, nous sommes repartis d'un moniteur existant, que nous avons encapsulé dans un composant logiciel, auquel nous sommes venu connecter d'autres composants, offrant la gestion des transactions emboîtées ouvertes. Le modèle à composants avait été choisi au départ pour répondre aux problèmes liés au besoin de compensation des transac-

tions emboîtées ouvertes. Ces travaux nous ont prouvé que son apport pouvait également être important dans la phase de conception des services techniques. Cela nous a permis de réfléchir à des solutions plus globales, et de concevoir un modèle de de réalisation de services techniques adaptables à base de composants. Dans cette seconde étape, en généralisant l'approche suivie précédemment, nous avons fourni une architecture complète pour l'adaptation dynamique des services techniques, prenant en compte le contexte d'exécution, et les besoins des applications.

La dernière partie de nos travaux concerne la réalisation de services propres aux applications nomades, et s'exécutant sur des terminaux légers. Dans le cadre de ces travaux, nous n'avons pas suivi le modèle de développement à base de composants cité précédemment pour plusieurs raisons : les travaux s'effectuant sur la même période, le modèle de conception ainsi que l'architecture d'adaptation n'était pas encore aboutis lors des travaux sur les applications de proximité; de plus, les terminaux nomades (PDA ou Smartphone) disponibles à cette période (2001), ne permettaient pas encore d'exécuter les plateformes considérées dans le travail sur l'adaptation. Cependant, nous avions besoin de données réelles pour nos travaux sur les algorithmes d'optimisation de l'évaluateur de requêtes dépendantes de la localisation. Il a donc été conçu de manière traditionnelle, à l'aide de langages objets. Le bilan de ce tavail est tout à fait satisfaisant, et nous a permis d'identifier les besoins en terme d'économie de bande passante (pour préserver l'autonomie des terminaux) et d'efficacité d'exécution (en ce qui concerne la recherche d'information). Cependant, le fait de ne pas avoir modélisé cet évaluateur de requête sous la forme d'un assemblage de composants, rend son évolutivité vers d'autres besoins applicatifs et d'autres optimisations plus délicat (il est impossible de remplacer rapidement une partie de l'évaluateur par d'autres "morceaux" applicatifs). Aussi, nous verrons dans les perspectives, que ce travail va maintenant se poursuivre, tirant ainsi tous les bénéfices des travaux sur l'adaptabilité dynamique.

5.1.2 Evolution du thème Systèmes d'Information Distribués

Ces années de recherche ont été intégralement consacrées à faire grandir un thème de recherche, celui des Systèmes d'Information Distribués (SID), au sein de l'équipe de Recherche Opérationnelle et Informatique du laboratoire LAMIH. Ce thème, créé par Arnaud Fréville, dont Didier Donsez a été le premier responsable, est devenu une part importante de l'équipe de recherche dont nous faisons partie, avec de jeunes enseignants chercheurs très motivés et trois thèses soutenues cette dernière année.

L'équipe a su également se faire connaître au travers de participations à des projets de recherche Régionaux (avec une participation aux différents contrats de plan état/régions (TACT COLORS puis TAC MOSAIQUES)), Nationaux (participation à des projets RNTL (IMPACT), RNRT (COMPiTV) et RNTS (COQUAS)), et internationaux (participation à un projet Européen ITEA (PEPiTA)), alors qu'elle n'était constituée que de trois maîtres de conférences et de trois doctorants.

Aujourd'hui, SID a atteint une certaine maturité, avec une recherche bi-polaire (travaux sur l'adaptabilité et les services dédiés aux applications nomades, présentés ici, mais également travaux sur l'étude de la sureté de fonctionnement des applications conçues à base de composants), et la présence de six maîtres de conférences et trois doctorants. Dans ce cadre, les travaux d'animation de la recherche et de gestion de projet au sein de SID furent très enrichissants et très passionnants.

