Université Lille 1
Sciences et Technologies

HABILITATION À DIRIGER DES RECHERCHES

POUR DES

SYSTÈMES MULTI-ROBOTS MOBILES AUTONOMES & FLEXIBLES

présentée par

BOURAQADI SAÂDANI Mohammed Nouraddine (Noury)

Soutenance prévue le
Vendredi 06 décembre 2013

Devant la commission d'examen composée de

Davide BRUGALI
Professeur à Università Degli Studi Di Bergamo, Responsable du Software for Experimental Robotics Lab - Bergamo (Italie)

Jacques FERBER
Professeur à l'Université de Montpellier 2, Équipe SMILE (LIRMM) - Montpellier

Michel OCCELLO (Rapporteur)
Professeur à l'Université Pierre Mendes France (Grenoble 2), Responsable de l'équipe COSY (LCIS) - Grenoble

Rachid ALAMI (Rapporteur)
Directeur de Recherche CNRS, Responsable de l'équipe RIS (LAAS) - Toulouse

Stéphane DUCASSE (Garant)
Directeur de Recherche INRIA, Directeur Scientifique de l'INRIA Lille, Responsable de l'équipe RMoD - Lille

Theo D'HONDT (Rapporteur)
Professeur à Vrije Universiteit Brussel, Software Languages Lab - Bruxelles (Belgique)

© 2014 Tous droits réservés.

Table des matières

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

© 2014 Tous droits réservés.

doc.univ-lille1.fr

Table des figures

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

© 2014 Tous droits réservés.

doc.univ-lille1.fr

à Pascale
Myriam & Nadia

© 2014 Tous droits réservés.

doc.univ-lille1.fr

Remerciements

J'adresse mes sincères remerciements à messieurs Rachid Alami, Théo D'Hondt et Michel Occello. Bien que leurs responsabilités soient très prenantes, ils ont eu l'extrême amabilité de prendre le temps de rapporter ce mémoire et de participer à la soutenance.

Mes remerciements vont également à messieurs Davide Brugali et Jacques Ferber. Ils se sont organisés pour faire le voyage jusqu'à Douai et me faire le plaisir de participer au jury, malgré des agendas bien chargés.

Monsieur Stéphane Ducasse a depuis de nombreuses années été un soutien sans faille et un partenaire exceptionnel. Il s'est toujours débrouillé pour se rendre disponible chaque fois que j'ai eu besoin de lui et en particulier pour être le garant de cette HDR. Je souhaite par ces quelques mots lui exprimer toute ma gratitude.

J'adresse aussi mes remerciements à mes différents partenaires et amis, notamment messieurs Serge Stinckwich et Georg Heeg. J'espère bénéficier encore longtemps de leur amitié, ainsi que de leur générosité intellectuelle.

Durant les 12 dernières années passées à l'Ecole des Mines de Douai, j'ai pu côtoyer de multiples personnes bienveillantes dans les différents départements et services de l'école. Je voudrais leur exprimer ma profonde reconnaissance. Cette HDR n'aurait pu voir le jour sans leur concours aussi bien sur les plans scientifique, technique, administratif, logistique et psychologique.

Toutes ces personnes qui mettent de l'huile dans les rouages à tous les niveaux, sont bien trop nombreuses pour être toutes nommées ici et je les prie de m'en excuser. Je vais me restreindre à quelques unes seulement au sein du Département Informatique et Automatique.

Merci à Philippe Hasbroucq chef du département qui a été le premier à me faire confiance.

Merci à mes complices scientifiques Luc Fabresse, Jannik Laval et Arnaud Doniec.

Merci à Christine Delille toujours efficace et avec le sourire en prime.

Merci à Anthony Fleury qui en plus de sa propre activité d'enseignant-chercheur trouve le temps d'aider les collègues.

Merci à Muriel Morgan présente chaque fois que j'ai eu besoin d'elle, notamment quand il

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

fallait gérer l'option ISIC.

Merci à Cécile Labarre pour ses conseils avisés et ses encouragements. J'espère sincèrement que notre collaboration scientifique dans le cadre du projet CAIRE sera la première d'une longue série.

En dernier lieu, je voudrais remercier tous les étudiants qui ont porté les travaux que je présente ici.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

Résumé

Mots-clés : *Architectures logicielles de contrôle de robots ; Systèmes Multi-Agents Robotiques ; Modularité ; Coordination ; Adaptation.*

Les travaux présentés dans cette HDR traitent des logiciels de contrôle des robots *mobiles* et *autonomes*. Un tel robot est capable de réaliser sans assistance humaine, les missions qui lui incombent en se déplaçant dans un environnement partiellement connu et changeant. Plus spécifiquement, je m'intéresse à la *flexibilité des logiciels* de contrôle des robots. Il s'agit de proposer des solutions qui permettent de prendre en compte la variabilité de l'environnement d'exécution des robots, mais également la diversité des robots, de leurs équipements et des missions qui leur sont assignées. Ces travaux sont directement en lien avec l'activité du groupe de travail "Architectures de contrôle pour la robotique" (GT 4) du GdR Robotique.

Les solutions que nous avons proposées permettent d'introduire de la flexibilité à différentes échelles des applications robotiques, mais également à différents moments de leur cycle de vie. Ainsi, nous avons proposé des modèles de programmation et des langages fusionnant la programmation par aspects et par composants. Ces solutions permettent de développer de manière modulaire les architectures de contrôle des robots. La modularité favorise la réutilisation et la maintenabilité de ces logiciels. Elle offre donc de la flexibilité pendant le développement et la maintenance.

Nous avons ensuite étendu la flexibilité apportée par les composants, à l'exécution des architectures de contrôle des robots. Pour ce faire, nous avons proposé des architectures capables de s'auto-adapter dynamiquement. Ainsi, un robot peut changer totalement son comportement suivant l'environnement dans lequel il est situé ou la tâche en cours de réalisation. Les adaptations peuvent aussi être déclenchées suite aux variations des ressources disponibles comme la mémoire ou la batterie.

L'auto-adaptation requiert de remplacer du code en cours d'exécution et de sauvegarder l'état des composants afin d'éviter la perte de données provisoirement inutilisées. Afin de limiter la quantité de RAM utilisée, il est nécessaire de transférer ces données sur disque. Les mémoires

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

virtuelles des systèmes d'exploitation s'avèrent inappropriées, car les composants comme les objets sont organisés en graphes dont les nœuds ne sont pas forcément dans les mêmes pages mémoire. Nous avons donc proposé une nouvelle mémoire virtuelle directement pilotable par les applications.

Une autre facette complémentaire de mes recherches concerne la coordination au sein des systèmes multi-robots. Une partie de ces travaux est en lien direct avec l'utilisation des composants dans les architectures de contrôle des robots. En effet, nous avons introduit une dimension de coopération dans les mécanismes d'adaptation des robots. Cette extension repose sur l'échange de composants logiciels entre les membres d'une flotte. En plus de coordonner les décisions d'adaptation, cette solution permet de minimiser la taille du code embarqué sur chaque robot.

