Université Joseph Fourier
- Grenoble -

Rapport scientifique présenté
pour l'obtention d'une
Habilitation à Diriger des Recherches

Spécialité : Informatique

Didier DONSEZ

Sujet du mémoire

Objets, composants et services : intégration de propriétés non fonctionnelles

Soutenue le 11 Décembre 2006

devant le jury

Président	M. Farid Ouabdesselam
Rapporteurs	M. Pierre Paradinas
	M. Philippe Pucheral
	M. Bertil Folliot
Examinateurs	M. Arnaud Fréville
	M. Pierre-Yves Cunin
	M. Jean-Bernard Stefani

Résumé

L'ajout de propriétés non fonctionnelles aux logiciels est nécessaire à leur déploiement sur les infrastructures opérationnelles des entreprises. Cet ajout complexifie la production du logiciel et aggrave la crise du logiciel. Mes recherches se sont intéressées depuis une dizaine d'années à l'ajout de propriétés non fonctionnelles importantes dans les différents paradigmes de développement qui se sont succédés, à savoir, les objets, les composants puis les services. Ces propriétés ont été tout d'abord pour les objets et les composants : la persistance, le contrôle de la concurrence et la fiabilité. La gestion du cycle de vie a été ensuite abordée pour les composants et les services. Mes recherches ont été menées à la croisée des domaines des bases de données, des systèmes d'exploitation, des intergiciels et du génie logiciel. Le résultat de mes travaux ouvre de nombreuses perspectives dans le domaine émergent des services « machine à machine ».

Remerciements

Je remercie sincèrement toutes les personnes qui ont contribué de près ou de loin à l'aboutissement de cette habilitation, en particulier :

Monsieur Farid Ouabdesselam, Professeur à l'Université Joseph Fourier de Grenoble, de me faire l'honneur de présider mon jury,

Messieurs Bertil Folliot, Professeur à l'Université Paris VI, Pierre Paradinas, Professeur au CNAM de Paris et Philippe Pucheral, Professeur à l'Université de Versailles – Saint Quentin, d'avoir accepté d'être les rapporteurs de mon habilitation,

Monsieur Arnaud Fréville, Professeur à l'Université de Valenciennes et du Hainaut-Cambrésis, d'avoir accepté d'examiner cette habilitation et surtout de m'avoir accordé sa confiance quand j'ai été nommé Maître de Conférences à Valenciennes, il y a déjà 10 ans,

Monsieur Jean-Bernard Stefani, Directeur du projet INRIA Sardes, d'avoir accepté d'examiner cette habilitation et m'accueillir cette année en délégation dans son équipe,

Je remercie également :

Messieurs Jacky Estublier et Pierre-Yves Cunin qui m'ont accueilli dans l'équipe Adele en 2001 et surtout pour m'avoir soutenu dans mes idées et mes projets,

Mes nombreux collègues académiques et industriels de Paris, Villetaneuse, Compiègne, Lille, Valenciennes,Grenoble et Valence, avec qui j'ai partagé idées, cours, formations et projets,

Les étudiants avec lesquels j'ai toujours le plaisir d'enseigner et d'apprendre,

Ma famille et mes amis d'ici, d'ailleurs et de très loin pour leur soutien constant, leur patience inaltérable, leur amitié (et plus si affinité) et leur amour,

Et une mention spéciale pour Paul, Eva et Patricia dont les weekends et les vacances ont largement été amputés par la rédaction de ce manuscrit.

So long, and thanks for all the fish.

Table des Matières

Table des Figures

Chapitre I

Introduction

> *"Software is invisible to most of the world. Although individuals, organizations, and nations rely on a multitude of software-intensive systems every day, most software lives in the interstitial spaces of society, hidden from view except insofar as it does something tangible or useful."*
> Grady Booch

La rédaction du rapport d'habilitation à diriger des recherches est traditionnellement l'occasion de prendre du recul par rapport au travail passé et en cours afin de faire une synthèse et une analyse sur tout ce que nous avons appris et réalisé dans les domaines de recherche visités et explorés. Ces quinze dernières années de recherche m'ont amené à aborder de nombreux domaines de recherche en suivant une ligne directrice à l'intersection des domaines de recherche sur les systèmes d'exploitation, sur les intergiciels, sur les systèmes de bases de données et sur le génie logiciel.

L'ajout de propriétés non fonctionnelles aux logiciels est nécessaire à leur déploiement sur les infrastructures opérationnelles des entreprises. Cet ajout complexifie la production du logiciel et aggrave la crise du logiciel. Mes recherches se sont focalisées sur l'étude de l'ajout de quelques propriétés non fonctionnelles dans les paradigmes de développement qui se sont succédés, à savoir, les objets, les composants puis les services. Ces propriétés ont été tout d'abord pour les objets et les composants : la persistance, le contrôle de la concurrence et la fiabilité. La gestion du cycle de vie a été ensuite abordée pour les composants et les services.

Ce chapitre retrace l'évolution générale du contexte de mes recherches sur ces quinze dernières années. Il poursuit par les problématiques qu'ont tentées de résoudre les communautés de chercheurs auxquelles je me suis rattaché. La suite de ce chapitre présente mon cheminement et la démarche que j'ai appliquée à mes travaux de recherche. Ce chapitre se termine la présentation de mes résultats et par le plan de route pour la suite de ce manuscrit d'habilitation à diriger des recherches en informatique.

I.1 Contexte général

Raccoon [Raccoon97] analysait l'évolution des paradigmes d'ingénierie du logiciel depuis une cinquantaine d'années et tentait de prévoir l'avenir (c'est à dire maintenant). Son analyse constatait un phénomène de vagues chacune divisée en 4 phases : innovation, croissance, maturité puis convention. Les vagues qu'il a identifiées sont : les instructions, les fonctions, les modules, les objets. Raccoon se risquait à prédire que la vague des années 2000-2010 serait celle des canevas (*frameworks*) et des patrons de conception (*pattern*)[1]. Raccoon constatait également l'influence des évolutions du matériel sur celles du logiciel (*« Hardware economics drive software economics »*) mais sans réellement relier l'impact des innovations du premier sur le second.

La suite de cette section présente les principales vagues de l'évolution du logiciel et du matériel (Figure I-1) qui sont apparues 10 ans après la rédaction de l'article de Raccoon. Cette section plante le décor de mes travaux de recherche depuis ma thèse de doctorat.

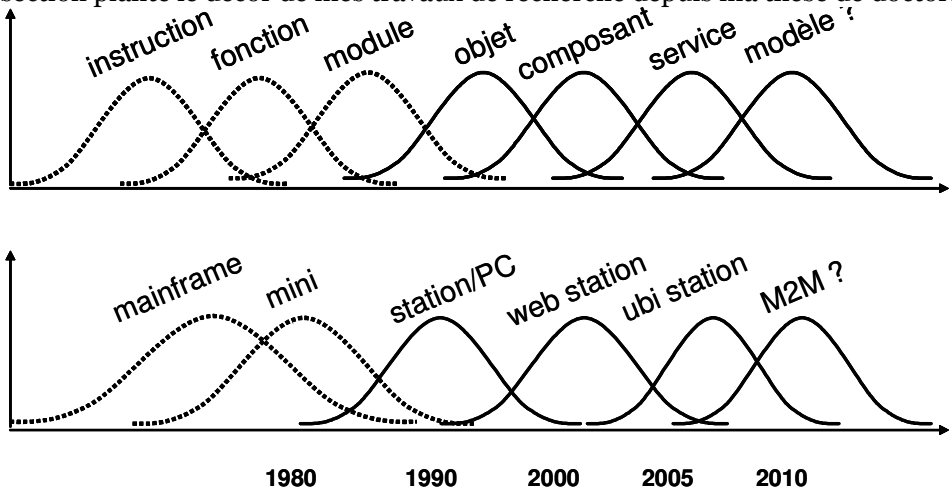

Figure I-1: Vagues du logiciel versus vagues de l'infrastructure

I.1.1 Les vagues de l'infrastructure

L'infrastructure matérielle et réseau des organisations a évolué selon plusieurs vagues. Les années 80 ont vu apparaître l'ordinateur personnel[2] et le poste dédié à un utilisateur. Le début des années 90 a été marqué par le règne sans partage de ce type de terminaux pour les personnels des organisations. Ce modèle d'architecture appelé client-serveur, confie une (grande) partie du traitement de l'application aux postes client reliés à des serveurs archivant les données (fichiers, bases relationnelles) par un réseau local (LAN).

Les années suivantes ont vues le développement du Web [Berners-Lee92] avec l'interconnexion des centres académiques et gouvernementaux. Les Etats Unis lançaient le programme « Information Highway » destiné à doper l'infrastructure réseau américain et mondial, et par la même occasion l'industrie informatique américaine [Colwell06]. Le réseau d'une organisation devient planétaire et les usagers de son système d'information

[1] Les patrons de conception (*design pattern*) [Gamma93,Gamma95] sont des structures de solutions récurrentes à des problèmes de conception dans des contextes particuliers. Les solutions de conception proposées par un patron particulier diffèrent presque toujours dans leurs détails tout en restant semblables dans leur principes.

[2] Ce fut l'IBM PC en 81, le Macintosh en 84 et les *stations de travail* Unix des 86.

sont alors directement ses clients (*extranet*) et non plus seulement ses agents (*intranet*)[3]. Les applications sont désormais architecturées selon le modèle multi-étage dans lesquels le terminal client se résume à un navigateur Web visualisant un document HTML calculée par un des serveurs de l'organisation. Ces serveurs se partagent les fonctions de présentation, de logique métier et de données. Ce modèle n'est en fait qu'un représentant d'une nouvelle discipline émergente, celle des intergiciels[4][Bernstein96, Emmerich00]. Pendant la même période se sont développés d'autres terminaux que l'usager emmène partout. Les assistants personnels (comme le Newton MessagePad d'Apple [Smith94], le PalmPilot d'US Robotics [McCandless97]), la téléphonie mobile [MoulyP94] et les cartes à puce [Guthery97, Rankl97] marquent le début de l'ère de l'informatique ubiquitaire. L'usager peut potentiellement accéder aux systèmes d'information des organisations au moyen d'une « ubi-station » constituée de nouveaux objets communicants [Weiser91]. Cependant la « bulle Internet » s'est un peu dégonflée, calmant la frénésie d'alors mais ne l'éteignant pas complètement.

Le début de ce millénaire a surtout vu le développement de l'Internet intra-entreprise (l'« EAI » pour *Enterprise Application Integration*), de l'Internet inter-entreprise (le « B2B » Business to Business) et des échanges pair à pair (P2P pour *peer-to-peer*). Le B2B a pris le relai de l'EDI (*Electronic Data Interchange*) [Langlois97] qui stagne depuis plusieurs années, dans l'automatisation des relations intra-organisations et des relations inter-organisations dans le commerce globalisé. La technologie prédominante se base sur le protocole HTTP devenu le canal de communication universel, et sur XML devenu la grammaire universelle de tous les langages d'échange. Les échanges pair à pair [Agre03, Androutsellis-Theotokis04] permettent désormais de se passer de l'infrastructure d'une organisation pour offrir un service souvent éphémère à une communauté souvent spontanée d'usagers. Ce modèle d'architecture part du principe que chaque nœud du réseau est symétriquement à la fois client et serveur de services. Le pair à pair s'est développé conjointement avec l'ADSL qui garantit désormais une connexion quasi-permanente à la toile avec une bande passante permettant l'échange de flux multimédia et isochrone. Les jeux massivement multi-joueurs[5] et la voix sur IP (VoIP) représentent désormais des applications importantes de l'Internet [Chaplin05]). En parallèle, l'infrastructure de l'organisation se complexifie avec l'apparition des grilles de calculs [Barroso03, Li05] et les serveurs en grappe de calcul (*cluster*) [Dongarra04, Wright05]. Cette infrastructure devient aussi de plus en plus virtuelle voir trans-organisationnelle avec l'externalisation des ressources de calcul et de stockage pour l'hébergement statique ou dynamique d'applications et de contenus [Nottingham01, Vakali03,Weihl04,Sahai04].

La nouvelle vague semble être celle de l'Internet des Choses (Internet of Things) [ITU05] pour lequel tous les objets enfouis de l'organisation [Thompson04] sont raccordés par l'Internet aux systèmes d'information des entreprises et des organisations. En effet, les fabricants d'équipements et de produits (électroménager, divertissement, appareillage industriel, véhicule, immeuble …) ont tendance à intégrer systématiquement la fonction de communication dans leurs produits. Les plus simples n'embarquent qu'un identifiant (étiquette RFID passive par exemple [Lahiri05]) alors que les plus complexes embarquent des capteurs [Hellerstein03, Zhao04] et parfois des actionneurs. Dans le premier cas, la sécurité, la traçabilité et la logistique en temps réel sont les trois principales applications. Dans le second cas, les deux principales applications sont la maintenance à distance (*e-maintenance*) qui réduit les coûts de SAV et de support, et

[3] Les services télématiques du Minitel avaient initiés ce mode de relation avec la clientèle mais seulement à l'échelle du pays via l'opérateur historique.
[4] En anglais *middleware*
[5] Le jeu Everquest regroupe plus de 300000 joueurs abonnés [Kushner05].

l'offre de services métier[6] à forte valeur ajoutée associés aux équipements, qui est vendue au client et permet de le fidéliser [Criéo2]. Cette nouvelle tendance appelée le M2M (*Machine-to-Machine*) [Lawton04] envisage des relations entre les nœuds de l'infrastructure indépendamment des usagers. Cette nouvelle vague sera plus particulièrement détaillée dans le chapitre 5 qui concerne mes perspectives de recherche.

I.1.2 Les vagues du logiciel

Les dernières années ont vu l'apparition successive de deux nouvelles vagues dans le développement de logiciel introduisant deux nouveaux paradigmes de programmation après celui de l'objet : les composants et les services. La prochaine vague semble être celle des modèles.

Ces paradigmes de programmation ont comme principal objectif la lutte contre la crise du logiciel [Dijkstra72]: le développeur doit à la fois faire face au passage à l'échelle et à la complexité des logiciels à mettre en œuvre. L'approche est à chaque fois de principalement réduire les nombres d'entités que doit manipuler le développeur d'application tout en augmentant le niveau fonctionnel et la qualité [Whittaker02]

La fin des années 80 a été marquée par l'émergence de l'approche objet pour le développement des applications. L'approche objet[7] [Meyer87,Snyder93,Abadi98] s'intéresse à modéliser finement les entités du monde réel de l'organisation. Les langages rigoureux [Black04] dans leur définition comme SmallTalk [Ingalls78] ou Eiffel [Meyer87] ont rapidement été supplantés par des langages « industriels »[8] comme C++ [Stroustrup85], Java [Gosling95] et C# [Ecma334] dont les concepteurs ont cherché le compromis pour une transition progressive des habitués de l'approche modulaire (c.a.d. le langage C) vers l'approche objet. L'approche objet prône l'isolation et la réutilisation du code par les principes d'encapsulation et d'héritage[9] pour passer à l'échelle sur des logiciels de plus en plus gros. L'approche objet reste néanmoins une approche pour la conception de programmes. Le déploiement de ces programmes dans l'environnement opérationnel requiert un surcoût[10] en lignes de code développées pour garantir des propriétés comme la distribution, la persistance, la cohérence, la gestion de la concurrence, la reprise sur panne, la gestion de la mémoire, l'équilibrage de charge ou au bien la sécurité. Ces propriétés que l'on qualifie de non-fonctionnelles par rapport à la fonction principale de l'application, restent alors entièrement à la charge du développeur.

Contrairement aux prévisions de Raccoon, la vague après l'objet a été l'approche à composant qui s'est développée au milieu des années 90 [Nierstrasz92,Szyperski98]. Cette approche différencie plusieurs acteurs dans la production du logiciel. Le développeur de composants suit généralement quelques patrons de conception (fabrique, inversion de contrôle, …) [Gamma95] et quelques règles et conventions de programmation pour produire des éléments de logiciel (les composants) réutilisables. Le développeur d'application réalise l'assemblage des composants et la configuration de propriétés fonctionnelles et non fonctionnelles (par exemple, transaction, degré d'isolation, …) éventuellement au moyen d'un langage

[6] Ces services s'apparentent au concept de helpdesk dans lequel l'entreprise prestataire mobilise des experts à la résolution d'un problème d'un client. Ce concept initialement appliqué à l'industrie informatique, s'applique à maintenant à d'autres domaines comme l'industrie automobile, …

[7] Inventé en 1965 à l'Université d'Oslo

[8] Nous les qualifierons ainsi car ils ont été soutenus par des industriels, respectivement AT&T, Sun, MicroSoft.

[9] Selon [Bézivin05] , l'héritage et l'instanciation sont les principes unificateurs de l'objet dans les langages de classes des années 80.

[10] Le surcoût d'implémentation de la seule propriété de persistance était estimé à 30 % du temps de développement.

d'architecture [Allen92,Medvidovic00]. Les propriétés non fonctionnelles sont configurées en grande partie au moment du déploiement du logiciel sur l'environnement d'exécution. La prise en charge des services non fonctionnels est réalisée par les conteneurs des composants de l'application. Le développement des conteneurs est confié à des spécialistes maîtrisant la complexité des services techniques et les techniques génératives. Ces spécialistes sont généralement les éditeurs de la plateforme d'exécution de l'application à composants qui forme alors la couche intergicielle reliant les composants de l'application. L'approche à composant a permis encore une fois de repousser la barrière de la complexité du logiciel. Cependant, elle introduit de nouveaux problèmes comme la cohérence de l'architecture globale du logiciel, la propagation des propriétés dans des compositions de composants, et l'extensibilité ou la personnalisation des services techniques requis par le logiciel.

Les dernières années ont vu l'émergence de l'approche à services [Bieber02,Sillitti02, Huhns05] popularisée par les Web Services. Cette approche propose de construire l'application à partir d'entités logicielles (éventuellement) opérées par des organisations tierces. Ces entités, appelés services, offrent des fonctionnalités décrites contractuellement. Le contrat, qui peut être syntaxique, sémantique, comportemental ou/et qualitatif [Beugnard99], permet de sélectionner (ou courter) les services pertinents pour l'application. Idéalement, le courtage est réalisé juste au moment de l'utilisation (juste à temps) et tout service respectant le contrat peut être substitué par un autre. L'application est alors une orchestration ou une chorégraphie de services échangeant messages de requêtes et de réponses. Les services sont considérés comme autonomes car l'usager de services n'a pas de prise sur ceux-ci en dehors de ce que définit le contrat signé et/ou de la qualité de service négociée par les contractants (*Service Level Agreement*) [Miller87]. De plus, les services utilisés n'ont pas nécessairement un cycle de vie (activité) synchronisé avec celui de l'application. La dynamique de l'approche à services suppose que de nouveaux services peuvent apparaître alors que l'orchestration/chorégraphie est entamée et que d'autres peuvent disparaître temporairement ou définitivement. La dynamique des services est prise en compte dans des plateformes d'exécution de services comme JINI [Arnold99], UPnP [Jeronimo03], DPWS [Jammes05] et OSGi [OSGi,Cervantes05] ou bien dans les *Enterprise Service Bus* (ESB) [Chappell04,Keen04,Schmidt05]. Ces derniers considèrent l'application comme un flot d'événements remis de manière asynchrone aux services intéressés. L'échelle de temps mesurant la dynamique peut varier de la minute pour un service embarqué dans une carte à puce (disponible que lorsque la carte est insérée dans un lecteur) à plusieurs mois pour une transaction commerciale de matières toxiques nécessitant l'obtention d'autorisations gouvernementales. Une fois de plus, la quête vers la réduction de la complexité réduit le nombre d'entités (i.e. des services) que manipule le développeur pour son application. Cependant, elle n'en réduit pas la complexité opérationnelle et rend difficile le contrôle de la cohérence globale.

2005 voit l'arrivée en force de la vague modèle [Selic03, Seidewitz03, Bézivin05, Estublier06] soutenue maintenant par de grands acteurs comme IBM et MicroSoft qui l'affichent ouvertement dans leurs stratégies. L'Ingénierie Dirigée par les Modèles (IDM)[11] propose de faciliter le développement des applications par l'utilisation de modèles productifs. Cette approche consiste, pour le développeur à définir un modèle[12] d'application et à le transformer vers un modèle exécutable sur une plateforme d'exécution par des techniques de transformation semi-automatiques. Avec cette

[11] En anglais *Model-Driven Engineering* (MDE)

[12] Les deux relations au cœur de l'IDM sont la *représentation* et la *conformité*. Le système étudié (M0) est représenté par un modèle (M1) conforme à un méta-modèle (M2) ; de plus, le méta-modèle (M2) est conforme à un méta-méta-modèle (M3) conforme avec lui-même.

approche, l'application peut être entièrement ou partiellement générée. La nouveauté de l'approche IDM par rapport aux méthodologies de développement des trente dernières années est qu'elle remet le modèle au centre de l'activité du développeur qui jusque là considérait le code comme primordial et le modèle comme « contemplatif » (spécification, documentation, communication, …). Cette approche peut être reliée à celle antérieure des langages spécifiques à un domaine[13] qui focalise et restreint le pouvoir d'expression du langage de développement à un domaine d'utilisation particulier (*problem domain*) [Wile99,Deursen00,Consel02].

I.2 Besoins

Au fil des années, le développeur s'est vu doté d'une *armada* de paradigmes de programmation et de paradigmes architecturaux pour faire face aux complexités combinées et grandissantes du logiciel et de l'infrastructure. Cependant même si les objectifs concernant la complexité de la partie fonctionnelle des applications ont été en grande partie atteints, la mise en œuvre des logiciels développés dans un contexte opérationnel reste une tâche complexe pour des développeurs métier. En effet, le logiciel déployé sur l'infrastructure requiert un certain nombre de propriétés dites non-fonctionnelles[14][Roman85]. Ces propriétés sont qualifiées de non-fonctionnelles car elles n'appartiennent pas à la préoccupation principale du développeur métier de l'application. La liste non exhaustive des propriétés non-fonctionnelles les plus couramment rencontrées dans les logiciels sont la distribution, la persistance, le contrôle de concurrence, la fiabilité, la distribution, l'authentification, le contrôle d'accès, la non-répudiation, l'audit, la facturation, la gestion des quotas, l'équilibrage de charge, la gestion de la mémoire, la gestion du cycle de vie, la migration, l'ininterruptibilité (*failover*), le temps critique, la qualité de service … Selon les catégories de logiciel, certaines propriétés seront plus importantes que d'autres. De plus, l'interprétation d'une propriété et ses technologies de mise en œuvre peuvent varier d'un nœud à l'autre de l'infrastructure. Par exemple dans une application bancaire, la persistance ne se traite pas de la même manière pour une carte à puce, pour un terminal de paiement commerçant ou pour un serveur de bases de données. L'échelle de temps de la persistance est de même différente entre un serveur de bases de données de production et un serveur d'entrepôts de données dans la grande distribution [Widom95,Inmon96,Kimball96,Zurfluh01] : classiquement 180 jours sur le premier [TpcC] contre plusieurs années pour le second [TpcH].

La mise en œuvre des propriétés non fonctionnelles dans le logiciel passe par l'utilisation de services non fonctionnels (aussi appelés services techniques). Ces services sont le plus souvent maîtrisables par des experts du domaine technique concerné (persistance, sécurité, QoS, dynamique …). Dans un premier temps, les communautés liées à ces services techniques ont cherché alors à rendre ces propriétés transparentes aux langages de développement, d'abord dans les langages modulaires puis dans les langages objets en se basant sur le sacro-saint principe d'orthogonalité entre les objets et la propriété technique. L'infusion des propriétés non fonctionnelles a été réalisée séparément les unes des autres. L'usage d'un langage étendu pour une propriété laisse néanmoins le développeur se débrouiller avec les services non infusés. La programmation orientée aspect [Kiczales97] a permis dans un premier temps d'infuser l'usage des services techniques sous forme d'aspect dans le code principal de l'application. Les conteneurs des composants ont encapsulé le code des services techniques en les rendant néanmoins

[13] En anglais *Domain Specific Language* (DSL). La littérature les appelle aussi micro-langages.
[14] Ces propriétés sont également recherchées pour le logiciel d'infrastructure qui administre l'infrastructure et les logiciels qui sont déployés dessus.

paramétrables par des développeurs profanes. Les plates-formes à composants industriels ont cependant figé les listes de services disponibles et leurs sémantiques. Du coté des plateformes à services, les propriétés non-fonctionnelles attachées aux services fonctionnels sont mises en œuvre par des services non-fonctionnels utilisant les mêmes mécanismes de découverte et de communication. Pour l'instant, l'ajout de ces propriétés par le développeur l'oblige à manipuler du code non-fonctionnel. Une tendance qui est issue de l'AOP est de ne pas distinguer le code fonctionnel du code non-fonctionnel. Cette approche unificatrice permet de considérer l'application selon les points de vue des différents acteurs de cycle de vie du logiciel. Par exemple, selon le point de vue d'un développeur de routeur téléphonique, l'automate d'établissement des connexions est sa préoccupation principale [Jacobson03][15]. Selon le point de vue du développeur de l'application de facturation, sa préoccupation principale est la comptabilité des unités de temps une fois la connexion établie. Ce deuxième point de vue sera considéré comme un service secondaire, technique ou non-fonctionnel par le premier développeur. De plus, ces deux services requièrent à leurs tours d'autres services comme la sécurité ou la persistance.

I.3 Domaines de Recherche et Contributions

Mes recherches ont été menées à l'intersection des domaines de recherche des Systèmes de Gestion de Bases de Données, des Systèmes d'exploitation, des Intergiciels et du Génie logiciel.

Mon souci au fil de ma recherche a été de comprendre comment utiliser et comment intégrer les propriétés non fonctionnelles dans le paradigme logiciel émergent du moment, à savoir les objets, les composants et enfin les services.

Les propriétés non fonctionnelles auxquelles je me suis intéressé sont :

- la persistance qui considère que les données créées ou modifiées par une activité demeurent après la terminaison de celle-ci,
- le contrôle de concurrence qui considère que plusieurs activités simultanées et concurrentes peuvent conduire à des incohérences dans le système d'information,
- la fiabilité qui considère que les pannes de l'infrastructure ou les abandons d'activité en cours d'exécution ne doivent pas nuire à la pérennité des résultats, ni à la cohérence du système,
- la gestion du cycle de vie qui considère que l'application doit réagir aux changements qui interviennent dans la vie des activités qui la composent.

Les trois premières propriétés étaient à la fondation de la communauté de recherche en systèmes de bases de données et dans une certaine mesure de la communauté des chercheurs en systèmes d'exploitation [Gray81]. Elles ont peu à peu infusé dans d'autres communautés et essaimé pour créer de nouveaux domaines comme celui des systèmes sûrs [Avizienis01]. La dernière propriété est apparue plus tardivement avec l'émergence des systèmes répartis à grande échelle. Les activités qu'elle recouvre, le déploiement et la reconfiguration des applications, sont devenues désormais des préoccupations majeures dans les domaines des intergiciels et du génie logiciel. Ces activités peuvent bénéficier de l'inséparable trio : persistance, contrôle de concurrence et fiabilité. Les communautés de

[15] Jacobson associe les aspects à des scénarios (*use cases* SDL ou UML) .

chercheurs ont proposé et continuent de proposer des solutions[16] pour garantir ces propriétés dans les logiciels en tout point de l'infrastructure de l'organisation (c.a.d. de la carte à puce à la grappe de serveurs).

L'étude des ces propriétés s'est fait en trois périodes qui correspondent aux trois dernières « vagues logicielles ». Je résume dans les sections suivantes mes contributions principales pour chacune de ces périodes. Les chapitres suivants du manuscrit détaillent ces périodes.

La période « objet »

Mes recherches de la période des objets furent guidées par le souci de rendre les objets persistants pour des applications s'exécutant sur une architecture distribuée. Cette période était marquée par un bouillonnement de propositions dans la communauté bases de données, pour la définition de mémoires d'objets persistants et de modèles de transactions avancées. Ma contribution fut la définition d'un canevas flexible pour construire des gérants d'objets persistants, distribués sur des réseaux de machines. Plusieurs modèles avancés de transaction pouvaient être mis en œuvre pour fiabiliser et pour synchroniser les traitements concurrents sur les objets répartis. Dans un deuxième temps, je m'étais focalisé sur la persistance et la fiabilisation des informations au travers de gérants d'objets persistants pour les cartes à puce. Ma vision était également de reconsidérer ces petits objets comme des ressources (voir des acteurs) à part entière dans les transactions opérant dans un système d'information grandement réparti.

La période « composant »

Pour le paradigme composant, je me suis intéressé à regarder comment spécialiser des modèles de composant existants à des domaines spécifiques. Cette recherche était dans la mouvance de l'ingénierie des conteneurs adaptables et des plateformes de composants. Mes contributions furent dans un premier temps des propositions pour simplifier l'usage des modèles avancées de transactions dans la programmation à composants EJB. Je me suis ensuite tourné vers des modèles de composants dédiés à des domaines d'application très spécifiques comme la télévision interactive et les applications basées capteurs. La télévision interactive fut alors l'occasion de m'intéresser à la gestion du déploiement de continu. Mes propositions furent le déploiement en continu des composants en fonction de leurs besoins, et la fiabilisation de l'activité de déploiement au moyen de modèles avancés de transactions.

La période « service »

Cette troisième période, qui se recouvre en partie avec la précédente, s'est intéressée principalement au cycle de vie des services. Ma préoccupation principale fut de regarder comme intégrer les changements de cycle de vie des services constituant les applications dans un contexte dynamique et sans fin d'une infrastructure parfois spontanée, voire éphémère et de plus en plus trans-organisationnelle. Mes travaux se sont placés dans le contexte de la recherche sur la reconfiguration à l'exécution et sur l'informatique autonomique. Mes contributions sont les définitions de plusieurs plateformes dynamiques pour des environnements d'exécution JEE et .NET. Une autre contribution fut également de regarder comment combiner l'approche à service et l'approche à composant dans le contexte des applications dynamiquement déployables et reconfigurables.

Perspectives

[16] Un des concepts les plus notoires est celui de la transaction. La transaction est une suite d'actions dont l'exécution est atomique, cohérente, isolée des autres transactions et dont les effets sont durables (propriétés ACID pour *Atomicity, Consistency, Isolation, Durability*).

Jusqu'à présent, je me suis intéressé à l'étude de ces propriétés à des points isolés de l'infrastructure matérielle des organisations et des entreprises : des cartes à puce aux serveurs d'entreprise, en passant par les capteurs, les terminaux de télévision interactives, les *gizmos* personnels communicants et les passerelles de services enfouies.

La mise en œuvre des futurs services qui traiteront de l' « Internet des Choses » nécessite de reconsidérer ces propriétés sur l'ensemble du spectre de l'infrastructure, à savoir de la nuée des simples capteurs à la ferme de serveurs de l'entreprise. Les intergiciels exécutant ces services devront accueillir et faire collaborer efficacement des applications développées selon les différents paradigmes logiciels métiers aux différents points de l'infrastructure. Ces intergiciels devront être complétés par des ateliers d'ingénierie des services M2M masquant la complexité de l'infrastructure aux usagers qui de plus en plus souhaiteront concevoir et développer sans délai leurs propres applications sans l'aide d'intermédiaires techniques. L'enjeu est majeur pour les entreprises car il concerne leurs capacités à réagir (*e-agility*) pour satisfaire le besoin de ses clients plus rapidement que la concurrence. Une approche préconisée et expérimentée par l'équipe est l'ingénierie dirigée par les modèles. Un des objectifs principaux est de limiter l'impact des évolutions du besoin client sur le temps de développement des environnements et des outils. Les travaux menés ces dernières années m'engagent dans la définition de cette prochaine génération d'intergiciels et d'ateliers de développement.

I.4 Démarche

La lecture du livre « Architecture des ordinateurs : une approche quantitative » [Hennessy90] de John Hennessy et David Patterson fut pour moi un guide de conduite pour mes recherches. Ce livre détaille l'approche suivie par les auteurs pour la définition de concepts RISC (*Reduced Instruction Set Computer*). Leur approche était d'analyser finement le besoin des applications et de tenir compte de l'évolution prévisible des technologies matérielles pour redéfinir les principes fondateurs des architectures des dernières générations de processeurs (Mips, Alpha, Sparc, Power, Pentium, Arm, …). J'ai essayé d'appliquer leurs démarches à mes propres recherches.

La recherche de problèmes réels rencontrés par les praticiens d'un domaine (développeurs, architectes) est un pré-requis à une recherche applicable et utilisable qui solutionne des problèmes non triviaux. Les partenariats avec les entreprises ont été ainsi souvent le point de départ permettant d'identifier des besoins réels et quotidiens de domaines applicatifs particuliers nécessitant des solutions adaptables à d'autres domaines ou généralisables à d'autres échelles du matériel. La participation et mon implication dans plusieurs projets partenariaux entre les entreprises et mes laboratoires d'accueil ont été l'occasion de développer des réalisations vérifiant les hypothèses et validant les travaux.

Pour conclure cette section, ma démarche s'est toujours inscrite dans un contexte de projet d'équipe plutôt que dans une aventure individuelle. L'image d'Epinal du chercheur enfermé dans sa tour d'ivoire ne m'a jamais séduite. Je crois plus que les progrès viennent des points de vue, des échanges et des collaborations avec les chercheurs provenant d'horizons parfois très différents. Cette dynamique d'équipe s'est concrétisée dès mon démarrage en doctorat, par mon implication dans des projets partenariaux avec des entreprises.

I.5 Lieux de résidence, périodes, compagnons d'exploration et données quantifiables

Le travail de recherche présenté dans ce mémoire s'est déroulé dans plusieurs laboratoires.

La période 1989-1994 consacrée à mon DEA et à ma thèse de doctorat s'est déroulée dans l'équipe RAPID dirigée par Pascal Faudemay, du laboratoire MASI de l'Université Paris VI. L'équipe RAPID s'intéressait à la conception conjointe de matériels et de logiciels pour l'accélération des recherches dans de grands ensembles de données. Mes compères de recherche étaient alors Philippe Homond, Eric Abécassis et Thierry Cruanes comme moi doctorants, et nous avons travaillé en projet à la réalisation d'un gestionnaire transactionnel d'objets persistants. J'ai soutenu ma thèse de Doctorat en Informatique le 30 septembre 1994 sur "WEA, un Gérant d'Objets Persistants pour des Environnements Distribués" à l'Université Paris VI au laboratoire MASI (URA CNRS) sous la direction de Georges Gardarin et Pascal Faudemay.

La courte mais enrichissante période 1994-1995 s'est déroulé à l'Université de Technologie de Compiègne en tant qu'ATER au laboratoire HEUDIASYC (URA CNRS 817). J'y ai découvert le goût du montage de projets exploratoires qui ne m'a jamais quitté depuis. Mes compères étaient (alors) de jeunes seniors Liming Chen et Marc Bui, et Félix Ramos alors doctorant.

L'année 1996 m'a fait voir le monde réel de l'informatique, celle des utilisateurs finaux, en tant que chef de projet informatique au CHRU de Lille. C'est à cet endroit que j'ai fait la connaissance de plusieurs chercheurs débutants sur une thématique montante : la carte à puce. Ceux-ci sont devenus mes compères de recherche par la suite : David Carlier, Sylvain Lecomte, Gilles Grimaud et Sébastien Jean de l'équipe RD2P du LIFL.

L'année 1996 a été également l'année de mon recrutement au poste de Maître de Conférences à l'Université de Valenciennes dans la branche informatique[17] du laboratoire LIMAV dirigé par Arnaud Fréville. Arnaud (que je remercie encore) m'a donné carte blanche pour faire démarrer et consolider un thème[18] nouveau sur les systèmes d'information et sur les objets communicants, en collaboration avec l'équipe RD2P du LIFL. Plusieurs maîtres de conférences m'ont rejoint sur ce thème, Smaïl Niar et Nadia Bennani, puis Sylvain Lecomte a été recruté sur ce thème et plusieurs doctorants ont choisi notre thème de recherche, Sergiy Nemchenko, Marie Thilliez et Colombe Hérault.

Depuis 2001, suite à ma mutation à l'Université Joseph Fourier, mes travaux se déroulent au sein de l'équipe ADELE dirigée par Jacky Estublier, du laboratoire LSR (UMR CNRS 5526) dans la fédération IMAG. Mon intégration dans cette équipe de Génie Logiciel m'a permis de regarder mes travaux sous un autre angle de vue et de progresser vers la recherche de l'*utilisabilité* de mes propositions. Les collaborations sont nombreuses et fortes avec tous les membres de l'équipe.

Pendant ces années, j'ai grandement participé et contribué à l'encadrement des thèses de doctorat de Sylvain Lecomte et Sébastien Jean soutenues au LIFL à l'Université de Lille 1 sous la direction de Vincent Cordonnier. J'ai démarré également les trois thèses financées de Sergiy Nemchencko, Marie Thilliez et Colombe Hérault avant de muter sur l'Université Grenoble 1. Deux thèses sont en cours : celle de Mikaël Désertot sur les

[17] Branche qui a rejoint en 1999 le LAMIH, UMR CNRS 8530
[18] Thème SID (Systèmes d'Information Distribués) de l'équipe ROI (Recherche Opérationnelle et Informatique) du LAMIH UMR CNRS 8530

serveurs d'entreprise dynamiques, celle de Cristina Marin co-encadrée avec Philippe Lalanda sur les services orientés capteurs. La thèse de Lionel Touseau débute sur les accords de niveau de services dans les plateformes dynamiques de services. Dix étudiants en DEA encadrés ou co-encadrés et une dizaine d'étudiants de DESS et ingénieurs ont participés pendant leurs stages directement aux différentes réalisations. Plusieurs dizaines de projets de maîtrise et de DESS/Master 2 Pro m'ont permis de défricher les domaines, d'expérimenter les technologies émergentes, d'outiller mes réalisations de recherche et de les utiliser par la suite dans mes enseignements. Plusieurs prototypes ont été réalisés pour la plupart dans le cadre de projets contractuels des équipes auxquelles j'appartenais ; les autres ont été réalisées pour « la cathédrale et le bazar » [Raymond98]. Ceux-ci seront listés au fur et à mesure dans les chapitres suivants. J'ai initié, participé et/ou piloté pour mes équipes d'accueil, plusieurs projets nationaux (Autoroutes de l'Information, RNTL IMPACT, CPER COLORS, RNTS COQUAS, RNRT COMPiTV, RNRT PISE) et européens (ITEA PEPiTA, ITEA OSMOSE, ITEA S4ALL, ITEA ANSO) ainsi que des contrats industriels. Ces projets m'ont permis de multiplier les relations industrielles qui ont été un terrain pour valider mes idées. Ces projets sont également listés au fur et à mesure dans les chapitres suivants. Mes travaux ont fait l'objet d'une trentaine de publications en conférences internationales et nationales en tant qu'auteur. Ces publications et les mémoires de thèse et de master sont référencés au fur et à mesure du texte.

I.6 Plan de route

La suite du manuscrit est structurée autour des trois paradigmes logiciels qui m'ont servi de support à l'étude des propriétés non fonctionnelles : la persistance, le contrôle de concurrence et la fiabilité et le cycle de vie dynamique sur lesquels j'ai travaillé. Le Chapitre II s'intéresse aux propriétés de persistance, de contrôle de concurrence et de fiabilité des objets. Le Chapitre III s'intéresse principalement à la spécialisation des modèles de composants dédiés à des contextes spécifiques. Le Chapitre IV est consacrée aux services et à leur dynamique.