D'un point de vue plus personnel, ce mémoire est aussi l'occasion de faire un premier bilan sur le métier d'enseignant chercheur. Ce travail fut très enrichissant sur bien des points, comme par exemple l'encadrement de doctorants, pour mener à bien un travail de longue haleine (généralement entre 3 et 4 ans). J'ai également particulièrement apprécié la prise en charge de la présidence du comité de programme de la conférence Mobilité et Ubiquité 2005 (UBIMOB'05) avec le professeur Joelle Coutaz de l'IMAG. Cette tâche, même si elle a demandé de nombreuses heures de travail, m'a permis de mieux comprendre le fonctionnement des comités de programme, ainsi que la difficulté d'équilibrer une conférence dans les différents thèmes qu'elle comporte. Cette expérience sera en partie renouvelée, puisque SID co-organise, avec l'université de Lille, l'édition 2006 des journées BDA [1].

5.2 Perspectives

Les perspectives de ces travaux sont de plusieurs ordres. Tout d'abord, concernant les travaux sur l'adaptabilité dynamique, de manière à définitivement valider l'approche proposée, il est important de continuer les tests et mesures déja effectués, et de les compléter en terme de :

- réponse à la montée en charge : dans l'architecture choisie, le coordinateur et l'annuaire sont deux éléments qui risquent d'avoir du mal à répondre à une demande importante de la part de plusieurs applications. L'étude de la montée en charge de notre système n'a pas encore été réalisée, puisque les tests ont été faits au travers de l'adaptation d'un service technique, pour répondre aux besoins d'une application, alors que l'annuaire ne comportait qu'une vingtaine de personnalités P1S différentes.
- fournir des mécanismes de déploiement : dans notre architecture d'adaptation, l'annuaire de services techniques fournit des références vers des composants implantant le service. Cependant, il se peut que le déploiement de ces composants doive être effectué sur des terminaux hétérogènes. Il convient donc de fournir des mécanismes permettant d'assurer le déploiement des services techniques sur les terminaux considérés. En effet, notre architecture permet de choisir le service technique à utiliser. Si les composants de ce service ne sont pas présents sur le terminal cible, il va falloir télécharger l'assemblage de composants et l'instancier, en tenant compte des caractéristiques de la plateforme à composants s'exécutant sur le terminal. Ce travail pourra se faire en collaboration avec l'équipe GOAL du LIFL et l'équipe LEOST de l'INRETS (qui traitent déjà des problèmatiques semblables **[FGM05]**) dans le cadre du projet du contrat de plan état région MOSAIQUES, débuté en janvier 2005.

Dans le cadre des travaux sur les applications de proximité, nous avons proposé un évaluateur de requêtes dépendantes de la localisation des utilisateurs. Cela correspondait à un besoin des utilisateurs avec l'apparition de terminaux de plus en plus nomades et connectés. Aujourd'hui, d'autres besoins se font également ressentir. On peut citer par exemple , en terme de fonctionnalité de l'évaluateur, le besoin d'évaluation de requêtes en continu **[TWB03]**. Ou encore, comme nous l'avons déjà montré dans la conclusion du chapitre 4, des besoins différents en terme d'optimisations sont également envisageables pour notre évaluateur de requêtes, notamment en fonction de l'architecture du réseau et des contraintes matérièlles **[MTV05]**. Il est donc important d'appliquer les principes d'adaptation des services techniques que nous avons conçus, à notre évaluateur de requêtes. Ce

1. BDA : Bases de Données Avancées

travail a débuté en octobre 2004, par un doctorat qui a pour but de fournir un évaluateur de requêtes capable de s'adapter aux applications et aux contraintes d'environnement, dans le cadre des applications ubiquitaires, basé sur l'approche préconisée dans les travaux de Colombe Hérault.

Dans le cadre du projet MOSAIQUES, nous travaillons maintenant sur la définition d'un méta-modèle pour l'adaptation des applications dans le domaine de l'informatique ubiquitaire. Une application est alors vue comme une association de services (techniques ou applicatifs). Au fil des discussions, il est apparu que la définition de ce modèle se heurtait à un certain nombre de définitions, qu'il était important de préciser dans le cadre de l'informatique ubiquitaire :

- définition du contexte (même si actuellement, toutes les définitions sont basées sur les travaux de A.K. Dey **[DEY01]**, les définitions qui en découlent divergent fortement. On peut par exemple citer les travaux menès à l'IMAG sur la formalisation du contexte dans les environnement momades **[KVG05]**, ou les travaux sur la définition et la représentation du contexte de l'école des mines de St Etienne **[BBB05]**,
- critère d'adaptation des applications : à partir de quel moment allons pouvoir décider d'adapter l'application,
- prise en compte des préférences utilisateurs : quelles vont être les interférences de ces préférences avec les décisions d'adaptation prises par le système.