Le travail précédent est basé sur l'hypothèse de l'existence d'une infrastructure de communication. Ceci n'est pas toujours le cas, notamment dans les applications de robotique de sauvetage. Aussi, nous avons proposé une solution où les robots constituent eux-même un réseau *ad hoc* et le maintiennent pendant qu'ils effectuent les tâches de la mission à proprement parlé. Leurs déplacements sont contraints de manière à maintenir la connectivité du réseau et à compenser les éventuelles déconnexions.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

Première partie

Introduction

© 2014 Tous droits réservés.

doc.univ-lille1.fr

1. Historique et contexte des travaux

Après ma formation d'ingénieur, j'ai souhaité acquérir de nouvelles connaissances scientifiques et techniques. Aussi, j'ai choisi d'effectuer un DEA, puis un doctorat dans un laboratoire qui aborde des problématiques scientifiques avec des retombées industrielles. C'est ainsi que j'ai rejoint le département informatique de l'École des Mines de Nantes, dans lequel j'ai eu la chance de voir le fonctionnement d'une équipe bien établie avec à sa tête un professeur (Pierre Cointe) de renommée internationale. Parallèlement, soucieux de garder un contact direct avec le monde de l'entreprise, je suis intervenu en tant que consultant indépendant dans différents projets en partenariat avec des infographistes et des architectes.

En 2001, à l'issue de mon post-doc effectué dans le cadre d'un projet avec France Télécom R&D, j'ai décidé de continuer à travailler dans le monde académique. J'ai donc décliné l'offre de rejoindre l'équipe Eclipse d'IBM St Nazaire en faveur d'un poste d'enseignant-chercheur au sein du DÉPT. INFORMATIQUE ET AUTOMATIQUE (DIA) de l'ÉCOLE DES MINES DE DOUAI. Le DIA venait d'être créé par rassemblement du département Informatique et de la quasi-totalité du département Productique. L'équipe informatique regroupait exclusivement des techniciens et des ingénieurs dédiés à l'enseignement et à la prestation de services, tant pour les services de l'état que pour le compte d'entreprises.

C'est dans ce contexte que je me suis attelé à la définition d'une thématique de recherche originale ainsi qu'au développement et à l'animation d'une équipe en devenir. J'étais conscient du défi que représentait mon projet, d'autant plus que j'étais jeune chercheur (2 ans après ma thèse). Mais, je tenais à vivre cette expérience rare et enrichissante que constitue le démarrage d'une nouvelle activité de recherche.

Douze ans plus tard, nous sommes 4 enseignants-chercheurs permanents en informatique : Arnaud Doniec, Luc Fabresse, Jannik Laval et moi même. En terme de co-encadrement de thèses, nous comptons 7 doctorants dont 5 ayant soutenus. Enfin, le groupe que j'anime a accueilli à ce jour 4 post-doctorants et 5 ingénieurs.

Le DIA contribue aujourd'hui par ses missions de formation, de recherche, de transfert de technologie, d'ingénierie et d'assistance technique à une large diffusion des Sciences et Technologies de l'Information et de la Communication (STIC) dans différents secteurs d'activités (industrie, services, transport, administration, enseignement...). Ces dernières années, il s'est structuré et renforcé sur le plan de la recherche, en mettant en place une organisation originale s'appuyant sur une Unité de Recherche et sur un Centre de Ressources, d'Ingénierie et de Développement.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

© 2014 Tous droits réservés.

doc.univ-lille1.fr

2. Robotique Mobile et Autonome

La seconde moitié du 20ème siècle a été profondément marquée par l'informatique. Nos sociétés ont été transformées avec ce qui est désormais appelé la révolution numérique. C'est le produit de la conjonction de deux phénomènes. Le premier est la réalisation de la conjecture de Moore [1] et la miniaturisation des ordinateurs qui en découle. Ces machines, dont les représentants les plus saillants sont les *smartphones* et les tablettes, ont désormais des prix abordables, ce qui a contribué à leur large diffusion et leur pénétration du quotidien de tout un chacun. Cette diffusion a été accélérée avec le développement des réseaux de télécommunication, qui constitue le deuxième phénomène à l'origine de l'ère du numérique. La quasi totalité des ordinateurs vendus à ce jours sont équipés d'interfaces réseau sans fil. Il est donc aisé de faire communiquer différents équipements voisins. Et grâce à l'Internet, des machines géographiquement distantes peuvent échanger des informations et réaliser des traitements les unes pour le compte des autres.

A l'instar du 20ème siècle, le 21ème va certainement connaître une révolution technologique liée à l'informatique. Il s'agit de la robotique *mobile* et *autonome* [SK08].

2.1 Définition de la robotique mobile et autonome

Le qualificatif *mobile* exclut les robots industriels traditionnels utilisés depuis les années 1950 dans les chaînes de production [Wik12a]. Ces robots sont confinés dans un espace totalement maîtrisé inaccessible aux humains. Toutes les situations et leurs enchaînements sont connus à l'avance. Les développeurs des logiciels de contrôle de tels robots peuvent décrire donc avec précision les séquences d'actions à réaliser pour chaque situation. Par opposition, un robot *mobile* évolue dans un environnement ouvert, partiellement connu et continuellement changeant. Il est donc quasiment impossible de prédire toutes les situations auxquelles le robot sera confronté et encore moins leurs successions du fait de l'explosion combinatoire.

Le qualificatif *autonome* exclut quant à lui les robots télé-opérés. La télé-opération nécessite de mobiliser une voire plusieurs personnes pour piloter un robot [Wik12d]. C'est l'usage qui en est fait dans le monde militaire qui est le plus médiatisé. Ainsi, en Irak et en Afghanistan, l'armée américaine a déployé aussi bien des *drones* [Wik12e] dans les airs, que des robots terrestres (comme le Packbot [Wik12b]). Ces robots sont pilotés à distance (voire de très loin pour les drones) pour notamment explorer des zones risquées ou manipuler des objets dangereux

1. Qualifiée de *loi* de Moore par abus de langage.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

(typiquement des explosifs). Par opposition, un robot *autonome* doit être doté d'un logiciel qui lui permet de prendre des décisions sans intervention humaine. Il doit donc mener à bien, seul, les missions qui lui incombent et ce, en toute sécurité pour les biens, les hommes, ainsi qu'en préservant sa propre intégrité.

La combinaison de la mobilité et de l'autonomie constitue un des grands défis de la robotique. Néanmoins, le niveau de connaissances actuel permet d'ores et déjà d'envisager la réalisation de produits, notamment pour le grand public. Un des plus célèbres représentants est le robot-aspirateur qui connait un grand succès commercial. La société iRobot, leader du marché, annonce avoir vendu 6 millions de Roomba depuis son lancement en 2002 [Wik12c]. Cet exemple et bien d'autres constituent les prémices de la révolution robotique de notre société. Les robots vont d'abord conquérir les tâches jugées fastidieuses, dégradantes ou dangereuses [2] avant d'envahir le reste de nos activités.