Chacun de ses chapitres est structuré selon les 3 sections décrivant successivement

- la problématique et l'état de l'art,
- mes travaux et mes contributions
- un bilan des résultats.

Dans chacun de ces thèmes, je ne prétends pas donner une couverture complète des sujets abordés. Je choisis d'entrer en profondeur dans certains sous thèmes. Chaque conclusion de chapitre mentionne les projets qui se sont rattachés aux travaux présentés.

Mes perceptives de recherche sont développées au Chapitre V.

Chapitre II

Objet, Persistance, Contrôle de Concurrence et Fiabilité

Historiquement, le problème de la persistance et de la fiabilité des données a amené à la fondation de la communauté de recherche sur les systèmes de bases de données. La motivation principale de cette communauté consiste à concevoir des systèmes capables de rendre pérennes les données bien après l'exécution des activités qui les ont créées ou modifiées et de permettre le partage cohérent de ces données entre plusieurs activités. Le contrôle de concurrence fut ensuite adressé à la fois par les communautés systèmes et bases de données pour garder cohérentes les données accédées par les activités concurrentes introduites par les premiers systèmes à temps partagé [Gray06]. L'objectif de conception de tels systèmes est la tolérance aux défaillances du matériel ou des applications tout en restant performant.

Ma thèse de doctorat s'est placée dans ce contexte pour la réalisation de plateformes flexibles et distribuées d'objets persistants. Les travaux qui ont suivi m'ont amené à revoir les propriétés de persistance et de fiabilité pour un type de machine très fortement contraint (en particulier la carte à puce) et dans les environnements nomades.

La section 1 rappelle les problématiques abordées par la communauté des systèmes bases de données et les recherches menées au début des années 90 sur les systèmes à objets persistants et sur les modèles avancées de transaction. La section 2 présentera mes contributions. La section 3 dresse un bilan sur le domaine et sur mes résultats.

II.1 Problématiques et Etat de l'art

Historiquement, les propriétés de persistance, de cohérence et de fiabilité ont été au cœur de la problématique de la communauté bases de données. Le modèle de bases de données hiérarchiques puis le modèle de bases de données réseaux (dont le représentant est le CODASYL [Flynn78]) représentent les données persistantes sous la forme de graphes orientés dans lesquels le programme procédural navigue en suivant les références à partir d'une ou de plusieurs racines.

Leur remplaçant[19], le modèle relationnel et son langage support, le SQL, normalisé presque vingt ans après l'article fondateur [Codd70], représentent les données sous la forme d'ensembles que le développeur manipule au moyen d'opérateurs. Le programme appelé requête est une composition d'opérateurs. L'idée sous-jacente est de laisser l'optimiseur [Jarke84] du système de gestion de bases de données (SGBD) relationnelles inférer le plan d'exécution des opérations d'accès aux données le plus adéquat pour la requête soumise. Cependant le développement d'applications requérant des langages généralistes, oblige leurs développeurs à nicher des requêtes SQL dans les langages procéduraux tels que PL/1 et C, posant alors des problèmes de typage (*impedance mismatch*) et nuisant aux performances (entre autre à cause de la double *bufferisation* des données et des conversions de format).

A la même période, alors que de nombreux traitements étaient réalisés en lot (*batch*), les applications sont devenues de plus en plus interactives. Le traitement interactif et en temps partagé des travaux des usagers a poussé les systèmes à synchroniser ceux-ci. Gray proposa alors un modèle de transactions [Gray80, Gray81] dont les propriétés ACID (Atomicité, Cohérence, Isolation et Durabilité) allaient marquer le point de départ pour un grand nombre de recherches dans le domaine des bases de données et des systèmes distribués.

La montée en puissance des langages objets dans l'industrie du logiciel, les besoins en persistance de nouveaux types d'applications et les nouveaux types d'architecture allaient guider les recherches dans les bases de données[20] et dans les systèmes transactionnels.

II.1.1 Nouvelles applications

Jusqu'alors la problématique des bases de données était principalement focalisée sur les applications orientées vers la gestion et l'administration des organisations et les relations entre clients et fournisseurs [Anon85, TpcC]. De nouveaux domaines applicatifs vont successivement intéresser la communauté bases de données.

De nouvelles applications souvent liées à l'ingénierie ont fait leur apparition dans des domaines comme la conception et la fabrication assistées par ordinateur (CAO/CFAO en mécanique[21], en micro-électronique, dans le BTP, dans le médical, en logiciel[22]…), la gestion électronique documentaire (GED) [Goldfarb90], les systèmes d'information géographique (SIG) [Stonebraker93] …

Ces applications se distinguent des applications plus traditionnelles de gestion sur plusieurs points :

- Un nombre d'usagers simultanés relativement faible : par exemple d'une dizaine à un millier en comparaison avec plusieurs centaines de milliers à l'heure pour les applications bancaires [Anon85, Gray85, Gray05] ou de réservation en ligne (Système de réservation aérien SABRE[23])
- Une durée des opérations assez longue : de plusieurs heures à plusieurs jours contre la minute pour les applications bancaires de type débit/crédit.

[19] Le marché des SGBDs relationnels n'a définitivement dépassé celui des SGBDs CODASYL qu'au milieu des années 90. Cependant, il existe toujours de nos jours des bases CODASYL patrimoniales qui sont en production.
[20] [Bernstein89, Silberschatz90, Silberschatz96, Bernstein98, Abiteboul05] sont les rapports des séminaires organisés par les « séniors » du domaine des bases de données autour des défis à venir pour ce domaine.
[21] Un des exemples les plus importants dans ce domaine est le logiciel CATIA de Dassault Systems.
[22] Ce que certains appellent du *metaware*.
[23] http://www.sabreairlinesolutions.com

- Une manipulation des données par le programme s'exécutant sur le poste de travail de l'usager (ingénieur, …) plutôt que sur le serveur.
- Un ensemble des données de travail important : plusieurs millions d'entités sont manipulées pour le routage d'un cœur de microprocesseur dans un atelier de CAO/VLSI contre 4 lignes dans la base de données du banc d'essai débit/crédit [Anon85].
- Une charge de travail caractérisée par des navigations en profondeur dans le graphe de données de travail [Carey93].
- Un degré important de contention entre des espaces de travail des différents usagers, lié à la nature collaborative des applications de conception et à la longue durée des requêtes.
- Une modélisation de données basée sur le paradigme d'héritage qui pousse les développeurs à préférer des méthodologies objets (OMT, Booch, …) et leurs langages support (ADA, SmallTalk, Objective-C mais surtout C++) pour garantir réutilisabilité et extensibilité.
- Des données parfois de grande taille telles que les objets multimédia dans la GED et les SIG.

Les besoins en persistance, en synchronisation et en fiabilité de ces nouvelles applications restaient insatisfaits par les systèmes de fichiers natifs des systèmes d'exploitation et par les systèmes de gestion de bases de données relationnelles. D'une part, les premiers obligeaient à pré-charger l'intégralité des données en mémoire primaire et n'offraient pas ou peu de mécanismes de synchronisation. D'autre part, les seconds qui avaient largement été optimisés pour les transactions de courte durée (*OLTP : OnLine Transaction Processing*) et pour les opérations ensemblistes, restaient inefficaces vis-à-vis de ce type de besoins.

II.1.2 Nouveaux postes de travail

Alors que la station de travail personnelle (PC) domine le paysage informatique pour l'accès aux systèmes d'information, de nouveaux types de terminaux vont apparaître. Les assistants personnels (le *Newton MessagePad* d'Apple [Smith94], le *PalmPilot* d'US Robotics [McCandless97]), les cartes à puce [Guthery97, Rankl97] et la téléphonie mobile[24][Mouly94] marquent le début de l'ère ubiquitaire pronostiquée par Mark Weiser [Weiser91]. L'usager peut potentiellement accéder aux systèmes d'information des organisations au moyen d'une « ubi-station » constituée d'un ensemble de nouveaux objets communicants comme les assistants personnels, les téléphones mobiles 2G et 3G, les appareils photo, les terminaux de TV interactive, les consoles de jeux, les appareils photo numérique, les web-stations de cybercafé, … Ces terminaux embarquent de plus en plus un fragment voire une poussière (pas toujours répliquée) de l'information attachée à leur porteur. Ces terminaux sont de plus en plus considérés comme à la fois des clients et des serveurs selon les principes du modèle pair-à-pair en se passant d'infrastructure fixe et fiabilisée (comme des serveurs d'entreprise).

Ces « postes de travail » mobiles (ou nomades) ont néanmoins un certain nombre de contraintes.

[24] Un téléphone cellulaire d'aujourd'hui a une capacité de stockage au moyen dix fois supérieure à un PC de bureau de 1990 pour un prix 5 fois moins élevé conformément à la loi de Moore [Moore65].

- L'autonomie énergétique guide les concepteurs du matériel à chercher des composants ayant le meilleur ratio performance CPU sur énergie consommée. Elle oblige aussi les développeurs de logiciel à concevoir ceux-ci dans le même esprit.
- La capacité en mémoire primaire est généralement réduite[25] par rapport à celle d'une station fixe
- Les communications avec d'autres systèmes se font de manière épisodique, sporadique, opportuniste et/ou éphémère. Celles-ci ne sont pas toujours très fiables et instantanées. Leurs coûts financier et énergétique sont non négligeables.
- Le terminal et ses composants (carte à puce, carte mémoire) sont à la merci de leur porteur et de son entourage. Pertes, vols, destructions sont des événements hautement probables par rapport au cas d'un serveur d'entreprise[26].

Cependant, ces « postes de travail » possèdent quelques avantages : les technologies de mémoire secondaires ont des propriétés comparables à celles des mémoires primaires comme l'accès direct qui autorise la lecture et l'exécution direct (*in place*).

Ces nouveaux postes de travail et leurs contraintes vont alors donner le cadre d'une partie de la recherche en systèmes et en bases de données. L'enjeu est alors d'embarquer (et même enfouir) une partie des systèmes d'information des organisations et de leurs applications dans ces objets.

II.1.3 Gérants d'objets persistants

Le problème[27] de la persistance des objets va créer un « schisme » dans la communauté bases de données entre deux écoles de pensée :

- D'une part, les « révolutionnaires » souhaitaient rendre persistants les objets des langages généralistes tels que SmallTalk et C++ et leur ajouter les fonctionnalités des SGBDs (concurrence, sécurité, manipulation ensembliste, …),
- D'autre part ; « les évolutionnistes » préféraient conserver l'existant (c.a.d. le modèle relationnel) et l'augmenter de quelques concepts empruntés à l'objet (principalement encapsulation, héritage). L'objectif était la conservation des bases relationnelles patrimoniales largement outillées et la migration progressive des applications vers l'objet.

Les révolutionnaires

Les révolutionnaires [Atkinson89] publient alors un manifeste qui liste des concepts obligatoires (appelés *golden rules* dans le manifeste) et facultatifs à supporter par les SGBDs orientés objets. Les concepts obligatoires sont : les objets complexes et composites, l'identité d'objet, l'encapsulation, l'héritage, la surcharge, l'extensibilité des types, la complétude[28] des langages de développement, la persistance orthogonale, la gestion de la mémoire secondaire, le contrôle de concurrence, la reprise sur panne et le requétage ensembliste. Les fonctionnalités souhaitables mais non obligatoires sont

[25] Seulement 512 octets de RAM statique pour une carte à puce JavaCard de 1996.
[26] L'incendie du siège du Crédit Lyonnais le 5 Mai 1996 a rappelé aux DSI que cette probabilité n'est jamais nulle.
[27] Carey et Dewitt [Carey96] dresse un bilan de la période 1986-1996 et nous donne rendez-vous en 2006.
[28] Toute l'application, de l'accès aux données à l'interface graphique, doit pouvoir être exprimée dans le ou les langages supportés par le système ce qui n'est pas le cas de SQL(:89 et même :92).

l'héritage multiple, la distribution, les transactions longues et le support des versions d'objets.

Ce groupe s'est structuré alors en consortium, l'Object Database Management Group [Cattell93]. L'ODMG avait pour but d'uniformiser les API langages de SGBDs orientés-objet. Les langages cibles ont été les langages procéduraux SmallTalk et C++ mais également un langage de requête ensembliste OQL[29] directement issu des travaux autour d'O2. L'API pour le langage Java fut ajoutée tardivement [Cattell00].

Une multitude de travaux académiques ont été réalisés dans cette mouvance et quelques uns ont essaimé vers des produits commerciaux. Une liste non exhaustive de ces systèmes est la suivante : Exodus [Carey86], GemStone [Bretl89], O2 [Deux91], Ode [Agrawal89], Jasmine [Ishikawa96], Iris [Fishman87], Eos [Biliris94a], ORION [Kim89], ObjectStore [Lamb91], QuickStore [White94], Texas [Singhal92], Orion/Ithasca [Kim90], Gemstone [Butterworth91], Versant [Wietrzyk98], PJama [Atkinson96], Encore [Hornick87], SHORE (Scalable Heterogenous Object REpository) [Carey94a], PerDiS [Ferreira99]. On peut noter que des systèmes tels que Geode [Pucheral88], Genesis [Batory88], DBGraph [Pucheral90] qui étaient plutôt orientés vers la construction modulaire de gérants d'objets persistants, ont été le point de départ de beaucoup de ces systèmes. C'était aussi le cas de WEA/YOODA développé par l'équipe RAPID détaillé par la section II.1.3.

Compte tenu des contraintes de taille de ce document, la suite résumera quelques unes des approches utilisées par les concepteurs de ces systèmes qui étaient toutes guidées par l'accès à haute performance aux objets persistants pour des bases dépassant largement l'espace de mémoire primaire des stations et des serveurs.

Lorsque ces systèmes sont utilisés selon l'architecture client-serveur, un des principes d'architecture est la migration de données vers les stations de travail (*Data Shipping*) pour y effectuer les traitements plutôt que l'envoi de ces traitements vers le ou les serveurs (*Query Shipping*) comme c'est encore le cas dans les serveurs SQL contemporains. Ce choix était guidé par la nature même de la charge de travail des applications cibles qui se caractérise par l'usage localisé d'un même ensemble de données de travail (*working set*) sur une durée relativement longue.

Le *data shipping* a conduit à la définition de nouveaux algorithmes de cache et de verrouillage des données [Franklin97]. A titre d'exemple, l'algorithme de *Callback Locking* [Howard88,Nelson88,Franklin92] autorise le gestionnaire de concurrence local au client à ne pas relâcher pas un verrou à la fin de transaction. Ce verrou est conservé par le gestionnaire local tant que le gestionnaire central ne demande pas l'invalidation du verrou pour l'accorder à un autre gestionnaire local.

Le *data shipping* a fait l'objet de plusieurs alternatives liées aux granularités choisies pour le transfert, pour le cache client et pour le verrouillage [DeWitt90]. Le grain peut être l'objet ou la page ou un groupe de pages pour les objets multi-pages. La granularité de l'objet permet un meilleur usage de la mémoire de la station de travail et une réduction de la contention entre les transactions. Cependant le grain de l'objet est inadapté pour les échanges réseau et ne permet pas de profiter des mécanismes relatifs aux MMU[30] des machines clientes tels que le *memory mapping* ou les *external mappers* nouvellement introduits dans les systèmes d'exploitation Mach, Chorus, Solaris. A l'opposé, le grain de la page est efficace pour le transfert et autorise l'usage de la MMU pour réaliser la déréférenciation et la pose implicite de verrous. Son inconvénient majeur est au niveau du verrouillage et sur l'usage de la mémoire primaire de la station. Le regroupement d'objets

[29] Qui a un fort parfum de SQL quand même !
[30] Memory Management Unit : unité matérielle chargée de la traduction des adresses virtuelles des processus vers les adresses en mémoire physique de la machine.

devient alors une nécessité pour limiter le chargement en mémoire des objets non accédés par le programme. Le regroupement peut être automatique ou guidé par le développeur lors de l'allocation d'un objet persistant. Le verrouillage adaptatif fait varier dynamiquement le grain du verrou entre la page et l'objet en fonction du degré de contention des transactions sur un ensemble de données [Carey94b]. Le chargement simultané de plusieurs pages (*bulk loading*) qui s'apparente au pré-chargement (*pre-fetching*) permet aussi de réduire le coût des transferts réseaux [Wiener94, Wiener95].

La persistance orthogonale aux types est une fonctionnalité importante qui a guidé de nombreux travaux. La persistance par allocation définit l'état transitoire ou persistant d'un objet au moyen de l'allocation. Cependant, elle pause alors le problème de la suppression explicite des objets et le problème des références pendantes (*dangling pointer*). La persistance par atteignabilité (ou transitivité) considère qu'un objet est/reste persistant quand il peut être référencé de manière transitive à partir d'une racine de persistance. La persistance par atteignabilité oblige alors le système à collecter les objets non atteignables dans le contexte difficile d'une mémoire transactionnelle et distribuée d'objets persistants [Yong94, Ferreira96, Amsaleg99, Brodie-Tzrrell04].

L'identifiant des objets peut être logique ou physique. L'identifiant logique ne dépend pas de l'emplacement physique (mémoire ou disque ou réseau) de l'objet alors que l'identification physique en dépend. L'identification logique permet de déplacer l'objet d'une page à l'autre ou d'une machine à l'autre pour effectuer des regroupements d'objets. L'accès à un objet par un identifiant logique nécessite une opération de déréférenciation qui convertit ce dernier en une adresse mémoire. La déréférenciation peut être précédée par le chargement en mémoire de la page ou de l'objet si celle-ci ou celui-ci font défaut dans le cache. Afin d'éviter une opération de déréferenciation lors de chaque accès, la technique du *swizzling* convertit la « valeur logique » de l'identifiant en une adresse mémoire qui est utilisée directement lors des accès suivants. Quand l'objet est modifié et journalisé, il faut au préalable reconvertir l'adresse mémoire vers la valeur logique de l'identifiant. Comme, cette opération dite de *unsizzling* est coûteuse, des systèmes comme ObjectStore et QuickStore ne réalisent cette opération qu'au rechargement de la page si celle-ci possède des références vers des pages qui ne pourront pas être réinstallées dans le même emplacement de la mémoire (virtuelle) du processus client. Certains travaux ont également anticipé l'apparition des processeurs à adressage virtuel 64 bits pour justifier une identification physique des objets [Shekita90, Gruber92].

L'architecture client-serveur est prédominante dans ces systèmes. Cependant des systèmes comme SHORE [Carey94a] et PerDiS [Shapiro99] proposent une architecture pair à pair pour un réseau de stations de travail qui peut être dépourvu d'un serveur de données. Chaque station est à la fois client et serveur d'un fragment de la base de données.

Les évolutionnistes

La réponse aux révolutionnaires partisans de l'objet persistant fut donnée par les évolutionnistes [Stonebraker90] qui préféraient augmenter le modèle relationnel prédominant par quelques concepts empruntés à l'objet (principalement encapsulation, héritage) tout en respectant les dogmes des SGBDs relationnels qui sont la séparation des données et des traitements et le rejet des langages procéduraux. Les propositions faites pour les SGBDs de 3ème génération étaient entre autres l'extensibilité des types avec les types de données abstraits (*ADT : Abstract Data Type* [Guttag77, BrodieS78]), les identifiants systèmes et l'accès au travers de ces identifiants, les vues modifiables et le support de client-serveur et bien sûr quelques concepts empruntés à l'objet comme l'héritage de type et l'encapsulation via des fonctions et des procédures de types (*stockées*).

Une des principales préoccupations de ce groupe était de permettre une migration progressive des applications et de la pratique de leurs concepteurs et développeurs vers les nouvelles fonctionnalités tout en conservant les bases relationnelles patrimoniales et en permettant même une co-existence.

Les travaux sur PostGres [Stonebraker91] et sur Starburst [Lohman91] ont défriché le terrain pour le groupe de travail sur SQL [Eisenberg99a, Eisenberg04] et pour les principaux éditeurs de SGBDs relationnels (SQL:92) qui étaient Oracle, IBM [Carey99], Informix et Sybase.

Ces systèmes se sont enrichis de l'ajout des ADT via des *plugins*[31] produits par des tiers. Ces *plugins* apportent les méthodes d'accès et des opérateurs propres aux types ajoutés au noyau du SGBD. Ces types sont par exemple des sons, des vidéos, des images, des données spatiales [DeWitt94], des séries temporelles, des séquences d'ADN, des empreintes digitales, …

L'héritage de type permet de définir un type à partir d'un autre type. Les objets sont des lignes de table typées et identifiées par un identifiant système (*row id*). Les références d'objets peuvent apparaître parmi les colonnes d'une table ou dans la définition d'un type. Elles peuvent être utilisées dans les clauses des requêtes SQL. L'héritage de tables supportée par Informix autorise qu'une requête porte simultanément sur une « super » table et ses tables dérivées contenant les objets appartenant à des sous types du type de la « super » table. Ces extensions ont été appelés objet-relationnel car elles gardent la forte empreinte du modèle relationnel.

Une colonne peut désormais être une collection [Beech93] de valeurs ou de références ; ce qui crée une rupture avec la sacro-sainte première forme normale. Les collections sont des listes ou tables ordonnées, des ensembles et des multi-ensembles (autorisant les doublons).

Les procédures stockées et les fonctions associées aux types limitent alors la dichotomie SGBD (données) et hors SGBD (logique métier et présentation). Initialement rédigées avec des langages procéduraux proches de SQL, fonctions et procédures stockées peuvent être désormais en langage Java [Eisenberg99b] (C# pour MS SQLServer) via des interfaces telles que SQLJ [Clossman98]. Le noyau du serveur embarque une machine virtuelle pour exécuter directement les procédures stockées. Les contraintes (*constraints*) et les déclencheurs (*triggers*) renforcent l'intégrité de la base de données et offrent les bases d'un mécanisme de programmation par aspect sur les requêtes SQL. Les vues modifiables combinées aux déclencheurs, autorisent à la fois la migration progressive d'une base relationnelle vers une base objet-relationnelle sans impact sur les applications patrimoniales mais également le développement de nouvelles applications selon le modèle objet-relationnel au dessus d'une base de données purement relationnelle.

Le bilan de ce « duel » est détaillée dans la conclusion de ce chapitre (section II.3.1).

II.1.4 Gestionnaires de Transactions

Gray introduit avec les transactions un modèle simple de synchronisation qui garantit les propriétés ACID (Atomicité, Consistance, Isolation, Durabilité) lorsque plusieurs exécutions accèdent de manière concurrente à un ensemble de données partagées [Gray81]. Ce modèle est le point de départ des nombreux travaux sur le contrôle de concurrence et la reprise sur panne pour des SGBDs et des moniteurs transactionnels centralisés ou distribués [Bernstein96].

[31] Ceux-ci sont appelés DataBlade pour IBM/DB2, Extenders pour Informix/Illustra [Ubell94], Data Cartridges pour Oracle et Plugins pour Sybase.

Le modèle des transactions est bien adapté à un contexte où les exécutions sont de courte durée et comportent un nombre limité de mises à jour. Cependant, ce modèle est limité sur plusieurs points [Gray81] : celui-ci ne supporte pas l'imbrication des exécutions de transaction. De plus, il est pénalisant pour les exécutions de longue durée. Ces limites ont été mises en évidence lors de la mise en œuvre sur les flots de travaux [Alonso97] et des applications collaboratives d'ingénierie pour lesquelles les exécutions durent longtemps et coopèrent souvent sur un ensemble de données qui peut être versionné, ou bien sur des applications de comptabilisation (*accounting*) dans les systèmes de télécommunication qui requièrent une disponibilité immédiate des ressources.

Plusieurs travaux [Elmagarmid92,Mohan94, Jajodia97] ont alors cherché à étendre ce modèle par des modèles avancés de transactions qui conservent les propriétés ACID ou alors relâchent certaines d'entre elles. Les modèles les plus notoires sont le modèle des transactions imbriquées strictes [Moss81], le modèle des transactions imbriquées ouvertes [Saheb99], le modèle des Sagas [Garcia-Molina87], le modèle ConTract [Wachter92], le modèle de transactions coopératives [Nodine92], le modèle de Workspace (*checkin-checkout*) [Lamb91] … Le formalisme ACTA [Chrysanthis94] a permis d'exprimer formellement ces modèles. Ce formalisme a inspiré directement ASSET [Biliris94b] qui est un exemple de système transactionnel flexible pour l'usage de ces modèles dans les applications.

Depuis, d'autres contextes sont apparus avec des besoins relativement similaires. Les applications mobiles considèrent généralement des réplicas de données qu'il faut synchroniser et garder cohérents dans un environnement où les communications avec un serveur fixe sont coûteuses et momentanément indisponibles [Serrano-Alvarado04]. Les services B2B sur le Web fournissent un autre contexte où il est difficile de coordonner la validation d'une transaction de longue durée entre plusieurs organisations autonomes [Little03].

II.2 Contributions

Ma thèse de doctorat se plaçait dans le contexte de la recherche en bases de données. Elle a fait l'objet de la réalisation de WEA une plateforme flexible et distribuée d'objets persistants. Cette plateforme est décrite dans la section II.2.1. Les travaux qui ont suivi m'ont amené à revoir les propriétés de persistance et de fiabilité pour un type de machine très fortement contraint qui est celui de la carte à puce. Ces travaux sont détaillés dans la section II.2.2.

II.2.1 Gérant flexible d'objets persistants

Au début des années 90, les Gérants d'Objets Persistants étaient principalement architecturés sur le modèle du client/serveur dans le but de partager les données persistantes. Cette structure suppose le plus souvent l'existence d'un réseau local pour l'échange de données (*data shipping*). Cependant, le client-serveur s'ajuste mal aux infrastructures très largement distribuées, aux clients nomades et aux traitements confidentiels qui ne peuvent être effectués que par des serveurs directement gérés par l'entreprise (*query-shipping*).

Ma thèse de Doctorat s'est intéressée à proposer une architecture flexible de gestionnaire d'objets persistants et transactionnels pouvant combiner à la fois l'échange de données et la soumission de traitements. La proposition faite est le modèle des Espaces de Travail (*Workspace*) [Donsez93,Donsez94a,Donsez95] qui permettait de composer des

architectures complexes d'applications en mettant en relation des espaces de données et d'exécution via des relations d'échange de données et via des relations de services. Chaque espace de travail est implémenté par un noyau d'accès transactionnel à des données persistantes; les données persistantes peuvent être stockées et jounalisées sur un disque local ou sur un disque distant géré par un autre Espace de Travail. Les Espaces de Travail communiquent suivant le principe des services pour l'échange de données (*data-shipping*) ou de requêtes de calcul (*query-shipping*).

Le Service de Données propose d'exporter des données persistantes d'un Espace de Travail serveur vers des Espaces de Travail clients qui exécutent les applications; ce service est récursif car il autorise un client à exposer le même service à d'autres clients. Cette propriété est utilisée pour construire plusieurs architectures de SGBDs. Un SGBD local mono-utilisateur est réalisé au moyen d'un simple Espace de Travail. Une architecture de SGBDs Client/Serveur sur un réseau local met en œuvre un espace de travail pour chaque serveur et un par client (Figure II-1 gauche): chaque espace de travail client importe les données exportées par les espaces de travail serveur. L'architecture pair à pair choisie par SHORE et PerDis est réalisée au moyen d'espaces de travail pairs (Figure II-1 droite). Chaque espace de travail exporte les données qu'il gère sur son disque local et importe les services de données des autres espaces pairs. Une architecture de SGBDs pour réseau largement réparti est réalisée en insérant un espace de travail mandataire (*proxy*) entre les espaces de travail client et les espaces de travail serveurs répartis sur le réseau distant (Figure II-2).

La gestion de la concurrence du Service de Données est conforme au modèle des transactions ACID. Elle est basée sur l'algorithme de *callback locking* pour la gestion des verrous et du cache des données. La journalisation peut être réalisée sur des disques locaux à la machine qui héberge l'espace de travail ou par l'espace de travail fournissant également le service de données. Les objets mis à jour sont « redescendus » vers l'espace de travail fournisseur sous la forme de différentiel (*delta*) d'objets ou de pages. La validation entre plusieurs espaces de travail suit la validation à deux phases. Le coordinateur de la validation est l'initiateur de la transaction. L'espace de travail fournisseur est en principe un processus localisé d'une machine distante. Cependant, dans le contexte des transactions emboitées, la transaction racine et les sous-transactions correspondent chacune à un espace de travail exécuté par un processus distinct. Les processus peuvent être colocalisés sur la même machine ou distribués sur un réseau.

Un espace de travail implémente également deux autres politiques de synchronisation sur les données qui permettent de construire des transactions imbriquées strictes [Moss81] et une politique de transactions « coopératives » (Figure II-3) [Donsez94b] similaires à celle des *workspaces* d'ObjectStore [Lamb91] qui réalisent des *chechin/checkout* de graphes d'objets avec la base principale. Un des avantages de notre modèle est qu'il offre la flexibilité d'appliquer différentes politiques aux différents niveaux de la hiérarchie d'espaces de travail. Par exemple, on peut définir un groupe d'utilisation ACID sur un espace régi par une politique *checkin/checkout*.

Le Service d'Opération réalise le calcul de requêtes sur le serveur. Ces services sont en général définis par les concepteurs de bases de données pour sécuriser les traitements et les données. Ce service est semblable aux procédures stockées du SQL. Cependant, l'espace de travail propose également un service mixte offrant à la fois l'exportation de données vers le client et la réalisation de requêtes sur le serveur. Le service mixte permet d'avoir des réplicas d'un ensemble de données seulement accessibles en consultation dans les espaces de travail client et un réplica accessible en modification dans l'espace du fournisseur. Le service synchronise les mises à jour de l'espace fournisseur avec les espaces clients.

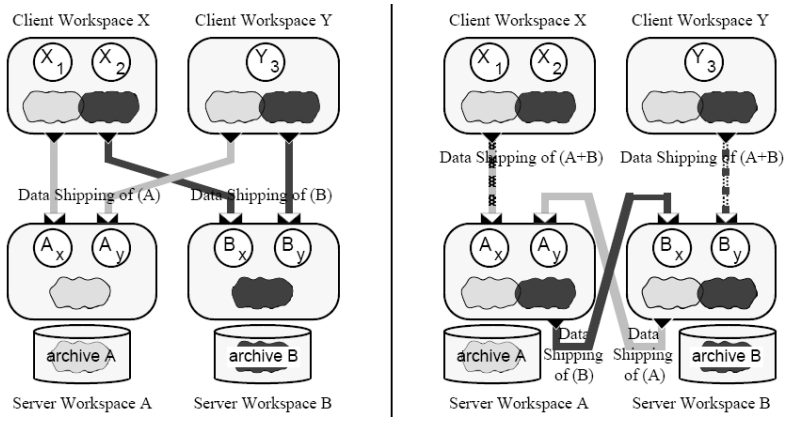

Figure II-1: Architectures par assemblage d'espaces de travail : le client-serveur (gauche), le pair à pair (droite).

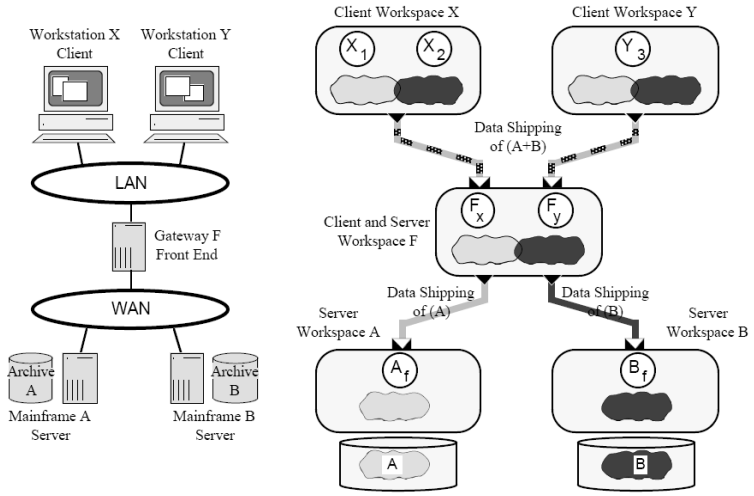

Figure II-2: Architecture d'espaces de travail pour réseau largement réparti.

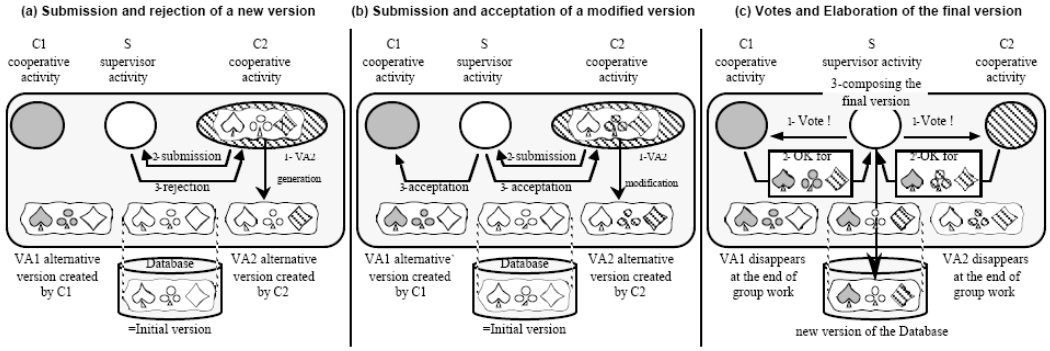

Figure II-3: Modèle de coopération dans un espace de travail.

Les Espaces de Travail font l'objet d'une implantation, WEA (Workspace Environment Architecture). Cette implantation repose sur l'utilisation de mécanismes systèmes alors émergents comme le *multi-threading* ou le *memory-mapping*. Un espace de travail peut abriter l'exécution de plusieurs groupes de *threads* chacun attaché à une transaction.

Cette fonctionnalité m'a conduit à préférer les *smart pointers* [Edelson92] au *swizzling* pour réaliser la déréférenciation des identifiants logiques d'objets. En effet, dans le contexte multi-thread et multi-transaction, le *swizzling* qui est propre à chaque transaction entraîne les recopies de pages (*copy-on-write*). Le surcoût par déréfenciation descend à 12 cycles (sur architecture Sparc) grâce à un cache efficace de traduction d'identifiants. Le *memory mapping* est utilisé entre autre pour réaliser la détection implicite des verrouillages. Les communications pour les services de données et d'opérations étaient réalisées au moyen d'un ORB pré-CORBA[32] [OMG94] utilisant l'échange asynchrone des messages de requête et de réponse. La génération des talons et squelettes étaient réalisés au moyen de macros CPP sur une version allégée de l'IDL CORBA 1.0. L'interface C++ de WEA offre également des collections persistantes.

Les perspectives données à ce WEA étaient la réplication et la migration de données entre plusieurs espaces de travail pour les domaines émergents du stockage, de la recherche et de la visualisation des objets multimédia (audio, vidéo) de grandes tailles [Homond95] et des documents semi-structurés SGML [Chen97] sur des réseaux à large échelle. Ces perspectives ont été réalisées dans le cadre du projet Tr@nsDoc financé lors de l'appel à projet « Autoroutes de l'Information ». Le modèle des espaces de travail a fait également l'objet d'une implémentation alternative, baptisé Yooda. Yooda a été réalisée par Eric Abécassis en thèse dans l'équipe RAPID [Abecassis95] et a été utilisé dans le produit SIG par la société APIC Systems qui était alors son employeur.

II.2.2 Persistance, Transaction et Cartes à puce

A partir de 1995, le boom de la téléphonie mobile et du paiement sécurisé pour le e-Commerce a dopé le marché de la carte à puce [Rankl97]. Cette machine, très fortement contrainte par sa mémoire, véhicule dans la poche de son porteur ou dans l'emplacement SIM[33] de son téléphone des données confidentielles avec un très haut niveau de sécurité. Cependant, la production de logiciels fiables pour carte à puce demandait un temps (*time-to-market*) incompatible avec les attentes des opérateurs de télécommunications mobiles désireux de satisfaire très rapidement les besoins de leur clientèle sur un marché très concurrentiel. L'arrivée des cartes de nouvelle génération [Guthery97] interprétant du *bytecode* JavaCard[34] allait démocratiser le développement d'applications encartées (*cardlets*) et intéresser les académiques à jeter un œil sur cette machine fortement contrainte par sa mémoire primaire et secondaire.

Les cartes de nouvelle génération possèdent deux caractéristiques intéressantes :

- Les données allouées par les applications encartées sont des graphes d'objets persistants, similaires à ceux gérés par les objets persistants à la place des fichiers ISO7816-4 des générations précédentes.
- Plusieurs applications relatives à différents fournisseurs de services peuvent être installées à distance (*OTA : Over-The-Air*) de manière sécurisée par l'émetteur de la carte.

Les problèmes identifiés pour ces cartes de nouvelle génération étaient alors :

[32] La vision du projet Eureka Software Factory [Fernstrom91] avait grandement inspiré la conception de notre ORB avant même la publication du premier document sur CORBA.
[33] Initialement, les concepteurs GSM prévoyaient plusieurs téléphones de portée et d'encombrement variables pour chaque abonné. La raison était du à la couverture du territoire très hétérogène en qualité. La carte devait donc pouvoir facilement migrer d'un combiné (*handset*) mobile à celui du véhicule.
[34] Plus exactement, la spécification JavaCard [JavaCard] ne reprend qu'un sous-ensemble du *bytecode* Java et limite fortement les possibilités de la machine virtuelle JavaCard VM : absence de ramasse-miette, absence de *thread*, absence de la réflexion, absence de la sérialisation, …

1. la nécessité de fiabiliser l'exécution des applications encartées pour garder les données cohérentes en cas de panne et de ce fait, éviter des failles de sécurité,
2. la nécessité de considérer l'application encartée comme un des éléments d'une application répartie sur des cartes, des terminaux et des serveurs,
3. la nécessité pour plusieurs applications encartées de coopérer de manière sécurisée dans la réalisation d'un service,
4. et la nécessité de pouvoir faire évoluer le logiciel et les données encartées de manière sécurisée.

Les solutions apportées ont été des algorithmes de reprise sur panne pour les systèmes d'exploitation carte, un moniteur transactionnel orienté carte et un modèle de partage sécurisé des données persistantes et évolutives pour les applications encartées.

II.2.2.1 Algorithmes de reprise sur panne pour les systèmes d'exploitation carte

Les données manipulées par les applications encartées sur les JavaCards sont des graphes d'objets alloués dans la mémoire secondaire de la carte qui peut-être de technologie FlashRAM ou EEPROM. Contrairement aux disques durs, ces technologies de mémoire secondaire sont directement adressables en place avec des niveaux de performance en lecture équivalents aux mémoires RAM. Dans le cas de la mémoire FlashRAM, l'écriture reste cependant beaucoup plus lente avec un facteur de blocage supérieur.

Un des problèmes des cartes est le risque de panne lié à la perte d'alimentation. Ce risque est principalement dû à l'arrachement de la carte du lecteur par le porteur ou bien à l'épuisement de la batterie du lecteur mobile. Ces pannes conduisent à des incohérences dans le graphe d'objets partiellement sauvegardé. Ces incohérences sont des sources de faille qu'exploitent les fraudeurs[35].