Les deux premiers points sont des choses auxquelles nous nous sommes déjà confrontés dans nos travaux sur l'adaptabilité des services techniques. La prise en compte de critères applicatifs (c'est à dire fonctionnels) rend toutefois ces définitions encore plus complexes (notamment du fait de la présence de l'utilisateur et de l'interface homme/machine), qui font que les décisions ne peuvent plus être prises uniquement sur des critères systèmes (comme cela est le cas dans notre travail sur les services techniques).

Les dernières perspectives concernent les problèmes de sureté des logiciels réalisés uniquement par assemblage de briques plus ou moins connues. Ces travaux sont très importantes dans le cadre du groupe SID. Nous avons défini des services comme étant des assemblages de composants répondant à certaines caractéristiques, qui sont énoncées dans des personnalités appelées P1S. Ces personnalités étant définies par le programmeur de services techniques, il devient donc important de définir des outils formels pour fournir les moyens de contrôler le travail effectué par ces assemblages. Ces outils doivent intervenir à plusieurs moments du cycle de vie de l'application. Le premier problème consiste à préciser ce qu'est la qualité de service d'un assemblage de composants : comment pouvons-nous évaluer de manière formelle cette qualité de service. Une fois cette qualité de service évaluée, il conviendra de contrôler si l'assemblage de composants fourni répond bien à cette demande de qualité, et de garantir, lors d'une adaptation, que le (ou les) composant(s) remplaçant offre une qualité de service au moins équivalente. Ces travaux, débutés au LAMIH avec la soutenance de la thèse de D. Petit **[PET03]**, sont au coeur de préoccupations de plusieurs autres équipes, et notamment RAIMBOW du laboratoire I3S **[BCE04]**.

Chapitre .6

Bibliographie

[ABG01] A. Andersen, G. Blair, V. Goebel, R. Karlsen, T. Stabell, W. Yu,
Artic Beans : configurable and Re-configurable Enterprise Component Architectures,
work in progresse session ACM / IFIP / USENIX Middleware'01, Allemagne, novembre 2001.

[ACD05] M. Anne, J.L. Crowley, V. Devin, G. Privar,
localisation Intra-batiment multi-technologies : RFID, Wifi et Vision,
, Actes de la deuxième conférence UBIMOB'05, Mobilité et Ubiquité, Grenoble, France, 2005.

[AP98] M. Abdallah, P. Pucheral,
A Non-Blocking Single-Phase Commit Protocol for Rigorous Participants,
Networking and Information Systems Journal (NIS), Vol.1, n° 2, December 1998.

[AVA94] G. Alonso, R. Vingralek, D. Agrawal, Y. Breitbart, A.E. Addabi, H.J. Schek, G. Weikum,
Unifying Concurrency Control and Recovery of Transactions,
Proceedings of Advances in Database Technology - EDBT'94. 4th International Conference on Extending Database Technology, Cambridge, United Kingdom, 1994.

[BBB05] O. Bucur, P. Beaune, O. Boissier,

Representing Context in a Multi-Agent System for Context-Based Decision Making, Proc. of International Workshop on Ambient Intelligence, Utrecht, Netherlands, July 2005.

[BCE04] M. Blay-Fornarino, A. Charfi, D. Emsellem, A.M. Pinna-Dery, M. Riveill, *Software Interactions*, Journal of Object Technology, 3(10) pp161-180, 2004.

[BCF97] J. Besancenot, M. Cart, J. Ferrié, R. Guerraoui, P. Pucheral, B. Traverson, *Les systèmes Transactionnels : concepts, normes et produits*, Edition Hermes,ISBN 2-86601-645-9, 1997.

[BCS04] E. Bruneton, T. Coupaye, J.B. Stefani, *The Fractal component Model*, Rapport technique 2.0.3, Objectweb Consortium, Février 2004.