2.2 Paysage socio-économique de la robotique

Différents organismes privés ou publics ont d'ores et déjà détecté le tournant robotique et commencent à le préparer activement. Le Japon et les Etats Unis sont précurseurs et co-leaders dans le domaine. Ils sont suivis par l'Union Européenne et par la Corée du Sud qui a massivement investi dans le domaine.

L'industrie nippone mène depuis plusieurs décennies une activité de R&D avec pour but de développer des robots compagnons. Les travaux les plus représentatifs concernant la robotique humanoïde avec notamment le robot Asimo de Honda [asi13] et la série de robots HRP développés dans le cadre d'un partenariat entre la société Kawada Industries et l'institut national japonais AIST (Advanced Industrial Science and Technology) [Kaw13].

Pour ce qui est des **Etats Unis d'Amérique**, deux moteurs stimulent le secteur de la robotique. Il y a d'une part la conquête spatiale avec la NASA. Les risques liés aux vols habités conjugués à une réduction drastique des budgets a conduit au recours à la robotique. Les exemples les plus médiatisés concernent l'exploration de la planète Mars avec des robots de type *rover* comme Curiosity [NAS13]. Le deuxième moteur de la robotique aux Etats Unis est le DARPA avec ses challenges médiatisés d'abord pour les véhicules autonomes [DAR04, DAR05, DAR07], puis pour les robots humanoïdes [DAR14]. A cela s'ajoute les applications militaires avec les drones et autres robots de déminage qui sont déjà sur le terrain ainsi que les exo-squelettes et les prothèses robotiques. Enfin, citons le lancement par le président Obama en juin 2011 de la *National Robotics Initiative (NRI)* [NSF11]. C'est l'un des 4 piliers du programme *Advanced Manufacturing Partnership* qui vise à financer des collaborations entre industrie et recherche dans le domaine de la robotique.

L'Union Européenne n'est pas en reste. Afin de soutenir la recherche en robotique, l'Union Européenne a lancé le réseau d'excellence EURON (EUropean RObotics research Network) depuis 1998 [Eur12b]. Le pendant industriel de ce réseau académique est la plate-forme technologique EUROP constituée en 2005 [Eur12c]. Cet édifice est complété par l'action de coordina-

2. Les 3D : *Dull, Dirty, Dangerous.*

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

tion EUROBOTICS démarrée en 2010 et dont le but est de faire la passerelle entre le monde académique et le monde industrielle [euR12a]. Cette action a été pérennisée en 2012, puis qu'EUROBOTICS a été transformée en une association à but non-lucratif.

En complément à ces actions trans-nationales, différents pays prennent des initiatives en faveur de la robotique. C'est le cas par exemple du ROBOCLUSTER [rob12a] au Danemark et du ROBOTDALEN [rob12c] en Suède et du NCCRROBOTICS en Suisse. Ces différentes structures visent à favoriser les collaborations entre la recherche académique et l'industrie.

En France, la robotique est considérée comme une technologie clé par le ministère de l'industrie [dl11]. Si cette citation n'a eu lieu qu'en 2011, elle a été précédée par différentes actions. L'une des premières est la création du Groupement de Recherche (GDR) en Robotique [CNR12] en 2007. Ce GDR vise à structurer et à fédérer la recherche académique en robotique tout en facilitant les interactions avec le monde industriel au travers du "Club des partenaires". De leur côté, les entreprises de la robotique personnelle et de services ont créé en 2009 le syndicat professionnel SYROBO [Syr12]. Deux années plus tard, SYROBO lance le salon INNOROBO dédié à la robotique [Inn12]. Cette manifestation, désormais annuelle, attire des exposants du monde entier, malgré un contexte de crise économique. Elle constitue ainsi un bon indicateur du dynamisme de cette branche de l'industrie.

Cela n'a pas échappé au monde de la finance qui manifeste son intérêt pour les robots avec la naissance du fond d'investissement français ROBOLUTION qui vise de financer des projets pour un total de 60 millions d'euros [rob12b]. Créé en janvier 2012, ce premier fond d'investissement en Europe totalement dédié à la robotique est un indicateur fort quant au potentiel économique de ce secteur.

Enfin, le monde politique est également sensibilisé au potentiel de la robotique pour le développement économique. Une étude sur la robotique a été menée conjointement par le PIPAM (Pôle Interministériel de Prospective et d'Anticipation des Mutations Economiques) et la DGCIS (Direction Générale de la Compétitivité de l'Industrie et des Services). Le rapport publié au printemps 2012 considère la robotique comme un élément clé de la compétitivité des entreprises industrielles [PD12]. Il déplore le peu d'actions publiques pour soutenir spécifiquement ce secteur. Un an plus tard, Arnaud Montebourg ministre du redressement productif, lance le plan robotique intitulé "France Robots Initiatives" [dRPdlSedlR13]. Il vise à hisser la France parmi les 5 pays leaders mondiaux de la robotique mobile à l'horizon 2020. Ce plan mobilise une enveloppe de 100 millions d'euros qui vise à soutenir les entreprises (notamment les startups) et à favoriser les échanges entre les laboratoires et les entreprises.

2.3 Références bibliographiques du chapitre

[asi13] Asimo by honda. http://asimo.honda.com/, mai 2013.

[CNR12] CNRS. Groupement de recherche en robotique. http ://www.gdr-robotique.org/, mar 2012.

[DAR04] DARPA. Grand challenge 2004. http://archive.darpa.mil/grandchallenge04/, 2004.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[DAR05] DARPA. Grand challenge 2005. http://archive.darpa.mil/grandchallenge05/, 2005.

[DAR07] DARPA. Urban challenge 2007. http://archive.darpa.mil/grandchallenge/, 2007.

[DAR14] DARPA. Humanoid challenge 2014. http://www.theroboticschallenge.org, 2014.

[dl11] Ministère de l'Industrie. Technologies clés 2015. http://www.industrie.gouv.fr/tc2015/, 2011.

[dRPdlSedlR13] Ministère du Redressement Productif and Ministère de l'Enseignement Supérieur et de la Recherche. France robots initiatives, Mars 2013.

[euR12a] euRobotics. eurobotics coordination action. http ://www.eurobotics-project.eu/, mar 2012.

[Eur12b] Euron. European robotics research network. http ://www.euron.org, mar 2012.

[Eur12c] Europ. European robotics technology platform. http ://www.robotics-platform.eu, mar 2012.

[Inn12] Innorobo. Innovation robotic summit. http ://www.innorobo.com, mar 2012.

[Kaw13] Industries Kawada. Humanoid robot hrp-4. http://global.kawada.jp/mechatronics/hrp4.html, May 2013.

[NAS13] NASA. Mars science laboratory, the next mars rover. http://www.nasa.gov/mission_pages/msl/index.html, May 2013.

[NSF11] NSF. National robotics initiative. http://www.nsf.gov/nri, Jun 2011.

[PD12] PIPAM and DGCIS. Le développement industriel futur de la robotique personnelle et de service en france, April 2012.