Nos propositions ont porté sur des algorithmes de reprise sur panne [Donsez98] garantissant la propriété d'atomicité et la durabilité des opérations groupées dans une transaction JavaCard. Ces algorithmes sont des adaptations des 2 algorithmes dits des pages ombres et de la journalisation avant [Bernstein87] pour les technologies de mémoire FlashRAM et EEPROM utilisées dans les cartes. Ces algorithmes étaient fortement contraints par la faible quantité de mémoire primaire disponible (c.a.d. 512 octets au total pour le système d'exploitation et la pile de l'application en 1998) et par l'érosion des points mémoire lors des écritures répétées.

II.2.2.2 Moniteur transactionnel carte

Le travail précédent avait été réalisé dans le contexte des transactions définies par la spécification JavaCard. Une des limites de ces transactions était que leurs portées n'excédent pas la durée d'une commande APDU qui est l'atome de dialogue avec une carte. Cette limite ne permettait pas d'utiliser la carte dans le contexte transactionnel d'une application distribuée sur plusieurs machines comme des serveurs, des terminaux ou bornes (de paiement) et même d'autres cartes. Plusieurs algorithmes de reprise sur panne ont été proposés afin que la délimitation d'une transaction puisse s'étendre sur

[35] Une des premières démonstrations de la JavaCard a été la très classique applet TicTacToe du JDK1.0.2. La logique de jeu en principe infaillible était encartée alors que l'IHM était exécutée par le PC hôte. Une des personnes assistant à la démonstration frauda au jeu en arrachant la carte du lecteur et battit la carte qui « raisonnait » avec une grille incohérente par rapport au nombre de tours de jeu.

plusieurs requêtes voire plusieurs (re)connexions de la carte. Un des objectifs était, dans un premier temps, de pouvoir utiliser la carte comme un esclave dans le protocole de validation à deux phases [Bernstein97].

L'objectif suivant était de montrer que les cartes à puce pouvaient être également utilisées pour coordonner l'aboutissement sécurisé d'une transaction entre plusieurs partenaires dans le cadre de nouvelles applications de e-Service ou de e-Commerce [Billard98]. Le travail a consisté à définir et à implanter un moniteur transactionnel minimal embarqué dans la carte [Lecomte97, Lecomte99]. Ce travail nommé COST-STIC a fait l'objet de la thèse de Sylvain Lecomte [Lecomte98D]. Une maquette de démonstration a été réalisée sur une JavaCard, avec un ORB Corba et un service de transaction suivant la spécification Object Transaction Service (OTS) de CORBA.

Une des conclusions de ce travail était que le modèle de transactions ACID plates et l'algorithme de validation à deux phases restaient inadaptés dans ce contexte opérationnel. Une piste explorée ultérieurement [Jean00a] a été le modèle des transactions emboîtées ouvertes (*Open Nested Transaction*) [WeikumS92].

II.2.2.3 Partage contrôlé et évolutif des données persistantes entre les applications encartées.

Une autre limite de la spécification JavaCard portait également sur le modèle de collaboration des applications encartées. Ce modèle est basé sur la publication par une application d'un objet partagé accessible par les autres applications. Ce modèle a selon nous, deux limitations. Premièrement, il oblige le développeur à mêler le code fonctionnel de l'objet avec le code de contrôle d'accès aux méthodes de l'objet. Deuxièmement, le modèle de données est alors souvent lié à l'application fournisseur et toute évolution de l'application qui passe par une désinstallation puis une réinstallation oblige à sortir les données de la carte, les convertir vers le nouveau modèle de données puis enfin réinstaller les données converties dans l'application fraîchement mise à jour. D'une part, ce mécanisme d'import-export, qui doit impérativement être sécurisé, n'est pas toujours possible quand les données doivent être conservées au secret uniquement dans la carte. D'autre part, ce mécanisme d'import-export oblige le développeur à le prévoir dans son application dès la conception de celle-ci.

L'indépendance des traitements et des données qui est la profession de foi des systèmes de gestion de bases de données, nous semblait un principe applicable à la collaboration des applications encartées. Nous avons proposé un modèle de carte hybride, c'est à dire à la fois, orientée données comme les cartes fichiers ISO 7816-4 et les cartes bases de données ISO 7816-7 [Paradinas94] et orientée traitement comme les cartes JavaCard. Cette carte hybride est architecturée selon le modèle des serveurs de bases de données relationnelles [Jean00b]. L'application encartée est alors considérée comme un ensemble[36] de procédures stockées [Eisenberg96]. Celle-ci dispose de son espace privé d'objets JavaCard persistants et d'un espace partagé de données relationnelles via une interface JDBC-SC semblable à JDBC. Le contrôle d'accès est étendu aux données relationnelles partagées comme aux procédures stockées. Un avantage indirect du modèle hybride est qu'il n'impose pas au concepteur de choisir dès la conception un des 2 modèles pour développer son application carte.

Ce travail a fait l'objet d'une partie de la thèse de Sébastien Jean [Jean01D]. Une maquette de démonstration a été réalisée au moyen d'une *cardlet* JavaCard contenant un interpréteur SQL et un contrôleur de droits (Figure II-4). Cette cardlet sert à la fois des

[36] Chaque commande APDU de l'application ou chaque méthode JavaCard RMI (JCRMI) [Vandewalle98] est associée à une procédure stockée.

ordres CQL de l'extérieur (APDU CQL dans la figure) et des invocations de méthode. Ces dernières (comme APDU A1::C1 sur la figure) peuvent utiliser un objet partagé implémentant une API JDBC-SC qui offre un sous-ensemble des fonctionnalités de l'API JDBC. (Java DataBase Connectivity)[Hamilton96]. Ce démonstrateur était complété par un pré-compilateur SCQLJ simplifiant l'utilisation de l'interface JDBC-SC de la *cardlet* de manière similaire à SQLJ [Eisenberg98, Clossman98] introduit par Oracle pour faciliter l'usage de JDBC dans les programmes Java. Le scénario d'usage concernait une application de partage de points de fidélité entre plusieurs commerçants affiliés au même programme de fidélisation du porteur de la carte.

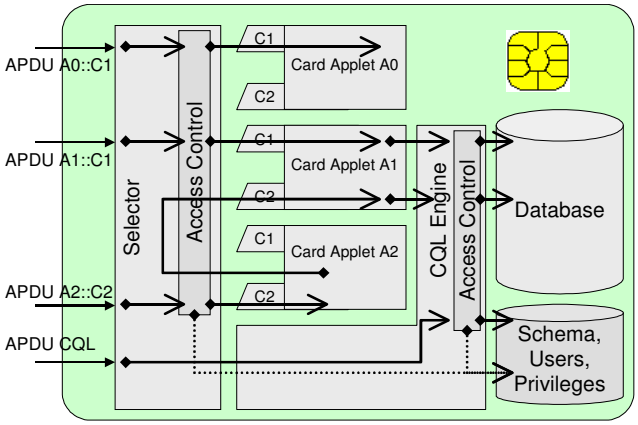

Figure II-4: Architecture de la carte hybride.

Une des perspectives du travail a été ensuite de rendre active la carte hybride afin de déclencher le traitement de règles actives [Paton99] en cas de mises à jour (*insert/delete/update*). Ces traitements pouvant être externalisés de la carte, la carte devait pouvoir s'affranchir de son rôle de serveur et devenir à son tour client en invoquant les traitements externes. Cette carte peut également héberger des traitements asynchrones souscrivant et émettant des messages [Donsez01a,Donsez01b] selon les principes des messageries inter-applicatives [Eugster03]. Les messages envoyés vers la carte sont tamponnés et redirigés vers le lecteur dans lequel la carte est insérée. Ces travaux avaient pour principal intérêt d'affranchir le développeur d'application carte de la contrainte imposant que les traitements soient limités à la durée de connexion de la carte dans un lecteur. Ces perspectives ont fait l'objet de la seconde partie de la thèse de Sébastien Jean sur la carte réactive AWARE [Jean01D].

II.3 Conclusion

Ce chapitre a présenté les résultats de nos travaux sur les systèmes à objets. La persistance et la fiabilité dans ces systèmes ont été le fil conducteur de ces travaux.

La section II.2.1 a présenté mes travaux sur ces deux propriétés dans le contexte d'applications à objets distribués sur divers types de réseaux de machines. Nous nous sommes intéressés à proposer une architecture flexible de gérants d'objets persistants et transactionnels autorisant plusieurs modèles d'architecture (client-serveur, proxy d'entreprise ou bien pair à pair) et plusieurs modèles avancés de transactions (plat, emboîtée ou bien coopérative).

La section II.2.2 a présenté la poursuite de ces travaux vers des machines fortement contraintes par leur matériel. Nous avons proposé des gestionnaires de persistance pour les cartes à microprocesseurs (dites cartes à puce). Il nous paraissait aussi opportun de considérer ces cartes comme des nœuds d'un réseau, pouvant participer à des transactions distribuées et jouant parfois un rôle central dans ces transactions en embarquant le contrôle de leurs validations.

II.3.1 Bilan du domaine

En 1996, Carey et DeWitt dressaient un premier bilan de la compétition entre révolutionnaires et évolutionnistes sur la décade 1986-1996 et donnaient rendez-vous en 2006 pour le bilan suivant [Carey96]. Le bilan est (selon moi) que les révolutionnaires ont perdu mais ils ont laissé des bonnes idées reprises par les évolutionnistes. Ces derniers n'ont cependant pas réussi à gagner la bataille de l'objet car pour le moment, le modèle relationnel pur reste la référence dans les développements d'applications avec J2EE (EJB2.x) et .NET bien que les langages supports (Java et C#) soient orientés objet. Cependant, le modèle objet-relationnel [Gray04] pourrait bien enfin finir par percer notamment avec les spécifications EJB3.0 et JDO qui, ironie du sort, ont largement être poussées par les anciens révolutionnaires. Il est intéressant de noter que les révolutionnaires se sont reconvertis vers le stockage et la recherche dans les corpus documentaires semi-structurés (i.e. XML). Il est également intéressant de noter la résurgence des langages de requêtes sur des ensembles d'objets avec l'effort de Microsoft d'intégrer les requêtes dans C#3.0 avec LINQ (Language INtegrated Query) afin de soulager l'*impedance mismatch* avec les bases de données relationnelles[37] et les corpus documentaires XML. Plusieurs agendas de recherche dans le domaine des bases de données semblent s'accorder pour dire que la nouvelle frontière sont la gestion des informations diffuses dans les milliards de capteurs et d'étiquettes RFID à venir [ITU] et les traitements des flux d'information provenant de ceux-ci. La Figure II-5 représente une extrapolation des vagues de Raccoon appliquées aux modèles de bases de données.

L'impact industriel des modèles avancés de transactions est resté insignifiant par rapport au modèle des transactions « plates » qui se retrouve dans tous les SGBDs , les moniteurs transactionnels et les services d'application (JEE et .NET) respectant la spécification XOpen/DTP [XOpen91]. Le modèle des transactions imbriquées strictes a été implémenté dans quelques produits orientés télécommunications suivant la spécification CORBA OTS (Object Transaction Service) de l'OMG. Cependant, Sun qui s'est largement appuyé sur OTS, a abandonné le modèle des transactions imbriquées strictes pour spécifier Java Transaction Service (JTS) du J2EE[38]. La résurgence de certains de ces modèles pourrait venir des Web Services pour lesquels le modèle des transactions plates est impraticable. Les propositions (XAML, BTP, WTP, WS-Transaction) [Little03] qui se succèdent depuis 5 ans, ont comme dénominateur commun la compensation d'actions qui est au cœur du modèle des transactions imbriquées ouvertes [Sahed99]. Ces modèles sont également exploités par les ateliers de développement logiciel comme Visual Studio et Eclipse mais sans un réel support efficace des systèmes de fichiers sous-jacents.

[37] LINQ ne semble pas adresser pour l'instant les bases de données objet-relationnelles.
[38] La raison est que ce modèle n'est par supporté par les principaux SGBDs relationnels.

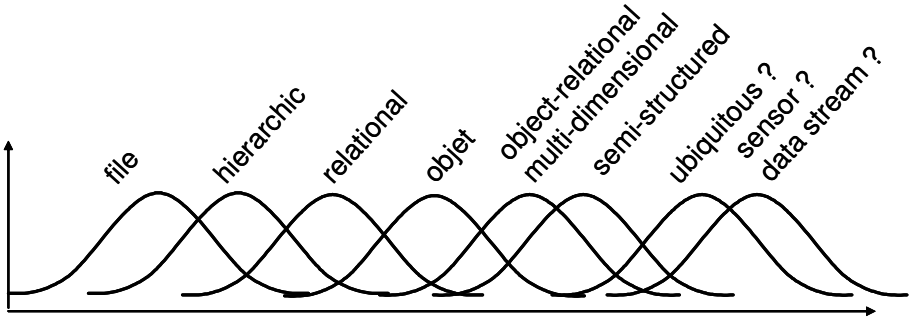

Figure II-5: Vagues de Raccoon appliquées aux bases de données

II.3.2 Bilan de ma recherche

Comme beaucoup, je faisais partie du camp des « vaincus » : la révolution objet avait été « avortée » par le conservatisme relationnel. Cependant, en travaillant à l'équipe RD2P, je pensais que le monde de la carte à puce qui venait juste d'adopter Java pour le développement des *cardlets*, pourrait accepter l'idée qu'une carte soit une mémoire transactionnelle d'objets Javacard persistants comme d'autres l'avaient proposé pour Java [Atkinson96]. Cependant comme avec le monde des applications Java, SQL qui reste le synonyme de persistance, semblait plus acceptable et rassurant pour une communauté qui venait de subir une révolution. Nos propositions se sont tournées vers l'intégration des principaux concepts avancés des bases de données relationnelles dans la carte.

Les expérimentations que nous menions, s'intéressaient également à l'intégration la carte dans les systèmes d'information des entreprises au moyen d'intergiciels en émergence (CORBA/OTS, EJB, JMS, SOAP …). Ces expérimentations furent pour moi l'occasion de me pencher nouveau sur le cas des transactions avancées à la fois pour la carte et pour le serveur d'entreprise qui dans le cadre du projet PEPiTA était le serveur J2EE JOnAS. Ce fut l'occasion de pratiquer le modèle à composants EJB très imparfait et en pleine maturation. Je me suis alors intéressé aux modèles de composants et aux technologies des conteneurs pour leurs mises en œuvre. Ma mutation dans l'équipe ADELE alors en pleine réflexion sur les composants, allait conforter l'orientation de mes recherches dans ce domaine qui font l'objet du chapitre suivant.

II.3.3 Projets et collaborations

Le projet Tr@nsDoc (labellisé et financé par l'appel à projet "Autoroutes de l'Information") a été l'occasion d'utiliser le modèle des Espaces de Travail dans le contexte émergent des Autoroutes de l'Information. L'objectif consistait à distribuer, diffuser, facturer des flots de données multimédia suivant des qualités de service (contraintes temps réel) et de stocker et rechercher dans des corpus documentaires semi-structurés. Ce projet réunissait mon équipe d'accueil à l'Université de Technologie de Compiègne, (animée par Liming Chen) et l'équipe de l'Université Paris VI (animée par Pascal Faudemay) dans laquelle j'avais passé ma thèse et deux partenaires industriels (les PME GECIP SA et STSI SA) avec qui j'étais en relation. Ce projet a été ma première expérience de montage de projet. Avec mon départ pour Lille puis Valenciennes, le projet s'est focalisé vers l'élaboration de méthodes d'indexation automatique de vidéos en utilisant de manière combiné les images, la bande son et éventuellement le script et des méthodes d'indexation semi-automatique par l'annotation des passages des vidéos par des spectateurs.

Les travaux autour de la carte à puce ont été réalisés en collaboration étroite avec le laboratoire de R&D de la société Gemplus (désormais Gemalto), dirigé par Pierre Paradinas et l'équipe RD2P du LIFL, dirigée par le Professeur Vincent Cordonnier. J'ai pu participer à l'encadrement scientifique des doctorants de l'équipe qui étaient David Carlier (Directeur technique de la société Bantry Technologies), Sylvain Lecomte (Professeur en Informatique à l'Université de Valenciennes) et Sébastien Jean (Maître de conférences en Informatique à l'IUT de Valence, Université Grenoble 2). Le projet ITEA PEPiTA détaillé dans la section III.3.2, fut également une occasion d'expérimenter la carte avec des serveurs d'applications avancées (J2EE).

Ses travaux ont fait l'objet de nombreux stages et projets d'études pour des élèves ingénieurs, des maîtrises et des DESS d'informatique, pour le développement des prototypes et des démonstrateurs.

II.3.4 Enseignements

Ces travaux de recherche m'ont permis de monter de nouveaux cours sur la sécurité et les cartes à puce pour le DESS TNSI de Valenciennes et pour l'option « Monétique et Traçabilité » de l'IUP MeSSI du Centre Joseph Fourier à Valence.

II.3.5 Publications

[Chen97] Liming Chen, Didier Donsez, Pascal Faudemay : Design of U-Doc, a research vehicle for hyper document retrieval on the Internet. Actes de Basque Intl Workshop on Information Technology - Data Management Systems, Biarritz, France, 2-4 Juillet 1997

[Donsez01a] Didier Donsez, Sébastien Jean, Sylvain Lecomte, Olivier Thomas : Asynchronous Use of Smart Card Services Using SOAP and JMS. Actes électroniques 3rd Gemplus Developer Conference (GDC'2001)

[Donsez01b] Didier Donsez, Sébastien Jean, Sylvain Lecomte, Olivier Thomas :Turning Multi-Application Smart Cards Services Available from Anywhere at Anytime : a SOAP/MOM Approach in the Context of JavaCards. Actes de la conférence eSmart 2001, Cannes, Septembre 2001, LNCS 2140, pp83-94

[Donsez93] Didier Donsez et Philippe Homond : WEA, Des Espaces de Travail Distribués à Objets Persistants. Actes des Journées des Jeunes Chercheurs en Systèmes à Mémoire Logiquement Partagée, Toulouse, Septembre 1993.

[Donsez94a] Didier Donsez, Philippe Homond et Pascal Faudemay : WEA, A Distributed Object Manager based on a Workspace Hierarchy. Actes de International Conference on Applications in Parallel and Distributed Computing , Caracas, Venezuela, Avril 1994.

[Donsez94b] Didier Donsez, Philippe Homond et Pascal Faudemay : A Cooperative Database System based on a Workspace Hierarchy. Actes de CODATA '94, Committee on Data for Science and Technology, Chambery, France, Septembre 1994 , pp247-258, Eds J-E Dubois, N. Gershon, ISBN-3-540-61457-5, Springer Verlag.

[Donsez95] Didier Donsez, Liming Chen et Pascal Faudemay: Shared Distributed Memory : the Workspace Model. Actes de European Research Seminar on Advances in Distributed Systems (ERSADS) L'Alpe d'Huez, France, Avril 1995.

[Donsez98] Didier Donsez, Gilles Grimaud, Sylvain Lecomte : Recoverable Persistent Memory for Smartcard. Actes de the 3th Smart Card Research and Advanced Application Conference (CARDIS) IFIP, UCL Louvain-la-Neuve , Belgique, 14-16 Septembre 1998, Springer-Verlag, LNCS 1820.

[Jean00a] 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.

[Jean00b] 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

[Jean01] Sébastien Jean, Didier Donsez et Sylvain Lecomte :Using some database principles to improve cooperation in multi-application smart cards. Proceedings of the IEEE Chilean Computer Society Symposium (CCSS'01), 2001.

[Lecomte97] Sylvain Lecomte et Didier Donsez : Gestion de Transactions pour les Cartes à Microprocesseur. Actes de la conférence NOTERE 97, Pau, France, Novembre 1997.

[Lecomte99] Sylvain Lecomte, Gilles Grimaud, Didier Donsez : Transactional Mechanisms for Open Smart Card. Actes de Gemplus Developer Conference (GDC'99), Paris, CNIT, Juin 1999

Chapitre III

Composants et Services Non Fonctionnels

La mise en œuvre d'applications orientées objet, persistantes et transactionnelles se révélait complexe. Le développeur métier devait particulièrement bien connaître l'API de programmation du système transactionnel et comprendre les mécanismes sous-jacents afin d'obtenir une application performante dans son contexte opérationnel. Généralement, les contraintes de performance obligeaient à revoir les choix transactionnels effectués lors de la conception et à faire évoluer le code source de l'application. La programmation orientée composant [McIlroy68, Nierstrasz92, Szyperski98] nous paraissait une solution intéressante qui soulagerait le développeur métier de l'utilisation des transactions avancées dans les applications.

Bien que plusieurs modèles de composants existaient, nous nous sommes alors focalisés sur le modèle émergent[39] des Enterprise JavaBeans (EJB) [Matena98] qui, bien que très imparfait, adressait les besoins non fonctionnels des applications d'entreprise. Une première étude s'intéressait à l'utilisation et l'intégration de modèles avancés de transactions dans les applications à base d'EJB. L'apparition des nouveaux contextes applicatifs mettant en œuvre des terminaux nomades, des terminaux de télévision interactifs et des passerelles industrielles, nous a poussés à étudier des modèles de composants dédiés à ces contextes. Le déploiement des composants dans ces plateformes devenait un enjeu majeur pour les plateformes d'exécution de ces composants. Nous avons étudié le déploiement de composants à la demande pour des applications de télévision interactive et la fiabilisation du déploiement d'applications sur un parc de machines.

Ce chapitre présente nos travaux relatifs aux plateformes à composants. La section 1 rappelle les grands axes de recherche menés autour des plateformes à composants. La section 2 présente mes contributions dans ce domaine. La section 3 dresse un bilan sur le domaine et sur mes résultats.

[39] Le JSR 19 définissant les EJB 2.0 a été initié en Juin 1999 et approuvé en Septembre 2001.

III.1 Problématiques et Etat de l'art

Les technologies objets ont apporté leurs contributions pour lutter contre la « crise du logiciel » [Dijkstra72,Cox90, Mahoney04]. Cependant, la réutilisation du logiciel n'a pas eu le succès escompté avec celles-ci. Une des raisons est sans doute le fait que le logiciel développé mélange sa structure, ses contraintes et ses fonctionnalités (ie. code). La réutilisation reste possible tant que les produits ont une structure similaire. Les changements majeurs dans la structure ne permettent pas une réutilisation aisée du logiciel [Pfister96]. De plus, l'ajout des objets implémentant les contraintes opérationnelles comme la fiabilité, la concurrence des accès, la distribution, la montée en charge ou bien la sécurité, rend la maintenance et la reconfiguration de ces objets complexes et parfois risquées pour la stabilité de l'application dans son environnement opérationnel. Les propositions [Cointe04] qui ont suivi visaient à clarifier l'architecture de l'application et à séparer les préoccupations [Hürsch95] des différents acteurs intervenant dans son développement. Les propositions notoires qui se côtoient [Pessemier2004], sont essentiellement la programmation orientée composant et la programmation orientée aspect [Kiczales96, Kiczales97, Pawlak05a]. Mes recherches s'étant principalement placées dans le contexte des composants, cet état de l'art porte essentiellement sur ces derniers.

La programmation orientée composant, qui représente une étape importante dans le domaine du génie logiciel, focalise principalement le développement du logiciel sur la réutilisation et l'intégration de composants. Ceux-ci n'ont parfois pas été conçus pour être composés avec d'autres composants (*COTS : Commercial Off The Shelf*) [Voas98]. Elle relève du concept de « *Programming-in-the-Large* » introduit par [DeRemer76], qui recommande de manipuler des entités logicielles de plus gros grain que les objets qui représentent très souvent les concepts atomiques manipulés par le logiciel. L'ingénierie des logiciels à composants s'intéresse également à la séparation des préoccupations afin de permettre l'intervention de différents acteurs et spécialistes technologiques tout au long du cycle de production et de maintenance (appelé cycle de vie) d'un logiciel. Elle permet ainsi un développement modulaire et une maintenance plus aisée des logiciels complexes. Pour leur part, les plateformes d'exécution des composants isolent ceux-ci des contingences des environnements opérationnels et préservent le développeur métier de leur manipulation.

III.1.1 Définition

La notion de composant logiciel n'est pas récente [McIlroy68]. Néanmoins les travaux autour des composants n'ont commencé à se développer qu'au début des années 90 [Nierstrasz92,]. La notion est porteuse de nombreuses définitions dans la littérature. Une des définitions les plus souvent citées est celle que donne Szyperski [Szyperski98] :

"*Un composant logiciel est une unité de composition ayant des interfaces spécifiées de façon contractuelle et possédant uniquement des dépendances de contexte explicites. Un composant peut être déployé de manière indépendante et il est sujet à composition par des tierces parties*."

La première phrase de cette définition insiste sur le caractère contractuel de ce que fournit le composant et de ce dont il a besoin pour être composé avec d'autres composants. La seconde phrase introduit à la fois une considération nouvelle qui est le déploiement et elle implique que plusieurs acteurs spécialisés ayant des préoccupations différentes, interviennent tout au long de cycle de vie d'un logiciel à base de composants.

III.1.2 Composition et Architecture

La définition insiste sur le développement du logiciel par composition d'unité logicielle. Cette composition est réalisée par la liaison des facettes et des réceptacles des composants. La liaison doit respecter les contraintes de compatibilité entre facette et réceptacle pour être valide. Les liaisons entre réceptacles et facettes peuvent refléter des types d'interaction différents. Un type d'interaction donné est généralement introduit par un modèle en fonction des besoins du domaine d'application qu'il cible. Par exemple, l'interaction peut représenter un flux d'événements asynchrones dans une application RFID, un flux de mesures selon le modèle producteur-consommateur dans une application basée capteurs ou bien encore un flux de type 1 vers N en diffusion synchrone dans une application de calcul sur grille de machines. La composition hiérarchique (ou récursive) [Batory92] considère qu'une composition des composants primitifs (c.a.d. l'atome de composition) est elle-même un composant (dit composite) disposant de ses propres facettes, ses propres réceptacles et ses propres propriétés de configuration. Elle est elle-même composable avec d'autres composants. La composition hiérarchique masque ainsi la complexité des niveaux inférieurs de la hiérarchie et assure ainsi la réutilisation de « bibliothèques » d'assemblages.

Les travaux sur les architectures logicielles et systèmes [Allen92, Perry92, Kruchten95, Garlan97,Bass98], sont contemporains de celles sur les composants. L'architecture d'un logiciel à composants définit les relations (i.e. liaisons) entre les composants ainsi que leurs configurations. Les langages spécialisés d'architectures (ADL) [Medvidovic00] servent à définir celles-ci. L'architecture du logiciel est généralement dissociée de l'environnement opérationnel sur lequel il sera déployé et exécuté. Cependant, les éléments de l'architecture sont parfois « décorés » par des informations décrivant les contraintes opérationnelles qui définissent les besoins non-fonctionnels de l'application.

III.1.3 Besoins et services non fonctionnels

Alors que les besoins fonctionnels expriment la logique métier du logiciel à développer, les besoins non fonctionnels[40] d'un logiciel rassemblent l'ensemble des qualités requises à son fonctionnement dans l'environnement opérationnel du client. Ces besoins sont par exemple la performance, le contrôle d'accès, la fiabilité, la persistance, le placement … Ces besoins sont généralement satisfaits par des services (ou intergiciels) techniques (également appelés services non-fonctionnels) comme des moniteurs transactionnels, des gestionnaires de bases de données ou des annuaires. La satisfaction de ces besoins dans un code métier est généralement considérée comme difficilement réalisable par un développeur métier [Kienzle02]. Elle est source d'erreurs, de failles ou de mauvaises performances quand ces besoins ne sont pas traités par des spécialistes.

Le principe de séparation des préoccupations (*separation of concerns*) [Hürsch95, Parnas72] qui a été popularisé par la programmation orientée aspect, apporte une première réponse en permettant plusieurs spécialistes du besoin non fonctionnel d'ajouter le code nécessaire sans pour autant le « diluer » dans le code fonctionnel.

Les plateformes à composants ont également suivi le principe de séparation des préoccupations pour qualifier les besoins fonctionnels de l'application et des composants qui la constituent. Le conteneur d'un composant est un élément technique qui réalise la liaison entre les instances du composants et les services techniques correspondants en fonction des propriétés non-fonctionnelles du composant. Les outils de génération [Czarnecki00,Batory03] des conteneurs sont nombreux et variés. Nous

[40] aussi appelés extra-fonctionnels

citerons parmi celles-ci la réécriture, l'injection de code, la réflexion, la programmation orientée aspect[41]. La génération peut être réalisée à la production, lors de la phrase de déploiement ou bien à l'activation du composant. L'application ainsi développée est déployée sur l'infrastructure matérielle de l'organisation qui n'en maîtrise parfois qu'une partie des éléments (serveurs, réseaux, terminaux usager et cartes à puce).

La plupart des modèles industriels définissent une liste figée de besoins non fonctionnels qui sont généralement liés au domaine métier. Plusieurs travaux se sont intéressés néanmoins à rendre les conteneurs flexibles afin de personnaliser la liste des besoins non-fonctionnels [Vadet01,Ducloso2]. La propagation des besoins non fonctionnels dans les compositions hiérarchiques de composants reste cependant un point de recherche ouvert.

III.1.4 Déploiement

La définition de Szyperski insiste également sur la nécessité de considérer le déploiement des composants comme une activité importante de l'approche à composants. Le déploiement logiciel regroupe l'ensemble des activités destinées à rendre le logiciel utilisable dans l'environnement opérationnel d'utilisateur du logiciel.

Le déploiement logiciel est généralement représenté comme un procédé dans lequel s'enchaînent des activités qui se partagent entre le fournisseur (i.e. l'éditeur) du logiciel et le consommateur (i.e. l'entreprise utilisatrice) de celui-ci [Coupaye00]. Ces activités sont la préparation d'une édition, son installation, son activation, sa désactivation, sa mise à jour, son adaptation, sa désinstallation et le repli de l'état associé [Carzaniga98].

Le déploiement d'un logiciel doit respecter les contraintes de réussite et de sûreté [Parrish01]. La contrainte de réussite assure que l'application a été correctement déployée tandis que la contrainte de sûreté assure que l'application déployée n'endommage pas le fonctionnement des autres applications déjà déployées dans le système.

Le déploiement logiciel a longtemps été négligé par les fournisseurs des plateformes d'exécution et par conséquent il a longtemps été réalisé de manière « artisanale ». Désormais, le déploiement prend de plus en plus d'importance dans les domaines du génie logiciel et des intergiciels avec la nécessité de déployer de manière automatique des applications distribuées et multi-paradigmes sur des parcs de grande échelle de machines hétérogènes [OMG03a,OASIS05].

III.1.5 Reconfiguration et adaptation

La maintenance de l'application comporte généralement des opérations de reconfiguration et d'adaptation de composants. La reconfiguration consiste à changer les valeurs des propriétés des composants ou à changer l'architecture de l'application en redéfinissant les connexions entre les composants. L'adaptation consiste en général à remplacer un composant par un autre qui peut éventuellement fournir et requérir des contrats différents du composant remplacé [Kefti04].

Généralement, la reconfiguration et l'adaptation requièrent de déconnecter le composant de l'application, de transférer son état courant vers un nouveau composant et puis de connecter ce dernier aux autres composants de l'application. Ces opérations nécessitent généralement l'arrêt de l'application.

[41] Dont le principe fondateur, la séparation des préoccupations (*separation of concerns*) [Parnas72], peut s'appliquer aux 4 paradigmes de programmation : objet, composant, service et modèle.

La maintenance dynamique consiste à introduire des modifications dans l'application au cours de son exécution sans nécessairement arrêter totalement l'application. Les principaux concepts au problème de la reconfiguration et de l'adaptation dynamiques [Kramer85,Hofmestier93,Magee96] sont décrits plus en détail dans [Buckley03]. La reconfiguration dynamique et l'adaptation dynamique sont des événements de plus en plus fréquents dans le cycle de vie des applications [Allen98] pour lesquels leurs concepteurs recherchent des propriétés comme la sensibilité au changement de l'environnement (*context-awareness*) [Dey00,Coutaz05] ou bien les propriétés autonomiques (*self-**) [Kephart03,Whisnant03].

III.1.6 Acteurs

La définition de Szyperski sous-entend également que plusieurs acteurs spécialisés ayant des préoccupations différentes, interviennent tout au long de cycle de vie d'un logiciel à base de composants.

Le développeur métier écrit le code fonctionnel du composant dans un langage généraliste (C, C++, Java, C#, …). Le modèle préconise parfois quelques conventions de programmation à respecter qui sont souvent liées à des principes et des patrons de conception [Gamma95] tels que la fabrique, l'interception, le mandataire, l'inversion de contrôle…. Il décrit de plus explicitement ses propriétés pour la configuration des composants et ses points d'entrée (facettes et réceptacles).

Le développeur d'application (dont le rôle s'approche de celui d'architecte) construit son application en réalisant une composition de plusieurs composants. La configuration des propriétés fonctionnelles des composants est généralement à la charge du développeur d'application tandis que la configuration des propriétés non fonctionnelles (par exemple, transaction, degré d'isolation, support de persistance…) est le plus souvent partagée entre le développeur d'application et l'administrateur de l'application.

Pour sa part, l'administrateur de l'application est l'acteur qui déploie l'application sur un environnement d'exécution opérationnel donné. Il est également en charge de reconfigurer et d'adapter l'application afin de corriger le mauvais comportement d'un composant, de répondre aux changements affectant l'environnement d'exécution, d'étendre l'application avec de nouvelles fonctionnalités mis à disposition par le producteur, d'améliorer ses performances ou bien de changer ses objectifs initiaux.

III.1.7 Modèles et plateformes

De nombreux modèles de composants ont été proposés, cependant peu sont opérationnels et ont été utilisés à grande échelle dans l'industrie du logiciel. Ces modèles sont parfois généralistes comme Fractal [Bruneton04] et Avalon [Apache99]. Cependant beaucoup sont spécialisés pour des niches applicatives spécifiques comme COM [Box98] et JavaBeans [Sun97] pour les applications sur poste de travail, CORBA Component Model (CCM) [OMG99], Enterprise JavaBeans (EJB) [Matena98] et Spring pour les applications d'entreprise, MBeans [Sun98] pour la supervision des applications et de machines, Common Component Architecture [CCA] pour les applications de calcul scientifique intensif (« *number crushing* »), PECOS [Genssler02], Robocop [Robocop] et SOFA [Plasil98] pour des applications embarquées (*consumer electronics*, …), LwCCM [OMG03b] et RTCOM [Tesanovic04] pour les applications temps réel [Wang05], Service Component Architecture (SCA) [BEA] et SCR [OSGi] pour les applications orientées services.

Les sous sections détaillent les trois modèles qui ont servi de support à mes travaux.

III.1.7.1 Enterprise JavaBeans

Les *Enterprise JavaBeans* (EJB) [Matena98,JSR220] sont la spécification d'un modèle de composants Java distribués destiné au développement de la logique métier des applications d'entreprise. Ce modèle est une des pièces maîtresses de la spécification *Java Enterprise Edition* (JEE)[42] qui regroupe l'ensemble des technologies relatives au développement et au déploiement des applications d'entreprise généralement conçues selon une architecture multi-étages.

Un composant appelé aussi *Enterprise Bean* (EB) est défini par son interface « maison » (*home*) et son interface métier. La première sert essentiellement à créer des instances du composant et à rechercher les instances qui existent déjà, tandis que la seconde est celle qui est offerte par les instances. Le composant peut être de 3 types au moins[43] : les *Entity Beans* qui représentent des sources de données persistantes, les *Session Beans* qui représentent la logique applicative et les *Message Driven Beans* qui représentent des consommateurs ou des souscripteurs de files de message asynchrones JMS.

Le conteneur d'EJB prend en charge les services non-fonctionnels suivants : le contrôle d'accès, la concurrence, les transactions ACID et la persistance. Les transactions sont les transactions « plates » du modèle XOpen/DTP [Bernstein97,Besancenot97] supportées par les principaux moniteurs transactionnels et les principaux gestionnaires de bases de données relationnelles. La persistance est essentiellement assurée grâce à des bases de données purement relationnelles. Bien que le langage support soit orienté objet, les EJB ne permettent hélas pas d'utiliser l'héritage ou le polymorphisme des objets. La spécification EJB 3.0 (JSR220) a été conçue pour simplifier la programmation des *Enterprise Beans* notamment par l'usage accru des annotations comme MicroSoft l'a fait quatre ans plus tôt avec sa plateforme .NET [Tha01]. Les EJB et la spécification JEE sont devenus en quelques années le principal support de développement des applications d'entreprise [Fowley04].

III.1.7.2 Fractal

Fractal [Bruneton04] est un modèle de composants permettant des compositions hiérarchiques. L'atome de composition est appelé composant primitif tandis que la composition est appelée composant composite. Le cadre de conception Fractal fournit une API permettant, entre autres, de définir des types de composants, de fabriquer des instances à partir de ces types, de configurer ces instances puis de les relier entre elles. Un composant Fractal fournit et requiert des interfaces fonctionnelles. Celles-ci sont relatives à la logique applicative. Les facettes sont appelées interfaces serveurs et les réceptacles, interfaces clientes. Le composant fournit également un ensemble d'interfaces de contrôle appelées contrôleurs. La liste des interfaces de contrôle inclut, entre autres, des interfaces dédiées à la gestion du cycle de vie, à la liaison des interfaces clientes et serveurs entre instances de composants, à la configuration interne et au contrôle du contenu pour les composants composites. Plusieurs instances d'un composant Fractal peuvent être créées à partir d'une fabrique associée à un type de composant. La spécification Fractal a fait l'objet de plusieurs implémentations alternatives pour Java et d'autres langages de classes. On peut citer parmi celles-ci, Julia, AOKell, Proactive, Fractalk, Think, FractNet [Seinturier06b], FractScript [Donsez06c], Flone, … Ces implémentations sont pour la plupart basées sur l'injection de bytecode ou l'AOP. Fractal est également la base

[42] Anciennement appelé J2EE

[43] Des types supplémentaires ont été introduits par divers JSRs. C'est par exemple le cas de *Enterprise Media Beans* (JSR86) qui représentent des sources de données multimédias dans une application JEE.

de nombreux travaux de recherche sur l'ingénierie des conteneurs [Seinturier05, Seinturier06a], sur l'adaptation [David05] et bien encore sur les connecteurs [Leclerc04].

III.1.7.3 Service Component Runtime / Service Binder

Service Component Runtime (SCR) est le modèle de composants orienté services qui se focalise essentiellement sur la dynamique des applications destinées à être exécutées sur la plateforme OSGi [OSGi]. Le composant SCR est décrit par le service (facette) qu'il fournit et par les services (réceptacles) qu'il utilise. Un service est un contrat fournissant une ou plusieurs interfaces Java (contrat syntaxique selon [Beugnard99]) qualifiées par des propriétés qui servent au courtage des services (contrat qualité de services selon Beugnard). Les services utilisés sont définis par l'interface requise et une expression de filtre sur les propriétés de courtage. Dans OSGi, les composants sont considérés comme autonomes car l'organisation initiatrice de l'application n'a pas nécessairement de prise sur l'administration de ceux-ci. De plus, le cycle de vie (activité) des composants n'est pas nécessairement synchronisé avec celui de l'application. Ainsi, les services utilisables (c.a.d. vérifiant le contrat de service requis par un réceptacle) peuvent apparaître alors que l'exécution de l'application est entamée et que d'autres qui sont en cours d'utilisation peuvent disparaître temporairement ou définitivement avant la terminaison de l'application. Dans ces conditions, le conteneur du composant prend en charge la liaison automatique avec les services qui apparaissent et la relâche des liaisons des services qui disparaissent. De plus, la liaison d'un réceptacle à un ou plusieurs services peut être considérée obligatoire à l'activation d'un composant. Le conteneur active et désactive[44] le composant en fonction de la présence ou de l'absence des services obligatoires.