[BDL02] N. Bennani, T. Delot, S. Lecomte, S. Nemchenko, D. Donsez, *Advanced transactional model for component-based model*, IEEE ISSADS, International Symposium on Advanced Distributed Systems, Guadalajara, Mexique, novembre 2002.

[BEK00] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelson, H.F. Nielson, S. Thatte, D. Winer, *Simple Object Access Protocol (SOAP) 1.1*, W3C note, 2000.

[BGM95] P.G. Bosco, E. Grasso, C. Moiso, *An Extended Transaction Model for an Object Distributed Architecture*, First International Workshop on High Speed Networks and Open Distributed Platforms, Berlin, Mai 1995.

[BIL95] D. Billard, *La reprise dans les systèmes transactionnels exploitant la sémantique des opérations typées*, Thèse de Doctorat, Univ. de Montpellier II, mai 1995.

[BP00] P. Bahl, V. Padmanabhan, *RADAR: An in-building RF-based user location and tracking system*, International Conference IEEE INFOCOM, Israel, 2000.

[BRU01] E. Bruneton, *Un support d'exécution pour l'adaptation des aspects non-fonctionnels des applications réparties*, Mémoire de Doctorat, Institut National Polytechnique de Grenoble, octobre 2001.

[BTP02] *OASIS Business Transaction Protocol*,

Version 1.0, An OASIS Committee Specification, 2002.

[CDE02] A. Chiquet, D. Donsez, J. Estublier, C. Hérault, S. Leblanc, S. Lecomte, P. Merle, O. Potonniée, T. Vessière,
Architecure Générale,
Délivrable du projet COMPiTV, 2002.

[CER03] A. Charfi, D. Emsellem, M. Riveill,
Dynamic component composition in .NET,
.NET: The programmer's Perspective: ECOOP Workshop 2003 publié dans le Journal of Object Technology, 3(2):p37-46, february 2004.

[CCC02] F. Cabrera, G. Copeland, B. Cox, T. Freund, J. Klein, T. Storey, S. Thatte,
Web Services Transaction (WS-Transaction),
Specification, IBM, Microsoft, BEA, 2002

[CCM01] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana,
Web Services Description Language (WSDL) 1.1,
W3C note, 2001.

[CP03] Y.-F. R. Chen, C. Petrie,
Ubiquitous Mobile Computing,
IEEE Internet Computing, Vol. 7, N° 2, 2003.

[CR90] P.K. Chrysanthis, K. Ramamritham,
ACTA: a framework for specifying and reasoning about transaction structure and behavior ,Proceedings of the ACM SIGMOD international conference on Management of data, Etats unis, 1990.

[CRM04] M. Cremene, M. Riveill, C. Martel, C. Loghin, C. Miron,
Adaptation dynamique de services,
1ère Conférence Francophone sur le Déploiement et la (Re)Configuration de Logiciels (DECOR), pages 53-64, Grenoble, octobre 2004.ISBN : 2-7261-1276-5

[DBC04] H. Duran Limon, G. Blair, G. Coulson,
Adaptative Ressource Management in Middleware : A survey,
IEEE distributed System Online, juillet 2004.

[DEY01] A.K. Dey,
Understanding and Using Context,
Personal and Ubiquitous Computing Journal, Volume 5 (1), 2001.

[DDS97] W. Du, J. Davis, M.-C. Shan, U. Dayal,
Flexible Compensation of Workflow Processes,
Hewlett-Packard Technical report, HPL-96-72(R.1), 1997.

[DUC02] F. Duclos
Environnement de gestion de services non fonctionnels dans les applications à composants,

Thèse de doctorat de l'université de Grenoble, 2002.

[DVD01] K. Dutta, D. VanderMeer , A. Datta, K. Ramamritham,
User Action Recovery in Internet SAGAs (iSAGAs),
Technologies for E-Services, Second International Workshop, Proceedings, TES 2001, Italy, 2001.

[FGM05] A. Flissi, C. Gransart, P. Merle,
Une infrastructure à composants pour les applications ubiquitaires, Actes de la deuxième conférence UBIMOB'05, Mobilité et Ubiquité, Grenoble, France, 2005.