[rob12a] Robocluster. http ://en.robocluster.dk/, mar 2012.

[rob12b] Robolution capital. http ://www.robolutioncapital.com/, mar 2012.

[rob12c] Robotdalen. http ://www.robotdalen.se/en/, mar 2012.

[SK08] Bruno Siciliano and Oussama Khatib, editors. *Handbook of Robotics*. Springer, 2008.

[Syr12] Syrobo. Syndicat de la robotique de service. http ://www.syrobo.org/, mar 2012.

[Wik12a] Wikipedia. Industrial robot. http://en.wikipedia.org/wiki/Industrial_robot, mar 2012.

[Wik12b] Wikipedia. Packbot. http://en.wikipedia.org/wiki/Packbot, mar 2012.

[Wik12c] Wikipedia. Roomba. http://en.wikipedia.org/wiki/Roomba, mar 2012.

[Wik12d] Wikipedia. Telerobotics. http://en.wikipedia.org/wiki/Telerobotics, mar 2012.

[Wik12e] Wikipedia. Unmanned aerial vehicle. http://en.wikipedia.org/wiki/Unmanned_aerial_vehicle, mar 2012.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

3. Travaux de Recherche

La robotique mobile et autonome est la préoccupation centrale de mes travaux. Plus spécifiquement, je m'intéresse à la facette logicielle des robots mobiles et autonomes. Afin d'alléger le discours, nous utiliserons à partir de maintenant le mot *robotique* (respectivement le mot *robot*) sans qualificatif pour désigner la *robotique mobile et autonome* (respectivement *robot mobile et autonome*).

3.1 Problématique abordée : Flexibilité des logiciels pour la robotique

3.1.1 Besoin de flexibilité

Les logiciels de contrôle des robots doivent répondre à de multiples contraintes [KS08, Bru07, Mal11]. Ils doivent prendre en compte la complexité, l'imprévisibilité et la dynamicité de l'environnement dans lequel baignent les robots. La diversité des robots et de leurs équipements (capteurs, actionneurs) est également une difficulté à laquelle sont confrontés les développeurs de logiciels pour contrôler les robots. A cela s'ajoute la diversité des missions que les utilisateurs assignent aux robots.

Une caractéristique commune à ces différentes contraintes est la *variabilité*. L'environnement est changeant. Les robots peuvent être différents par leur niveau d'équipement et les caractéristiques des matériels et logiciels qu'ils embarquent. Il en est de même des missions qui peuvent différer par leur objectif, le nombre de robots utilisés, ou encore par le degré d'homogénéité de la flotte robotique.

Afin de prendre en compte cette variabilité intrinsèque à la robotique, nous nous intéressons à la *flexibilité* des moyens de production des logiciels de contrôle des robots. Il s'agit de fournir aux développeurs des solutions pour adapter *facilement* les fonctionnalités d'un logiciel existant (par exemple le remplacement d'un lidar par un scanner 3D à infra-rouges de type Kinect sur la plate-forme robotique) ou d'en ajouter de nouvelles (par exemple ajout d'un bras manipulateur à une plate-forme mobile).

Nous abordons la question de la flexibilité de la production de logiciels robotiques selon deux dimensions liées, mais distinctes :

- le *processus* de production du logiciel ;
- l'*échelle* à laquelle travaillent les développeurs.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

3.1.2 Flexibilité du processus de production du logiciel de contrôle de robot

Suivant la dimension processus de production des logiciels robotiques, la question flexibilité est abordée aussi bien au niveau démarche globale qu'au niveau de chacune des différentes phases du cycle de vie des logiciels. Cela consiste à se poser pour chaque phase deux questions :

- Comment prendre en compte des besoins et des contraintes divers et changeants tout en minimisant l'effort des acteurs humains ?
- Quels outils et abstractions utiliser pour que le logiciel produit puisse opérer dans différents contextes ?

Par exemple, en phase de conception, on se posera les questions du paradigme à utiliser, de l'architecture et des schémas de conception [1]. En phase de développement, l'effort fourni par les développeurs et la flexibilité du logiciel produit dépendent de l'environnement de développement ainsi que du langage de programmation. Les bénéfices dus au langage de programmation varient suivant qu'il est plus ou moins de haut-niveau, avec des constructions plus ou moins adaptées au domaine applicatif et plus ou moins proches du paradigme utilisé en conception. Les questions liées à la flexibilité touchent également les autres phases telles que le déploiement, l'exécution et l'administration. Bien entendu, le paradigme choisi en conception et le langage de programmation adopté pour le développement ont un impact conséquent sur la flexibilité à ces phases. Mais, les outils et les infrastructures utilisés ont également une importance loin d'être négligeable.

3.1.3 Flexibilité et échelle de travail des développeurs de systèmes robotiques

L'échelle de travail lors de l'étude et la réalisation de systèmes robotiques correspond au niveau d'abstraction adopté par les développeurs. Cette dimension a son importance dans les *systèmes multi-robots*, c'est-à-dire des systèmes où plusieurs robots sont conjointement utilisés pour la réalisation d'une unique mission. Lorsqu'on s'intéresse à un tel système au niveau macroscopique, on se penche sur la coordination et la coopération entre plusieurs robots. En revanche, le niveau microscopique concerne le logiciel de contrôle spécifique à un unique robot.

Bien entendu, les deux niveaux des systèmes robotiques sont liés. Le logiciel de contrôle de chaque robot a un impact sur le système multi-robots. Inversement, la structuration de robots en un système cohérent, implique de contraindre le fonctionnement du logiciel de contrôle de chaque robot, afin d'atteindre l'objectif global à toute la flotte robotique.

La préoccupation de flexibilité touche les deux niveaux des systèmes multi-robots. La flexibilité du logiciel de contrôle d'un robot détermine par exemple la facilité avec laquelle on pourrait utiliser le même robot dans des systèmes différents ou à jouer des rôles différents dans un même système. La flexibilité d'un système multi-robots englobe quant à elle la possibilité de faire varier le nombre de robots, leur infrastructure matérielle et logicielle, ainsi que la répartition des rôles sans mettre en péril la mission à réaliser.

1. *design pattern.*

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

3.2 Approche adoptée

Afin de répondre au besoin de flexibilité dans les logiciels pour la robotique, nous avons adopté une approche structurée en 3 axes complémentaires :

- Modèles de programmation modulaire
- Langages dynamiques
- Systèmes multi-agents

Dans cette section, nous décrivons brièvement les différents axes de notre approche. Nous montrons aussi la pertinence de chaque axe vis à vis de notre problématique. Ainsi, nous faisons le lien entre les axes de notre approche et les dimensions de la flexibilité des logiciels pour la robotique.

3.2.1 Modèles de programmation modulaire

Définition

La *modularisation* consiste à fragmenter un logiciel [Par72] dans le but d'améliorer la flexibilité du processus de développement et de maintenance. L'idée sous-jacente est que développer un fragment [2] est plus simple que de s'attaquer frontalement à un monolithe. En effet, les *fragments* constituant un logiciel peuvent être réalisés de manières différentes (développement *ex nihilo*, réutilisation ou adaptation de l'existant), par des personnes différentes, éventuellement en parallèle.