Ce modèle à composants qui s'inspire très largement de ServiceBinder [Cervantes04a] reste néanmoins un modèle à composants qui n'adresse pas des services non fonctionnels importants tels que la sécurité, la persistance, la concurrence, … Ce modèle de composant fait actuellement l'objet du JSR 291 et donne le point de départ à d'autres modèles de composants dynamiques comme *iPOJO*, *Enterprise Component Runtime* (ECR), *Spring* ou bien encore *EasyBeans*.

III.2 Recherches et contributions

Ma première rencontre avec les composants fut par le biais du modèle des JavaBeans ajouté aux API de Java pour simplifier le développement des IHM des applications (comme l'antique *BeanBox*). Mon intérêt pour les composants a grandi avec la publication de la spécification du modèle des *Enterprise JavaBeans* qui faisait entrer le langage Java dans les développements d'applications d'entreprise. Malgré les innombrables critiques sur ses imperfections, ce modèle de composants d'entreprise avait le mérite de simplifier la prise en charge des transactions et de la persistance avec des bases de données relationnelles. Je me suis plus particulièrement intéressé à ce modèle pour faciliter l'usage des modèles avancés de transactions dans le cadre de nouvelles applications.

Parmi ces nouvelles applications, la télévision interactive semblait être une voie d'avenir pour l'usage du commerce électronique par le plus grand nombre. Ce domaine imposait des contraintes sur l'ingénierie des applications puisqu'une partie de la logique métier de l'application ne pouvait être exécutée que sur le terminal de l'abonné. Il m'apparaissait nécessaire d'unifier le modèle de composants servant à développer la partie serveur et la

[44] La désactivation entraîne la destruction de l'état contrairement à Fractal ou les EJB pour lesquels le composant est réactivé avec son état antérieur.

partie terminal de l'application. Comme le modèle des EJB correspondait aux besoins de la partie serveur, il me semblait opportun d'adapter le modèle des EJB aux contraintes de la partie terminal.

Mon arrivée dans l'équipe ADELE m'a fait reconsidérer cette hypothèse. Je me suis intéressé aux conteneurs flexibles de composants pour un type d'application émergeant dans l'équipe : celui des applications basées capteurs. En parallèle, la plateforme de déploiement et d'exécution OSGi offrait un socle pour construire des modèles de composants et pour les déployer. L'équipe ayant été pionnière dans le domaine du déploiement, je me suis intéressé alors à l'appliquer aux applications de télévision interactive et aux applications basées capteurs. La fiabilisation des déploiements me semblait un scénario intéressant pour l'application des concepts avancés des transactions.

La première sous-section présente mes contributions dans la prise en compte des transactions avancées de modèles de composants. La sous-section suivante présente mes contributions sur des modèles de composants dédiés. La troisième sous-section présente mes contributions dans le domaine du déploiement.

III.2.1 Composants et Transactions Avancées

Un des apports majeurs du modèle de composants EJB est la simplification de la mise en œuvre de la persistance et de la fiabilité dans le développement des applications d'entreprise. Ses propriétés sont décrites par le développeur dans le descripteur de déploiement du *bean* et le conteneur fait les appels aux API[45] JTS et JDBC en fonction des valeurs du descripteur. Cependant, le modèle des transactions « plates » qu'utilise les EJB se révélait limité dans les contextes applicatifs émergents comme les opérations multi-sites sur le Web (Web service de vente de voyages multimodaux, …), les applications hautes performances de comptabilité des infrastructures de télécommunication ou pour la prise en compte de systèmes de persistance non transactionnels tels que les annuaires LDAP, les dépôts de documents WebDAV, les systèmes de fichiers servant les medias des sites Web ou bien les cartes à puce des usagers (notamment les cartes de fidélité). Les modèles avancés de transactions offraient des solutions à ces contextes. Par exemple, le modèle des transactions emboîtées ouvertes (*open nested transactions*) permet de compenser la réservation d'une location de voiture par une opération d'annulation en cas d'abandon d'une transaction globale portant sur plusieurs services Web des voyagistes [PEPiTA99].

La prise en main de modèles avancés de transactions par le développeur restait néanmoins encore plus difficile que pour le modèle des transactions ACID. Notre proposition fut donc de déléguer la prise en charge des transactions avancées au conteneur EJB. Le développeur n'a plus alors qu'à décrire et à paramétrer le modèle de transactions souhaité. Notre réalisation s'est limitée aux modèles de transactions emboîtées fermées (*close nested transactions*) [Moss81] et aux modèles des transactions emboitées ouvertes [Weikum92] (*open nested transactions*). Nous avons introduit de nouveaux attributs au descripteur de *bean* pour les transactions emboîtées fermées et ouvertes [Bennani02]. Ces attributs permettent de lancer une transaction racine ou une sous transaction fille lors de l'invocation de la méthode ou rendent obligatoire la présence d'un contexte transactionnel racine ou emboîté afin de démarrer une sous transaction fille. Les attributs pour les transactions emboîtées ouvertes sont complétés par une référence vers un *bean* servant à réaliser l'opération de compensation pour la méthode invoquée. Ce *bean* de compensation doit implémenter la même interface métier que le *bean* invoqué : chaque méthode réalise l'opération de compensation de la méthode ayant

[45] Les API JCA, JDO et JPA sont apparues ultérieurement à nos travaux.

la même signature. Une instance de *bean* de compensation est créée lors de l'invocation. Celle-ci n'est invoquée par le gestionnaire de compensation du moniteur transactionnel étendu qu'en cas d'abandon de la transaction mère.

Ces travaux ont fait l'objet de la thèse de Sergiy Nemchenko [Nemchenko04D]. L'expérimentation a été menée sur le conteneur EJB et sur le moniteur transactionnel du serveur J2EE open-source JOnAS[46] par l'ajout des transactions emboîtées fermées et ouvertes dans le cadre du projet ITEA PEPiTA. En parallèle, l'équipe de systèmes distribués dirigée par Frantisek Plasil qui était partenaire du projet proposait un nouveau modèle de transactions avancées baptisées Bourgogne Transactions [Prochazka00]. D'autres systèmes comme ASSET [Biliris94b], ArgunaTS [Houston01] offraient des moniteurs transactionnels étendus cependant aucune intégration n'avait été entreprise pour simplifier l'usage des transactions avancées dans les modèles de composants. Cette recherche m'a permis aussi de soulever de nombreuses questions sur le contrôle de concurrence, sur les compensations en cascade dans les modèles de transactions emboîtées ouvertes et sur leur utilité dans le contexte des services Web qui n'en étaient qu'à leur genèse [Donsez01c,Donsez01d].

III.2.2 Composants dédiés

La technologie des composants prenait pied petit-à-petit pour les développements d'applications d'entreprise. L'apparition des nouveaux contextes applicatifs mettant en œuvre des terminaux nomades, des terminaux de télévision interactifs et des passerelles industrielles, nous a ensuite conduits à étudier les modèles de composants pour le développement des applications liées à ces contextes.

Plusieurs approches ont été envisagées. La première consistait à adapter un modèle de composants établi, aux contraintes environnementales et opérationnelles des contextes ciblés. La seconde était de définir des conteneurs flexibles et d'y ajouter les propriétés non-fonctionnelles requises par le domaine visé. La première approche a le double avantage de disposer des développeurs ayant déjà de l'expérience sur ce modèle, et d'unifier le modèle de composants pour le développement de la partie serveur et de la partie terminal de l'application. Tandis que la seconde approche a l'avantage de définir un modèle de composants taillé sur mesure pour le domaine cible.

La première approche fut appliquée au développement d'applications de la télévision interactive afin de garder un modèle de composants homogènes avec la partie de l'application qui est hébergée sur les serveurs de l'opérateur de télévision interactive. Le seconde approche fut appliquée à celui des applications de collecte de données capteurs exécutées par des passerelles industrielles. Ce choix était guidé par la recherche du modèle de composants minimaliste destinés à des machines fortement contraintes par la mémoire et la CPU.

III.2.2.1 Des Enterprise Beans sur des terminaux de télévision interactive

L'abandon progressif de la télévision analogique pour la télévision numérique satellitaire et terrestre laissait supposer que l'offre de services pouvait se développer fortement [Press93,O'Driscoll00]. Ceux-ci sont généralement associés à un événement audio-visuel (compétition sportive, jeu, …) et de ce fait, ils devraient être développés en un minimum de temps. L'adoption du langage Java [JavaTV] par les industriels de télévision

[46] Ce moniteur qui était au cœur de JOnAS a été par la suite détaché pour devenir le projet JOTM d'ObjectWeb.

interactive (iTV) amenait à repenser les méthodes de développement des applications s'exécutant en partie sur les serveurs des opérateurs iTV et en partie sur les terminaux iTV de leurs abonnés. La programmation à composant nous paraissait une solution propice à la réduction des temps de développement.

Notre approche était de choisir un modèle de composants satisfaisant pour développer la partie serveur et la partie terminal de l'application. Nous avions donc choisi le modèle des EJB bien adapté à la partie serveur et de le spécialiser en fonction des contraintes opérationnelles du terminal. Les deux principales contraintes étaient liées aux modes de communication entre le terminal et le serveur. Le serveur diffuse des données à l'ensemble des terminaux des abonnés par une liaison satellite dite descendante [Sarginson96] et le terminal de l'usager ne peut entrer en communication que rarement avec les serveurs de l'opérateur via une liaison modem bas débit et facturé au temps de connexion.

L'application exécutée par le terminal reste un assemblage standard de *Session Beans* (SB), d'*Entity Beans* (EB) et de *Message Driven Beans* (MDB) colocalisés sur la machine virtuelle du terminal. Les *beans* peuvent néanmoins invoquer des *Session Beans* localisés sur le serveur J2EE de l'opérateur moyennant le coût de communication de la liaison montante. Notre étude s'intéressait à redéfinir les services non fonctionnels requis par les conteneurs d'EB et de MDB dans le contexte de la diffusion de la liaison descendante. Le serveur diffuse périodiquement l'état de tous les EB selon le principe du système de fichier en carrousel de la spécification DSM-CC [Balabanian96] déjà utilisé pour des données comme le Télétexte. Les messages JMS destinés aux MDB sont également diffusés sur ce principe.

Cette idée avait fait l'objet du sujet de DEA de Colombe Hérault [Hérault01M] et la base du projet RNRT COMPiTV. Colombe Hérault financée sur ce projet a poursuivi son DEA par une thèse sur les services non fonctionnels dans les modèles à composants [Herault05D].

III.2.2.2 Composants pour les serveurs embarqués

Les fabricants d'équipements sont de plus en plus prompts à intégrer la fonction de communication dans les équipements qu'ils produisent. Leurs motivations sont d'une part, la maintenance à distance (*e-maintenance*) qui réduit les coûts de SAV et de support et, d'autre part, l'offre de services métier à forte valeur ajoutée associés aux équipements. Ces services sont vendus au client et ils permettent de le fidéliser. Ils s'appuient sur une infrastructure matérielle et logicielle qui réalise la médiation des mesures acquises par les capteurs disséminés dans le monde réel (ie l'usine, l'entrepôt ou le magasin du client du service) jusque dans le système d'information de l'entreprise (client et fournisseur de services). Cette infrastructure est distribuée sur un parc de machines et de plates-formes d'exécution hétérogènes couvant à la fois des passerelles embarquées et des serveurs d'entreprise.

L'approche à composant qui était acquise pour les développements côté serveur d'application n'offrait pas de modèle adéquat pour la partie passerelle des services orientés vers la médiation des mesures acquises sur les capteurs. Nous nous sommes alors focalisés sur la proposition d'un modèle de composants dédié aux développements de ce type de service. Nos relations avec la Direction Technique de Schneider Electric nous a permis de constater que les applications sont essentiellement basées sur la collecte, le filtrage, l'agrégation, le contrôle et l'analyse d'un flot de mesures. L'état de l'art des modèles de composants n'offrait pas vraiment de réponse aux besoins des applications de médiation de mesures.

Notre proposition portait sur la définition d'un modèle à composant dédié au développement des services [Marin05a]. Ce modèle que nous avions initialement baptisé SensorBean, reprend les idées de ServiceBinder (cf. section III.1.7.3) pour automatiser les liaisons entre les composants. Il introduit un connecteur [Mehta00] suivant le patron producteur-consommateur en plus des connecteurs de type client-serveur synchrones ou asynchrone que proposent les EJB, Fractal ou ArchJava [Aldrich02] et des connecteurs de type événementiel qu'ajoute CCM, Dream, PECOS ou bien RoboCop. Le connecteur de type producteur-consommateur est adapté à la construction d'applications traitant des flots de mesures. Comme dans le cas d'un connecteur événementiel, le producteur pousse les données produites vers le consommateur. Cependant, le consommateur a la possibilité de réclamer au producteur de produire une nouvelle donnée. Le connecteur réalise également un contrôle de flot basé sur le contenu pour éviter l'engorgement du consommateur. Par exemple, le producteur peut ne pousser une mesure que s'il constate un écart suffisant entre les mesures produites. Dans le cas des réseaux de capteurs, le producteur représente un ou plusieurs capteurs physiques qui acquièrent des mesures. Les mesures sont poussées vers les consommateurs qui peuvent éventuellement déclencher des actionneurs. Ce connecteur est typé par la nature des mesures qu'il transporte afin de vérifier la compatibilité entre le producteur et le consommateur contrairement aux connecteurs de flux binaires de SOFA [Plasil98] ou de Bip [Emonet06].

La projection de ce connecteur peut être effectuée sur le service WireAdmin[47] d'OSGi et sur le service Data Distribution Service[48] [OMG04] de l'OMG. Dans le contexte du WireAdmin d'OSGi, les connecteurs peuvent être instanciés dynamiquement entre les producteurs et les consommateurs. Les travaux sur les ADL et sur ServiceBinder nous ont également inspirés la définition d'un ADL dédié à la construction d'applications dynamiques basés sur l'échange de flots de mesures. Cet ADL baptisé WireAdminBinder, automatise la connexion et la déconnexion des producteurs et des consommateurs en fonction de leur présence et de leur absence et en fonction des propriétés de courtage comme par exemple, le type de données échangées.

III.2.3 Déploiement de composants

A mon arrivée à Grenoble, l'équipe ADELE qui m'accueillait était une des rares équipes de recherche à s'intéresser au déploiement de logiciels à grande échelle. Les domaines applicatifs que nous étudions dans l'équipe, apportaient de nouveaux terrains d'expérience et des problèmes de déploiement à résoudre.

La télévision interactive requiert le déploiement dynamique « à la demande » des composants sur les terminaux de télévision interactive. Le *edge computing* requiert une gestion autonomique du déploiement des applications J2EE entre les serveurs d'origine et les serveurs frontières. Le déploiement de logiciels à grande échelle qui est susceptible de subir des erreurs, nécessite d'être fiabilisé par des systèmes de reprise. Et enfin, les applications dont les composants sont développés selon des modèles différents, nécessitent d'être déployées sur des environnements d'exécution hétérogènes de façon transparente.

Les quatre sous-sections suivantes présentent mes recherches et mes contributions relatives au déploiement de composants.

[47] Nous avons développé la première implémentation *open-source* de WireAmin pour Oscar/Felix. Celle-ci fait partie de la base de code d'Apache Felix

[48] Nous avons développé la première implémentation *open-source* de DDS en Java. Celle-ci fait partie de la base de code de JacORB [Brose97].

III.2.3.1 Déploiement de composants à la demande

Le déploiement d'applications sur les terminaux de télévision interactive est contraint par plusieurs hypothèses. Premièrement, l'opérateur iTV met à disposition plusieurs centaines d'applications téléchargeables par les terminaux des abonnés. Cependant, ceux-ci ne peuvent pas installer l'ensemble des applications simultanément du fait d'une capacité de stockage[49] limitée. Deuxièmement, les réseaux satellitaires et terrestres à diffusion sont utilisés pour diffuser régulièrement les fichiers contenant le binaire des applications selon le principe du système de fichier en carrousel déjà cité.

Les applications iTV, qui sont appelées XLet, sont conçues selon l'API JavaTV [JavaTV]. Elles sont constituées d'un graphe de classes qu'il est nécessaire de charger (déployer) entièrement avant le démarrage effectif de l'application. Quand ce graphe comporte plusieurs centaines de classes, le démarrage nécessite alors plusieurs dizaines de secondes.

Notre proposition appelée OSGiTV [Chomat03] fut donc de déployer les applications selon l'approche à composant orienté services de ServiceBinder (cf Figure III-1). Celle-ci permet un déploiement incrémental des composants accélérant ainsi les temps de démarrage. Les composants sont isolés par les interfaces et les classes qui forment les contrats de service de ServiceBinder pour éviter des dépendances de code entre les implémentations. Ainsi le binaire d'un composant n'est déployé que lorsqu'un composant client se lie à une de ses facettes. La plateforme d'exécution du terminal iTV comporte un gestionnaire de déploiement qui est guidé par au moins 3 sources d'ordre d'activation immédiate des composants principaux des applications : l'opérateur peut envoyer ce type d'ordre pour des applications comme la « grille de programme » (EPGXlet sur la figure), l'abonné peut au travers de l'application « grille de programme » demander le démarrage d'une application, et enfin, l'insertion d'une carte à puce d'abonné peut déclencher le déploiement des services cartes. Dans l'exemple de la Figure III-1, le service GamblingCardProxy est un CardService OCF [Hansmann00] dédié à l'application TCasinoXLet. Celui-çi est déployé dynamiquement quand la carte est insérée et que la liste des applications encartées est connue.

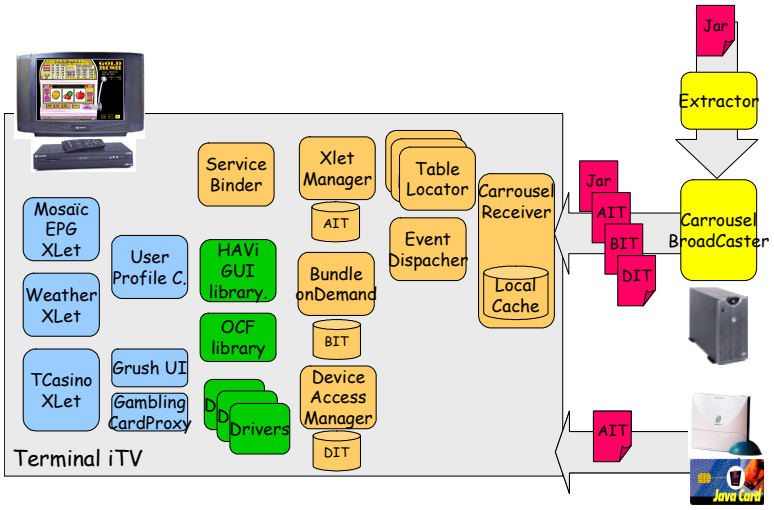

Figure III-1: Plateforme OSGiTV pour le développement à la demande d'applications de télévision interactive.

[49] Bien que les terminaux haut de gamme incluent un disque dur de grande capacité, le problème de la mise à jour reste ouvert.

La plateforme OSGiTV s'intéressait à déployer à la demande non seulement les applications mais également les constituants de l'environnement d'exécution. Ces derniers constituent le *firmware* dont la mise à jour requiert encore le redémarrage de la plateforme et donc l'interruption du fonctionnement.

III.2.3.2 Déploiement autonomique

Le *edge computing* [Akamai01,Weihl04] est un modèle d'infrastructure dans lequel l'opérateur ou le distributeur de contenu [Saroiu02,Vakali03] rapproche les ressources de calcul et de stockage des utilisateurs finaux afin de soulager son réseau et d'améliorer les temps de réponse aux requêtes des usagers sur le site d'un fournisseur de service [Vakali03]. Ce dernier bénéficie alors de l'infrastructure à la demande de l'opérateur quand la montée en charge ne peut pas être absorbée par ses propres serveurs. Dans le cas d'applications JEE, rapprocher les ressources de calcul revient à déployer en partie ou totalement une application à la demande du client (cf. section III.2.3.1) sur une machine du point d'accès proche de l'usager. Cependant, ce déploiement peut être pénalisant s'il est systématique.

L'informatique autonomique [Horn01,Kephart03] nous a semblé être une voie prometteuse en vue de diminuer les coûts de maintenance et d'augmenter la qualité de services d'un parc de serveurs de proximité (*edge servers* en anglais). Notre étude portait sur l'intégration d'une boucle de contrôle autonomique pour le déploiement d'applications JEE entre les serveurs d'origine des fournisseurs et les serveurs de proximité de l'opérateur [Désertot05b]. Les propriétés autonomiques sur lesquelles nous nous sommes focalisés étaient l'auto-configuration et partiellement l'auto-réparation. Plusieurs stratégies de déploiement [Désertot06b,Désertot06d] basées sur des règles Evénement-Condition-Action [Kephart04], ont été définies, simulées et expérimentées pour une application de e-Commerce semblable à celle du TPC-W [TpcW]. Notre approche reste néanmoins applicable (et appliquée) à d'autres contextes d'administration de grille de calcul [Li05]. Cette étude avait fait l'objet du DEA de Clément Escoffier [Escoffier04M].

III.2.3.3 Fiabilisation du déploiement

Lors du déroulement du procédé de déploiement, des erreurs et des situations exceptionnelles peuvent mettre le système dans un état incohérent. Il est nécessaire d'analyser l'état du système et d'agir pour le remettre dans un état cohérent. La plupart des systèmes de déploiement réalisent cela de manière ad-hoc et choisissent généralement de revenir à l'état antérieur.

Nous nous sommes intéressés à rechercher les modèles, les outils et les méthodes permettant de traiter les problèmes de reprise sur erreur dans les systèmes de déploiement. Les modèles de transactions ont constitué le point de départ de notre réflexion. Notre modèle de déploiement transactionnel [Marin04] s'inspire des modèles de transactions imbriquées ouvertes. Une transaction de déploiement peut être aussi bien défaite que compensée en cas d'erreurs. Le choix entre l'un ou l'autre qui fait l'objet d'une stratégie est réalisé dynamiquement en fonction du contexte du déploiement. Par exemple, en cas de panne de communication, une stratégie peut chercher à retenter un téléchargement en échec car 95% des composants ont été téléchargés plutôt que de défaire l'ensemble.

Ce système de déploiement a été prototypé au moyen de procédés exprimés dans le langage APEL [Dami98] pour exécuter les opérations élémentaires de déploiement, leurs opérations de contingence, ainsi que les opérations de compensation. Cette étude avait fait l'objet du DEA de Cristina Marin [Marin04M].

III.2.3.4 Déploiement de composants hétérogènes

Le contexte de services basés capteurs (cf. section III.2.2.2) introduit la nécessité de déployer des applications dont les composants sont développés selon des modèles différents, sur des environnements d'exécution hétérogènes. Les composants de collecte sont par exemple des bundles OSGi [OSGi] ou des DLL OPC (*OLE for Process Control*) [Opc98] tandis que les composants d'analyse sont des Enterprise Beans JEE. Les cycles de vie de composants comme les opérations élémentaires diffèrent d'un modèle à l'autre.

Nous avons proposé un modèle de déploiement adressant l'hétérogénéité des plateformes d'exécution. Ce modèle s'appuie sur la définition de dépendances et de contraintes extensibles pour calculer les plans de déploiement. Ce modèle est moins raffiné que le modèle [OMG03a] spécifié par l'OMG pour les applications distribuées à base de composants. Cependant, il traite de la reconfiguration dynamique de l'application qui n'est pas abordée par l'OMG D&C. Ce modèle fut l'objet d'une de nos contributions au projet ITEA Osmose [Osmose, Lestideau05].

III.3 Conclusion

Ce chapitre a présenté les résultats de nos travaux sur les modèles à composants. Le support des besoins non fonctionnels dans ces modèles a été le fil conducteur de ces travaux.

La section III.2.1 a présenté l'intégration de services transactionnels avancés dans un modèle de composants d'entreprise, celui des Enterprise JavaBeans.

La section III.2.2 a présenté la poursuite de nos travaux vers la recherche de modèles de composants et de services non-fonctionnels dédiés à des contextes applicatifs particuliers. Ces contextes étaient les terminaux de télévision interactive et les passerelles industrielles dédiées à la collecte de mesures acquises par des réseaux de capteurs.

La section III.2.3 a présenté nos travaux sur le déploiement d'applications à composants dans les contextes précédemment cités. Nous nous sommes également intéressés à appliquer les principes transactionnels pour fiabiliser le déploiement d'applications.

III.3.1 Bilan du domaine

La recherche sur les composants est active depuis une décade. L'adoption de cette approche est de plus en plus acquise pour les développements d'applications d'entreprise JEE. Cependant, un grand nombre de développeurs JEE hésitent à se lancer dans l'utilisation des *Enterprise Beans* (par ignorance souvent). L'introduction des EJB 3.0 devrait peut être conduire un grand nombre à franchir le pas. De son coté, la plateforme .NET reste ancré sur les concepts des objets malgré l'ajout au langage (CLI) [Ecma335] des événements, la définition d'unités de conditionnement et de plusieurs technologies de projection de persistance objet-relationnel ou objet-XML.

Ce constat nous a motivés pour proposer une implémentation de Fractal [Escoffier05, Escoffier06] pour la plateforme .NET. L'idée première de cette implémentation était de mesurer le degré d'isomorphisme qui pouvait y avoir entre 2 implémentations d'un même modèle de composants basés sur l'AOP et dans 2 environnements d'exécution différents, à savoir Java et .NET [Seinturier06b]. Parallèlement, on peut constater que la programmation orientée aspect est loin d'être adoptée par la communauté des

développeurs. Cette méthodologie semble être une technologie principalement utilisée pour la génération de conteneurs des composants.

Une tendance pour l'approche à base de canevas est également de limiter l'adhérence du code fonctionnel avec un environnement d'exécution afin de faciliter la portabilité entre environnements d'exécution (JavaCard, J2ME, OSGi, J2SE et J2EE par exemple) et simplifier la mise au point (test unitaire, déboggage, …). L'émergence et le succès grandissant de canevas comme Spring ou iPOJO, basés sur les POJOs (*Plain Old Java Objects*) en sont une preuve. Parallèlement, la plupart des modèles existants migrent vers l'utilisation des annotations[50] [Newkirk02,Cepa04,Pawlak05b] pour définir les composants directement dans le code fonctionnel. C'est par exemple le cas de EJB3.0 [JSR220] avec JEE 5.0 ou bien des Fraclets [Rouvoy06] pour le modèle Fractal.

Cette décade a été également celle de la prise en compte des problèmes de déploiement de logiciels. Le déploiement est une préoccupation prenant de plus en plus d'importance pour les concepteurs de canevas d'exécution comme le prouve, par exemple, la bataille d'experts autour des JSR 277 et 291 sur la modularité dans la plateforme Java. Cependant, ce domaine reste encore ouvert à de nombreuses recherches. Par exemple, la fiabilisation du déploiement (installation et reconfiguration) que nous avions abordée (cf. section III.2.3.3) est un chantier quasi-vierge en ce qui concerne les composants hiérarchiques ou bien les services dynamiques.

Enfin, la résurgence [Nierstrasz05] des langages dynamiques (souvent appelé langages de script comme JavaScript, Ruby, Groovy, PHP) [Ousterhout98] pourrait bien influer et dynamiser l'adoption de modèles de composants. En fait, ces langages qui sont pratiqués par une immensité de développeurs experts ou novices, ont la mauvaise réputation de rendre la mise au point et la maintenance du logiciel développé très difficiles. L'approche à composant apporte à ces langages la modularité requise pour le développement de très gros logiciels [Donsez06c].

III.3.2 Projets et collaborations

Le projet PEPiTA (Platform for Enhanced Provisioning of Terminal-independent Application) proposait d'étendre les spécifications J2EE afin de prendre en compte de nouveaux services de persistance, des clients légers et nomades tels que les cartes à puce, les décodeurs de TV numérique et les téléphones cellulaires. Le projet propose une implantation open-source de ces spécifications sur la base du serveur EJB JOnAS initialement développé par Bull. Notre contribution au projet portait sur l'extension des transactions aux transactions imbriquées strictes (Close Nested) et ouvertes (Open Nested) afin d'intégrer des fonctionnalités de type Workflow et BTP (Business Transaction Processing) aux EJB (cf. section III.2.1). Nous avons également fourni plusieurs démonstrateurs utilisant des terminaux WAP et les dernières générations de cartes à puce Bull CP8. Ce projet qui était financé par le programme Eureka/ITEA de 1999 à 2001, a financé la thèse de Sergiy Nemchencho, le stage post doctoral d'Emmanuel Adam et de 6 stagiaires de Master 2. Les partenaires du projet PEPiTA étaient Bull, Evidian, Alcatel, France Telecom, Bantry Technologies, Université de Grenoble, Université de Louvain, l'Université de Valenciennes et l'Université Charles de Pragues. Le projet PEPiTA s'est poursuivit avec le projet IMPACT financé par le RNTL 2001.

Le projet COMPiTV [COMPiTV] financé par le programme RNRT 2001 s'intéressait à proposer une plateforme à composants CCM au dessus d'OSGi ayant des services non-fonctionnels adaptés au mode de communication par diffusion. Le DEA de Colombe

[50] Les annotations sont présentes depuis la version 1.0 du canevas .NET. Elles nous été introduites que depuis la version 5.0 de la plateforme Java et l'édition J2ME ne supporte pas pour l'instant.

Hérault (cf. section III.2.2.1) fut à l'origine de ce projet. Il a financé la thèse de Colombe Hérault co-encadrée par Sylvain Lecomte et Nadia Bennani. Dans ce projet, la contribution de l'équipe portait sur la proposition de déploiement à la demande de composants (cf. section III.2.3.1). Notre choix s'était porté sur ServiceBinder plutôt que CCM. Les partenaires de ce projet étaient Canal + Technologies, Gemplus, l'Université de Valenciennes, l'Université Lille 1 et l'Université Grenoble 1.

Le partenariat avec l'équipe de Philippe Lalanda, de la Direction Technique de Schneider Electric, démarré en 2002, portait sur la définition de modèles de composants dédiés pour les passerelles industrielles (cf. section III.2.2.2) [Baude06]. Ce partenariat a permis de lancer le projet RNRT PISE (Passerelle Internet Flexible et Sécurisée) suite aux premiers résultats des DEAs de Mikaël Désertot et de Stéphane Chomat. Le projet PISE a finalement démarré en Septembre 2004 et il se termine en Décembre 2006. Les partenaires sont Schneider Electric, France Telecom R&D, TRIALOG, l'INRIA Sophia-Antipolis et l'équipe ADELE du LSR.

Le projet ITEA OSMOSE [OSMOSE] s'intéressait à la définition de services non fonctionnels pour les plateformes d'exécution J2EE, CCM et OSGi. La contribution de l'équipe ADELE au projet portait sur la plateforme OSGi OSCAR et sur les outils de déploiement (cf. section III.2.3.3, cf III.2.3.4). Nourredine Belkhatir et moi animions le groupe de travail relatif au déploiement.

Ces projets ont été l'occasion de collaborer avec les équipes R&D des sociétés Gemplus, Canal + Technologies, Bull, Bull CP8, Schneider Electric et France Telecom.

Les étudiants en DEA que j'ai encadré ou co-encadré sur les sujets relatifs aux composants étaient: Colombe Hérault, Marie Thilliez, Stéphane Chomat, Mikaël Desertot Cristina Marin, Olivier Ducas et Clément Escoffier. Les étudiants encadrés pendant leurs stages de DESS étaient Stéphane Chomat et Sabrina Lefièvre.

III.3.3 Enseignements

Ces travaux de recherche m'ont permis de monter de nouveaux cours de Master 2 Professionnel sur les systèmes distribuées, les technologies de services Internet (J2EE, .NET, Web Services) et les services M2M. Ces cours sont principalement regroupés dans les projets eCOM, GICOM [Boyer02] et M2M du Master 2 Professionnel Génie Informatique de l'institut IMA (Informatique et Mathématiques Appliqués) et la troisième année de la filière RICM de PolyTech'Grenoble.

III.3.4 Publications

[Baude06] Francoise Baude, André Bottaro, Jean-Michel Brun, Antonin Chazalet, Arnaud Constancin, Didier Donsez; Leven Gurgen, Philippe Lalanda, Virginie Legrand, Vincent Lestideau, Sylvain Marié, Cristina Marin, Alain Moreau, Vincent Olive : Extension de passerelles OSGi pour la grande échelle: Modèles et outils. Actes électroniques de l'Atelier de travail OSGi, Ubimob'06, 3e Journées Francophones Mobilité et Ubiquité, 5 septembre 2006, Paris, 5 pages, https://hal.archives-ouvertes.fr/hal-00097266

[Bennani02] Nadia Bennani, Thierry Delot, Sylvain Lecomte, Sergiy Nemchenko, Didier Donsez : Advanced Transactional Model for Component-Based Model. Actes de la conférence IEEE ISADS (International Symposium on Advanced Distributed Systems) 2002, Guadalajara, Mexique, ISBN 970-27-0358-1, pp171-182

[Boyer02] Fabienne Boyer, Sébastien Chassande-Barrioz, Didier Donsez, David Féliot, Sacha Krakowiak. GICOM : un atelier pour l'expérimentation des technologies de systèmes distribués d'entreprise. Actes du colloque Technologies de l'Information et de la

Communication dans les Enseignements d'ingénieurs et dans l'industrie, Lyon 13-15 novembre, 2002, pp 423-424. http://docinsa.insa-lyon.fr/tice/2002/ca/ca102.html

[Chomat03] Stéphane Chomat, Didier Donsez : OSGiTV: une plateforme de déploiement d'applications de télévision interactive basée sur OSGi. Actes de Conférence Française des Systèmes d'Exploitation (CFSE'03), La Colle sur Loup, France, Octobre 2003

[COMPiTV] Livrables du projet COMPiTV (Plateforme à composants pour les services de télévision interactive), http://www.rnrt.org/rnrt/projets/res_01_47.htm

[Désertot05b] Mikaël Desertot, Clément Escoffier, Didier Donsez : Autonomic Management of J2EE Edge Servers. Actes de 3rd International Workshop on Middleware for Grid Computing, Co-located with Middleware 2005, Grenoble, France, November 28-29th 2005

[Désertot06b] Mikael Desertot, Clément Escoffier, Philippe Lalanda, Didier Donsez : Autonomic Management of Edge Servers. Actes de International Workshop on Self-Organizing Systems, New Trends in Network Architectures and Services IWSOS'06, 18-20 September 2006, Passau, Germany, LNCS 4124, ISBN: 3-540-37658-5, pp 216-229, http://dx.doi.org/10.1007/11822035_18

[Désertot06d] Mikaël Désertot, Clément Escoffier, Didier Donsez : Towards an Autonomic Approach for Edge Computing. Concurrency and Computation: Practice and Experience, John Wiley & Sons, Novembre 2006 (publication papier début 2007), 16 pages, http://www3.interscience.wiley.com/cgi-bin/abstract/113466340/ABSTRACT

[Donsez01c] Didier Donsez, Pierre Yves Gibello, Advanced Transaction Models for EJB and Web Services. Présentation invitée à la Conférence ObjectWeb, ENST, Paris, France, 30-31 Octobre 2001

[Donsez01d] Didier Donsez : Les protocoles avancés: transaction BTP. Séminaire IN'TECH Thème Web Services, INRIA Rhône-Alpes, Montbonnot, France, 5 Décembre 2001, http://stream-serv.inrialpes.fr/intech/web_services/d_donsez.ram

[Donsez06c] Didier Donsez : Usage des langages de script pour des composants adaptables. Actes des Journées Composants 2006 (JC2006), 4-6 octobre 2006, Perpignan, France, pp 70-78.

[Ketfi02] Abdelmajid Ketfi, Humberto Cervantes, Richard S. Hall, Didier Donsez : Composants Adaptables au dessus d'OSGi. Actes des Journées Systèmes à Composants Adaptables et extensibles, Grenoble 17-18 octobre 2002.

[Lestideau05] Vincent Lestideau, Didier Donsez : On-Demand Service Installation and Activation with OSGi. Présentation à la Conférence ObjectWeb, INRIA, Lyon, France, Janvier 2005. https://wiki.objectweb.org/ObjectWebCon05/attach?page=Sessions%2Fondemand_osgi.pdf

[Marin04] Cristina Marin, Didier Donsez, Noureddine Belkhatir : Gestion transactionnelle de la reprise sur erreurs dans le déploiement. Actes de la 1ère conférence francophone sur le Déploiement et la Reconfiguration de logiciels, (DECOR 2004), pp199-210, Grenoble, 28-29 Octobre 2004, ISBN 2-7261-12776-5.

[Marin05a] Cristina Marin, Mikaël Désertot, Didier Donsez : SensorBean : Un modèle à composant pour les services basés capteurs. Actes des Journées Composants (JC'05), Le Croisic, France, 5-8 Avril 2005

[OSMOSE] Livrables du projet OSMOSE (Open Source Middleware for. Open Systems in Europe), program ITEA (Information Technology for European Advancement), http://www.itea-osmose.org/

[PEPiTA] Livrables du projet PEPiTA (Platform for Enhanced Provisioning of Terminal-independent Applications), program ITEA (Information Technology for European Advancement), http://pepita.objectweb.org/

[Seinturier06b] Lionel Seinturier, Nicolas Pessemier, Clément Escoffier, Didier Donsez : Towards a Reference Model for Implementing the Fractal Specifications for Java and the .NET Platform. Fractal CBSE workshop at ECOOP 2006, Nantes, 3 Juillet 2006, à paraître dans Springer Verlag LNCS Serie.

Chapitre IV

Services et Dynamicité

La programmation orientée composant représente une étape importante dans le domaine du génie logiciel. Elle se focalise principalement sur le développement du logiciel par la composition de composants réutilisables. L'application est sous le contrôle d'une administration qui peut décider de la reconfiguration de son architecture.

L'émergence d'applications utilisant plusieurs entités logicielles contrôlées par plusieurs domaines d'administration disjoints, allaient changer la manière de concevoir et de construire les applications. La nouvelle approche de développement appelée Architecture Orientée Service (AOS) propose la construction d'une application par coordination de ces entités appelés services.

Cette approche énonce un certain nombre de propriétés que doivent posséder les services utilisés au sein d'une application orientée service. Le couplage faible, la substituabilité et la liaison retardée sont des propriétés souvent citées pour démarquer les services des composants. Cependant, une propriété discriminante est qu'il n'existe pas toujours d'administration centrale des services engagés dans la réalisation d'une application. Ainsi, l'administrateur d'une application ne maîtrise souvent ni le cycle de vie (disponibilité / indisponibilité) ni la qualité des services qui sont a priori régis par des tiers. De plus, l'application doit pouvoir être sensible aux changements de cycle de vie des services qui interviennent en cours d'exécution. L'expression de la variabilité dynamique au sein de l'application a motivé en grande partie mes recherches durant ces 6 dernières années.