[GCL99] Y. Goland, T. Cai, P. Leach, Y. Gu, S. Albright,
Simple Service Discovery Protocol,
Internet Draft draft-cai-ssdp-v1-03, disponible à l'adresse: http://www.upnp.org/download /draft$_c ai_s sdp_v 1_0 3$.txt, 1999.

[GDL05] H. Grine, T. Delot, S. Lecomte,
Adaptive Query Processing in Mobile Environment
in the 3rd International workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC 2005), Grenoble, 2005.

[GEI01] K. Geihs,
Middleware Challenges Ahead,
IEEE Computer, 34(6): pp24-31, June 2001.

[GHL05] H. Grine, C. Hérault, S. Lecomte, T. Delot,
Localisation de Services Techniques dans un Modèle à Composants,
Actes des 4èmes Journées composants (JC), Le Croisic, avril 2005.

[GRA81] J. Gray,
The Transaction Concept: Virtues and Limitations,
Very Large DataBases 1981, pp 144-154, Septembre 1981.

[GRI04] H. Grine,
Localisation de Services Non-Fonctionnels dans un Modèle à Composants,
DEA en informatique, Université de Valenciennes et du Hainaut Cambrèsis, Juin 2004.

[GS87] H. Garcia-Molina, K. Salem,
SAGAS,
Preceedings ACM International Conference on Management of Data, pp 249-259, San-Francisco, mai 1987.

[HC01] G. Heineman, W. Council,
Component Based Software Engineering, putting the pieces together,
ISBN 0-201-70485-4, Addison Weysley, 2001.

[HBD02] C. Hérault, N. Bennai, T. Delot, S. Lecomte, M. Thilliez,

Adaptability of non-functional services for component model: application to the M-Commerce, IEEE ISSADS, International Symposium on Advanced Distributed Systems, Guadalajara, Mexique, novembre 2002.

[HER05] C. Hérault, *Adaptabilité des Services Techniques dans le Modèle à composants*, Thèse de doctorat en informatique, Université de Valenciennes et du Hainaut Cambrèsis, Juin 2005.

[HILL99] A. Hills,
Wireless andrew, IEEE Spectrum, Vol 36(6), pp49-53, 1999

[HL03] C. Hérault, S. Lecomte,
Adaptabilité des services techniques dans un modèle à composants,
3ème Conférence Française sur les Systèmes d'Exploitation, CFSE, La Colle Sur Loup, France, octobre 2003.

[HL04] C. Hérault, S. Lecomte,
Gestion de personnalités de services techniques dans un modèle à composant,
rapport interne au LAMIH numéro 14/2004. Soumis à la revue l'Objet, 2004.

[HL04b] C. Hérault, S. Lecomte,
Gestion Dynamique des Services Techniques pour Modèle à Composants,
1ère Conférence Francophone sur le Déploiement et la (Re)Configuration de Logiciels (DECOR), pages 135-146, Grenoble, octobre 2004. ISBN : 2-7261-1276-5

[HLD04] C. Hérault, S. Lecomte, T. Delot,
New technical services using the component model for applications in heterogeneous environment.,
Innovative Internet Community Systems (I2CS), Mexique, juin 2004.

[HNL05] C. Hérault, S. Nemchenko, S. Lecomte,
A Component-Based Transactional Service, Including Advanced Transactional Models,
Advanced Distributed Systems: 5th International School and Symposium, ISSADS 2005, Guadalajara, Mexico, Janvier 2005, Revised Selected Papers, publié dans Lecture Notes in Computer Science, Volume 3563 / 2005, ISSN: 0302-9743

[HR83] T. Haerder, A. Reuter,
Principles of Transaction-Oriented Database Recovery,
ACM Computing Surveys, Vol. 15, 1983.

[HS99] K. Hinckley, M. Sinclair,
"Touch-sensing input devices",
ACM CHI'99, International Conference on Human Factors in Computing Systems, pp. 223-230, 1999.

[HW91] C. Hasse, G. Weikum,

A Performance Evaluation on Multi-Level Transaction Management,
Proceedings of the VLDB Conference, 1991.