La modularisation est un *graal* poursuivi par les travaux en génie logiciel depuis leur origine. Elle a débouché sur une succession de modèles dont le représentant phare est actuellement la *programmation par objets* [MNC+89]. Mes travaux ont porté sur deux paradigmes post-objet : les *aspects* et les *composants*.

Aspects

Le concept d'aspect est né comme une approche de programmation : la *programmation par aspects* [3] [KLM+97, BL01]. Il a été ensuite généralisé à tout le cycle de développement logiciel. On parle désormais de *développement logiciel orienté aspect* [4] [FEAC05].

L'idée maîtresse de ce paradigme consiste à isoler et à expliciter des fragments, appelés *aspects*, qui sont transversaux à d'autres fragments du même logiciel. Un aspect comporte des traitements qui permettent de modifier la structure ou l'exécution d'autres fragments en des points particuliers appelés *points de jonction* (*join points*). La modification de la structure peut consister par exemple en l'insertion de champs ou de méthodes dans certaines classes. La modification de l'exécution quant à elle se traduit par l'insertion de traitements dans le flot de l'exécution,

2. Nous utilisons le mot *fragment* au lieu de *module* qui est largement utilisé en informatique avec des sens plus restreints que ce que nous entendons.
3. *AOP* pour *Aspect-Oriented Programming*
4. *AOSD* pour *Aspect-Oriented Software Development*

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

par exemple avant ou après l'exécution de certaines méthodes, la lecture ou encore l'écriture de certains champs.

Les aspects correspondent typiquement à des propriétés dites extra-fonctionnelles ou d'infrastructure comme par exemple le protocole de communication distante ou la politique de stockage des données. Avec les paradigmes plus classiques tels que la programmation par objets, de telles facettes se trouvent mélangées et dispersées dans les logiciels. En représentant de telles propriétés sous forme d'aspects, il devient possible d'isoler chacune d'elles, ce qui offre une nouvelle dimension de modularisation.

Composants

Un *composant* est un fragment logiciel dont on a explicité à la fois les *ports* de connexion avec les autres composants et les dépendances vis à vis du contexte d'exécution (OS, bibliothèques, ...) [SGM02]. Avec ce paradigme, une application est un assemblage de composants complémentaires, dont les ports compatibles sont connectés. L'architecture définie lors de la conception est ainsi directement matérialisée lors du développement.

Le but ultime du paradigme composants est de restreindre la réalisation d'un logiciel à la définition d'une architecture. Un ensemble d'outils se chargerait ensuite de retrouver les composants adéquats et de les assembler pour produire l'exécutable. Les composants seraient recherchés dans des bibliothèques [5] et choisis sur la base exclusive de leurs spécifications.

Ainsi, l'approche à base de composants promeut une démarche *descendante* pour réaliser des applications. Elle s'appuie pour cela sur *l'inversion de contrôle*. Avec les objets, chaque objet se charge de construire lui même ses connexions vers les autres objets. Alors qu'avec les composants, chaque composant se contente de spécifier ce qu'il requiert sous forme de ports. C'est l'utilisateur du composant qui décide des connexions.

Jusqu'à présent, nous avons parlé de modularisation *horizontale*, dans le sens où tous les composants se situent au même niveau. Cependant, nombre de modèles de composants dits hiérarchiques [LW05, BCS02, FDH08] encouragent également une modularisation *verticale*. Il s'agit de permettre de construire des composants *composites*, autrement dit des composants eux-mêmes constitués d'autres composants.

Modularisation et flexibilité des logiciels

La modularisation qu'elle soit à base d'aspects ou de composants permet aux développeurs de diviser le logiciel "pour mieux régner". Lors du développement, un logiciel complexe serait ainsi divisé en fragments plus simple à réaliser individuellement. Certains fragments déjà existants et donc ayant déjà fait leurs preuves pourraient être réutilisés. Cela permettrait d'accélérer le développement tout en assurant un bon niveau de fiabilité. Par ailleurs, la production peut être échelonnée de différentes manières dans le temps, suivant les priorités du client (commencer par les fragments les plus critiques) et la taille de l'équipe (parallélisation du travail). Enfin, l'évolution d'un logiciel devrait idéalement se traduire par le remplacement de certains fragments et la révision d'une partie des connexions entre fragments.

5. On parle de *composants sur étagère* (*Components Off The Shelf (COTS)*)

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

La modularisation ouvre également la voie pour la flexibilité du déploiement et de l'exécution du logiciel. Certains fragments peuvent être remplacés *à chaud* pendant que les autres continuent à fonctionner. Ces remplacements peuvent être effectués pour la mise à jour d'applications, afin de minimiser les temps d'arrêt [6]. Remplacer dynamiquement des fragments peut également être nécessaire pour des applications sensibles au contexte [7] [ADB+99]. C'est le cas des robots mobiles qui, du fait de leurs déplacements, sont confrontés à un environnement variable dans le temps et dans l'espace. Ils doivent alors changer de comportement pour prendre en compte le contexte dans lequel ils opèrent.

3.2.2 Langages dynamiques

Définition

Comme pour nombre de notions en informatique, il n'existe pas de définition catégorique sur ce qu'est un langage de programmation dynamique [Erd08, Pau07]. Néanmoins, il est admis qu'un tel langage réalise à l'exécution des traitements que d'autres langages effectuent à la compilation [Wik12].

Le flou de cette définition provient selon nous du fait qu'il existe un continuum entre les langages fortement statiques et ceux fortement dynamiques. Un langage est plus ou moins dynamique suivant qu'il réalise plus ou moins de traitements à l'exécution. L'archétype de ces traitements est la vérification de type. Cependant, d'autres mécanismes du langages peuvent être effectués à l'exécution comme par exemple la résolution de nom ou le remplacement de code à chaud. De telles capacités reposent sur la propriété de *réflexion*.

La Réflexion

D'après Brian Smith, la réflexion est "*la capacité d'une entité à s'auto-représenter et plus généralement à se manipuler elle-même, de la même manière qu'elle représente et manipule son principal sujet.*" [Smi84]. La réflexion [8] caractérise donc la propriété d'un système capable de raisonner et d'agir sur lui-même [Zim96].

Deux formes de réflexion peuvent être distinguées : *l'introspection* et *l'intercession*. Lorsqu'un système réflexif se contente de s'observer et de répondre à des questions sur son état, nous parlons *d'introspection*. Quand il s'auto-modifie, nous parlons *d'intercession*.

Introspection et intercession se basent sur la *réification*. Celle-ci consiste à représenter les éléments constituants un système sous forme de données manipulables par ce même système. Ainsi, la réification est utilisée pour la construction de l'auto-représentation d'un système réflexif.