Ce chapitre présente nos travaux relatifs aux plateformes dynamiques de services. La section 1 rappelle les grands axes de recherches menés autour des plateformes dynamiques de services. La section 2 présentera mes contributions dans ce domaine. La section 3 dresse un bilan sur le domaine et sur mes résultats.

IV.1 Problématiques et Etat de l'art

L'Architecture Orientée Service (AOS)[51] est une nouvelle approche de conception des applications par coordination d'entités logicielles indépendantes appelées services. Cet état de l'art présente les principes de cette approche et les propriétés attendues pour des services. Nous nous intéressons à la dynamique des services qui apparaissent et disparaissent au cours de l'exécution de l'application et nous listerons plusieurs plateformes d'exécution des services dynamiques.

IV.1.1 Historique

Le besoin de faire collaborer les systèmes d'information des partenaires commerciaux ou des organisations existent depuis deux décennies au moins. L'EDI (*Electronic Data Interchange*) [Langlois97] fut une première initiative à grande échelle de normaliser à la fois la technologie d'échange et la sémantique des messages échangés pour un domaine d'application particulier (assurance maladie, construction automobile, …). La faible connectivité des entreprises et la complexité de mise en œuvre des technologies sous-jacentes ont réservé longtemps l'EDI aux 20000 plus grandes entreprises mondiales et leurs fournisseurs.

La généralisation du protocole HTTP pour le transport de documents et pour la soumission de requêtes ainsi que la spécification du langage XML[52] [Box01]] pour la définition de documents structurés ont favorisé le développement des échanges informatiques des documents contractuels entre les entreprises. En s'inspirant du précurseur XML-RPC, le W3c jetait les bases des services Web avec la spécification SOAP (Simple Object Access Protocol) [W3C98] et avec WSDL (Web Services Description Language) [W3C00][53]. Un service Web représente généralement un point d'accès au système d'information d'une entreprise pour les systèmes d'information de ses clients et de ses fournisseurs.

Initialement prévues pour la collaboration inter-organisation, les technologies de services Web ont rapidement été appliquées à l'EAI (Enterprise Application Integration) [Hohpe03] qui s'intéresse à la collaboration[54] des différents systèmes d'information d'une même organisation. Le succès est tel que la plupart des technologies EAI propriétaires ont quasiment été balayées du marché.

Parallèlement, l'idée d'une informatique ubiquitaire [Weiser91] commençait à se matérialiser avec les intergiciels Jini [Arnold99] et UPnP (Universal Plug and Play) [Jeronimo03] pour l'interconnexion des équipements de l'environnement de l'usager. Le monde des équipementiers s'accorda sur la nécessité de définir des interfaces standards au dessus de ces protocoles. La notion de service permettait de garantir la substitualité des équipements provenant de différents équipementiers. Le service rempli par un équipement sert alors à la fois à sa découverte dans le réseau et à sa manipulation.

[51] En anglais, *Service-Oriented Architecture* (SOA) ou bien *Service-Oriented Computing* (SOC).
[52] XML (eXtended Markup Language) est une spécification du W3C qui se basait sur une simplification drastique de la norme ISO 8879:1986 SGML (Standard Generalized Markup Language) défini par Charles Goldfarb en 1974 [Goldfard91].
[53] Bien que SOAP et WSDL aient la faveur des grands éditeurs IT, les praticiens industriels se sont également tournés vers des technologies plus simples telles que REST [Fielding00] pour invoquer et décrire des services Web [Hinchcliffe05]. Ces dernières ont notamment un certain succès auprès des développeurs AJAX et Flash.
[54] Le terme utilisé est le plus souvent « intégration ».

IV.1.2 Définition

L'architecture orientée service (AOS) [Bieber01, Sillitti02, Huhns05] définit le service comme brique de base de construction des applications. Cette entité logicielle qui peut éventuellement être gérée par une organisation tierce, offre des fonctionnalités décrites contractuellement. Le contrat[55] qui peut être syntaxique, sémantique, comportemental ou/et qualitatif [Beugnard99] est généralement implémenté par plusieurs fournisseurs de service.

Une implémentation d'un service est rendue disponible auprès des applications par sa publication (ou enregistrement) dans un annuaire de services. L'opération de courtage est l'opération de recherche et de sélection des services pertinents dans l'annuaire pour le client de services en fonction d'un contrat donné. Cette opération est confiée à un tiers appelé courtier qui joue l'intermédiaire entre les implémentations de services et leurs usagers. L'AOS recommande de réaliser le courtage juste au moment de l'utilisation (juste à temps)[56] : toute implémentation de service respectant le contrat recherché peut ainsi être choisie pour réaliser le service. L'application est généralement construite comme une orchestration ou une chorégraphie [Peltz03] des services sélectionnés. Celle-ci échange des messages de requêtes et de réponses synchrones ou asynchrones avec eux. Cette application peut elle-même implémenter un contrat de service et participer à son tour, à une chorégraphie.

Les services sont souvent considérés comme autonomes ou faiblement couplés car l'organisation qui contrôle l'application initiatrice, n'a pas de prise sur les services utilisés en dehors de ce que définit le contrat et/ou du contrôle de la qualité de service négociée par les contractants quand les services sont fournis par des organisations tierces. Les propriétés non-fonctionnelles (sécurité, fiabilité, …) font partie de la qualité de service négociée. La mise en œuvre peut requérir l'usage de services non-fonctionnels associés. Par exemple, la fiabilisation d'une transaction opérant sur plusieurs services Web nécessite un service Web de coordination [Little03].

L'usage d'un service peut faire l'objet d'un accord de niveau de service (*service-level agreement*) entre l'usager d'un service et le fournisseur du service[57] [Miller87]. Une fois que l'accord est négocié entre les parties, les contractants s'engagent à utiliser et à fournir le service selon les termes de l'accord, à gratifier l'usage du service et à subir des pénalités en cas de non-respect des termes de l'accord. L'accord est généralement négocié puis conclu lors du courtage et sa gestion (*service-level management*) peut-être confiée à un tiers dans un souci d'équité entre usager et fournisseur. Ce tiers appelé moniteur [Keller02] ou surveillant de service est chargé de la surveillance, de la facturation des usages ainsi que de l'application des pénalités [Leopoldi02, Oldham06].

IV.1.3 Dynamique des Services

Un service n'a pas nécessairement un cycle de vie (activité) synchronisé avec celui de l'application initiatrice. En effet, de nouveaux services pertinents peuvent apparaître alors que l'orchestration menée par l'application est entamée. Il en va de même pour des services réservés ou en cours d'utilisation qui peuvent disparaître temporairement ou

[55] Le concept de contrat est apparu avec [Helm90,Meyer92] pour les langages objets afin d'imposer la fiabilité et la sûreté de fonctionnement. Les contrats dans le langage Eiffel [Meyer91] et dans des extensions à Java [Lackner02] s'expriment en termes de pré-conditions sur les paramètres des méthodes, post-conditions sur leurs retours et d'invariants de classe.

[56] La littérature parle aussi de liaison retardée (*late-binding*).

[57] Le nombre des participants peut être supérieur à 2 et l'accord peut porter sur l'usage conjoint de plusieurs services.

définitivement avant la conclusion de l'application. Ces événements sont de moins en moins considérés comme des événements exceptionnels qui conduisent à l'abandon de l'application. Par exemple, un système d'exploitation considère comme normal l'insertion ou l'arrachement d'une clé USB possédant un système de fichiers et réagit en fonction. L'architecte de l'application (ou de l'intergiciel) doit donc se préoccuper de ces événements s'il souhaite en tirer les bénéfices comme par exemple la sensibilité au contexte, la mise à jour incrémentale, …

La prise en charge de la dynamique reste néanmoins une préoccupation secondaire mal appréhendée par les architectes qui se focalisent surtout sur la partie fonctionnelle de l'application. Par exemple, un développeur de plugin Eclipse n'est guère concerné par le fait que les plugins dont dépend le sien puissent être mis à jour dynamiquement alors qu'il est nécessaire d'en tenir compte pour un fonctionnement cohérent après les mises à jour. L'architecture orientée service dynamique (AOSD) se focalise sur l'expression des variations dans l'architecture de l'application en fonction de la dynamique des services qui la composent ou peuvent la composer. Un des objectifs majeurs est d'assurer un fonctionnement en continu de l'application en minimisant l'interruption du service rendu. Les travaux dans ce domaine se rattachent au problème général de la reconfiguration et de l'adaptation introduit dans le chapitre précédent (cf. section. III.1.5). Les langages d'expression d'architecture [Magee96] traitent un spectre plus ou moins large des comportements possibles de l'architecture face aux variations notifiées.

IV.1.4 Modèles et plateformes

Les services Web ont été les principaux promoteurs de l'architecture orientée service. Ils sont souvent synonymes dans une grande partie des travaux. Il existe cependant un grand nombre de plateformes alternatives qui se réclament de cette approche. Cette sous-section est consacrée à la présentation des plateformes orientées services qui ont servi de base à nos travaux.

IV.1.4.1 Services Web

Les services Web ont réussis à être la plateforme consensuelle d'interopérabilité entre applications réparties, là où CORBA et Java RMI ont échoué. Les services reposent principalement sur des technologies basées sur XML pour la structure et le contenu de messages échangés entre services (SOAP), pour la description (fonctionnelle et non-fonctionnelle) des services (WSDL), pour la découverte et le courtage des services (UDDI, WS-Discovery) et pour leurs orchestrations (BPEL). L'ensemble de ces technologies qui sont appelées WS-*, s'articulent néanmoins de manière plus au moins cohérentes [Motahari-Nezhad06].

Bien que faiblement couplés les uns aux autres, les services Web peuvent être utilisés avec des garanties de qualité de service associées aux propriétés non-fonctionnelles classiques comme la fiabilité, la sécurité, … WS-Policy [W3C02] permet d'exprimer des politiques d'usage du service sous forme d'assertions. WS-Agreement [Oldham06] et d'autres [Lamanna03,Ludwig03,Dan04] s'intéressent plus particulièrement à la définition d'objectifs de niveau de service pour la création de contrats, d'accords et de garanties à partir des offres, entre le fournisseur de service et l'usager.

Les services Web autorisent une conception flexible d'applications selon plusieurs modèles d'interaction entre les services. Le modèle requête-réponse synchrone peut côtoyer le modèle d'envoi asynchrone de messages acheminés vers les services concernés par leurs contenus. Alors que le modèle synchrone est employé principalement pour les applications transactionnelles ou interactives, le modèle asynchrone s'applique plus

facilement à des procédés déclenchés par l'arrivée de messages transportés par un bus à messages. Ce dernier modèle est préconisé par les *Enterprise Service Bus* (ESB) [Chappell04] pour la conception d'applications à grande échelle.

IV.1.4.2 Services ambiants

L'informatique ambiante[58] s'est très tôt inspiré de l'architecture orientée service pour décrire les équipements comme un ensemble de services faiblement couplés à l'application usager. Cette application est un point de contrôle tel qu'une télécommande spécifique ou générique [Donsez06b,Donsez07]. Un grand nombre de technologies de découverte dynamique des équipements dans des réseaux ambiants ont été définies [Zhu05, Edwards06]. Cependant, seuls Jini et UPnP couvrent fonctionnellement la définition de services. Jini et UPnP supportent l'ajout et le retrait dynamique d'équipements du réseau ambiant SOHO (*Small Office Home Office*) [Marple04].

Jini [Arnold99] est le représentant emblématique des plates-formes de développement d'applications distribuées et spontanées. Les équipements Jini implémentent des services dont le contrat est une interface Java qualifiée par des paires d'attributs-valeurs textuelles. La liaison à un service est limitée par la durée du bail négocié à la connexion ou renégocié. Plusieurs canevas ont été construits au dessus de Jini comme par exemple Rio qui gère les niveaux d'accord de services ou bien les JavaSpaces [Freeman99] qui offrent un modèle de communication par échange transactionnel d'objets qui s'inspire largement de Linda [Carriero89].

Jini a été progressivement supplanté par la spécification UPnP [UPnP,Jeronimo03] qui a le soutien de l'industrie des équipements. Un équipement UPnP est un ou plusieurs services Web « ancestraux » (c'est-à-dire antérieurs à SOAP 1.1 et WSDL 1.0). Ils sont limités par le typage des paramètres des actions. Ces services disposent aussi de variables d'état qui peuvent notifier leurs changements de valeur. Le forum UPnP standardise quelques profils d'équipement type dans un environnement SOHO. Le successeur d'UPnP, *Devices Profile for Web Services* (DPWS) [Jammes05], s'appuie sur les derniers standards WS-* afin d'unifier l'infrastructure d'accès aux équipements avec celle des services Web utilisés par les fournisseurs de services à valeur ajoutée externes au domicile.

IV.1.4.3 OSGi

La spécification OSGi définit un canevas de déploiement et d'exécution dynamique de services colocalisés au sein d'une machine virtuelle Java. Définie initialement pour les passerelles domotiques et immotiques, cette spécification est utilisée également pour les applications à plugin telles qu'Eclipse et Lotus et pour les noyaux de plusieurs serveurs JEE (JOnAS, WebSphere, Geronimo). L'OSGi Alliance, qui est le consortium industriel contrôlant cette spécification, s'intéresse d'une part à la définition de la machine de déploiement et d'exécution des services Java et d'autre part, à standardiser un certain nombre de contrats de service usuels ou dédiés à des domaines d'application comme la télématique (*car automation*) et la téléphonie mobile.

La programmation d'applications pour la plateforme OSGi suit les principes de la programmation orientée service dynamique. Le contrat de services OSGi est syntaxique et qualitatif. La syntaxe est décrite pour une ou plusieurs interfaces Java versionnées. Ces interfaces sont de plus qualifiées par des propriétés de courtage non négociables. La spécification n'adresse pas de qualités de service ni de propriétés non-fonctionnelles

[58] Les adjectifs « ubiquaire », « pervasive » et « adhoc » sont également utilisées dans la littérature.

relatives pour les services. Ces dernières peuvent néanmoins être gérées de manière adhoc [Tournier05,Touseau06M]. Le contrat de service peut néanmoins évoluer dynamiquement sans préavis par l'ajout ou le retrait d'interfaces et par le changement des valeurs des propriétés de courtage.

Les services sont conditionnés dans des unités de déploiement appelés *bundles*. La plateforme gère l'ensemble complet des étapes du cycle de vie de ces *bundles*. Un service peut être enregistré dans un annuaire de services dès que le *bundle* est activé. Le service est alors visible des applications[59] hébergées par la plateforme. La plateforme OSGi notifie aux applications les changements intervenant dans le registre de services et sur le cycle de vie des *bundles*.

L'application OSGi (qui peut elle-même fournir des services) référence directement (c.a.d. sans proxy) les objets implémentant les services et invoque les méthodes sur les objets sans surcoût. La dernière spécification a défini un modèle de composants orientés services présenté à la section III.1.7.3 afin de soulager le développeur de la gestion de la dynamique des services [Cervantes02]. OSGi n'adresse pas directement la distribution des services. Cependant, la spécification OSGi définit des ponts de communication avec des intergiciels comme Jini et UPnP [Donsez05,Donsez06b,Donsez07] limités à des niches applicatives comme le SOHO. D'autres ponts adressant d'autres protocoles de découverte et d'invocation et d'autres modes de communication [Donsez06a,Brady06] ont été proposés pour adresser la construction d'applications très largement distribuées ou bien selon des modèles événementiels. Un des points difficiles est de rendre « sans-couture » les communications inter-passerelles afin de rendre la distribution transparente du point de vue du développeur de services OSGi. De nombreux travaux s'inscrivent dans la distribution de la passerelle OSGi [Bottaro06].

IV.2 Contributions

Ma courte expérience de chef de projet au CHRU de Lille sur le projet ICAR-Telematics m'a permis d'appréhender la difficulté d'intégrer des applications pilotées par des organisations différentes (hôpitaux, laboratoires, médecins libéraux, …) via une multitude de protocoles dédiés à chaque domaine métier (EDISanté, HL7, DICOM, …) [Beuscart96]. Dérivant de SGML, la technologie XML m'avait semblée très tôt une voie d'avenir pour simplifier l'intégration notamment avec l'unification du support de la représentation des données échangées entre les partenaires. L'architecture orientée service qui en grande partie en découle semblait pouvoir simplifier la réalisation des applications multipartenaires. Mes recherches se sont focalisées alors plutôt vers le contexte des services ambiants dans lesquels les partenaires d'une application apparaissent et disparaissent dynamiquement et éventuellement sans préavis. L'étude de ces services dans plusieurs contextes applicatifs comme la photo-numérique, le commerce mobile de proximité et l'hospitalisation à domicile, m'ont fait m'intéresser aux plateformes d'exécution de ces services dynamiques puis à l'ingénierie de ceux-ci.

La première sous-section présente mes contributions sur les plateformes dynamiques de services tandis que la suivante présente ceux sur l'ingénierie des services dynamiques.

[59] La spécification OSGi considère que plusieurs applications s'exécutent simultanément sur la même machine virtuelle Java. Ces applications co-exécutées qui peuvent appartenir à des organisations différentes, peuvent avoir des permissions différentes (accès au modem GPRS, accès à un coupe-circuit électrique, …).

IV.2.1 Plateformes dynamiques de services

Cette sous-section s'intéresse aux travaux menés sur les plateformes d'exécution de services capables de réagir aux changements. Le premier travail concerne la conception de noyau dynamique de serveurs Java d'applications d'entreprise. Le second concerne l'étude de plateformes dynamiques de services dans l'environnement .NET. La troisième porte sur l'étude de l'introduction de l'orientation service dans un modèle à composants.

IV.2.1.1 Noyau dynamique de serveurs d'applications d'entreprise.

L'informatique à la demande [Rappa04,Crawford05] est un modèle économique dans lequel l'opérateur d'une infrastructure matérielle met à disposition de ses clients une partie de son réseau et de ses serveurs de calcul et de stockage. Le client n'est pas obligé de surdimensionner son infrastructure pour répondre à quelques surcharges sporadiques. Cette mise à disposition est réalisée dynamiquement en fonction de leurs besoins immédiats. Pour cela, les applications et les services techniques associés du client doivent pouvoir être déployés en continu sur les nœuds des grappes de serveurs de l'opérateur de l'infrastructure, sur ses serveurs de proximité dans le cas du *edge computing* présenté dans la section 0 ou encore sur les routeurs « intelligents » de l'*Application-Oriented Network* promu par Cisco. Bien que tous les serveurs JEE[60] réalisent un déploiement dynamique des applications JEE, il n'en va pas généralement de même pour les serveurs techniques requis par les applications[61].

Nous nous sommes donc intéressés à définir et à proposer l'architecture d'un noyau flexible de serveur applicatif JEE autorisant à la fois le déploiement dynamique des applications JEE et celui des services techniques associés [Désertot05a, Désertot06c]. Alors que d'autres travaux s'intéressaient à l'approche à composants pour structurer les serveurs d'applications [Fleury03, Herness04, Jordan04, Abdelatif06], nous nous sommes tournés vers l'approche à services pour définir ce noyau de serveur. Notre approche consiste à traiter les applications et les services techniques de manière uniforme. Ce sont des services faiblement couplés qui sont conditionnés dans des unités de déploiement livrées de manière indépendante. Leurs cycles de vie sont indépendants les uns des autres. La liaison entre une application et un service technique (respectivement un service technique avec un service technique) est réalisée à la demande et de façon retardée. Les dépendances de liaisons sont inférées à partir des manifestes des applications. Par exemple, le conteneur d'une application ne se lie au service transactionnel de la plateforme que si l'application utilise les transactions. De plus, le service transactionnel n'est installé et activé que si une des applications actives requiert un fonctionnement transactionnel.

La validation a été menée par la rétro-conception de la base de code de JOnAS qui est constituée de 800 000 lignes de code Java. L'implémentation réalisée qui est appelée « JOnAS on Demand », repose sur la plateforme de services OSGi. Le maintien de la compatibilité avec les unités de déploiement des applications JEE et l'API de déploiement définie par le JSR 88, nous a obligé à spécialiser les chargeurs de classes de la plateforme de services OSGi Felix [Hall04] qui servent normalement au support du déploiement de bundles OSGi. Les chargeurs spécialisés pour des unités JEE infèrent les informations concernant le graphe de chargement des classes à partir des manifestes de déploiement JEE.

[60] Une liste des serveurs JEE et de leurs fonctionnalités est dressée et mise à jour par le site The Server Side [TSS]

[61] Les serveurs les plus avancés ne couvrent qu'une petite partie du cycle de vie des services techniques, à savoir l'activation et l'arrêt. L'installation, la mise à jour et le retrait sont ignorés.

Ces restrictions peuvent être toutefois levées quand les unités JEE sont reconditionnés sous la forme de bundles OSGi avant la phase de déploiement : les applications et les services techniques peuvent être conditionnés et déployés de manière uniforme sous la forme de *bundles* OSGi. C'est ce que montre EasyBeans/OSGi, une implémentation d'un conteneur EJB3.0 à laquelle nous avons participée en collaboration avec l'équipe JEE de Bull. La validation a été faite dans le contexte du *edge-computing* et dans celui des applications taillées sur mesure à la demande pour lesquelles le client ne veut payer qu'un sous-ensemble des logiciels et des sous-composants présents dans le catalogue d'un éditeur ou d'un fournisseur de services.

Ces travaux ont fait l'objet de la thèse de Mikael Désertot. « JOnAS on Demand » sert désormais de guide à la prochaine version du serveur JOnAS[62]. Les propositions que nous avons faites sont désormais reprises par les principaux éditeurs de serveurs JEE pour leurs prochaines éditions et l'Alliance OSGi vient de fonder un groupe d'experts dédiés à ce sujet.

IV.2.1.2 Plateformes dynamiques de services sur .NET

La spécification OSGi définit une plateforme orientée service qui permet le déploiement dynamique et incrémental d'applications Java. La technologie OSGi est maintenant employée comme mécanisme général de gestion du cycle de vie des constituants d'une ou de plusieurs applications se partageant la même machine virtuelle Java. A bien des égards, la plateforme du .NET [Thai01] de Microsoft améliore la plateforme Java notamment en introduisant un conditionnement unifié et versionné et en ajoutant la notion de domaine d'application. Cependant, elle manque de support explicite pour réaliser des applications extensibles dynamiquement comme celles qui ont été rendues possibles pour la plateforme Java grâce à la technologie OSGi.

Il nous a semblé intéressant d'étudier et de réaliser une plateforme dynamique de services pour .NET en s'appuyant sur les principes de la plateforme OSGi [Escoffier05,Escoffier06]. Le premier résultat fut le constat d'impossibilité de décharger de manière fine les classes chargées par un domaine d'application, comme le permet la plateforme Java avec la destruction d'un chargeur de classe.

Ce manque de flexibilité de la part de la plateforme .NET nous a amené à proposer et à implémenter plusieurs alternatives d'architectures de plateformes dynamiques de services. Ces alternatives ont été comparées entre elles et avec OSGi sur Java selon plusieurs critères. Ces critères étaient la possibilité de décharger incrémentalement les classes attachées à un service, la possibilité d'avoir des classes d'implémentation non visibles par les autres services, la possibilité de relier les services par des liaisons directes sans introduire de mandataires (proxy) et enfin le partage des membres statiques.

La première alternative (Figure IV-1, en haut à gauche) déploie les services dans le même domaine d'application ce qui ne permet pas un déchargement indépendant des services et ce qui rend visibles les classes d'implémentation. La seconde alternative (Figure IV-1, en haut à droite) exécute les services dans des domaines séparés ce qui oblige d'une part à répliquer les classes définissant le contrat dans les domaines d'application et d'autre part à réaliser les liaisons entre services avec des paires talon-squelette .NET Remoting requérant la sérialisation des paramètres des méthodes invoquées. La troisième alternative (Figure IV-1, en bas) améliore la seconde en faisant partager les contrats via le domaine spécial destiné aux partages des *assemblies*. Cependant, il n'est alors plus possible de mettre à jour les contrats partagés sans

[62] La prochaine édition du serveur JOnAS fait l'objet d'une collaboration en ObjectWeb, Bull, l'Université de Pékin et le consortium OrientWare.

redémarrer la CLR (Common Language Runtime). La quatrième approche est une hybridation des alternatives 1 et 2 ou 1 et 3 pour former des regroupements de services dont les cycles de vie sont corrélés.

Aucune des 4 alternatives ne satisfaisait individuellement l'ensemble de ces critères définis tandis qu'OSGi les satisfait. Cette étude a été complétée par l'étude d'une cinquième alternative qui modifiait le chargeur de classes de la CLR disponible dans l'implémentation source visible de .NET appelée Rotor [Stutz03]. Ce travail s'est trouvé dans une impasse car l'implémentation courante de Rotor ne recyclait pas les classes déchargées. Les caractéristiques des alternatives étudiées sont synthétisées pour le tableau de la Figure IV-2.

Ce travail qui était l'objet du projet de Magistère de Clément Escoffier [Escoffier04M], a été réalisé sur la version 1 de .NET. Il est resté valide pour la version 2 sortie postérieurement et il semble que la version 3 de .NET n'adressera toujours pas le déchargement fin des classes bien que le besoin se fasse de plus en plus sentir notamment dans la communauté des développeurs de plugins pour des applications comme Visual Studio.

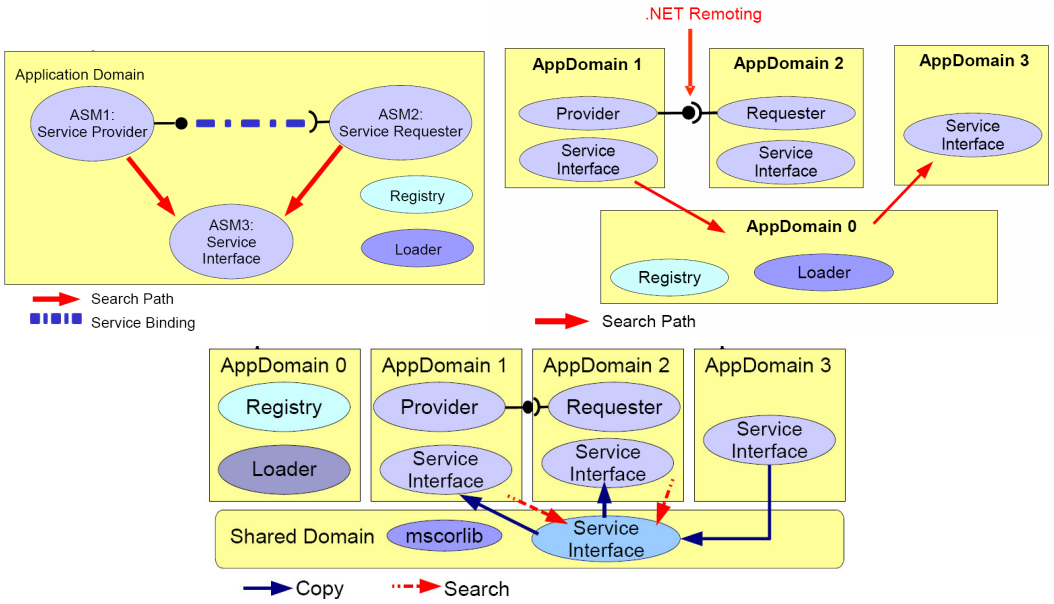

Figure IV-1: Alternatives d'architecture de plateforme dynamiques de services .NET

Aspect étudié	*Alternative 1: 1 seul domaine d'application*	*Alternative 2: multi domaine d'application*	*Alternative 4 hybride 1+2*	*Alternative 3 utilisation du shared domain*	*Alternative 5 modification de la CLR ROTOR*	*OSGi (JVM)*
Apparition dynamique des services	Oui	Oui	Oui	Oui	Oui	Oui
Chargement dynamique	Oui	Oui	Oui	Oui	Oui	Oui
Déchargement dynamique	Non	Oui	Par groupe de service	Non	Non	Oui

Aspect étudié	*Alternative 1: 1 seul domaine d'application*	*Alternative 2: multi domaine d'application*	*Alternative 4 hybride 1+2*	*Alternative 3 utilisation du shared domain*	*Alternative 5 modification de la CLR ROTOR*	*OSGi (JVM)*
Invocation sans proxy	Oui	Non	Parfois	Non	Oui	Oui
Comportement des attributs statiques	Attributs statiques uniques	1 copie par domaine	1 copie par domaine	1 copie par domaine	Attributs statiques uniques	Attributs statiques uniques
Modification de la CLR	Non	Non	Non	Non	Oui : politique de chargement des classes	JVM standard depuis 1.1

Figure IV-2: Tableau récapitulatif des alternatives pour OSGi.NET

IV.2.1.3 Approche mixte composant et service

La plateforme OSGi ne disposait pas de modèle de composants avant les travaux de Humberto Cervantes et de Rick Hall sur ServiceBinder (cf. section III.1.7.3). ServiceBinder est un modèle de composants suivant le principe de l'orientation service : les composants sont faiblement couplés et leurs cycles de vie sont indépendants les uns des autres. Ce modèle reste néanmoins limité à la gestion automatique des liaisons et du cycle de vie des composants. Parallèlement, le modèle Fractal est un modèle de composants autorisant des compositions hiérarchiques de composants. Il est de plus extensible par l'ajout de nouveaux contrôleurs ou par la spécialisation de contrôleurs existants. Fractal dispose également d'un ADL dont les éléments peuvent être étendus. Cependant, Julia, l'implémentation de référence de Fractal en Java, ne disposait pas de formats pour les unités de déploiement. Julia ne permettait pas non plus un déploiement en continu comme le permet OSGi avec ses services.

Nous avons proposé d'introduire le modèle à composants basé sur Fractal pour la plateforme de services OSGi. La motivation de cette proposition baptisée FROGi [Cervantes04b,Désertot06a] était double. D'un côté, FROGi offre aux développeurs OSGi un modèle à composants extensible qui facilite le développement des services ; ces derniers restent toutefois compatibles avec les services « patrimoniaux ». D'un autre côté, FROGi introduit les principes de l'orientation service pour les relations de composition et de liaison du modèle Fractal. De plus, FROGi bénéficie de l'infrastructure de déploiement d'OSGi afin de faciliter le conditionnement versionné de compositions Fractal et de déployer celles-ci en continu.

Nous sommes partis de l'idée que les services sont des entités logicielles à gros grain qui sont mises en œuvre dans une orchestration conçue sur la seule connaissance des contrats que doivent fournir les services engagés à l'exécution, tandis qu'un composant est une entité logicielle participant à une composition bien définie lors la production. Un service peut éventuellement être le résultat d'une composition de composants. Les extensions sur Julia et sur l'ADL Fractal consistent d'une part à qualifier certaines interfaces serveurs avec des propriétés de courtage et d'introduire des liaisons des interfaces clientes basées sur le courtage. Nous avions introduit également une « directive » dans l'ADL permettant d'indiquer les frontières entre les parties de la composition qui doivent être conditionnées séparément. Le cycle de vie d'une composition conditionnée indépendamment, peut soit

être assujetti au cycle de vie du composite englobant (cas de la Figure IV-3), soit géré de manière autonome comme le propose ServiceBinder.

FROGi offre un compromis entre l'approche composant et l'approche service. Il modère ainsi l'usage du courtage à grande échelle des services dans une application et permet d'introduire un couplage faible entre les composants d'une composition. Ce travail a été mené conjointement avec Mikael Désertot et Humberto Cervantes.

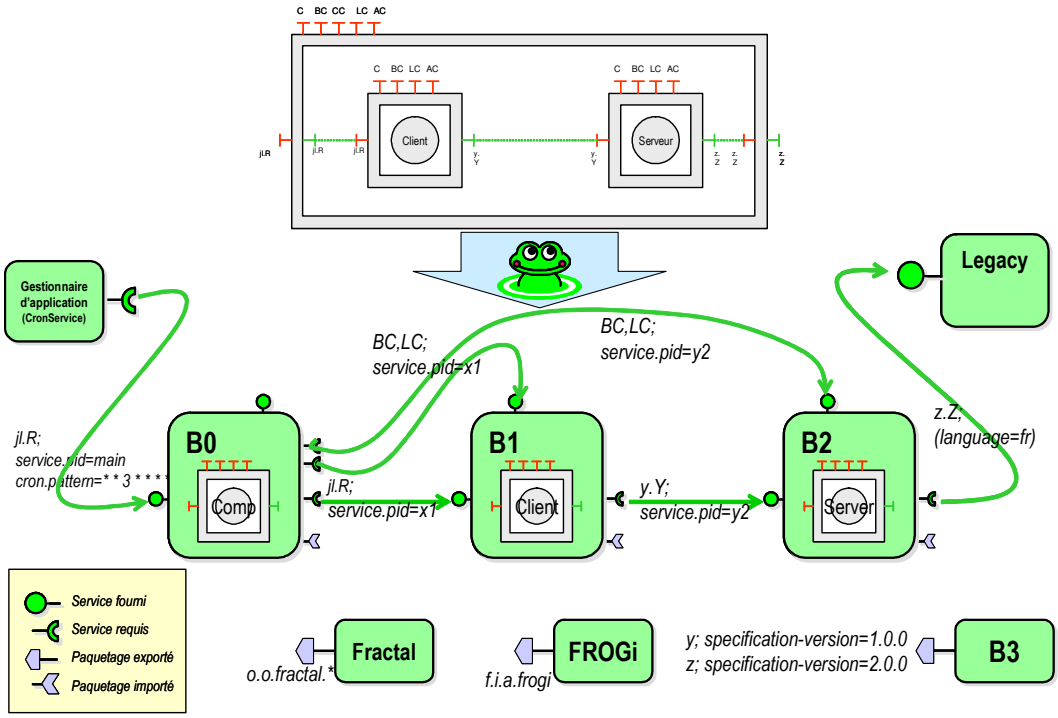

Figure IV-3: Exemple de transformation orientée service d'une composition Fractal.

IV.2.2 Ingénierie des services spécifiques

Le développement des applications dynamiques directement au dessus de plateformes orientées services est une tâche difficile pour le développeur. En effet, il est le plus souvent un expert métier dont la formation initiale est rarement le génie logiciel. Une ambition du projet PISE [Baude06] était d'offrir à ce développeur les outils d'ingénierie logicielle lui permettant de développer rapidement l'application nécessaire. Un des défis pour le fournisseur de ces outils est de pouvoir faire évoluer ceux-ci avec les évolutions des besoins du développeur métier et éventuellement d'être réutilisable dans un autre domaine métier avec le minimum d'impact sur les outils. Le projet PISE offrait le cadre applicatif de la distribution électrique pour valider nos recherches.

L'émergence de l'ingénierie dirigée par les modèles (ie *Model Driven Engineering*) et l'apparition des premiers outils de génération dans les principaux ateliers logiciels nous avaient fait nous tourner vers l'utilisation des modèles pour générer des applications à services pour des domaines d'application spécifiques. L'idée clé était de remonter au niveau méta-modèle à la fois le méta-modèle des concepts des usagers et le méta-modèle de l'espace technologique des services dynamiques. Dans le cadre de PISE, les concepts liés à la distribution électrique sont des équipements, des pilotes de bus de terrain et des services métier à valeur ajoutée, et l'espace technologique est celui des services OSGi. L'objectif est de minimiser l'impact sur les transformateurs de modèle en cas d'évolution

des spécifications des concepts métiers ou en cas de changement de technologies de services (par exemple, DPWS au lieu d'OSGi). Cette approche s'inspire des travaux sur la fédération de domaines (fonctionnels et non-fonctionnels) menés dans l'équipe Adèle depuis plusieurs années [Estublier05].

Ces travaux [Marin05b, Marin05c] font l'objet de la thèse de Cristina Marin et ont été validés dans le contexte de la passerelle industrielle PISE sur la plateforme OSGi à la fois pour les « très classiques » services requête-réponse mais également pour des services producteur-consommateur adaptés au traitement de flux de mesures acquises par des capteurs (cf. section III.2.2.2). L'atelier évolutif se concrétise sous la forme d'une suite de plugins d'édition et de génération pour l'atelier Eclipse. Ces travaux se poursuivent par la thèse de Jian Qi Yu (contexte domotique) et le DRT d'Aurélien Poitou (DPWS) sous la direction de Philippe Lalanda.

IV.3 Conclusion

Ce chapitre a présenté les résultats de nos travaux sur les services. La prise en compte de la dynamique dans les plateformes d'exécution a été le fil conducteur de ces travaux. La section IV.2.1 a présenté nos contributions sur les plateformes dynamiques de services tandis que la section IV.2.2 était consacrée à l'ingénierie des applications pour des domaines spécifiques s'exécutant sur ces plateformes.

La suite de cette section présente un bilan de mes recherches. Elle présente les projets et les collaborations que j'ai menées sur ce domaine. Ces recherches m'ont permis de créer et d'enrichir mes cours en master et en cycle ingénieur.

IV.3.1 Bilan

L'architecture orientée service (AOS) est devenue ces dernières années le courant dominant dans le domaine de l'ingénierie du logiciel. L'AOS est cependant un domaine en pleine maturation[63] et les technologies associées sont encore en pleine effervescence. Les technologies des services Web qui ont favorisé l'émergence de cette approche de conception, éclipsent cependant les travaux menés sur les services sur d'autres technologies au point d'être considérée comme un synonyme de l'approche dans l'esprit des chercheurs et des industriels. Or les services Web adressent seulement les applications distribuées et l'usage d'XML pour faire dialoguer des entités logicielles ne suffit pas pour se revendiquer de l'AOS.

La technologie OSGi qui se revendique de l'AOS dès sa genèse, devient peu à peu la référence pour l'application de l'AOS au développement d'applications (Java) non distribuées. OSGi dont le succès est sans doute lié à son adoption par la communauté Eclipse [Gruber04], pourrait bien devenir le canon de conception des applications Java à partir de la version 7.0 malgré les réticences des pères de la plateforme Java. Un des prochains défis sera alors d'effacer la rupture qui existe entre l'AOS pour les applications distribuées et l'AOS pour les applications colocalisées. Coté .NET le grand compétiteur de la plateforme Java, il fut frappant pour nous d'observer que l'AOS colocalisée n'est toujours pas adressée dans les dernières éditions de .NET.

L'architecture orientée service n'adresse pas directement la prise en charge des contraintes opérationnelles. Celles-ci doivent être prises en charge par le développeur par l'usage explicite de services non-fonctionnels qui peuvent eux-mêmes être contractualisés

[63] Les conférences du domaine (ICSOC, SCC) sont à peine à la quatrième édition.

et implémentés par des services. L'usage des modèles de composants supportant ces services est donc une nécessité. La ligne de démarcation entre composants et services n'est plus alors très claire et reconcevoir une application constituée de services et de composants oblige souvent à déplacer cette ligne avec les technologies sous-jacentes. Ce constat fut à l'origine de notre travail sur FROGi afin de réconcilier les 2 approches et à l'origine du projet PISE.

Une contrainte opérationnelle qui prend beaucoup d'importance avec l'AOS, est le changement incrémental qui s'opère dans l'application au cours de son exécution. La dynamique de services vue par la plupart des travaux est souvent restreinte à la phase de découverte et d'attachement. Très peu font l'hypothèse qu'un service utilisé peut disparaitre temporairement ou définitivement. Ce comportement est difficile à appréhender par le développeur. Les propositions de modèles de composants orientés services (cf section III.1.7.3) solutionnent en partie ce problème. Cependant, ceux-ci restent limités par l'expressivité des réactions à appliquer en cas de changement. Une voie que nous envisageons avec le M2R et la thèse de Lionel Touseau, est de contractualiser les changements dynamiques dans les termes d'accords de niveau de services. Ce travail s'insère dans le domaine général des architectures dynamiques.