[JIR03] Jironde Project,
"http://jotm.objectweb.org/jironde.html",
Sous projet du consortium ObjectWeb.

[JK97] S. Jajodia, L. Kerschberg,
Advanced Transaction Models and Architectures
, Kluwer Academic Publishers, Boston, 1997.

[JON01] JOnAS Project,
"http://jonas.objectweb.org",
Java Open Application Server, Projet du consortium ObjectWeb.

[KAR03] R. Karlsen,
An adaptive transactional system - framework and service synchronisation,
Dans les actes de The international Symposium of distributed Objects and Applications, DOA'2003, Italie, 2003.

[KCB02] F. Kon, F. M. Costa, G. S. Blair, R. H. Campbell,
The case for reflective middleware,
Communication ACM vol. 45 pp33-38, 2002.

[KLS90] H.F. Korth, E. Levy, A. Silberschatz,*A Formal Approach to Recovery by Compensating Transactions*,
Proceedings of the VLDB Conference, 1990.

[KVG05] M. Kirsch-Pinheiro, M. Villanova-Oliver, J. Gensel, H. Martin,
Context-Aware Filtering for Collaborative Web Systems: Adapting the Awareness Information to the Users Context,
20th ACM Symposium on Applied Computing (SAC 2005), , ACM Press, pp. 1668-1673, Santa Fe, New Mexico, USA, March 2005.

[LEC98] S. Lecomte,
COST-STIC : Cartes Orientées Services transactionnels et Systèmes Transactionnels Intégrants des cartes,
Thèse de doctorat de l'université de Lille 1, Novembre 1998.

[LS98] J. Liang, S. Sédillot,
MAAO OTS: An OMG Object Transaction Service based on CORBA,
White Paper INRIA, Révision 1.1, Mars 1998.

[MAE87] P. Maes,
Concepts and experiments in computational reflection,
Conference on Object Oriented Programming Systems Languages and Applications, pp 147-155, ACM Press, 1987.

[MOS85] J. E. B. Moss,

Nested Transaction : an Approach to Reliable Distributed Computing,
MIT Press series in Information systems, 1985.

[MHC01] N. Minar, M. Hedlund, C. Shirky, T. O'Reilly, D. Bricklin, D. Anderson, J. Miller, A. Langley, G. Kan, A. Brown, M. Waldman, L. Cranor, A. Rubin, R. Dingledine, M. Freedman, D. Molnar, R. Dornfest, D. Brickley, T. Hong, R. Lethin, J. Udell, N. Asthagiri, W. Tuvell, B. Wiley,
Peer-to-Peer:Harnessing the Power of Disruptive Technologies,
Edité par Andy Oram, première édition Février 2001, ISBN: 0-596-00110-X.

[NEM04] S. Nemchenko, *Modèle de transactions avancées et modèle à composants*,
Thèse de doctorat en informatique, Université de Valenciennes et du Hainaut Cambrèsis, septembre 2004.

[MER01] B. McKee, D. Ehnebuske, D. Rogers,
UDDI Version 2.0 API Specification,
disponible sur le site UDDI.org, 2001.

[MGL02] B. Marquet, C. Gustave, A. Lefebvre, S. Nemchenko, S. Chassande-Barrioz,
Secured services in a multi-tier architecture,
World Telecommunications Congress, France, 2002.

[MMG01] R. Marvie, P. Merle, JM. Geib, S. Leblanc,
TORBA: vers des contrats de courtage,
Electronic Journal on Networks and Distributed Processing, num. 11, Mars 2001, p1-18.

[MSK04] P. K. McKinley, S. M. Sadjadi, E. P. Kasten, B. H. C. Cheng,
Composing adaptive software,IEEE Computer, pages 56-64, Juillet 2004.

[MTV05] M. Scholl, M. Thilliez, A. Voisard,
Location-based Mobile Querying in Peer-to-Peer Networks,
OTM Workshop on Context-Aware Mobile Systems, 2005.

[OHE99] R. Orfali, D. Harkey, J. Edwards,
Client/Serveur Survival Guide, 3ième édition,
Edition Vuibert, 1999, ISBN 2-711-786498.