Suivant la nature des entités réifiés, on distingue la *réflexion structurelle* et la *réflexion comportementale* [Coi88]. "Cette distinction a été faite principalement parce qu'il est beaucoup plus facile d'implanter la réflexion de structure efficacement que la réflexion comportementale." [Mal97, page 74]

6. *down-time*
7. *Context Aware*
8. Certains auteurs utilisent le terme *réflexivité*. Nous préférons *réflexion* plus proche de l'anglais *reflection*.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

La réflexion structurelle d'un système correspond à la face statique du système puisqu'elle donne accès aux structures de données qui constituent le système. Elle se traduit par la réification des entités utilisées pour la construction du système. Par exemple, dans un langage à classes de la famille de SMALLTALK, la réflexion de structure peut se manifester par la possibilité de modifier la représentation des classes.

La réflexion comportementale correspond à la face dynamique du système. Elle se traduit par la réification des entités utilisées pour le fonctionnement du système. Par exemple, la réflexion de comportement peut se manifester dans un système par la réification de la pile d'exécution où de l'interprète du système.

Langages dynamiques et flexibilité des logiciels

En déférant à l'exécution certains traitements, les langages dynamiques introduisent un support à la flexibilité dans le processus de développement. Ils permettent d'analyser l'exécution des programmes et éventuellement de les modifier à chaud, sans besoin de redémarrage. Les développeurs peuvent alors traquer les bogues et les corriger sans perdre les contextes où les dysfonctionnements sont observés. Nous sortons ainsi des cycles classiques comme le cycle en V ou en cascade. La frontière entre les phases de codage et de test devient flou, ce qui rejoint les approches agiles et notamment le développement dirigé par les tests [9] [Bec01].

La dynamicité des langages offre également un niveau de flexibilité concernant les constructions langagières mêmes. Il est possible d'introduire de nouvelles constructions, ou de modifier celles déjà existantes en fonction des besoins des développeurs. Les métaclasses explicites [Coi87, BC89, Bou99a, Bou04a] et les traits [SDNB03, DWBN07, BDNW08] sont deux exemples représentatifs de telles modifications.

Les capacités réflexives caractéristiques des langages dynamiques rendent les outils de développement plus flexibles. En effet, de tels outils manipulent directement des entités d'exécution de premier ordre qui peuvent être introspectées ou modifiées par l'intermédiaire de requêtes *méta* fournies par le langage lui-même. Dès lors, il est aisé d'adapter les outils ou d'en développer de nouveaux.

La réflexion facilite également la maintenance et l'évolution des systèmes réflexifs. En effet, les développeurs disposent de plus de flexibilité et peuvent intervenir à deux niveaux différents. Un logiciel réflexif dispose d'une part du *niveau de base* qui correspond au services applicatifs d'un logiciel et d'autre part du *méta-niveau*, dans lequel les mécanismes d'exécution sont matérialisés et donc accessibles aux développeurs. Les accès séparés à ces deux niveaux constituent une nouvelle dimension pour améliorer la modularité des systèmes. Elle peut même servir de fondation à la programmation aspects [BL05].

9. Test Driven Development

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

3.2.3 Systèmes multi-agents

Définition

Le domaine des Systèmes Multi-Agents (SMA) s'intéresse à la conception de réseaux d'entités autonomes et coopérantes [Fer95, WJ95, BD01]. Ces entités appelées *agents* peuvent être purement logicielles. Elles peuvent être également physiquement incarnées. Les agents se caractérisent par une grande autonomie de décision. Cette liberté couvre leurs interactions avec les autres agents, ainsi que leurs actions sur l'environnement et leur interprétation de celui-ci.

Coordination et coopération dans les systèmes multi-agents

Le recours aux SMA est particulièrement approprié dans les situations où différents agents doivent se *coordonner* et éventuellement *coopérer* afin d'atteindre des objectifs individuels ou un but partagé. La *coordination* a typiquement lieu quand il s'agit d'utiliser une ressource à tour de rôle ou d'ordonner les actions réalisées par chaque agent. La *coopération* va plus loin dans le sens où les agents partagent au moins un sous-ensemble de leurs buts et se coordonnent pour les réaliser.

Coordination et coopération requièrent que les agents communiquent. Cette communication est indirecte quand l'échange d'information passe par l'environnement. Ce mécanisme appelé *stigmergie*, est utilisé chez les fourmis ou les termites qui déposent des phéromones dans l'environnement. Leurs congénères réagissent aux phéromones en adaptant leurs comportements.

Bien que débouchant sur des résultats très intéressants [DCL95], la stigmergie présente le défaut de ne permettre qu'une communication limitée. Les messages sont de bas niveau et véhiculent peu d'information. Par ailleurs, la transposition sur des robots reste délicate du fait de la difficulté de produire des phéromones robotiques.

Aussi, je me suis tourné vers des modèles qui reposent sur une communication directe. Ils permettent l'échange d'information complexe sur la base de *protocoles* qui spécifient les types de messages échangés et les réactions possibles à chacun de ces types [BD01]. Un exemple de protocole est célèbre *Contract Net*. Il permet à un groupe d'agents de partager la charge de travail en échangeant des informations sur les tâches à réaliser et les capacités des agents prêts à contribuer.

SMA et robotique

Les SMA peuvent être utilisés en robotique à deux échelles complémentaires. Sur un plan microscopique, le logiciel de contrôle d'un robot peut être structuré sous la forme d'un SMA, où chaque agent régirait une ou plusieurs parties logiques ou physiques (capteurs, actionneurs, calculateurs) [OPP96]. Cela revient à utiliser les agents comme vecteur de la modularisation des logiciels, complémentaires aux composants et aux aspects.

Mes travaux ont porté sur une utilisation des SMA au niveau *macroscopique*. A cette échelle, le parallèle entre les SMA et les Systèmes Multi-Robots (SMR) [Par08] se fait naturellement. En effet, un SMR peut être vu comme un SMA où chaque agent est physiquement incarné dans un

© 2014 Tous droits réservés.
doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

corps robotique. Les différents robots constituent une "société" d'agents qui doit accomplir une mission commune.

SMA et flexibilité des systèmes

Les SMA représentent un vecteur intéressant pour la flexibilité à une échelle différente de celle des deux précédentes approches. Alors que la modularité et les langages dynamiques nous permettent d'aborder la flexibilité au niveau "microscopique" d'un agent pris individuellement, les SMA mettent l'accent sur les interactions entre des agents.

Une facette intéressante des SMA est qu'ils permettent d'ajuster le comportement global du système en faisant varier l'organisation de la société d'agents, et en particulier les rôles des agents et leurs modes d'interactions. Ces ajustements peuvent être réalisés par les concepteurs pour traiter des missions différentes par les tâches à réaliser ou par l'environnement. Ils peuvent aussi être la conséquence de multiples décisions prises individuellement par les agents en fonction de leur charge de travail ou de leurs ressources.

Un autre intérêt des SMA est la redondance qui favorise la tolérance aux pannes. Plusieurs agents peuvent jouer le même rôle ou bien disposer des mêmes capacités. La défaillance de l'un peut donc être comblée par les autres.