IV.3.2 Projets et collaborations

Ma réflexion sur les services a été nourrie par les discussions et les expérimentations faites dans le cadre du projet PISE et avec l'équipe JOnAS dans le cadre de la thèse de Mikaël Désertot.

Le projet RNRT PISE s'intéressait à la définition d'une passerelle de services à valeur ajoutée dans le contexte de la supervision de la distribution électrique. La contribution de l'équipe ADELE portait à la fois sur la définition de la plateforme d'exécution des services dédiés et sur la définition des méthodes d'ingénierie des ateliers.

Le projet ServiceBinder-NG soutenu par le LAFMI (Laboratoire Franco-Mexicain d'Informatique) a été l'occasion de continuer une collaboration fructueuse en idées avec Humberto Cervantes qui avait préparé sa thèse dans l'équipe ADELE.

Ces projets ont été l'occasion de collaborer avec les équipes R&D des sociétés Schneider Electric, France Telecom et Bull.

Les étudiants que j'ai encadrés ou co-encadrés sur ces sujets relatifs aux services étaient: Mikaël Desertot (thèse), Cristina Marin (thèse), Clément Escoffier (Magistère et M2R) et Lionel Touseau (M2R et thèse).

IV.3.3 Enseignements

Ces travaux de recherche m'ont permis de monter de nouveaux cours de Master 2 Professionnel sur les systèmes distribuées, les technologies de services Web et sur les intergiciels adaptables (OSGi). Ces cours sont principalement regroupés dans les projets eCOM, GICOM [Boyer02] et M2M du Master 2 Professionnel Génie Informatique de l'UFR IMA (Informatique et Mathématiques Appliqués) et la troisième année de la filière RICM de PolyTech'Grenoble, et l'unité d'enseignement MAD du Master 1 d'Informatique.

IV.3.4 Publications

[Baude06] Francoise Baude, André Bottaro, Jean-Michel Brun, Antonin Chazalet, Arnaud Constancin, Didier Donsez; Leven Gurgen, Philippe Lalanda, Virginie Legrand, Vincent Lestideau, Sylvain Marié, Cristina Marin, Alain Moreau, Vincent Olive : Extension de

passerelles OSGi pour la grande échelle: Modèles et outils. Actes électroniques de l'Atelier de travail OSGi, Ubimob'06, 3e Journées Francophones Mobilité et Ubiquité, 5 septembre 2006, Paris, 5 pages, https://hal.archives-ouvertes.fr/hal-00097266

[Beuscart96] Régis .J. Beuscart, Didier Donsez, Rick Palo : Integration in Medical Information Systems. Tutoriel du Thirtheenth Internationational Congress Medical Informatics Europa (MIE'96), Copenhague, Danemark, 19-22 Août 1996.

[Cervantes02] Humberto Cervantes, Didier Donsez, Richard Hall : Dynamic Application Frameworks using OSGi And Beanome. Actes de la conférence IEEE ISADS (International Symposium on Advanced Distributed Systems) 2002, Guadalajara, Mexique, ISBN 970-27-0358-1, pp129-139

[Cervantes04b] Humberto Cervantes, Mikaël Désertot, Didier Donsez : FROGi : Déploiement de composants Fractal sur OSGi. Actes de la 1ère conférence francophone sur le Déploiement et la Reconfiguration de logiciels, (DECOR 2004), pp147-158, Grenoble, 28-29 Octobre 2004, ISBN 2-7261-12776-5.

[Désertot05a] Mikaël Désertot, Didier Donsez : Infusion of OSGi Technology into a J2EE Application Server. OSGi World Congress 2005, Paris, 11-15 octobre 2005.

[Désertot06a] Mikaël Désertot, Humberto Cervantes, Didier Donsez : FROGi: Fractal components deployment over OSGi. Proceedings of Software composition 5th International Symposium on Software Composition (SC 2006), 25-26 March 2006, Vienna, Austria (Satellite event of ETAPS 2006), LNCS 4089, ISBN: 3-540-37657-7, pp 275-290, http://dx.doi.org/10.1007/11821946_18

[Désertot06c] Mikael Desertot, Didier Donsez and Philippe Lalanda, "A Dynamic Service-Oriented Implementation for Java EE Servers". Actes de 3th IEEE International Conference on Service Computing (SCC'06), 18-22 September 2006, Chicago, USA

[Donsez05] Didier Donsez : Mise en oeuvre d'UPnP sur OSGi. Deuxièmes Journées Francophones: Mobilité et Ubiquité 2005 (UbiMob 05), Grenoble, France, 31 mai 2005 (7 heures), ISBN:1-59593-172-4, http://doi.acm.org/10.1145/1102613.1102655

[Donsez06a] Didier Donsez, Gaël Thomas : Propagation d'événements entre passerelles OSGi. Atelier de travail OSGi, Ubimob'06, 3e Journées Francophones Mobilité et Ubiquité, 5 septembre 2006, Paris, 4 pages

[Donsez06b] Didier Donsez : Courtage et déploiement dynamiques de composants pour l'infrastructure d'équipements UPnP. Actes des 3e Journées Francophones Mobilité et Ubiquité (Ubimob'06), 5 - 8 septembre 2006, Paris, pp115-118.

[Donsez07] Didier Donsez : On-Demand Component Deployment in the UPnP Device Architecture". Proceedings of the 4th IEEE Consumer Communications and Networking Conference (CCNC) 2007, Las Vegas,11-13 Janvier 2007, http://www.ieee-ccnc.org/2007

[Escoffier05] Clément Escoffier, Didier Donsez : Implémentation de plates-formes dynamiques de services avec .NET. Actes de Conférence Française des Systèmes d'Exploitation (CFSE'05), Le Croisic, France, 5-8 Avril 2005, pp 51-62

[Escoffier06] Clément Escoffier, Didier Donsez, Richard S. Hall : Developing an OSGi-like Service Platform for .NET. Proceedings of the 3rd IEEE Consumer Communications and Networking Conference (CCNC) 2006, Las Vegas,8-10 Janvier 2006, http://www.ieee-ccnc.org/2006/

[Marin05b] Cristina Marin, Didier Donsez, Philippe Lalanda : Approche IDM pour le développement des services basés capteurs. Actes des Premières journées sur l' Ingénierie Dirigée par les Modèles (IDM05), 30 Juin et 1 Juillet 2005, Paris, France, ISBN 2-7261-1284-6, http://planetmde.org/idm05/actes.pdf

[Marin05c] Cristina Marin, Philippe Lalanda, Didier Donsez : A MDE Approach for Power Distribution Service Development. Proceedings of the 3rd International Conference on Service Oriented Computing 2005 (ICSOC05), Amsterdam, The Netherlands,December 12-15, 2005, LNCS 3826, ISSN: 0302-9743, http://dx.doi.org/10.1007/11596141_48

Chapitre V

Direction de Recherche et Perspectives

> *"The most profound technologies are those that disappear. They weave themselves into the fabric of every day life until they are indistinguisable from it."* Mark Weiser

Mes recherches ont principalement porté sur l'étude de quelques propriétés non-fonctionnelles à des points isolés de l'infrastructure matérielle des organisations et des entreprises : des cartes à puce aux serveurs d'entreprise, en passant par les capteurs, les terminaux de télévision interactives, les *gizmos* personnels communicants et les passerelles de services enfouies. La réunion des systèmes très hétérogènes constitue l'infrastructure de la prochaine vague du *e-Business* basée sur la mise en œuvre de l'« Internet des Choses ».

Ce chapitre présente mes perspectives de recherche concernant la définition et le développement de la prochaine génération de plateformes de services destinées à exécuter les services basés sur l'exploitation des données acquises par les nuées de capteurs. La section 1 présente le contexte émergent de ces nouveaux services appelés « machine-à-machine » (M2M) qui me servira de champs d'expérimentation. La section 2 présente les verrous à lever. La section 3 présente les directions des recherches à mener.

V.1 Contexte : les services machine à machine

L'étiquette électronique radio fréquence (RFID) [Lahiri05,Borriello05] semble être la force d'entraînement principale pour la prochaine génération des technologies de l'information (IT) avec sa vision de l'"Internet des choses" ("*Internet of things*") [ITU05]. Ce mouvement est soutenu par les principaux leaders mondiaux de la grande distribution et les grands acteurs des technologies de l'information. Plus globalement, ce mouvement annonce les prémices de la prochaine vague du e-Business qui s'appuiera sur la création de nouveaux services à forte valeur ajoutée utilisant les données acquises dans les équipements reliés par des réseaux capillaires (*SANETs* : *Sensor-Actuator NETworks*) [Zhao04].

A titre d'exemple, les données (ou mesures) acquises par les capteurs peuvent être une tension sur un appareil de mesure électrique, la température dans une pièce ou la présence d'une étiquette RFID associée à un objet ou à une personne en un lieu, le rythme cardiaque d'un patient hospitalisé à domicile, le niveau d'un conteneur de collecte de déchets, le niveau de stock de boissons dans un distributeur automatique, la fréquence journalière des bourrages d'une photocopieuse en location, la position GPS d'un véhicule accidenté, celle d'un conteneur sur une zone de fret portuaire, les anomalies de fonctionnement d'un ascenseur, le taux de fumée et le flux vidéo d'un couloir dans un immeuble ancien … Les services qui découlent de l'exploitation de ces informations, sont plus communément appelés M2M (« Machine-à-Machine ») [Lawton04].

Ces services M2M offrent aux entreprises de nouvelles opportunités de modèles économiques (ie *pay as you use*), d'amélioration de la qualité du service rendu à leurs clients (particuliers ou entreprises) et de satisfaction de leurs obligations légales ou contractuelles. Ils s'imposent comme le nouvel outil d'efficacité (ie *just-in-time*) et de productivité pour les entreprises et organisations *e-agiles*.

A titre d'exemple, un fabricant de conteneurs de déchet ajoutant des étiquettes RFID dans ses produits est également devenu le comptable des déchets jetés par les possesseurs des conteneurs pour des sociétés de ramassage des ordures. Les camions bennes sont équipés de balances électroniques pour peser les conteneurs de déchet et de lecteurs RFID pour identifier leurs propriétaires. Les informations collectées permettent d'optimiser les tournées des camions et d'effectuer une facturation individualisée des déchets jetés en fonction du poids et des erreurs dans le tri sélectif.

V.2 Verrous

Ces services M2M s'appuient essentiellement sur l'intégration des données du monde réel dans les systèmes d'information des entreprises en « temps réel » non critique (ie. *online* ou *near-realtime*). L'état de l'art actuel dans le domaine du génie logiciel et des intergiciels ne permet pas la mise en place rapide et évolutive de services performants et simples à administrer, indispensables à la satisfaction des fournisseurs et de leurs clients.

Les solutions existantes sont réalisées encore de manière empirique en accolant des briques technologiques récupérées dans les deux domaines de métiers concernés : celui de l'informatique embarquée et celui des technologies de l'information. Ces solutions qui fonctionnent sur des projets pilotes de petite échelle avec des équipes de spécialistes hautement qualifiés, ne sont guère viables sur des projets de plus grande échelle avec les équipes d'opérationnels car elles sont lourdes, figées et inadaptées aux enjeux. Ces solutions requièrent de plus une collaboration entre des équipes pluridisciplinaires, pluri-technologiques et pluri-organisations, ce qui peut être un facteur d'échec d'un projet M2M.

Une tendance actuelle pour l'architecture de ces services est de rapprocher des capteurs une partie du contrôle et des traitements des applications en les déléguant à des passerelles Internet situées entre les réseaux capillaires et les infrastructures IT de l'entreprise et ses partenaires [Lalanda04, Heinzelman04]. Il apparaît néanmoins que le développement de telles applications sur de telles architectures se complexifie à cause de la grande variété des plates-formes mises en œuvre et du caractère fluctuant de l'environnement (ie les équipements sont connectés et déconnectés de manière souvent imprévisible, …).

Les expérimentations et les discussions que j'ai menées depuis 4 ans avec nos partenaires industriels (Schneider Electric, FT RD, Thales, EDF, Bull …) dans le cadre des projets PISE, OSMOSE, S4ALL, ANSO et RFIDi nous ont permis d'identifier une infrastructure matérielle et logicielle relativement commune aux services M2M.

Cette infrastructure appelée *Edge-Premise-Server* (*EPS*) comporte plusieurs types de nœuds [Frischbier06] supportant chacun des paradigmes particuliers de composants logiciels (JEE, OSGi, J2ME,.NET) et d'intergiciels de médiation (cf Figure V-1).

- Le niveau *edge* correspond aux passerelles OSGi sur J2ME/CDC (ou des passerelles OPC sur Compact .NET ou Micro .NET) enfouies dans l'entrepôt, le magasin ou le container. Celles ci qui sont directement connectées aux récepteurs RFID et aux capteurs physiques, "remontent" les événements et les mesures vers les serveurs JEE (resp .NET) du centre opérationnel de l'entreprise. Le niveau *edge* est généralement en charge des fonctions d'automatisme en réaction aux mesures et aux événements avec éventuellement des contraintes de type temps-réel. Par exemple, le nœud *edge* dans un entrepôt automatisé pilote en temps réel des convoyeurs et des portiers dans lequel circulent également les personnels.
- Le niveau *premise* est un intermédiaire introduit entre les *edges* et les serveurs quand des fonctions de corrélation des événements et de coordination entre plusieurs passerelles sont requises. Le niveau *premise* apporte la centralisation des fonctions d'administration et de sécurité pour un groupe (géographique) de passerelles, par exemple sur un entrepôt, un immeuble, une zone portuaire …. Le niveau *premise* peut être répliqué et étagé pour supporter la charge et les défaillances. Ces fonctions sont le plus souvent réalisées de manière autonome du fait qu'aucun personnel qualifié n'est présent sur les lieux et que la liaison entre zone géographique et le centre d'opérations n'est pas garantie ou intermittente (par exemple, une plateforme pétrolière isolée, une boutique reliée par l'ADSL, …).
- Le niveau *server* est un cluster de machines à haute performance dans le centre opérationnel de l'entreprise. Ce niveau réalise l'archivage à long terme des informations collectées et consolidées, des rapports d'activité résumés et détaillés et de l'exportation de l'information vers les tiers. Les composants sont plutôt développés selon la spécification JEE.

Cette infrastructure est très hétérogène sur plusieurs plans. Sur le plan matériel, la passerelle qui est en contact avec le monde réel, doit être une machine à bas coût notamment pour atteindre des marchés de masse comme l'automobile ou la domotique tandis que la configuration du centre serveur constitué de fermes de serveurs en lame peut atteindre plusieurs millions d'euros. Le plan des canevas de développement est d'autant plus hétérogène que chaque point de l'infrastructure privilégie un canevas particulier voire métier et qu'il existe aussi de multiples canevas pour développer une portion de l'application à un point de l'infrastructure. Enfin, sur le plan de l'administration de l'infrastructure, les différents points sont rarement gérés d'une manière unifiée et globale.

Il est donc nécessaire d'offrir aux architectes et développeurs à la fois les outils d'ingénierie logicielle et les environnements d'exécution (ie intergiciels) associés [Heinzelman04] permettant la mise en place rapide, évolutive, autonomique et sûre de ces services destinés aux experts métiers des domaines d'application [Bonnet00].

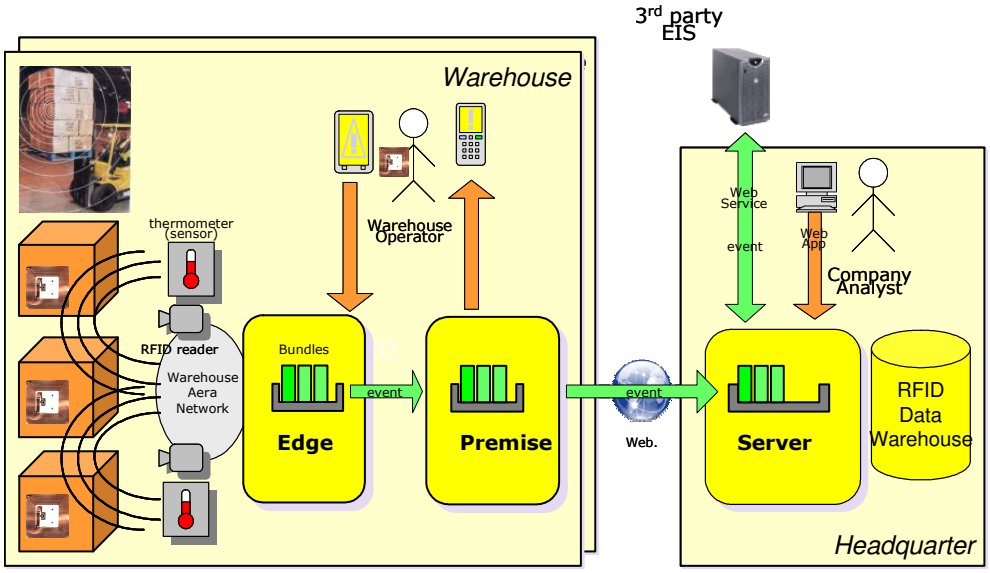

Figure V-1: Infrastructure EPS pour les services RFID

V.3 Perspectives de recherche

Mes perspectives de recherche concernent la définition des outils d'ingénierie logicielle et les environnements d'exécution pour la mise en œuvre des services M2M dans des infrastructures complexes du type *Edge-Premise-Server*. Ainsi, les plateformes dynamiques de services de la prochaine génération devront :

- exécuter des applications empruntant des concepts et des propriétés (non-fonctionnelles) aux canevas de développement patrimoniaux ou à venir,
- intégrer la gestion de la qualité de services entre les constituants des applications,

Les sections suivantes détaillent ces deux points qui constituent mes perspectives de recherche.

V.3.1 Applications multi-canevas

L'architecte d'une infrastructure EPS est confronté le plus souvent au dilemme suivant. Le niveau *premise,* qui est à l'intersection entre le monde de l'embarqué et le monde du système d'information de l'entreprise, doit pouvoir exécuter à la fois des composants OSGi et des composants JEE ainsi que les services techniques associés (transactions, sécurité, …). Dans de telles infrastructures, l'architecte doit généralement faire le choix entre l'une ou l'autre des technologies créant ainsi une rupture dans le mode opératoire de l'infrastructure et dans les paradigmes de développement utilisés pour développer les composants distribués sur les différents nœuds de l'infrastructure. Quand les niveaux *edge* et *premise* sont fusionnés pour des questions de coût et d'encombrement de matériel, le problème s'aggrave car le nœud doit exécuter en plus des composants temps-réel pour le pilotage des automates. L'architecte doit alors effectuer des choix technologies très en amont de la phase de conception de son projet de services M2M et la remise en cause des choix effectués devient alors difficile.

Les constituants d'une application déployée en tout point de cette infrastructure doivent donc pouvoir emprunter des éléments de programmation et des propriétés non-fonctionnelles dans les différents canevas de développement habituels des développeurs métiers. A titre d'exemple, un *SessionBean* JEE, déployé sur un nœud *premise,* devrait pouvoir utiliser des services dynamiquement offerts par des bundles OSGi déployés sur ce nœud. De même, l'authentification réalisée par un service technique relié au conteneur EJB d'un *SessionBean* précédent devrait permettre à un bundle OSGi d'exploiter le contexte d'authentification pour contrôler l'accès aux méthodes des services qu'il fournit. Le dilemme n'est résolu qu'en partie par les intergiciels multi-personnalités (i.e. schizophrènes) qui n'exposent qu'une seule personnalité de l'intergiciel à la fois à l'application. Selon moi, une première solution est l'utilisation de conteneurs flexibles capables à la fois de supporter des codes patrimoniaux programmés selon des modèles établis comme les EJB (cf. section III.1.7.1), Hibernate, CCM, SCR (cf. section III.1.7.3) ou émergents comme Spring, iPOJO ou bien SCA.

Un point délicat pour la conception de ces conteneurs concerne l'hétérogénéité de concept de cycle de vie des entités logicielles dans l'ensemble de ces canevas. Dans les plateformes dynamiques de services, ce concept est fondamental car il conditionne les relations entre les constituants de l'application en vue de reconfigurer celle-ci ou d'effectuer un déploiement en continu. Or, la notion de cycle de vie est rarement présente dans les canevas cités ou alors elle reste une préoccupation subalterne que les développeurs négligent. Selon moi, il est donc envisageable d'un part, d'étendre ou de raffiner les canevas afin de les rendre sensibles au cycle de vie des autres constituants de l'application, et d'autre part, d'abstraire la notion de cycle de vie [Ducas04M] afin de la manipuler au niveau des modèles dans les ateliers définis selon l'approche IDM.

Une autre perspective concerne la prise en compte du renouveau des langages dynamiques pour le développement et le déploiement des constituants de l'application M2M. Les langages dynamiques, plus connus sous le nom de « langages de script » [Ousterhout98], sont généralement des langages à vocation généraliste mais qui se retrouvent dans des niches technologiques (programmation Web, administration système, …). Ces langages ont cependant un avantage indéniable sur les langages comme Java,C#,C++: ils sont compris et utilisés par un nombre incalculable de développeurs occasionnels ou d'analystes métiers[64] sans réelle compétence informatique. En dehors des considérations sur la syntaxe du langage, ceux-ci simplifient grandement la chaine d'outils requises pour développer et (re)déployer les programmes. Je pense que ces langages serviront au développement de l'essentiel des composants qui seront déployés tout le long de l'infrastructure M2M. L'idée est de rendre la main aux usagers finaux non informaticiens pour le développement rapide (voire *agile*) des services M2M. Cette idée était à l'origine des outils d'informatique décisionnels apparus vers 1995 et qui a quasi fait disparaitre l'informaticien d'infocentre qui traduisait en SQL-92 les besoins des analystes métiers. Cette tendance a d'ailleurs conduit les architectes des principales plateformes (Java et .NET) à travailler à l'évolution des machines virtuelles pour le support efficace des langages dynamiques [Hamilton05]. L'usage généralisé de ces langages pourrait avoir un impact sur les modèles de composants qui n'autorisent actuellement que des évolutions statiques des contrats des composants. J'ai, d'ors et déjà, commencé à expérimenter l'évolution dynamique des contrats pour les modèles Fractal et SCR [Donsez06c]. Ces expérimentations se poursuivront dans le contexte des services M2M.

[64] Le développeur occasionnel réalise des macros VB pour Excel, des actions JavaScript dans une page HTML ou bien le traitement d'un formulaire en PHP. Leur nombre dans le monde semble dépasser les 100 millions tandis que le nombre de développeurs professionnels est plafonné à 15 millions.

V.3.2 Accords de niveaux de services

Quand un client requiert des garanties sur la qualité des services qui lui sont fournis ou bien quand la fourniture du service fait l'objet d'une compensation (généralement financière), l'usage d'un service peut faire l'objet d'un accord de niveau de service (*service-level agreement*) entre l'usager d'un service et le fournisseur du service (cf. section IV.1.2). Ces accords qui sont pratiqués depuis plusieurs années dans le domaine des réseaux d'opérateurs, font leur apparition dans d'autres domaines d'activité : distribution électrique, sécurité, santé … Les critères de qualité de services sur lesquels portent les accords sont généralement spécifiques au domaine adressé. Par exemple, la gigue est un critère importante dans les transissions réseau. De même, le délai de recouvrement des pannes pour des serveurs Web et le temps de réponse des requêtes font partie des accords avec les hébergeurs d'applications et de contenus.

La gestion des accords de niveau de service (*service-level management*) constitue une part non négligeable des services M2M. Par exemple, la dérégulation de l'électricité aux Etats-Unis et les grands *blackouts* new-yorkais et californiens poussent les industriels consommateurs d'électricité de qualité à utiliser de plus en plus des services de supervision et d'audit de leur approvisionnement en électricité. Ces services de supervision peuvent à leur tour faire l'objet d'accords de niveau de services. Il est intéressant de noter que l'offre des services est parfois freinée, voire retirée, du fait de l'impossibilité de mettre en place des systèmes de gestion des accords afin de prouver le respect des termes de l'accord en cas de procès.

La gestion des accords de niveau de service est ainsi un des éléments clés d'une plateforme de services M2M. La gestion porte aussi bien entre usagers et services distants qu'entre usagers et services colocalisés au sein d'un même nœud. Avec le projet de master recherche de Lionel Touseau [Touseau06M], je me suis plus spécialement intéressé à définir des accords de services sur la dynamique des services et la gestion automatique de ces accords dans le modèle de composants SCR. Plusieurs perspectives à ces recherches qui se feront dans le cadre de la thèse de Lionel Touseau [Touseau09D].

Une première perspective est d'étendre la gestion des accords aux autres modèles de communication utilisés par les services M2M. Ces modèles, dont nous avons discuté à la section III.2.2.2, sont la publication-souscription d'événements et les flots de mesures. Pour ces deux modèles, l'accord est généralement négocié pour réguler le flot d'événements ou bien au contraire garantir un délai maximum avant rafraichissement d'une mesure. Le rôle de l'usager et le rôle du fournisseur dans l'accord peuvent être inversés (émetteur ou récepteur et respectivement producteur et consommateur) selon les contextes applicatifs.

Une autre perspective est la prise en compte du temps critique dans les accords de niveau de service. Celle-ci est particulièrement importante pour les nœuds de type *edge* qui sont au « contact » des capteurs/lecteurs et des actionneurs, en amont de la chaîne de collecte et qui font cohabiter et collaborer à la fois de services temps réel dur liés aux applications d'automatique, des services temps réel doux destinés à remonter des alertes vers le système IT et des services non-temps réel qui réalisent les configurations et la collecte des statistiques. Cette perspective me conduira à étudier une plateforme dynamique de services s'appuyant au dessus d'une machine virtuelle temps réel [Bollella00].

D'une manière plus générale, j'étudierai également l'usage de l'isolation dans ces plateformes pour un meilleur contrôle des ressources. Introduite par la plateforme .NET sous le nom de domaine d'application, l'isolation permet d'exécuter plusieurs activités colocalisées dans une même machine virtuelle (Java ou CLI) dans des espaces de

ressources séparés. Elle permet également de réserver des ressources (CPU, mémoire, …) et contrôler leurs usages. Cette fonctionnalité d'isolation [Czajkowski03] maintes fois reportée dans les éditions précédentes de Java devrait être présente dans la version 7 de la plateforme Java. Cette fonctionnalité devrait avoir un impact non négligeable sur l'architecture de plateformes comme celle d'OSGi et sur la conception des applications qui s'exécuteront dessus comme nous l'avons déjà constaté avec la plateforme .NET (cf section IV.2.1.2).

Une autre piste de recherche serait de fiabiliser la plateforme de services OSGi au niveau de la machine virtuelle avec une approche semblable à celle de la Virtual Virtual Machine (VVM) [Folliot01][Thomas05]. Un des talons d'Achille de la plateforme OSGi est la conservation de références vers des objets instanciés par un bundle devant être arrêté ou mis à jour. Cette conservation peut être du à la mauvaise programmation du bundle usager de l'objet comme c'est souvent le cas avec les plugins Eclipse, ou bien de même intentionnel afin de perturber le fonctionnement globale de la plateforme OSGi. Une approche « à la VVM » consisterait à définir une VMlet (déployable dynamiquement) qui surveilleraient les bundles clients et la relâche des références inter-bundles selon des principes semblables au ramasse-miette interne de la machine virtuelle. Pour des questions de performance, cette surveillance ne s'appliquerait que sur les bundles non-qualifiés par l'opérateur de la passerelle. Cette approche par VMlet pourrait s'appliquer également à la gestion des accords de services par la surveillance des références inter-bundles faisant l'objet d'un accord de service.

Une autre perspective concerne l'usage de l'informatique autonomique [Kephart03] pour la gestion des accords de niveau de service. Dans le contexte de l'usage des services éphémères et d'accords de courte durée, la négociation, la conclusion et la gestion de l'accord doivent être réalisées sans requérir l'intervention d'opérateurs humains tant que le non-respect des termes de l'accord n'entraine pas de lourdes pénalités ou des conséquences dramatiques. L'informatique autonomique me semble une voie prometteuse pour établir les accords et gérer ceux-ci à grande échelle.

V.4 Conclusion

Ce dernier chapitre a présenté mes perspectives de recherche dans le contexte des services M2M. Ceux-ci soulèvent de nombreux et nouveaux problèmes de recherche. L'identification de ces problèmes n'en est qu'au début. Celle-ci remonte des premières expérimentations « in-vitro » que j'ai démarrée en laboratoire et que je compte appliquer l'an prochain pour des pilotes expérimentaux dans des environnements opérationnels.

Les recherches que je compte mener concernent, d'une part, les supports de multiples canevas métiers dans le développement des services M2M déployés en tout point de l'infrastructure matérielle et logicielle, et d'autre part, la gestion des accords de niveau de services dans les plateformes d'exécution des services M2M. Plus globalement, ces perspectives dépassent le cadre de l'infrastructure EPS des services M2M. Elles concernent plus globalement l'injection de serveurs dynamiques d'applications dans tous les objets à la frontière du monde réel, c'est-à-dire, dans la passerelle domotique du particulier, la passerelle industrielle d'une usine ou dans le téléphone mobile d'un des 2 milliards d'abonnés GSM.

Mon parcours et mes travaux ont été menés à la confluence des domaines de recherche sur les intergiciels, sur les bases de données et sur le génie logiciel. Nous sommes confiants dans la définition de cette prochaine génération d'intergiciels et d'ateliers de

développement nécessaires au développement et au déploiement à grande échelle des futurs services M2M.

Chapitre VI

Références

[Abadi98] Martin Abadi, Luca Cardelli : A Theory of Objects. Pub. Springer, 1998, ISBN 0387947752

[Abdelatif06] Takoua Abdellatif : Apport des architectures à composants pour l'administration des intergiciels. Thèse de Doctorat d'Informatique, Institut National Polytechnique de Grenoble, Septembre 2006.

[Abecassis95] Eric Abecassis : Le gérant d'objets VROOM. Thèse de Doctorat d'Informatique, Université Pierre et Marie Curie (Paris VI), Paris, France, Mai 1995

[Abiteboul05] Serge Abiteboul, Rakesh Agrawal, Philip A. Bernstein, Michael J. Carey, Stefano Ceri, W. Bruce Croft, David J. DeWitt, Michael J. Franklin, Hector Garcia-Molina, Dieter Gawlick, Jim Gray, Laura M. Haas, Alon Y. Halevy, Joseph M. Hellerstein, Yannis E. Ioannidis, Martin L. Kersten, Michael J. Pazzani, Michael Lesk, David Maier, Jeffrey F. Naughton, Hans-Jörg Schek, Timos K. Sellis, Avi Silberschatz, Michael Stonebraker, Richard T. Snodgrass, Jeffrey D. Ullman, Gerhard Weikum, Jennifer Widom, Stanley B. Zdonik : The Lowell database research self-assessment. Communications of the ACM Volume 48, Number 5, 2005, pp 111-118

[Agrawal89] Rakesh Agrawal, Narain H. Gehani: ODE (Object Database and Environment): The Language and the Data Model. Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data, Portland, Oregon, May 31 - June 2, 1989, pp 36-45

[Agre03] Philip E. Agre : P2P and the Promise of Internet Equality. Communications of the ACM, Volume 46, Number , 2003, pp 39-42.

[Akamai01] Akamai - Turbo-charging Dynamic Web Sites with Akamai EdgeSuite, Technical Report, Akamai Technologies, 2001, http://www.akamai.com/en/resources/pdf/whitepapers/

[Aldrich02] Jonathan Aldrich, Craig Chambers, David Notkin: ArchJava: connecting software architecture to implementation. Proceedings of the 22rd International Conference on Software Engineering (ICSE), 19-25 May 2002, Orlando, Florida, USA, pp 187-197. [Allen92] Robert B. Allen, David Garlan: A Formal Approach to Software Architectures. Algorithms, Software, Architecture - Information Processing '92, Volume 1. Proceedings of the IFIP 12th World Computer Congress, Madrid, Spain, 7-11 September 1992, ISBN 0-444-89747-X: 134-141

[Allen98] Robert Allen, Rémi Douence, David Garlan : Specifying and Analyzing Dynamic Software Architectures. Proceedings of the 1st International Conference on Fundamental Approaches to Software Engineering (FASE), Lisbon, Portugal, March 28 - April 4, 1998, LNCS 1382, pp 21-37

[Alonso97] Gustavo Alonso, C. Mohan : Workflow Management: The Next Generation of Distributed Processing Tools. Chapter 2 in Advanced Transaction Models and Architectures, S. Jajodia, L. Kerschberg (Eds.), Kluwer Academic Publishers, 1997.

[Amsaleg99] Laurent Amsaleg, Michael J. Franklin, Olivier Gruber: Garbage collection for a client-server persistent object store. ACM Transactions on Computer Systems (TOCS), Volume 17, Number 3, August 1999, pp 153-201

[Androutsellis-Theotokis04] Stephanos Androutsellis-Theotokis, Diomidis Spinellis: A Survey of Peer-to-Peer Content Distribution Technologies. ACM Computing Surveys 36, 4 (Dec. 2004), 335-371.

[Anon85] Anon et al.: A Measure of Transaction Processing Power. Datamation, April Issue, 1985

[Apache99] Apache Apache Project: The Avalon Framework, http://avalon.apache.org/

[Arnold99] Ken Arnold, Bryan Osullivan, Robert W. Scheifler, Jim Waldo, Ann Wollrath, Bryan O'Sullivan: The Jini Specification, Addison-Wesley Pub, 1 edition (June 1999), ISBN 0201616343

[Atkinson89] Malcolm P. Atkinson, François Bancilhon, David J. DeWitt, Klaus R. Dittrich, David Maier, Stanley B. Zdonik: The Object-Oriented Database System Manifesto. Proceedings of the First International Conference on Deductive and Object-Oriented Databases (DOOD'89), Kyoto research Park, Kyoto, Japan, 4-6 December, 1989, ISBN 0-444-88433-5, pp 223-240

[Atkinson96] Malcolm P. Atkinson, Mick J. Jordan, Laurent Daynès, Susan Spence: Design Issues for Persistent Java: A Type-Safe, Object-Oriented, Orthogonally Persistent System. Proceedings of the 7th Workshop on Persistent Object Systems (POS), Cape May, New Jersey, USA, 1996. Morgan Kaufmann 1997, ISBN 1-55860-447-2, pp 33-47

[Avizienis01] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell: Fundamental Concepts of Dependability. Research Report No 1145, LAAS-CNRS, April 2001.

[Balabanian96] Vahe Balabanian, Liam Casey, Nancy Greene., Chris Adams: An introduction to digital storage media-command and control. IEEE Communications Magazine, Volume: 34, Issue: 11, November 1996, pp 122-127.

[Barroso03] Luiz André Barroso, Jeffrey Dean, Urs Hölzle: Web Search for a Planet: The Architecture of the Google Cluster. IEEE Micro, Volune 23, No. 2, March-April 2003, pp. 22-28.

[Bass98] Len Bass, Paul Clements, Rick Kazmanm: Software Architecture In Practice. Pub. Addison-Wesley, ISBN 0-201-19930-0.

[Batory03] Don Batory: The Road to Utopia: A Future for Generative Programming. International Seminar on Domain-Specific Program Generation, Dagstuhl Castle, Germany, March 23-28, 2003, LNCS 3016, pp 1-17

[Batory88] Don S. Batory, J. R. Barnett, J. F. Garza, K. P. Smith, K. Tsukuda, B. C. Twichell, T. E. Wise: GENESIS: An Extensible Database Management System. IEEE Transactions on Software Engineering (TSE), Volume 14, Number 11, November 1988, pp 1711-1730.

[Batory92] Don S. Batory, Sean W. O'Malley: The Design and Implementation of Hierarchical Software Systems with Reusable Components. IEEE Transactions on Software Engineering and Methodology, Volume 1, Number 4, October 1992, pp 355—398

[Baude06] Francoise Baude, André Bottaro, Jean-Michel Brun, Antonin Chazalet, Arnaud Constancin, Didier Donsez; Leven Gurgen, Philippe Lalanda, Virginie Legrand, Vincent Lestideau, Sylvain Marié, Cristina Marin, Alain Moreau, Vincent Olive : Extension de passerelles OSGi pour la grande échelle: Modèles et outils. Actes électroniques de l'Atelier de travail OSGi, Ubimob'06, 3e Journées Francophones Mobilité et Ubiquité, 5 septembre 2006, Paris, 5 pages, https://hal.archives-ouvertes.fr/hal-00097266

[BEA] BEA Systems, IBM, IONA, Oracle, SAP AG, Siebel Systems, Sybase: Service Component Architecture Specification. http://www-128.ibm.com/developerworks/library/specification/ws-sca

[Beech93] David Beech: Collections of Objects in SQL3. Proceedings of the 19th International Conference on Very Large Data Bases (VLDB), August 24-27, 1993, Dublin, Ireland, pp 244-255.

[Bennani02] Nadia Bennani, Thierry Delot, Sylvain Lecomte, Sergiy Nemchenko, Didier Donsez : Advanced Transactional Model for Component-Based Model. Actes de la conférence IEEE ISADS (International Symposium on Advanced Distributed Systems) 2002, Guadalajara, Mexique, ISBN 970-27-0358-1, pp171-182

[Berners-Lee92] Tim J. Berners-Lee, Robert Cailliau, Jean-Francois Groff, Bernd Pollermann: World-Wide Web: An Information Infrastructure for High-Energy Physics. Présentation à Artificial Intelligence and Software Engineering for High Energy Physics in La Londe, France, January 1992.

[Bernstein89] Philip A. Bernstein, Umeshwar Dayal, David J. DeWitt, Dieter Gawlick, Jim Gray, Matthias Jarke, Bruce G. Lindsay, Peter C. Lockemann, David Maier, Erich J. Neuhold, Andreas Reuter, Lawrence A. Rowe, Hans-Jörg Schek, Joachim W. Schmidt, Michael Schrefl, Michael Stonebraker: Future Directions in DBMS Research - The Laguna Beach Participants. SIGMOD Record 18(1): 17-26 (1989)

[Bernstein96] Philip A. Bernstein, Eric Newcomer: Principles of Transaction Processing for Systems Professionals. Morgan Kaufmann 1996

[Bernstein96] Philip A. Bernstein,. Middleware: A Model of Distributed System Services. Communications of ACM. Vol. 39, No. 2, 1996, pp 86-98.

[Bernstein97] Philip A. Bernstein, Vassos Hadzilacos, Nathan Goodman: Concurrency Control and Recovery in Database Systems. 1987, Addison-Wesley, ISBN 0-201-10715-5

[Bernstein98] Philip A. Bernstein, Michael L. Brodie, Stefano Ceri, David J. DeWitt, Michael J. Franklin, Hector Garcia-Molina, Jim Gray, Gerald Held, Joseph M. Hellerstein, H. V. Jagadish, Michael Lesk, David Maier, Jeffrey F. Naughton, Hamid Pirahesh, Michael Stonebraker, Jeffrey D. Ullman: The Asilomar Report on Database Research. SIGMOD Record 27(4): 74-80 (1998)

[Besancenot97] Jérôme Besancenot, Michèle Cart, Jean Ferrié, Rachid Guérraoui, Philippe Pucheral, Bruno Traverson : Les systèmes transactionnels: concepts, normes et produits. Ed. Hermes, 1997, ISBN 2-86601-645-9

[Beugnard99] Antoine Beugnard, Jean-Marc Jézéquel, Noël Plouzeau, Damien Watkins : Making Components Contract Aware. IEEE Computer, 32(7), 1999, pp 38–45.