[OMG02] Spécification de l'OMG,
Additional Structuring Mechanisms for the OTS Specification,
version 1.0, Object Management Groupe, 2002.

[OMG03] *Transaction Service Specification*,
Version 1.4, spécification CORBA Services, OMG, 2003.

[OMH05] B. Ozakar, F. Morvan, A. Hameurlain,
Query Optimization: Mobile Agents versus Accuracy of the Cost Estimation,
International Journal of Computer Systems Science & Engineering, CRL

Publishing Ltd9 De Montfort Mews Leicester Vol. 20 N. 3, p. 161 - 168, mai 2005.

[PAB00] K. Partridge, L. Arnstein, G. Borriello, T. Whitted,
Fast intrabody signaling,
Conference on Wireless and Mobile Computer Systems and Applications, Monterey, Canada, 2000.

[PET03] D. Petit,
Génération automatique de composants logiciels sûrs à partir de spécifications formelles B,
Mémoire de Doctorat, Université de Valenciennes et du Hainaut-Cambrésis, Valenciennes, décembre 2003.

[PRI01] M. Prinkey,
An Efficient Scheme for Query Processing on Peer-to-Peer Networks,
Rapport technique, http://aeolusres.homestead.com/files/index.html, Aeolus Research, Inc., 2001.

[RM00] M. Riveill, P. Merle,
Programmation par composants,
Technique de l'ingénieur, traité informatique, numéro H2 759, novembre 2000.

[RM03] R. Rouvoy, P. Merle,
Abstraction of transaction demarcation in component oriented platforms,
ACM / IFIP / USENIX Middleware'03, Bresil, Juin 2003.

[SAL98] Salutation Consortium,
White Paper : Salutation Architecture : Overview,
disponible à l'adresse http://www.salutation.org/whitepaper/originalwp.pdf, 1998.

[SDK01] A Seydim, M. Dunham, V. Kumar,
"Location Dependent Query Processing",
Proceedings du 2nd ACM international workshop on Data engineering for wireless and mobile access, Etats Unis, 2001.

[SMI84] B.C. Smith,
Reflection and Semantics in Lisp,
n Proceedings of the 14th Annual ACM Symposium on Principles of Programming Languages, pages 23-35, 1984.

[SUN99] S. Cheung,
Java Transaction Service (JTS),
Version 1.0, spécification Sun Microsystems Inc., USA, 1999.

[SVF05] N. Samara, A. Vervisch-picois, M. françois,
"La localisation en intérieur à l'aide de récepteur GPS : vers un système de

positionnement universel?",
Actes de la deuxième conférence UBIMOB'05, Mobilité et Ubiquité, Grenoble, France, 2005.

[TAQ01] D. Taquet,
Fédération de Systèmes d'Information dans le cadre du Home Area Network,
DEA en informatique, Université de Valenciennes et du Hainaut Cambrèsis, Juin 2002.

[TCD03] M. Thilliez, Y. Colmant, T. Delot,
Query Evaluation in ISLANDS,
Démonstration aux 19ièmes journées Bases de Données Avancées, Lyon, Octobre 2003.

[TD04] M. Thilliez, T. Delot,
Evaluation de requêtes dépendantes de la localisation dans les réseaux mobiles,
Premières Journées Francophones: Mobilité et Ubiquité 2004, ACM International Conference Proceeding Series, Nice, France, pp. 115-122, juin 2004, ISBN 1-58113-915-2.

[TD04b] M. Thilliez, T. Delot,
Evaluating Location Dependent Queries Using ISLANDS,
Advanced Distributed Systems: Third International School and Symposium, ISSADS 2004, Guadalajara, Mexico, Janvier 2004. ISBN 3-540-22172-7/0302-9743.

[TDL03] M. Thilliez, T. Delot,S. Lecomte,
Hybrid Peer-To-Peer Model in Proximity Applications,
IEEE Computer Society (Ed.), Actes du 17ième International Conference on Advanced Information Networking and Applications, Chine, pp. 306-311, janvier 2003, ISBN 0-7695-1906-7.

[TDL04] M. Thilliez, T. Delot,S. Lecomte,
A Metadata-Based Positioning Solution to Evaluate Location-Dependent Queries in Wireless Environments,
Actes des 20èmes journées de Bases de Données Avancées, Montpellier, octobre 2004.