Enfin, le passage à l'échelle est également caractéristique des SMA. Les organisations d'agents peuvent reposer sur des systèmes de prise de décision locale, où chaque agent décide seul des actions qu'il doit réaliser. Il prend en compte pour cela son voisinage constitué d'autres agents et de la partie de l'environnement dans laquelle il est situé. En supprimant le besoin d'un responsable central, cette approche supprime un goulot d'étranglement de communication et de calcul. Il n'y a plus de nœud critique dont la panne ou la déconnexion handicape la totalité du système, comme c'est le cas dans les architectures centralisées.

3.3 Directions de recherche suivies

L'approche que j'ai adoptée dans mes travaux se décline en 3 axes : modularité, langages dynamiques et SMA (voir la section 3.2, page 25). Ceux-ci constituent le cadre de pensée qui a guidé mes travaux sur les systèmes multi-robots flexibles. C'est dans ce cadre que se situent les trois directions de recherche que j'ai suivies et que je décrit dans la présente section.

3.3.1 Cartographie des travaux

Dans notre quête de systèmes multi-robots flexibles, nous avons suivi trois directions de recherche complémentaires :

– **Langages et modèles de programmation réflexifs**. Cette direction vise la *modularisation* des applications robotiques, en s'appuyant sur la réflexivité directement, ou indirectement. L'utilisation indirecte de la réflexion consiste à étendre un langage réflexif avec les concepts liés à la programmation par composants et par aspects.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

– **Infrastructures modulaires**. Cette direction a pour objectif de proposer un ensemble cohérent d'infrastructures qui ciblent différents moments dans le cycle de vie des logiciels de contrôle des robots. Le but est de faciliter la mise au point de ces logiciels, leur packaging, leur déploiement ainsi que la gestion des ressources à l'exécution en utilisant la *modularisation* et les langages *dynamiques*.
– **Modèles de coordination décentralisée**. L'objet de cette direction est la coordination de flottes robotiques vues comme des *systèmes multi-agents* physiques qui prennent des décisions de manière locale et autonome. L'idée est de fournir aux développeurs un cadre pour faire émerger l'organisation du système, sur la base de règles suivies par chaque robot du système.

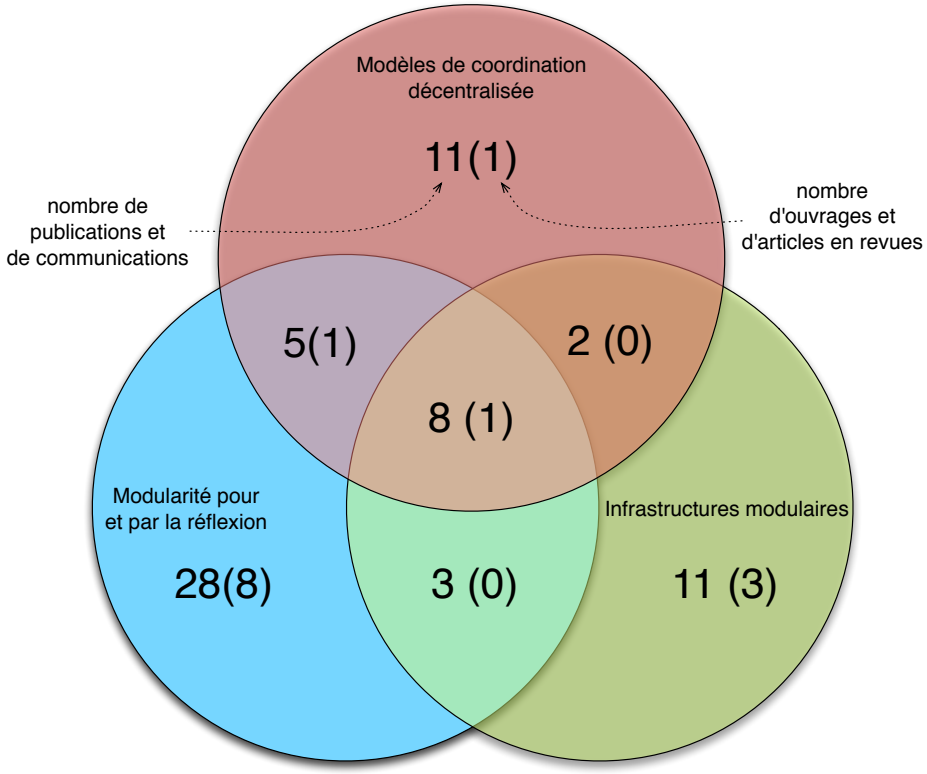

FIGURE 3.1 – Principales publications et communications classées par thème

Comme le montre la carte des publications et communications [10] donnée par la figure 3.1, mes travaux ont porté sur chacune des 3 précédentes directions prises séparément, mais également sur leurs intersections. Chaque cercle comporte au centre le nombre de papiers qui traitent exclusivement du thème en question. Les articles qui relèvent de plusieurs thématiques apparaissent quant à eux dans les intersections. Les nombres entre parenthèses représentent le nombre d'ouvrages, chapitres et articles en revue pour chaque partie.

10. Le détail est listé en section B.4

© 2014 Tous droits réservés.
doc.univ-lille1.fr

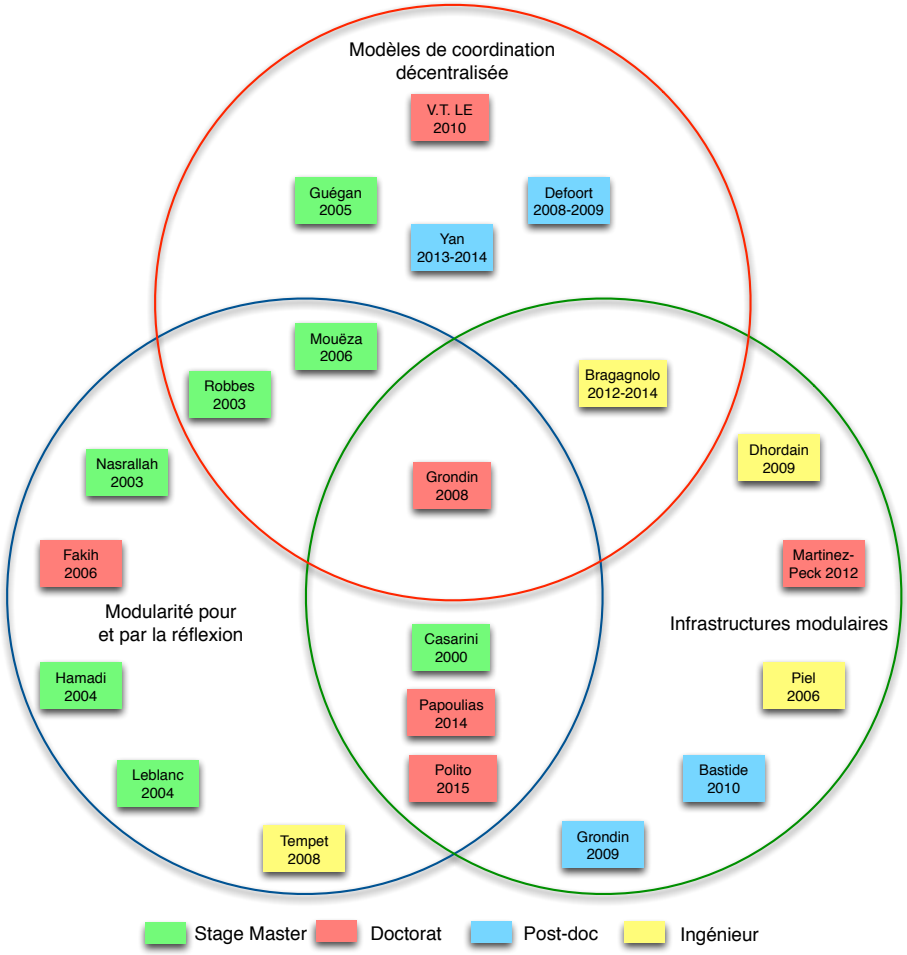

FIGURE 3.2 – Carte thématique des encadrements

© 2014 Tous droits réservés.