[Beuscart96] Régis .J. Beuscart, Didier Donsez, Rick Palo : Integration in Medical Information Systems. Tutoriel du Thirtheenth Internationational Congress Medical Informatics Europa (MIE'96), Copenhague, Danemark, 19-22 Août 1996.

[Bézivin05] Jean Bézivin : On the Unification Power of Models, Software and System Modeling 4(2): 171-188 (2005)

[Bieber01] Guy Bieber, Jeff Carpenter: Introduction to Service-Oriented Programming. OpenWings whitepaper, Septembre 2001, http://www.openwings.org/download/specs/ServiceOrientedIntroduction.pdf

[Biliris94a] Alexandros Biliris, Euthimios Panagos: EOS: An Extensible Object Store. Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, May 24-27, 199, pp 517

[Biliris94b] Alexandros Biliris, Shaul Dar, Narain H. Gehani, H. V. Jagadish, Krithi Ramamritham: ASSET: A System for Supporting Extended Transactions. Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, May 24-27, 1994, pp 44-54

[Billard98] David Billard: Multipurpose Internet Shopping Basket. Proceedings of the Workshop on Business Process Reengineering and Supporting Technologies for Electronic Commerce, Ninth International Workshop on Database and Expert Systems Applications (DEXA), Vienna, Austria, August 24-28, 685-690

[Black04] Andrew P. Black: Post-Javaism. IEEE Internet Computing, Volume 8, Number 1, January/February 2004, pp 96, 93-95 (2004)

[Bonnet00] Philippe Bonnet, Johannes Gehrke, Praveen Seshadri: Querying the physical world. IEEE Personal Communication, Volume 7, October 2000, pp pp. 10-15

[Bollella00] Greg Bollella, Ben Brosgol, Steve Furr, David Hardin, Peter Dibble, James Gosling, Mark Turnbull: The Real-Time Specification for Java. Pub. Addison Wesley, June 06, 2000, ISBN 0-201-70323-8

[Borriello05] Gaetano Borriello: RFID: tagging the world (Special issue). Communations of the ACM, Vol. 48, N° 9, September 2005

[Bottaro06] André Bottaro, Anne Gérodolle, Philippe Lalanda : Pervasive spontaneous composition. 1st IEEE International Workshop on Services Integration in Pervasive Environments, in conjunction with the IEEE ICPS'06, June 29, 2006, Lyon, France

[Box01] Don Box: A Brief History of SOAP. O'Reilly XML.com, April 2001, http://webservices.xml.com/pub/a/ws/2001/04/04/soap.html

[Box98] Don Box: Essential COM. Pub. Addison-Wesley, January 1998. ISBN 0-201-63446-5

[Boyer02] Fabienne Boyer, Sébastien Chassande-Barrioz, Didier Donsez, David Féliot, Sacha Krakowiak. GICOM : un atelier pour l'expérimentation des technologies de systèmes distribués d'entreprise. Actes du colloque Technologies de l'Information et de la Communication dans les Enseignements d'ingénieurs et dans l'industrie, Lyon 13-15 novembre, 2002, pp 423-424. http://docinsa.insa-lyon.fr/tice/2002/ca/ca102.html

[Brady06] Bob Brady: Developing Collaborative Tools With Equinox and ECF. presentation at the EclipseWorld 2006, Cambridge, MA, September 2006.

[Bretl89] Robert Bretl, David Maier, Allen Otis, D. Jason Penney, Bruce Schuchardt, Jacob Stein, E. Harold Williams, Monty Williams: The GemStone Data Management System. Object-Oriented Concepts, Databases, and Applications 1989, ACM Press and Addison-Wesley 1989, ISBN 0-201-14410-7, pp 283-308,

[Brodie78] Michael L. Brodie, Joachim W. Schmidt: What is the Use of Abstract Data Types?. Proceeding of Fourth International Conference on Very Large Data Bases (VLDB), September 13-15, 1978, West Berlin, Germany, pp 140-141

[Brodie-Tzrrell04] William Brodie-Tyrrell, Henry Detmold, Katrina E. Falkner, David S. Munro: Garbage Collection for Storage-Oriented Clusters. Proceeding of the Twenty-Seveth Australasian Computer Science Conference (ACSC2004), Dunedin, New Zealand, January 2004, pp 99-108.

[Brose97] Gerald Brose, JacORB: Implementation and Design of a Java ORB . Proceeding. of IFIP WG 6.1 International Working Conference on Distributed Aplications and Interoperable Systems (DAIS'97,), September 30 - October 2, Cottbus, Germany

[Bruneton04] Eric Bruneton, Thierry Coupaye, Matthieu Leclercq, Vivien Quéma, Jean-Bernard Stefani: An Open Component Model and Its Support in Java. Proceedings of the 7th International Symposium on Component-Based Software Engineering (CBSE), Edinburgh, UK, May 24-25, 2004,. LNCS 3054, pp 7-22

[Butterworth91] Paul Butterworth, Allen Otis, Jacob Stein: The Gemstone Object Database Management System. Communications of the ACM 34(10): 64-77 (1991)

[Carey86] Michael J. Carey, David J. DeWitt, Daniel Frank, Goetz Graefe, M. Muralikrishna, Joel E. Richardson, Eugene J. Shekita: The Architecture of the EXODUS Extensible DBMS. Proceedings of the International Workshop on Object-Oriented Database Systems

(OODBS), September 23-26, 1986, Asilomar Conference Center, Pacific Grove, California, USA, ISBN 0-8186-0734-3, pp 52-65

[Carey93] Michael J. Carey, David J. DeWitt, Jeffrey F. Naughton: The 007 Benchmark. SIGMOD Conference 1993, pp 12-21

[Carey94a] Michael J. Carey, David J. DeWitt, Michael J. Franklin, Nancy E. Hall, Mark L. McAuliffe, Jeffrey F. Naughton, Daniel T. Schuh, Marvin H. Solomon, C. K. Tan, Odysseas G. Tsatalos, Seth J. White, Michael J. Zwilling: Shoring Up Persistent Applications. Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, May 24-27, 1994, pp 383-394

[Carey94b] Michael J. Carey, Michael J. Franklin, Markos Zaharioudakis: Fine-Grained Sharing in a Page Server OODBMS. Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, May 24-27, 1994, pp 359-370

[Carey96] Michael J. Carey, David J. DeWitt: Of Objects and Databases: A Decade of Turmoil. Proceedings of 22th International Conference on Very Large Data Bases (VLDB), September 3-6, 1996, Mumbai (Bombay), India. ISBN 1-55860-382-4, pp 3-14

[Carey99] Michael J. Carey, Donald D. Chamberlin, Srinivasa Narayanan, Bennet Vance, Doug Doole, Serge Rielau, Richard Swagerman, Nelson Mendonça Mattos: O-O, What Have They Done to DB2? Proceedings of 25th International Conference on Very Large Data Bases (VLDB99), September 7-10, 1999, Edinburgh, Scotland, pp 542-553

[Carriero89] Nicholas Carriero, David Gelernter: Linda in Context. Communications of the ACM 32(4): 444-458 (1989)

[Carzaniga98] Antonio Carzaniga, Alfonso Fuggetta, Richard S. Hall, André Van Der Hoek, Deniis Heimbigner, Alexander L.Wolf: A Characterization Framework for Software Deployment Technologies. Technical Report CU-CS-857-98, Dept. of Computer Science, University of Colorado, April 1998, http://serl.cs.colorado.edu/~carzanig/papers/CU-CS-857-98.pdf

[Cattell93] Rick G. Cattell: The Object Database Standard: ODMG-93. Pub. Morgan Kaufmann 1993

[CCA] The Common Component Architecture Forum: Common Component Architecture Core Specification. http://www.cca-forum.org/docs/specification.html

[Cepa04] Vasian Cepa, Mira Mezini: Language Support for Model-Driven Software Development. Special issue of the Journal Science of Computer Programming on Model Driven Architecture: Foundations and Applications. Elsevier Science, 2004

[Cervantes02] Humberto Cervantes, Didier Donsez, Richard Hall : Dynamic Application Frameworks using OSGi And Beanome. Actes de la conférence IEEE ISADS (International Symposium on Advanced Distributed Systems) 2002, Guadalajara, Mexique, ISBN 970-27-0358-1, pp129-139

[Cervantes04a] Humberto Cervantes, Richard S. Hall: Autonomous Adaptation to Dynamic Availability Using a Service-Oriented Component Model. Proceedings of the 26th International Conference on Software Engineering (ICSE), 23-28 May 2004, Edinburgh, United Kingdom, pp 614-623

[Cervantes04b] Humberto Cervantes, Mikaël Désertot, Didier Donsez : FROGi : Déploiement de composants Fractal sur OSGi. Actes de la 1ère conférence francophone sur le Déploiement et la Reconfiguration de logiciels, (DECOR 2004), pp147-158, Grenoble, 28-29 Octobre 2004, ISBN 2-7261-12776-5.

[Cervantes05] Humberto Cervantes, Richard S. Hall: Chapter I: Service Oriented Concepts and Technologies, in the book Service-Oriented Software System Engineering: Challenges and Practices, (ISBN 1-59140-426-6) edited by Zoran Stojanovic and Ajantha Dahanayake, Idea Group Publishing, 2005.

[Chaplin05] Heather Chaplin, Aaron Ruby, Smartbomb : The Quest for Art, Entertainment, and Big Bucks in the Videogame Revolution, Algonquin Books, November 2005, ISBN 1565123468

[Chappell04] David A. Chappell, Enterprise Service Bus, Pub. O'Reilly, June 2004, ISBN 0-596-00675-6

[Chen97] Liming Chen, Didier Donsez, Pascal Faudemay : Design of U-Doc, a research vehicle for hyper document retrieval on the Internet. Actes de Basque Intl Workshop on Information Technology - Data Management Systems, Biarritz, France, 2-4 Juillet 1997

[Chomat03] Stéphane Chomat, Didier Donsez : OSGiTV: une plateforme de déploiement d'applications de télévision interactive basée sur OSGi. Actes de Conférence Française des Systèmes d'Exploitation (CFSE'03), La Colle sur Loup, France, Octobre 2003

[Chrysanthis94] Panos K. Chrysanthis, Krithi Ramamritham: Synthesis of Extended Transaction Models Using ACTA. ACM Transactions on Database Systems (TODS), Volume 19, Number 3, September 1994, pp 450-491

[Clossman98] Gray Clossman, Phil Shaw, Mark Hapner, Johannes Klein, Richard Pledereder, Brian Becker: Java and Relational Databases: SQLJ (Tutorial). Proceedings of the ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA, pp 500

[Codd70] Edgar. F. Codd: A Relational Model of Data for Large Shared Data Banks. Communications of the ACM (CACM), Vol 13, Number 6, pp 377-387, 1970.

[Cointe04] Pierre Cointe, Jacques Noyé, Rémi Douence, Thomas Ledoux, Jean-Marc Menaud, Gilles Muller, and Mario Südholt. Programmation post-objets : des langages d'aspects aux langages de composants. RSTI L'Objet, 10(4):119--143, 2004

[Colwell06] Robert P. Colwell: The Pentium Chronicles. Pub. Wiley, 2006, ISBN : 0-471-73617-1

[COMPiTV] Livrables du projet COMPiTV (Plateforme à composants pour les services de télévision interactive), http://www.rnrt.org/rnrt/projets/res_01_47.htm

[Consel02] Charles Consel: Domain-Specific Languages: What, Why, How. Second Workshop on Language Descriptions, Tools and Applications (LDTA 2002), Grenoble, France, 13 April 2002 in Electronic Notes in Theoretical Computer Science (ENTCS). 65(3)

[Coupaye00] Thierry Coupaye, Jacky Estublier: Foundation of Entreprise Software Deployment. Proceedings of the Conference on Software Maintenance and Reengineering, 29 February - 3 March, 2000, Zurich, Switzerland.

[Coutaz05] Joëlle Coutaz, James L. Crowley, Simon Dobson, David Garlan: Context is key. Communication of the ACM, Volume 48, Number 3, March 2005, pp 49-53

[Cox90] Brad J. Cox: Planning the software industrial revolution: IEEE Software 7 (1990) pp 25-33.

[Crawford05] Catherine H. Crawford, G. Paul Bate, Luba Cherbakov, Kerrie Holley, Charles Tsocanos: Toward an on demand service-oriented architecture. IBM Systems Journal, Volume 44, Number 1, 2005, pp 81-108

[Crié02] Dominique Crié : La relation client. Fidélité, fidélisation, produits fidélisants. Ed. Vuibert, 2002, ISBN 271176995X

[Czajkowski03] Grzegorz Czajkowski, Laurent Daynès, Ben Titzer: A Multi-User Virtual Machine. Proceedings of the General Track: 2003 USENIX Annual Technical Conference, June 9-14, 2003, San Antonio, Texas, USA, pp 85-98

[Czarnecki00] Krzysztof Czarnecki, Ulrich W. Eisenecker: Generative Programming - Methods, Tools, and Applications. Pub. Addison Wesley, 2000, ISBN 0201309777

[Dami98] Samir Dami, Jacky Estublier, Mahfoud Amiour: Apel: A Graphical Yet Executable Formalism for Process Modeling. Proceedings of Automated Software Engineering, Volume 5, Number 1, January 1998, pp 61-96

[Dan04] Asit Dan, Doug Davis, Robert Kearney, Alexander Keller, Richard P. King, Dietmar Kuebler, Heiko Ludwig, Mike Polan, Mike Spreitzer, Alaa Youssef: Web Services on Demand: WSLA-Driven Automated Management. IBM Systems Journal, Volume 43, Number 1, 2004, pp 136-158

[David05] Pierre-Charles David: Développement de composants Fractal adaptatifs : un langage dédié à l'aspect d'adaptation, Thèse de Doctorat d'Informatique, Université de Nantes, Juillet 2005.

[DeRemer76] Frank DeRemer, Hans H. Kron: Programming-in-the-Large Versus Programming-in-the-Small. IEEE Transactions on Software Engineering (TSE), Volume 2, Number 2, June 1976, pp 80-86

[Désertot05a] Mikaël Désertot, Didier Donsez : Infusion of OSGi Technology into a J2EE Application Server. OSGi World Congress 2005, Paris, 11-15 octobre 2005.

[Désertot05b] Mikaël Desertot, Clément Escoffier, Didier Donsez : Autonomic Management of J2EE Edge Servers. Actes de 3rd International Workshop on Middleware for Grid Computing, Co-located with Middleware 2005, Grenoble, France, November 28-29th 2005

[Désertot06a] Mikaël Désertot, Humberto Cervantes, Didier Donsez : FROGi: Fractal components deployment over OSGi. Proceedings of Software composition 5th International Symposium on Software Composition (SC 2006), 25-26 March 2006, Vienna, Austria (Satellite event of ETAPS 2006), LNCS 4089, ISBN: 3-540-37657-7, pp 275-290, http://dx.doi.org/10.1007/11821946_18

[Désertot06b] Mikael Desertot, Clément Escoffier, Philippe Lalanda, Didier Donsez : Autonomic Management of Edge Servers. Actes de International Workshop on Self-Organizing Systems, New Trends in Network Architectures and Services IWSOS'06, 18-20 September 2006, Passau, Germany, LNCS 4124, ISBN: 3-540-37658-5, pp 216-229, http://dx.doi.org/10.1007/11822035_18

[Désertot06c] Mikael Desertot, Didier Donsez and Philippe Lalanda, "A Dynamic Service-Oriented Implementation for Java EE Servers". Actes de 3th IEEE International Conference on Service Computing (SCC'06), 18-22 September 2006, Chicago, USA

[Désertot06d] Mikaël Désertot, Clément Escoffier, Didier Donsez : Towards an Autonomic Approach for Edge Computing. Concurrency and Computation: Practice and Experience, John Wiley & Sons, Novembre 2006 (publication papier début 2007), 16 pages, http://www3.interscience.wiley.com/cgi-bin/abstract/113466340/ABSTRACT

[Deursen00] Arie van Deursen, Paul Klint, Joost Visser: Domain-Specific Languages: An Annotated Bibliography. SIGPLAN Notices 35(6): 26-36 (2000)

[Deux91] O. Deux: The O2 System. Communications of the ACM 34(10), 1991, pp 34-48

[DeWitt90] David J. DeWitt, Philippe Futtersack, David Maier, Fernando Vélez: A Study of Three Alternative Workstation-Server Architectures for Object Oriented Database Systems. Proceedings of the 16th International Conference on Very Large Data Bases (VLDB), August 13-16, 1990, Brisbane, Queensland, Australia. Proceedings. Morgan Kaufmann, ISBN 0-55860-149-X, pp 107-121

[DeWitt94] David J. DeWitt, Navin Kabra, Jun Luo, Jignesh M. Patel, Jie-Bing Yu: Client-Server Paradise. Proceedings of the 20th International Conference on Very Large Data Bases, (VLDB'94), September 12-15, 1994, Santiago de Chile, Chile, pp 558-569

[Dey00] Anind K. Dey: Providing Architectural Support for Building Context-Aware Applications. PhD thesis, College of Computing, Georgia Institute of Technology, 2000.

[Dijkstra72] Edsger W. Dijkstra: The Humble Programmer, CACM Volume 15, Number 10, October 1972, 1972 Turing Award Lecture pp 859-866

[Dongarra04] Jack Dongarra: High Performance Computing Trends, Supercomputers, Clusters and Grids. Third International Conference on Grid and Cooperative Computing (GCC 2004), Wuhan, China, October 21-24, 2004. LNCS 3251, ISBN 3-540-23564-7

[Donsez01a] Didier Donsez, Sébastien Jean, Sylvain Lecomte, Olivier Thomas : Asynchronous Use of Smart Card Services Using SOAP and JMS. Actes électroniques 3rd Gemplus Developer Conference (GDC'2001)

[Donsez01b] Didier Donsez, Sébastien Jean, Sylvain Lecomte, Olivier Thomas :Turning Multi-Application Smart Cards Services Available from Anywhere at Anytime : a SOAP/MOM Approach in the Context of JavaCards. Actes de la conférence eSmart 2001, Cannes, Septembre 2001, LNCS 2140, pp83-94

[Donsez01c] Didier Donsez, Pierre Yves Gibello, Advanced Transaction Models for EJB and Web Services. Présentation invitée à la Conférence ObjectWeb, ENST, Paris, France, 30-31 Octobre 2001

[Donsez01d] Didier Donsez : Les protocoles avancés: transaction BTP. Séminaire IN'TECH Thème Web Services, INRIA Rhône-Alpes, Montbonnot, France, 5 Décembre 2001, http://stream-serv.inrialpes.fr/intech/web_services/d_donsez.ram

[Donsez05] Didier Donsez : Mise en oeuvre d'UPnP sur OSGi. Deuxièmes Journées Francophones: Mobilité et Ubiquité 2005 (UbiMob 05), Grenoble, France, 31 mai 2005 (7 heures), ISBN:1-59593-172-4, http://doi.acm.org/10.1145/1102613.1102655

[Donsez06a] Didier Donsez, Gaël Thomas : Propagation d'événements entre passerelles OSGi. Atelier de travail OSGi, Ubimob'06, 3e Journées Francophones Mobilité et Ubiquité, 5 septembre 2006, Paris, 4 pages

[Donsez06b] Didier Donsez : Courtage et déploiement dynamiques de composants pour l'infrastructure d'équipements UPnP. Actes des 3e Journées Francophones Mobilité et Ubiquité (Ubimob'06), 5 - 8 septembre 2006, Paris, pp115-118.

[Donsez06c] Didier Donsez : Usage des langages de script pour des composants adaptables. Actes des Journées Composants 2006 (JC2006), 4-6 octobre 2006, Perpignan, France, pp 70-78.

[Donsez07] Didier Donsez : On-Demand Component Deployment in the UPnP Device Architecture". Proceedings of the 4th IEEE Consumer Communications and Networking Conference (CCNC) 2007, Las Vegas,11-13 Janvier 2007, http://www.ieee-ccnc.org/2007

[Donsez93] Didier Donsez et Philippe Homond : WEA, Des Espaces de Travail Distribués à Objets Persistants. Actes des Journées des Jeunes Chercheurs en Systèmes à Mémoire Logiquement Partagée, Toulouse, Septembre 1993.

[Donsez94a] Didier Donsez, Philippe Homond et Pascal Faudemay : WEA, A Distributed Object Manager based on a Workspace Hierarchy. Actes de International Conference on Applications in Parallel and Distributed Computing , Caracas, Venezuela, Avril 1994.

[Donsez94b] Didier Donsez, Philippe Homond et Pascal Faudemay : A Cooperative Database System based on a Workspace Hierarchy. Actes de CODATA '94, Committee on Data for Science and Technology, Chambery, France, Septembre 1994 , pp247-258, Eds J-E Dubois, N. Gershon, ISBN-3-540-61457-5, Springer Verlag.

[Donsez95] Didier Donsez, Liming Chen et Pascal Faudemay: Shared Distributed Memory : the Workspace Model. Actes de European Research Seminar on Advances in Distributed Systems (ERSADS) L'Alpe d'Huez, France, Avril 1995.

[Donsez98] Didier Donsez, Gilles Grimaud, Sylvain Lecomte : Recoverable Persistent Memory for Smartcard. Actes de the 3th Smart Card Research and Advanced Application Conference (CARDIS) IFIP, UCL Louvain-la-Neuve , Belgique, 14-16 Septembre 1998, Springer-Verlag, LNCS 1820.

[Duclos02] Frédéric Duclos : Environnement de Gestion de Services Non-Fonctionnels dans les Applications à Composants. Thèse de Doctorat en Informatique, Université Joseph Fourier, Octobre 2002.

[Ecma334] ECMA: Standard ECMA-334, C# Language Specification, 4th edition. June 2006, http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf

[Ecma335] ECMA: Standard ECMA-335, Common Language Infrastructure (CLI), 4th edition. June 2006, http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-335.pdf

[Edwards06] W. Keith Edwards , “Discovery Systems in Ubiquitous Computing”, IEEE Pervasive Computing, Volume 05, Number 2, April-June, 2006, pp 70-77

[Eisenberg04] Andrew Eisenberg, Jim Melton, Krishna G. Kulkarni and Jan-Eike Michels and Fred Zemke, SQL: 2003 has been published, SIGMOD Record, 33 (1) 2004, pp 119-126

[Eisenberg98] Andrew Eisenberg, Jim Melton: SQLJ Part 0, Now Known as SQL/OLB (Object-Language Bindings). SIGMOD Record 27(4): 94-100 (1998)

[Eisenberg99a] Andrew Eisenberg, Jim Melton: SQL: 1999, formerly known as SQL 3. SIGMOD Record 28(1): 131-138 (1999)

[Eisenberg99b] Andrew Eisenberg, Jim Melton: SQLJ-Part 1: SQL Routines Using the Java Programming Language. SIGMOD Record 28(4): 58-63 (1999)

[Elmagarmid92] Ahmed K. Elmagarmid (Ed.): Database Transaction Models for Advanced Applications. Pub. Morgan Kaufmann 1992, ISBN 1-55860-214-3

[Emmerich00] Wolfgang Emmerich: Software Engineering and Middleware: A Roadmap, In The Future of Software Engineering, Anthony Finkelstein (Ed.), ACM Press, 2000, ISBN 1-58113-253-0, pp. 117-129.

[Emonet06] Rémi Emonet, Dominique Vaufreydaz, Patrick Reignier, Julien Letessier : O3MiSCID, a Middleware for Pervasive Environments, 1st IEEE International Workshop on Services Integration in Pervasive Environments, June 29, 2006, Lyon, France

[Escoffier05] Clément Escoffier, Didier Donsez : Implémentation de plates-formes dynamiques de services avec .NET. Actes de Conférence Française des Systèmes d’Exploitation (CFSE'05), Le Croisic, France, 5-8 Avril 2005, pp 51-62

[Escoffier06] Clément Escoffier, Didier Donsez, Richard S. Hall : Developing an OSGi-like Service Platform for .NET. Proceedings of the 3rd IEEE Consumer Communications and Networking Conference (CCNC) 2006, Las Vegas,8-10 Janvier 2006, http://www.ieee-ccnc.org/2006/

[Estublier05] Jacky Estublier, Germán Vega, Anca D. Ionita: Composing Domain-Specific Languages for Wide-Scope Software Engineering Applications. Proceedings of the 8th International Conference on Model Driven Engineering Languages and Systems (MoDELS/ex UML), Montego Bay, Jamaica, October 2-7, 2005, LNCS 3713. pp 69-83

[Estublier06] Jacky Estublier, Jean-Marie Favre, Mireille Blay-Fornarino : L'ingénierie dirigée par les modèles : Au delà du MDA. Ed. Hermès - Lavoisier , 2006, ISBN : 2-7462-1213-7

[Eugster03] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, Anne-Marie Kermarrec : The Many Faces of Publish/Subscribe. ACM Computing Surveys. Volume 35, Number 2 , June 2003, pp 114-131.

[Fernstrom91] Chister Fernström: The Eureka Software Factory: Concepts and Accomplishments. Proceedings of the 3rd European Software Engineering Conference (ESEC), Milan, Italy, October 21-24, 1991, LNCS 550, pp 23-36

[Ferreira96] Paulo Ferreira, Marc Shapiro: Larchant: Persistence by Reachability in Distributed Shared Memory Through Garbage Collection. Proceedings of the 16th International Conference on Distributed Computing Systems (ICDCS). May 27-30, 1996, Hong Kong, pp 394-401

[Ferreira99] Paulo Ferreira, Marc Shapiro, Xavier Blondel, Olivier Fambon, João Garcia, Sytse Kloosterman, Nicolas Richer, Marcus Roberts, Fadi Sandakly, George Coulouris, Jean Dollimore, Paulo Guedes, Daniel Hagimont, Sacha Krakowiak: PerDiS: Design, Implementation, and Use of a PERsistent DIstributed Store. Advanced Distributed Computing: From Algorithms to Systems. LNCS 1752, 1999, pp427-452

[Fielding00] Roy Thomas Fielding: Architectural styles and the design of network-based software architectures. PhD Thesis, University of California, Irvine, 2000. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[Fishman87] Daniel H. Fishman, David Beech, H. P. Cate, E. C. Chow, Tim Connors, J. W. Davis, Nigel Derrett, C. G. Hoch, William Kent, Peter Lyngbæk, Brom Mahbod, Marie-Anne Neimat, T. A. Ryan, Ming-Chien Shan: Iris: An Object-Oriented Database Management System. ACM Transactions on Office Information Systems (TOIS), Volume 5, Number 1, January 1987, pp 48-69

[Fleury03] Marc Fleury, Francisco Reverbel: The JBoss Extensible Server, Middleware 2003. Proceedings of the ACM/IFIP/USENIX International Middleware Conference, Rio de Janeiro, Brazil, June 16-20, 2003,. LNCS 2672, pp 344-373

[Flynn78] Michael J. Flynn, Jim Gray, Anita K. Jones, Klaus Lagally, Holger Opderbeck, Gerald J. Popek, Brian Randell, Jerome H. Saltzer, Hans-Rüdiger Wiehle: Operating Systems, An Advanced Course, Springer 1978, pp 393-481

[Folliot01] Bertil Folliot, Ian Piumarta, Lionel Seinturier, Carine Baillarguet, Christian Khoury, Arthur Léger, Frédéric Ogel : Beyond Flexibility and Reflection: the Virtual Virtual Machine Approach. NATO Advanced Research Workshop, Environments, Tools and Applications for Cluster Computing (IWCC), Mangalia, Romania, September 1-6, 2001, LNCS 2326, pp. 17–26.

[Fowley04] Martin Fowler, Don Box, Anders Hejlsberg, Alan Knight, Rob J. High, John Crupi : The great J2EE vs. microsoft.NET shootout. Companion to the 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA), Vancouver, BC, Canada, October 24 - 28, 2004, pp 143-144.

[Franklin92] Michael J. Franklin, Michael J. Carey : Client-server caching revisited. In Proceedings of the International Workshop on Distributed Object Management, August 1992.

[Franklin97] Michael J. Franklin, Michael J. Carey, Miron Livny: Transactional Client-Server Cache Consistency: Alternatives and Performance. ACM Transactions on Database Systems (TODS) , Volume 22, Number 3, September 1997, pp 315-363.

[Freeman99] Eric Freeman, Susanne Hupfer, Ken Arnold: JavaSpaces Principles, Patterns, and Practice. Pub. Addison-Wesley Professional, 1. June 1999, ISBN 0-201-30955-6

[Frischbier06] Sebastian Frischbier, Kai Sachs, Alejandro Buchmann: Evaluating RFID Infrastructures. Workshop RFID Intelligente Funketiketten - Chancen und Herausforderungen, Erlangen, Germany, July 2006

[Gamma93] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Design patterns: Abstraction and reuse of object-oriented design. In European Conference on Object-Oriented Programming Proceedings (ECOOP'93), volume 707 of Lecture Notes in Computer Science. Springer-Verlag, July 1993

[Gamma95] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Design Patterns: Elements of Reusable Object Oriented Software. Pub. Addison-Wesley, Reading, MA, 1995, ISBN-13: 978-0-201-63361-0

[Garcia-Molina87] Hector Garcia-Molina, Kenneth Salem: Sagas. Proceedings of the Association for Computing Machinery Special Interest Group on Management of Data 1987 Annual Conference, San Francisco, California, May 27-29, 1987, pp 249-259

[Garlan97] David Garlan, Robert T. Monroe, David Wile: ACME: An Architecture Description Interchange Language. Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative Research (CASCON), November 10-13, 1997, Toronto, Ontario, Canada, pp 169-- 183

[Genssler02] Thomas Genssler, Alexander Christoph, Michael Winter, Oscar Nierstrasz, Stéphane Ducasse, Roel Wuyts, Gabriela Arévalo, Bastiaan Schönhage, Peter O. Müller, Christian Stich: Components for embedded software: the PECOS approach. Proceedings of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES 2002), Grenoble, France, October 8-11, 2002. ISBN 1-58113-575-0, pp 19-26

[Goldfarb91] Charles F. Goldfarb, Yuri Rubinsky: The SGML Handbook, Pub. Oxford University Press, 1991, ISBN 0198537379

[Gosling96] James Gosling, William N. Joy, Guy L. Steele Jr.: The Java Language Specification. Pub. Addison-Wesley 1996

[Gray04] Jim Gray: The Next Database Revolution. Keynote at the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004 1-4

[Gray05] Jim Gray: A Measure of Transaction Processing 20 Years Later. IEEE Data Engineering Bulletin. 28(2): 3-4 (2005)

[Gray06] Interview de Jim Gray Sur Channel 9, Mars 2006, http://channel9.msdn.com/Showpost.aspx?postid=168181

[Gray80] Jim Gray: A Transaction Model. Proceedings of the 7th International Colloquium on Automata, Languages and Programming (ICALP 1980) , Noordweijkerhout, The Netherland, July 14-18, 1980, LNCS 85, pp 282-298

[Gray81] Jim Gray: The Transaction Concept: Virtues and Limitations (Invited Paper). Proceeding of 7th International Conference on Very Large Data Bases (VLDB), September 9-11, 1981, Cannes, France, pp 144-154

[Gray85] Jim Gray, Pete Homan, Harald Sammer, Bob Good, Dieter Gawlick: One Thousand Transactions per Second. Spring COMPCON 1985, ISBN 0-8186-0613-4, pp 96-101

[Gruber05] Olivier Gruber, B. J. Hargrave, Jeff McAffer, Pascal Rapicault, Thomas Watson: The Eclipse 3.0 Platform: Adopting OSGi technology. IBM Systems Journal, Volume 44, Number 2, 2005, pp 289-300

[Gruber92] Olivier Gruber, Laurent Amsaleg: Object Grouping in Eos. International Workshop on Distributed Object Management (IWDOM), Edmonton, Alberta, Canada, August 19-21, 1992. pp 117-131.

[Guthery97] Scott B. Guthery: Java Card (Industry Report). IEEE Internet Computing Volume 1, Number 1, 1997, pp 57-59.

[Guttag77] John V. Guttag: Abstract Data Type and the Development of Data Structures. Communications of the ACM Volume 20, Number 6, 1977, pp 396-404

[Hall04] Richard S. Hall: A Policy-Driven Class Loader to Support Deployment in Extensible Frameworks. Proceedings of Second International Working Conference on Component Deployment, (CD 2004), Edinburgh, UK, May 20-21, 2004, LNCS 3083, pp 81-96

[Hamilton05] Graham Hamilton, Mark Reinhold, Gilad Bracha: Evolving the Java Language. JavaOne 2005, TS-7955, , San Francisco , May 2005.

[Hamilton96] Graham Hamilton, Rick Cattell: JDBC: A Java SQL API. JavaSoft Specification, 1997 Version 1.2

[Hansmann00] Uwe Hansmann, Martin S. Nicklous, Thomas Schäck, Frank Seliger: Smart Card Application Development Using Java. Pub. Springer, 2000, ISBN 3-540-65829-7

[Heinzelman04] Wendi B. Heinzelman, Amy L. Murphy, Hervaldo S. Carvalho, Mark A. Perillo: Middleware to Support Sensor Network Applications. IEEE Network, , Volume18, Number 1, Jan/Feb 2004, pp. 6- 14

[Hellerstein03] Joseph M. Hellerstein, Wei Hong, Samuel Madden: The Sensor Spectrum: Technology, Trends, and Requirements. SIGMOD Record 32(4): 22-27 (2003)

[Helm90] Richard Helm, Ian M. Holland, Dipayan Gangopadhyay: Contracts: Specifying Behavioural Compositions in Object-Oriented Systems. Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications / European Conference on Object-Oriented Programming, 21-25 October 1990, Ottawa, Canada,. (OOPSLA/ECOOP 1990) pp 169-180

[Hennessy90] John L. Hennessy, David A. Patterson, Computer Architecture: A Quantitative Approach, Pub. AP Professional, April 1990, ISBN 1558600698

[Herault05] Colombe Hérault : Adaptabilité des services techniques dans le modèle à composants. Thèse de Doctorat d'Informatique, Université de Valenciennes, Juin 2005.

[Herness04] Eric N. Herness, Rob J. High, Jason R. McGee: WebSphere Application Server: A foundation for on demand computing. IBM Systems Journal, Volume 43, Number 2, 2004, pp 213-237

[Hinchcliffe05] Dion Hinchcliffe: SOA / Web Services: WSDL - Describing Web Services in 2005 - How to make Web services consumable today, SYS-CON online magazine, Oct. 15, 2005, http://au.sys-con.com/read/136203_1.htm

[Hofmestier93] C. R. Hofmeister: Dynamic Reconfiguration of Distributed Applications. PhD Thesis, Computer Science Department, University of Maryland, 1993

[Hohpe03] Gregor Hohpe, Bobby Woolf, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Pub. Addison-Wesley Professional, 2003, ISBN 0321200683

[Homond95] Philippe Homond : Implémentation d'une mémoire virtuelle distribuée. Thèse de Doctorat d'Informatique, Université Pierre et Marie Curie (Paris VI), Paris, France,, Septembre 1995

[Horn01] Paul Horn, Autonomic Computing: IBM's Perspective on the State of Information Technology, IBM Whitepaper, http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf

[Hornick87] Mark F. Hornick, Stanley B. Zdonik: A Shared, Segmented Memory System for an Object-Oriented Database. ACM Transactions on Office Information Systems (TOIS), Volume 5, Number 1, January 1987, pp 70-95

[Houston01] Iain Houston, Mark C. Little, Ian Robinson, Santosh K. Shrivastava, Stuart M. Wheater: The CORBA Activity Service Framework for Supporting Extended Transactions. Middleware 2001, IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg, Germany, November 12-16, 2001, LNCS 2218, pp 197-215

[Howard88] John H. Howard, Michael L. Kazar, Sherri G. Menees, David A. Nichols, Mahadev Satyanarayanan, Robert N. Sidebotham, Michael J. West: Scale and Performance in a Distributed File System. ACM Transactions on Computer Systems,6 (1), February 1988.

[Huhns05] Michael N. Huhns, Munindar P. Singh: Service-Oriented Computing: Key Concepts and Principles. IEEE Internet Computing 9(1): 75-81 (2005)

[Hürsch95] Walter L. Hürsch, Cristina Videira Lopes: Separation of Concerns. Technical report NU-CCS-95-03, College of Computer Science, Northeastern University, Boston, February 1995

[Ingalls78] Daniel Ingalls: The Smalltalk-76 Programming System. Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages (POPL), Tucson, Arizona, January 1978, pp 9-16

[Inmon96] William H. Inmon: Building the Data Warehouse, 2nd Edition. Pub. John Wiley & Sons, Inc., ISBN n°0471-14161-5, USA, 1996

[Ishikawa96] Hiroshi Ishikawa, Yasuo Yamane, Yoshio Izumida, Nobuaki Kawato: An Object-Oriented Database System Jasmine: Implementation, Application, and Extension. IEEE Transactions on Knowledge and Data Engineering, Volume 8, Number 2, April 1996, pp 285-304

[ITU05] International Telecommunication Union. The Internet of Things, Executive Summary. ITU Internet Reports 2005, November 2005 http://www.itu.int/osg/spu/publications/internetofthings/InternetofThings_summary.pdf

[Jacobson03] Ivar Jacobson: Use Cases and Aspects-Working Seamlessly Together. Journal of Object Technology 2(4): 7-28 (2003)

[Jajodia97] Sushil Jajodia (Editor), Larry Kerschberg (Editor), Advanced Transaction Models and Architectures , Ed Kluwer Law International, June 1997, 1997, ISBN 0-7923-9880-7

[Jammes05] François Jammes, Antoine Mensch, Harm Smit: Service-Oriented Device Communications using the Devices Profile for Web Services. Proceedings of the 3rd International Workshop on Middleware for Pervasive and Ad-hoc Computing (MPAC), Grenoble, France, November 2005

[Jarke84] Matthias Jarke, Jürgen Koch: Query Optimization in Database Systems. ACM Computing Survey 16(2),1984, pp 111-152.

[JavaCard] Sun Microsystems: JavaCard 2.0 Language Subset and Virtual Machine Specification. Rev 1.0 final, October 1997.

[Jean00a] 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.

[Jean00b] 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

[Jean01] Sébastien Jean, Didier Donsez et Sylvain Lecomte :Using some database principles to improve cooperation in multi-application smart cards. Proceedings of the IEEE Chilean Computer Society Symposium (CCSS'01), 2001.