[THI04] M. Thilliez,
Requêtes dépendantes de la localisation en environnement mobile : Expression, Evaluation et Optimisation ,
Thèse de doctorat en informatique, Université de Valenciennes et du Hainaut Cambrèsis, Décembre 2004.

[THI01] M. Thilliez,
Commerce electronique de proximité,
DEA en informatique, Université de Valenciennes et du Hainaut Cambrèsis, Juin 2001.

[TL02] T. Thai, H.Q. Lam,
.NET Framework Essentials, 2nd edition,
O'Reilly & Associates, ISBN 05-960-030-21, 2002.

[TSU01] S. Tsur,
Are Web Services the next revolution in E-Commerce?, Proceedings of the VLDB Conference, 2001.

[TWB03] D. Touzet, F. Weis, M. Banâtre,
PERSEND: enabling continuous queries in proximate environments,
Proceedings of the Workshop on Mobile and Ubiquitous Information Access, pages 31-38, Udine, Italy, 2003.

[UPN00] UPnP Forum,
Understanding UPnP : a White Paper,
téléchargeable à l'adresse $http://www.upnp.org/download/UPNP_UnderstandingUPNP.doc$,
Juin 2000.

[VGP97] J. Veizades, E. Guttman, C. E. Perkins, S. Kaplan,
Service Location Protocol, RFC 2165,
disponible à l'adresse http://www.ietf.org, 1997.

[WAL00] J. Waldo,
The Jini Specifications,
édité par Ken Arnold (2nd Edition) chez Addison-Wesley Professional, ASIN: 0201726173, 2000.

[WHF92] R. Want, A. Hopper, V. Falcao, J. Gibbons,
"The active badge location systems",
ACM Transactions on Informations Systems, Vol. 10(1), pp91-102, 1992.

[WSA97] R. Want, B. Schilit, N. Adams, R. Gold, K. Petersen, D. Goldberg, J. Ellis, M. Weiser,
The parctab ubiquitous computing experiment,
Mobile Computing, chap. 2, ISBN 0-7923-9697-9, 1997.

[YG03] B. Yang, H. Garcia-Molina,
"Designing a Super-Peer Network",
IEEE International Confonference on Data Engineering, 2003.

Résumé

Depuis la fin des années 1990, le développement des terminaux nomades et des réseaux sans fil s'est considérablement accéléré. Cela a provoqué l'apparition de nouvelles applications, très largement réparties, et offrant de nouveaux services, aussi bien aux usagers (applications de commerce électronique, télévision interactive, applications de proximité), qu'aux entreprises (développement du commerce B2B).
Avec l'apparition de ces nouvelles applications, les services techniques, qui prennent en charge un certain nombre de contraintes systèmes, ont également dû évoluer. Nos travaux ont principalement porté sur l'étude de mécanismes permettant de concevoir des services techniques adaptés à ces nouvelles applications. Cela a été réalisé en trois phases :

- Dans un premier temps, nous avons étudié la possibilité d'utiliser le modèle de programmation par composants, pour réaliser un service de gestion de transactions distribuées offrant la possibilité d'utiliser le modèle de transactions emboîtées ouvertes (ce modèle transactionnel étant très intéressant dans le cadre des applications de commerce B2B).
- Puis, nous avons généralisé le mécanisme de réalisation des services techniques en utilisant le modèle à composants, de manière à pouvoir facilement faire évoluer ces services aux travers de différentes "personnalités". Dans le cadre de ce travail, nous avons également proposé une architecture permettant de choisir la bonne personnalité du service technique, en fonction des besoins applicatifs, et des contraintes de l'environnement d'exécution.
- Enfin, nous avons proposé un service de requêtes dépendantes de la localisation de l'utilisateur, pour applications de proximité. Ce service, utilisant une localisation approchée d'un utilisateur, propose de retrouver des informations, dans un environnement dynamique formé de plusieurs terminaux nomades, capables de capacité de communication.

L'ensemble de ces travaux ont abouti aux soutenances de plusieurs thèses de doctorat, ainsi que de plusieurs DEA.