[Jeronimo03] Michael Jeronimo , Jack Weast: UPnP Design by Example: A Software Developer's Guide to Universal Plug and Play. Pub. Intel Press, ISBN 0971786119, 2003

[Jordan04] Mick J. Jordan, Grzegorz Czajkowski, Kirill Kouklinski, Glenn Skinner: Extending a J2EETM Server with Dynamic and Flexible Resource Management. Proceedings of the ACM/IFIP/USENIX International Middleware Conference (Middleware 2004), Toronto, Canada, October 18-20, 2004, LNCS 3231 , pp 439-458

[JSR220] Java Community Process: JSR 220 : Enterprise JavaBeans 3.0. May 2006, http://jcp.org/en/jsr/detail?id=220

[Keen04] Martin Keen, Patterns: Implementing an SOA Using an Enterprise Service Bus. IBM Redbook, 2004, ISBN 0738490008 http://www.redbooks.ibm.com/redpieces/pdfs/sg246346.pdf

[Kefti04] Abdelmadjid Ketfi : Une approche générique pour la reconfiguration dynamique des applications à base de composants logiciels. Thèse de Doctorat en Informatique, Université Joseph Fourier, Décembre 2004.

[Keller02] Alexander Keller, Heiko Ludwig: Defining and Monitoring Service Level Agreements for Dynamic e-Business. Proceedings of the 16th USENIX System Administration Conference (LISA'02), Philadelphia, PA, Nov. 2002.

[Kephart03] Jeffrey O. Kephart, David M. Chess: The Vision of Autonomic Computing. IEEE Computer, Volume 36, Number 1, January 2003, pp 41-50

[Kephart04] Jeffrey O. Kephart, William E. Walsh: An Artificial Intelligence Perspective on Autonomic Computing Policies. Fifth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY), 2004: pp 3-12

[Ketfi02] Abdelmajid Ketfi, Humberto Cervantes, Richard S. Hall, Didier Donsez : Composants Adaptables au dessus d'OSGi. Actes des Journées Systèmes à Composants Adaptables et extensibles, Grenoble 17-18 octobre 2002.

[Kiczales97] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, John Irwin: Aspect-Oriented Programming. Proceedings of the European Conference on Object-Oriented Programming (ECOOP), Finland. Springer-Verlag LNCS 1241. June 1997, pp 220-242.

[Kienzle02] Jörg Kienzle, Rachid Guerraoui: AOP: Does It Make Sense? The Case of Concurrency and Failures. Proceedings of the 16th European Conference on Object-Oriented Programming (ECOOP), Malaga, Spain, June 10-14, 2002. LNCS 2374, pp 37-61.

[Kim89] Won Kim, Nat Ballou, Hong-Tai Chou, Jorge F. Garza, Darrell Woelk: Features of the ORION Object-Oriented Database System. Object-Oriented Concepts, Databases, and Applications. ACM Press and Addison-Wesley 1989, ISBN 0-201-14410-7, pp 251-282.

[Kim90] Won Kim, Jorge F. Garza, Nat Ballou, Darrell Woelk: Architecture of the ORION Next-Generation Database System. IEEE Transactions on Knowledge and Data Engineering, Volume 2, Number 1, March 1990, pp 109-124

[Kimball96] Ralph Kimball: The Data Warehouse Toolkit. Pub. John Wiley, 1996, ISBN 0471153370

[Kramer85] Jeff Kramer, Jeff Magee: Dynamic Configuration for Distributed Systems. IEEE Transactions on Software Engineering (TSE), Volume 11, Number 4, April 1985, pp 424-436.

[Kruchten95] Philippe Kruchten: The 4+1 View Model of Architecture. Volume 12, Number 6, November 1995, pp 42-50

[Kushner05] David Kushner. Engineering EverQuest: online gaming demands heavyweight data centers. IEEE Spectrum, July 2005, Volume: 42 , Issue: 7, pp 34-39

[Lackner02] Martin Lackner, Andreas Krall, Franz Puntigam: Supporting Design by Contract in Java. Journal of Object Technology 1(3): 57-76 (2002)

[Lahiri05] Sandip Lahiri: RFID Sourcebook. Pub. IBM Press, August 2005; Pages: 304, ISBN 0131851373.

[Lalanda04] Philippe Lalanda: An E-Services Infrastructure for Power Distribution. IEEE Internet Computing 9(3): 52-59 (2005)

[Lamanna03] D. Davide Lamanna, James Skene, Wolfgang Emmerich: SLAng: A Language for Defining Service Level Agreements. 9th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS), 28-30 May 2003, San Juan, Puerto Rico, pp 100-106

[Lamb91] Charles Lamb, Gordon Landis, Jack A. Orenstein, Daniel Weinreb: The ObjectStore Database System. Communications of the ACM 34(10): 50-63 (1991)

[Langlois97] Marc Langlois et al. : Guide Pratique de l'EDI, Principaux langages EDI, Méthodologie de mise en œuvre, Environnement technique d'utilisation, Ouvrage collectif sous la direction de Marc Langlois, Ed WEKA, Juin 1997, 9ème Complément, ISBN 2-7337-0078-2

[Lau05] Kung-Kiu Lau Zheng Wang: A taxonomy of software component models. Proceeding of , 2005. 31st EUROMICRO Conference on Software Engineering and Advanced Applications, 30 Aug.-3 Sept. 2005, pp 88- 95.

[Lawton04] George Lawton: Machine-to-Machine Technology Gears Up for Growth. IEEE Computer 37(9): 12-15 (2004)

[Leclerc04] Matthieu Leclercq, Vivien Quéma, Jean-Bernard Stefani: DREAM: a component framework for the construction of resource-aware, reconfigurable MOMs. Proceedings of the 3rd Workshop on Adaptive and Reflective Middleware, Toronto, Ontario, Canada, October 19, 2004, pp 250-255

[Lecomte97] Sylvain Lecomte et Didier Donsez : Gestion de Transactions pour les Cartes à Microprocesseur. Actes de la conférence NOTERE 97, Pau, France, Novembre 1997.

[Lecomte99] Sylvain Lecomte, Gilles Grimaud, Didier Donsez : Transactional Mechanisms for Open Smart Card. Actes de Gemplus Developer Conference (GDC'99), Paris, CNIT, Juin 1999

[Leopoldi02] Rick Leopoldi: IT Services Management - A Description of Service Level Agreements, RL Information Consulting Whitepaper, May 2002, http://www.itsm.info/SLA%20description.pdf

[Lestideau05] Vincent Lestideau, Didier Donsez : On-Demand Service Installation and Activation with OSGi. Présentation à la Conférence ObjectWeb, INRIA, Lyon, France, Janvier 2005. https://wiki.objectweb.org/ObjectWebCon05/attach?page=Sessions%2Fondemand_osgi.pdf

[Li05] Maozhen Li, Mark Baker: The Grid: Core Technologies. Pub. John Wiley ISBN : 0-470-09417-6 (2004)

[Little03] Mark C. Little: Transactions and Web services. Communications of the ACM (CACM), Volume 46, Number 10, October 2003, pp 49-54

[Lohman91] Guy M. Lohman, Bruce G. Lindsay, Hamid Pirahesh, K. Bernhard Schiefer: Extensions to Starburst: Objects, Types, Functions, and Rules. Communications of the ACM 34(10): 94-109 (1991)

[Ludwig03] Heiko Ludwig, Alexander Keller, Asit Dan, Richard P.King, Richard Franck: WSLA Language Specification, Version 1.0. IBM Research Report, 2003, 110 pages, http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf

[Magee96] Jeff Magee, Jeff Kramer: Dynamic Structure in Software Architectures. Proceedings of the Fourth ACM SIGSOFT Symposium on Foundations of Software Engineering (FSE), San Francisco, California, USA, October 16-18, 1996, pp 3-14

[Mahoney04] Michael S. Mahoney: Finding a History for Software Engineering. IEEE Annals of the History of Computing, vol. 26, no. 1, pp. 8-19, Jan-Mar, 2004.

[Marin04] Cristina Marin, Didier Donsez, Noureddine Belkhatir : Gestion transactionnelle de la reprise sur erreurs dans le déploiement. Actes de la 1ère conférence francophone sur le Déploiement et la Reconfiguration de logiciels, (DECOR 2004), pp199-210, Grenoble, 28-29 Octobre 2004, ISBN 2-7261-12776-5.

[Marin05a] Cristina Marin, Mikaël Désertot, Didier Donsez : SensorBean : Un modèle à composant pour les services basés capteurs. Actes des Journées Composants (JC'05), Le Croisic, France, 5-8 Avril 2005

[Marin05b] Cristina Marin, Didier Donsez, Philippe Lalanda : Approche IDM pour le développement des services basés capteurs. Actes des Premières journées sur l' Ingénierie Dirigée par les Modèles (IDM05), 30 Juin et 1 Juillet 2005, Paris, France, ISBN 2-7261-1284-6, http://planetmde.org/idm05/actes.pdf

[Marin05c] Cristina Marin, Philippe Lalanda, Didier Donsez : A MDE Approach for Power Distribution Service Development. Proceedings of the 3rd International Conference on Service Oriented Computing 2005 (ICSOC05), Amsterdam, The Netherlands,December 12-15, 2005, LNCS 3826, ISSN: 0302-9743, http://dx.doi.org/10.1007/11596141_48

[Marple04] Dave Marples, Stan Moyer: Home Networking and Appliances. in Diane Cook, Sajal Das, Smart Environments: Technologies, Protocols and Applications. Pub. John Wiley, 2004, ISBN 0-471-54448-5

[Matena98] Vlada Matena, Mark Hapner: Enterprise JavaBeans, Version 1.0. Sun Microsystems Specification, March 1998.

[McCandless97] Michael McCandless: Internet Services: The PalmPilot and the Handheld Revolution. IEEE Expert 12(6): 6-8 (1997)

[McIlroy68] M. Douglas McIlroy: Mass Produced Software Components, NATO Software Engineering Conference, Garmisch, Germany, 7th to 11th October 1968, pp. 138-150

[Medvidovic00] Nenad Medvidovic, Richard N. Taylor: A Classification and Comparison Framework for Software Architecture Description Languages. IEEE Transactions on Software Engineering 26(1): 70-93 (2000)

[Mehta00] Nikunj R. Mehta, Nenad Medvidovic, Sandeep Phadke: Towards a taxonomy of software connectors. Proceedings of the 22nd International Conference on Software Engineering, June 4-11, 2000, Limerick Ireland. pp 178-187.

[Meyer87] Bertrand Meyer: Reusability: The Case for Object-Oriented Design. IEEE Software Volume 4, Number 2, March 1987, pp 50-64

[Meyer91] Bertrand Meyer: Eiffel: The Language Prentice-Hall 1991, ISBN 0132479257

[Meyer92] Bertrand Meyer: Applying Design by Contract. IEEE Computer 25(10): 40-51 (1992)

[Mili95] Hafedh Mili, Fatma Mili, Ali Mili: Reusing Software: Issues and Research Directions. IEEE Transactions on Software Engineering, Vol.21, no.6 pp.528-562, June 1995

[Miller87] George W. Miller: Service Level Agreements: Good Fences Make Good Neighbors. Proceeding of the 13th International Computer Measurement Group Conference, Orlando, FL, USA, December 7-11, 1987

[Mohan94] C. Mohan: Advanced Transaction Models - Survey and Critique. Tutorial presented at ACM SIGMOD International Conference on Management of Data, Minneapolis, May 1994.

[Moore65]. Gordon E. Moore: Cramming More Components onto Integrated Circuits. Electronics, volume 38, number 8 (1965), http://www.intel.com/research/silicon/moorespaper.pdf.

[Moss81] J. Elliot B. Moss: Nested transactions: an approach to reliable distributed computing. Ph.D. Thesis, Massachusetts Institute of Technology, Technical Report MIT/LCS/TR-260, 1981

[Motahari-Nezhad06] Hamid R. Motahari Nezhad, Boualem Benatallah, Fabio Casati, Farouk Toumani, Web Services Interoperability Specifications, IEEE Computer , Volume 39, Number 5, pp. 24- 32, May 2006.

[Mouly94] Michel Mouly, Marie-Bernadette Pautet: The Evolution of GSM. Mobile Communications: Advanced Systems and Components, 1994 International Zurich Seminar on Digital Communications, Zurich Switzerland, March 8-11, 1994, LNCS 783, ISBN 3-540-57856-0, pp 13-20

[Nelson88] Michael N. Nelson, Brent B. Welch, John K. Ousterhout: Caching in the Sprite Network File System. ACM Transactions on Computer Systems, 6(1):134--154, February 1988. http://citeseer.ist.psu.edu/nelson88caching.html

[Newkirk02] James Newkirk, Alexei A. Vorontsov: How .NET's Custom Attributes Affect Design. IEEE Software, Volume 19, Number 5, September/October 2002, pp 18-20

[Nierstrasz05] Oscar Nierstrasz, Alexandre Bergel, Marcus Denker, Stéphane Ducasse, Markus Gälli, Roel Wuyts: On the Revival of Dynamic Languages. 4th International Workshop on Software Composition (SC), Edinburgh, UK, April 9, 2005, LNCS 3628, pp. 1-13

[Nierstrasz92] Oscar Nierstrasz, Simon J. Gibbs, Dennis Tsichritzis: Component-Oriented Software Development. 160-165, CACM, Volume 35, Number 9, September 1992.

[Nodine92] Marian H. Nodine, Sridhar Ramaswamy, Stanley B. Zdonik: A Cooperative Transaction Model for Design Databases. Database Transaction Models for Advanced Applications 1992, pp 53-85

[Nottingham01] M. Nottingham, X. Liu:, Edge Architecture Specification. Akamai and Oracle, 2001, http://www.esi.org/architecture_spec_1-0.html

[OASIS05] Organization for the Advancement of Structured Information Standards (OASIS): OASIS Solution Deployment Descriptor TC. 2005, http://www.oasis-open.org/committees/sdd/charter.php

[O'Driscoll00] Gerard O'Driscoll: The essential guide to Digital Set-Top Boxes and Interactive TV. Pub. Prentice Hall, 2000, ISBN 0130173606

[Oldham06] Nicole Oldham, Kunal Verma, Amit P. Sheth, Farshad Hakimpour: Semantic WS-Agreement Partner Selection. Proceedings of 15th International World Wide Web Conference (WWW), 2006, http://lsdis.cs.uga.edu/library/download/OVSH-WWW06.pdf

[OMG03a] Object Management Group: OMG Specification for Deployment and Configuration of Component-based Distributed Applications (OMG D&C). June 2003, http://www.omg.org/docs/ptc/03-07-02.pdf

[OMG03b] Object Management Group: Lightweight Corba Component Model (CCM). OMG Final Adopted Specification, ptc/03-11-03, November 2003, http://www.omg.org/docs/ptc/03-11-03.pdf

[OMG04] Object Management Group: Data Distribution Service for Real-Time Systems Specification, Version 1.0. December 2004, http://www.omg.org/docs/ptc/04-12-02.pdf

[OMG91] Object Management Group: Common Object Request Broker Architecture (CORBA) Specification 1.0, October 1991,

[OMG99] Object Management Group, CORBA Component Model (CCM), August 1999.

[Opc98] OPC Foundation: OPC Overview Version 1.0. October 27, 1998, http://www.opcfoundation.org

[OSGi] OSGi Alliance: The OSGi Specifications. http://www.osgi.org

[OSMOSE] Livrables du projet OSMOSE (Open Source Middleware for. Open Systems in Europe), program ITEA (Information Technology for European Advancement), http://www.itea-osmose.org/

[Ousterhout98] John K. Ousterhout: Scripting: Higher-Level Programming for the 21st Century. IEEE Computer, Volume 31, Number 3, March 1998, pp 23-30

[Papazoglou03] Mike .P. Papazoglou, D. Georgakopoulos: Special Issue on Service-Oriented Computing, Communication of the ACM, vol. 46, no. 10, 2003.

[Paradinas94] Pierre Paradinas, Jean-Jacques Vandewalle: A Personal and Portable Database Server: the CQL Card. Proceedings of the First International Conference on Applications of Databases (ADB),Vadstena, Sweden, June 21-23, 1994, LNCS 819, pp 444-457.

[Parnas72] David Lorge Parnas: On the Criteria To Be Used in Decomposing Systems into Modules. Communications of the ACM 15(12): 1053-1058 (1972)

[Parrish01]. Allen S. Parrish, Brandon Dixon, David Cordes: A Conceptual Foundation for Component-Based Software Deployment. Journal of Systems and Software, Volume 57, Number 3, July 2001, pp 193-200

[Paton99] Norman W. Paton (Ed.): Active Rules in Database Systems. Pub. Springer, New York, 1999, ISBN 0-387-98529-8

[Pawlak05a] Renaud Pawlak, Jean-Philippe Retaillé, Lionel Seinturier, Foundations of AOP for J2EE Development. Pub. APress, 2005, ISBN 1-59059-507-6

[Pawlak05b] Renaud Pawlak: Spoon: Annotation-Driven Program Transformation - The AOP Case. Proceedings of First Workshop on Aspect-Orientation for Middleware Development. Middleware 2005, Grenoble, France

[Peltz03] Chris Peltz: Web Services Orchestration and Choreography. IEEE Computer, Volume 36, Number 10, October 2003, pp 46-52

[PEPiTA] Livrables du projet PEPiTA (Platform for Enhanced Provisioning of Terminal-independent Applications), program ITEA (Information Technology for European Advancement), http://pepita.objectweb.org/

[Perry92] Dewayne E. Perry, Alexander L. Wolf: Foundations for the Study of Software Architecture. ACM SIGSOFT Software Engineering Notes, 17:40--52, October 1992

[Pessemier2004] Nicolas Pessemier, Lionel Seinturier, Laurence Duchien: Components, ADL & AOP: Towards a Common Approach. Proceedings of the Workshop on Reflection, AOP, and Meta-Data for Software Evolution (RAM-SE'04-ECOOP'04), Oslo, June 15, 2004. Fakultät für Informatik, Universität Magdeburg 2004, pp 61-69.

[Pfister96] C Cuno Pfister, Clemens Szyperski: Why Objects are Not Enough. Proceedings of the International Component Users Conference, Munich, Germany, 1996. SIGS

[Plasil98] František Plášil, Dušan Bálek, Radovan Janecek: SOFA/DCUP: Architecture for Component Trading and Dynamic Updating. Proceedings of the 4th International Conference on Configurable Distributed Systems (ICCDS), May 4-6, 1998, Annapolis, Maryland, USA

[Press93] Larry Press: The Internet and Interactive Television. Communications of the ACM, 36, 12, Dec. 1993.

[Prochazka00] Marek Prochazka: Advanced Transactions in Enterprise JavaBeans. Second International Workshop on Engineering Distributed Objects (EDO), Davis, CA, USA, November 2-3, 2000, LNCS 1999, pp 215-230

[Pucheral88] Philippe Pucheral, Jean-Marc Thévenin, Hermann Steffen: Géode: un gestionnaire d'objets extensible orienté grande mémoire. Quatrièmes Journées Bases de Données Avancées (BDA), 17-20 Mai 1988, Bénodet, France, pp 267-284

[Pucheral90] Philippe Pucheral, Jean-Marc Thévenin, Patrick Valduriez: Efficient Main Memory Data Management Using the DBGraph Storage Model. 16th International Conference on Very Large Data Bases (VLDB), August 13-16, 1990, Brisbane, Queensland, Australia. Proceedings. Morgan Kaufmann, ISBN 0-55860-149-X, pp 683-695

[Raccoon97] L. B. S. Raccoon: Fifty Years of Progress in Software Engineering, ACM SIGSOFT Software Engineering Notes Vol 22, No 1 January 1997, pp 88-104, http://uweb.txstate.edu/~mg43/CS5391/Papers/Introduction/progress50years.pdf

[Rankl97] Wolfgang Rankl, Wolfgang Effing: Smart Card Handbook. Pub. John Wiley, 1997, ISBN 0-47196-720-3.

[Rappa04] Michael A. Rappa: The utility business model and the future of computing services. IBM Systems Journal, Volume 43, Number 1, 2004, pp 32-42

[Raymond98] Eric S. Raymond: The Cathedral and the Bazaar, First Monday, vol. 3, 1998.

[Robocop] ITEA Project ROBOCOP, "Robust Open Component Based Software Architecture or Configurable Devices Project", http://www.extra.research.philips.com/euprojects/robocop

[Roman85] Roman Gruia-Catalin: A Taxonomy of Current Issues in Requirements Engineering. IEEE Computer 18(4): 14-23 (1985) pp 14-22.

[Rouvoy06] Romain Rouvoy, Nicolas Pessemier, Renaud Pawlak, Philippe Merle: Using Attribute-Oriented Programming to Leverage Fractal-Based Developments: at the 5th Fractal workshop at ECOOP 2006, July 2006, In 5th International ECOOP Workshop on Fractal Component Model, Nantes, France, July 2006

[Sahai04] Sahai, Akhil; Felix, Wu (Eds.): Utility Computing. Proceedings of the15th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM 2004, Davis, CA, USA, November 15-17, 2004, LNCS Vol. 3278, ISBN 3-540-23631-7.

[Saheb99] Malik Saheb, Ramzi Karoui, Simone Sédillot: Open Nested Transaction: A Support for Increasing Performance and Multi-tier Applications. Proceedings of the Eight International Workshop on Foundations of Models and Languages for Data and Objects (FMLDO), Schloß Dagstuhl, Germany, September 27-30, 1999, pp 115-138

[Sarginson96] P.A. Sarginson:MPEG-2, Overview of the System Layer". BBC R&D report, 1996.

[Saroiu02] Stefan Saroiu, Krishna P. Gummadi, Richard J. Dunn, Steven D. Grivvle, Henry M. Levy: An Analysis of Internet Content Delivery Systems. Proceedings of the 5th Symposium on Operating System Design and Implementation (OSDI), December 9-11, 2002, Boston, Massachusetts, USA 2002.

[Schmidt05] Marc-Thomas Schmidt, Beth Hutchison, Peter Lambros, Rob Phippen: The Enterprise Service Bus: Making service-oriented architecture real. IBM Systems Journal, Volume 44 Issue 4, 2005, https://www.research.ibm.com/journal/sj/444/schmidt.pdf

[Seidewitz03] Ed Seidewitz: What Models Mean. IEEE Software 20(5): 26-32 (2003)

[Seinturier05] Lionel Seinturier: Réflexivité, aspects et composants pour l'ingénierie des intergiciels et des applications réparties. Thèse d'habilitation à diriger des recherches en Informatique, Université Pierre et Marie Curie (Paris VI), Paris, France, 13 décembre 2005.

[Seinturier06a] Lionel Seinturier, Nicolas Pessemier, Laurence Duchien, Thierry Coupaye: A Component Model Engineered with Components and Aspects. Proceedings of the 9th International SIGSOFT Symposium on Component-Based Software Engineering (CBSE), Stockholm, Sweden, June 2006, LNCS 4063.

[Seinturier06b] Lionel Seinturier, Nicolas Pessemier, Clément Escoffier, Didier Donsez : Towards a Reference Model for Implementing the Fractal Specifications for Java and the .NET Platform. Fractal CBSE workshop at ECOOP 2006, Nantes, 3 Juillet 2006, à paraître dans Springer Verlag LNCS Serie.

[Selic03] Bran Selic: The Pragmatics of Model-Driven Development. IEEE Software 20(5): 19-25 (2003)

[Serrano-Alvarado04] Patricia Serrano-Alvarado, Claudia Roncancio, Michel E. Adiba: A Survey of Mobile Transactions. Distributed and Parallel Databases, Volume 16, Number 2, September 2004, pp 193-230

[Shekita90] Eugene J. Shekita, Michael J. Zwilling: Cricket: A Mapped, Persistent Object Store. Proceedings of the Fourth International Workshop on Persistent Objects (POS), 23-27 September 1990, Martha's Vineyard, MA, USA, pp 89-102.

[Silberschatz90] Abraham Silberschatz, Michael Stonebraker, Jeffrey D. Ullman: Database Systems: Achievements and Opportunities - The "Lagunita" Report of the NSF Invitational Workshop on the Future of Database System Research held in Palo Alto, California, February 22-23, 1990. pp 6-22

[Silberschatz96] Abraham Silberschatz, Michael Stonebraker, Jeffrey D. Ullman: Database Research; Achievements and Opportunities into the 21st Century. SIGMOD Record 25(1): 52-63 (1996)

[Sillitti02] Alberto Sillitti, Tullio Vernazza, Giancarlo Succi: Service Oriented Programming: A New Paradigm of Software Reuse. Proceedings of the 7th International Conference on Software Reuse: Methods, Techniques, and Tools, (ICSR-7), Austin, TX, USA, April 15-19, 2002,. LNCS 2319, pp 269-280

[Singhal92] Vivek Singhal, Sheetal V. Kakkad, Paul R. Wilson: Texas: An Efficient, Portable Persistent Store. POS 1992. Proceedings of the Fifth International Workshop on Persistent Object Systems, San Miniato (Pisa), Italy, 1-4 September, 1992, pp 11-33

[Smith94] Walter R. Smith, The Newton Application Architecture. Proceedings of the 39th IEEE Computer Society International Conference, pp. 156–161, San Francisco, 1994.

[Snyder93] Alan Snyder: The Essence of Objects: Concepts and Terms. IEEE Software, Volume 10, Number 1, January 1993, pp 31-42

[Stonebraker90] Michael Stonebraker, Lawrence A. Rowe, Bruce G. Lindsay, Jim Gray, Michael J. Carey, Michael L. Brodie, Philip A. Bernstein, David Beech: Third-Generation Database System Manifesto - The Committee for Advanced DBMS Function. SIGMOD Record 19(3): 31-44 (1990)

[Stonebraker91] Michael Stonebraker, Greg Kemnitz: The Postgres Next Generation Database Management System. Communications of the ACM 34(10): 78-92(1991)

[Stonebraker93] Michael Stonebraker, James Frew, Kenn Gardels, Jeff Meredith: The Sequoia 2000 Benchmark. Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993, pp 2-11

[Stroustrup86] Bjarne Stroustrup: The C++ Programming Language, First Edition. Pub. Addison-Wesley 1986

[Stutz03] David Stutz, Ted Neward, Geoff Shilling: Shared Source CLI Essentials. Pub. O'Reilly, March 2003, ISBN 0-596-00351-X

[Sun00] Sun Microsystems: Java TV Specification 1.0, 2000, http://java.sun.com/products/javatv/

[Sun97] Sun Microsystems: JavaBeans Specification , Version 1.01, 1997.

[Sun98] Sun Microsystems:, Java Management Extensions Specification http://java.sun.com/products/JavaManagement/

[Szyperski98] Clemens Szyperski: Component Software: Beyond Object-Oriented Programming. Pub.Addison Wesley, 1998, ISBN 0-201-17888-5

[Tesanovic04] Aleksandra Tesanovic, Dag Nyström, Jörgen Hansson, Christer Norström: Aspects and components in real-time system development: Towards reconfigurable and reusable software. Journal of Embedded Computing, February 2004.

[Thai01] Thuan L. Thai,Hoang Lam: .NET Framework Essentials. Pub.. O'Reilly, 2001, ISBN 0-596-00165-7

[Thomas05] Gaël Thomas: Applications Actives : Construction dynamique d'environnements d'exécution flexibles homogènes. Thèse de Doctorat d'Informatique, Université Pierre et Marie Curie (Paris VI), Paris, France, 2005

[Thompson04] Craig Thompson: Everything Is Alive. IEEE Internet Computing, January-February 2004, pp 83-86

[Tournier05] Jean-Charles Tournier: Qinna : une architecture à base de composants pour la gestion de la qualité de service dans les systèmes embarqués mobiles. Thèse de Doctorat d'Informatique, Institut National des Sciences Appliquées de Lyon (INSA de Lyon), 2005

[TpcC] Transaction Processing Performance Council: The TPC Benchmark™C (TPC-C), http://www.tpc.org/tpcc/default.asp

[TpcH] Transaction Processing Performance Council: The TPC Benchmark™H (TPC-H), http://www.tpc.org/tpch/default.asp

[TpcW] Transaction Processing Performance Council: The TPC Benchmark™W (TPC-W), http://www.tpc.org/tpcw/default.asp

[TSS] The Server Side: Application Server Matrix, document en ligne http://www.theserverside.com/tt/articles/article.tss?l=ServerMatrix

[Ubell94] Michael Ubell: The Montage Extensible DataBlade Achitecture. Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, May 24-27, 1994., pp 482

[UPnP] UPnP Forum, http://www.upnp.org

[Vadet01] Matthieu Vadet, Philippe Merle : Les conteneurs ouverts dans les plates-formes à composants. Actes des Journées Composants, Besançon, 25-26 Octobre 2001

[Vakali03] Athena Vakali, George Pallis: Content Delivery Networks: Status and Trends, IEEE Internet Computing, Volume 7, Number 6, November/December 2003, pp 68-74

[Vandewalle98] Jean-Jacques Vandewalle, Eric Vétillard: Developing Smart Card-Based Applications Using Java Card. Proceedings of the the 3rd International Conference on Smart Card Research and Applications (CARDIS), Louvain-la-Neuve, Belgium, September 14-16, 1998, LNCS 1820, pp 105-124.

[Voas98] Jeffrey M. Voas: The Challenges of Using COTS Software in Component-Based Development (Guest Editor's Introduction). IEEE Computer 31(6): 44-45 (1998)

[W3C00] World Wide Web Consortium: Web Services Description Language (WSDL) Version 1.0. 2000, http://www.w3.org/TR/wsdl

[W3C02] World Wide Web Consortium: Web Services Policy Framework (WS-Policy). Version 1.0. December 18, 2002, http://www.w3.org/Submission/WS-Policy/

[W3C98] World Wide Web Consortium: Simple Object Access Protocol (SOAP) Version 1.0. 1998, http://www.w3.org/TR/soap

[Wachter92] Helmut Wächter, Andreas Reuter: The ConTract Model. In Database Transaction Models for Advanced Applications 1992, pp 219-263

[Wang05] Shengquan Wang, Sangig Rho, Zhibin Mai, Riccardo Bettati, Wei Zhao: Real-Time Component-based Systems. Proceedings of IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), March 2005.

[Weihl04] Andy Weihl, Jay Parikh, William E. Weihl: Edge computing: Extending Enterprise Applications to the Edge of the Internet. Proceedings of the 13th International World Wide Web Conference (WWW), May 2004, pp 180-187

[Weikum92] Gerhard Weikum, Hans-Jörg Schek: Concepts and Applications of Multilevel Transactions and Open Nested Transactions Database Transaction Models for Advanced Applications 1992:, In Database Transaction Models for Advanced Applications, ed. A.K. Elmagarmid, Morgan Kaufmann, 1992, pp 515-553

[Weiser91] Mark Weiser: The Computer for the Twenty-First Century. Scientific American, pp. 94-10, September 1991, http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html

[Whisnant03] Keith Whisnant, Zbigniew Kalbarczyk, Ravishankar K. Iyer: A system model for dynamically reconfigurable software. IBM Systems Journal 42(1): 45-59 (2003)

[White94] Seth J. White, David J. DeWitt: QuickStore: A High Performance Mapped Object Store. Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, May 24-27, 1994, pp 395-406

[Whittaker02] James A. Whittaker, Jeffrey M. Voas. 50 years of software: key principles for quality. IT Professional, Volume 4, Issue 6, Nov.-Dec. 2002 Page(s):28 - 35

[Widom95] Jenifer Widom: Research Problems in Data Warehouse. Proc. of 4th international Conference on Information and Knowledge Management (CIKM), November, 1995.

[Wiener94] Janet L. Wiener, Jeffrey F. Naughton: Bulk Loading into an OODB: A Performance Study. of the 20th International Conference on Very Large Data Bases, (VLDB), September 12-15, 1994, Santiago de Chile, Chile, pp 120-131.

[Wiener95] Janet L. Wiener, Jeffrey F. Naughton: OODB Bulk Loading Revisited: The Partitioned-List Approach. Proceedings of 21th International Conference on Very Large Data Bases (VLDB'95), September 11-15, 1995, Zurich, Switzerland, pp 30-41

[Wietrzyk98] Vlad Ingar Wietrzyk, Mehmet A. Orgun: VERSANT Architecture: Supporting High - Performance Object Databases. of the 1998 International Database Engineering and Applications Symposium (IDEAS), Cardiff, Wales, U.K., Junly 8-10, 1998, pp 141-149

[Wile99] David S. Wile, J. Christopher Ramming: Guest Editorial: Introduction to the Special Section Domain-Specfic Languages (DSL). IEEE Transactions on Software Engineering, Volume 25, Number 3, 1999, pp 289-290

[Wright05] Jane Wright: Blades Have the Edge. IEEE Spectrum, April 2005, pp 25-29

[XOpen91] The X/Open CAE Specification. Distributed Transaction Processing: The XA Specification. X/Open Document Number: XO/CA/91/300. December 1991

[Yong94] Voon-Fee Yong, Jeffrey F. Naughton, Jie-Bing Yu: Storage Reclamation and Reorganization in Client-Server Persistent Object Stores. Proceedings of the Tenth International Conference on Data Engineering (ICDE), February 14-18, 1994, Houston, Texas, USA, pp 120-131

[Zhao04] Feng Zhao, Leonidas Guibas. Wireless Sensor Networks: An Information Processing Approach. Pub. Morgan Kaufmann, May 2004, ISBN 1-55860-914-8

[Zhu05] Fen Zhu, Matt W. Mutka, Lionel M. Ni: Service Discovery in Pervasive Computing Environments. IEEE Pervasive Computing, Volume 04, Number 4 , Oct-Dec, 2005, pp. 81-90

[Zurfluh01] Gilles Zurfluh, Elisabeth Métais, Marie-Christine Rousset, Frédéric Bret, Irène Guessarian, Mohand Hacid, Olivier Teste, Sylviane Schwer, Didier Donsez, Thierry

Cruanes, Zohra Bellahsene: Entrepôt de données pour l'aide à la décision, dans le livre Ingénierie des systèmes d'information, Editeurs Camille Rosenthal-Sabroux, Corine Cauvet, Hermes Informatique et SI, pp 175-208, 2001, ISBN 2-7462-0219-0

Annexe A
Liste des encadrements

Cette annexe présente une liste des encadrements majeurs.

VI.1 Doctorats

[Leconte98D] Sylvain LECOMTE, "COST STIC : Des Cartes Orientés Services Transactionnels et des Systèmes Transactionnels Intégrant des Cartes", Thèse de Doctorat d'Informatique soutenue en Septembre 1998 au Laboratoire LIFL UMR CNRS 8022, Equipe RD2P (Univ. Lille 1). co-encadrement avec Vincent Cordonnier (Directeur) de Septembre 1996 à Septembre 1998.

[Jean01D] Sébastien JEAN, " Modèles et Architectures d'Interaction interne et externe pour Cartes à Microprocesseur Ouvertes", Thèse de Doctorat d'Informatique soutenue en Septembre 1998 au Laboratoire LIFL UMR CNRS 8022, Equipe RD2P (Univ. Lille 1). co-encadrement avec Vincent Cordonnier (Directeur) de Septembre 1998 à Septembre 2001.

[Nemchenko04D] Sergiy NEMCHENKO, « Transactions Avancées pour plateformes serveur à composants », Thèse de Doctorat d'Informatique démarrée juin 2000 et soutenue en Septembre 2004 au Laboratoire LAMIH UMR CNRS 8530, Equipe ROI (Univ. Valenciennes), co-encadrement avec Sylvain LECOMTE jusqu'à ma mutation à Grenoble en Septembre 2001.

[Desertot07D] Mikaël DESERTOT, « Architecture hièrarchique des serveurs de composants et de services non fonctionnels », Démarré Septembre 2003, soutenance prévue Février 2007, Laboratoire LSR UMR CNRS 5526, Equipe Adèle (Univ. Grenoble 1), co-encadrement avec Philippe LALANDA

[Marin07D] Cristina MARIN, « Modèle à Composants Logiciels Dynamiques pour Passerelles Embarquées de Services », Démarré Septembre 2004, Laboratoire LSR UMR CNRS 5526, Equipe Adèle (Univ. Grenoble 1), co-encadrement avec Philippe LALANDA

[Touseau09D] Lionel TOUSEAU, « Accord de niveau de services dans les plateformes dynamiques de services », Démarré Septembre 2006, Laboratoire LSR

UMR CNRS 5526, Equipe Adèle (Univ. Grenoble 1), co-encadrement avec Philippe LALANDA

VI.2 DEAs, Masters Recherche et Magistères

[Cruanes91M] Thierry CRUANES, «Transactionnement Dataflow de requêtes dans une Base de Données Distribuée», DEA de Systèmes Informatiques, au Laboratoire MASI UA 818, Equipe RAPID (Univ. Paris 6), Juin 1991, co-encadrement avec Pascal FAUDEMAY.

[Jean98M] Sébastien JEAN, «Dualité traitements/données dans la carte à microprocesseur : la carte active JavaScript», DEA d'Informatique, au Laboratoire LIFL UMR CNRS 8022, Equipe RD2P (Univ. Lille 1), Juin 1998, co-encadrement avec Jean-Marie PLACE.

[Thilliez01M] Marie THILLIEZ, «Commerce Electronique de Proximité», DEA AISIH option Informatique, au Laboratoire LAMIH UMR CNRS 8530, Equipe ROI (Univ. Valenciennes), Juin 2001, co-encadrement avec Nadia BENNANI.

[Hérault01M] Colombe HERAULT, «EJBiTV – Plateforme à composants de type EJB pour le développement d'applications de TV interactive», DEA AISIH option Informatique, au Laboratoire LAMIH UMR CNRS 8530, Equipe ROI (Univ. Valenciennes), Juin 2001, co-encadrement avec Sylvain LECOMTE.

[Chomat03M] Stéphane CHOMAT, «OSGi et programmation aspect : vers une construction de composants par composition», DEA «Information, Systèmes et Communications» au Laboratoire LSR UMR CNRS 5526, Equipe Adèle (Univ. Grenoble 1), Juin 2003.

[Désertot03] Mikaël DESERTOT, «Un modèle à composant pour les applications embarquées contraintes», DEA «Information, Systèmes et Communications» au Laboratoire LSR UMR CNRS 5526, Equipe Adèle (Univ. Grenoble 1), Juin 2003.

[Ducas04M] Olivier DUCAS, «Modèle de composant léger et dynamique pour les serveurs embarqués et légers», Master 2 Recherche «Systèmes d'Information» au Laboratoire LSR UMR CNRS 5526, Equipe Adèle (Univ. Grenoble 1), Juin 2004.

[Marin04M] Cristina MARIN, «Gestion de la cohérence et des transactions avancées liées au déploiement», Master 2 Recherche « Systèmes et Logiciels » au Laboratoire LSR UMR CNRS 5526, Equipe Adèle (Univ. Grenoble 1), Juin 2004, co-encadrement avec Noureddine BELKHATIR.

[Escoffier04M] Clément ESCOFFIER, «Étude et réalisation d'une plateforme domotique sur .Net». Magistère 2ème année d'Informatique au Laboratoire LSR UMR CNRS 5526, Equipe Adèle (Univ. Grenoble 1), Septembre 2004.

[Escoffier05M] Clément ESCOFFIER, «Stratégies de migration et de réplication des composants fonctionnels et non-fonctionnels dans le contexte du Edge-Computing». Master 2 Recherche « Systèmes et Logiciels » au Laboratoire LSR UMR CNRS 5526, Equipe Adèle (Univ. Grenoble 1), Juin 2005. co-encadrement avec Mikaël Désertot.

[Touseau06M] Lionel TOUSSEAU, «Accord de Niveau de Services dans les plateformes dynamiques de Services», Master 2 Recherche « Systèmes et Logiciels » au Laboratoire LSR UMR CNRS 5526, Equipe Adèle (Univ. Grenoble 1), Juin 2006.