Deux cartes complémentaires sont celles des figures 3.2 et 3.3. La figure 3.2 positionne sur mes directions de recherche, les étudiants que j'ai encadrés. La liste de ceux-ci figure dans l'annexe B.3 (page 119). La figure 3.3 quant à elle donne la répartition sur mes directions de recherche des projets et des collaborations, également décrits dans l'annexe D (page 139).

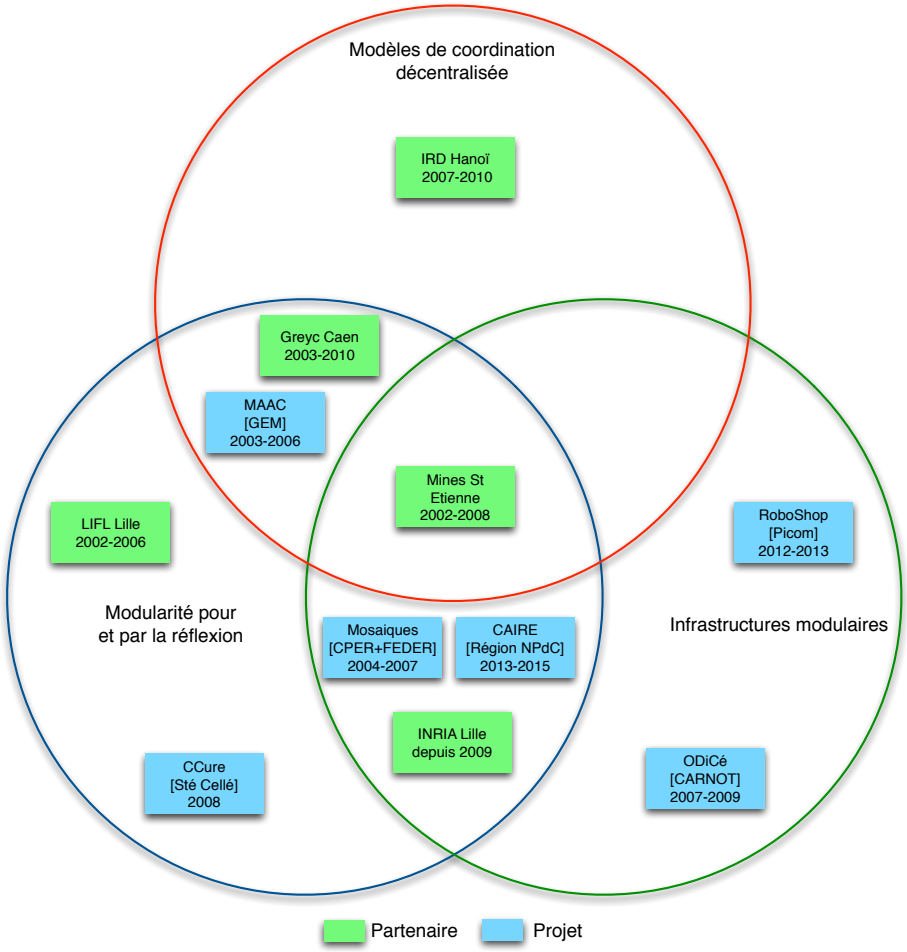

FIGURE 3.3 – Carte thématique des projets et partenariats

3.3.2 Modularité pour et par la réflexion

Cette direction de travaux est le prolongement naturel de mon doctorat [Bou99a]. En effet, ma thèse a porté sur la composition de métaclasses explicites [BLR98], ainsi que sur leur utilisation comme support à la programmation par aspects [Bou99b].

Après mon arrivée à l'ÉCOLE DES MINES DE DOUAI en 2001, j'ai poursuivi ce travail [BL05, Bou04b] tout en élargissant mon domaine d'étude pour inclure les composants logiciels [FBDH12, BF09]. Cette ouverture vers les composants s'est concrétisée notamment par les thèses de Houssam Fakih [Fak06] et de Guillaume Grondin [Gro08].

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

La thèse de Houssam Fakih [Fak06] que j'ai co-encadrée avec Laurence Duchien [11], a porté sur l'intégration du paradigme composant avec la programmation par aspects [FBD04, FB05]. Dans ce travail, nous avons proposé de représenter les aspects sous forme de composants du méta-niveau. Afin de permettre l'introspection et l'intercession des composants applicatifs, ces derniers sont dotés de ports *méta* qui permettent de tisser les aspects par assemblage. Ce travail a été réalisé avec le modèle de composants FRACTAL [BCLS04] du consortium ObjectWeb [12]. La validation expérimentale a reposé sur FRACTALK [13], notre projection en Smalltalk du modèle Fractal, initiée dans le stage master d'Ali Hamadi [Ham04].

L'utilisation de FRACTALK s'est étendue à d'autres travaux que nous avons menés et notamment la thèse de Guillaume Grondin [Gro08] que j'ai co-encadrée avec Laurent Vercouter [14] sous la direction d'Olivier Boissier [15]. Ce travail qui est à l'intersection de mes directions de recherche porte sur l'adaptation d'architectures à base de composants pour le contrôle d'agents robotiques. Il englobe le modèle MADCAR qui relève des modèles de programmation réflexifs. MADCAR introduit une représentation *abstraite* d'architectures où les composants sont spécifiés à l'aide de contraintes. Un moteur d'adaptation qui relève du méta-niveau de chaque agent, sélectionne dans une bibliothèque une architecture, puis des composants en utilisant un moteur satisfaction de contraintes [Col90]. Ainsi, en fonction des évolutions de l'environnement, MADCAR révise l'assemblage de composants tout en assurant la cohérence de l'état de l'application [GBV08, GBV06b].

3.3.3 Infrastructures modulaires

Le besoin de valider expérimentalement les différentes propositions m'a conduit à consacrer une part non-négligeable de mon activité aux infrastructures. Par exemple, le besoin de prendre en compte la répartition, typique des systèmes multi-robots, nous a conduit à développer l'intergiciel orienté-service UBIQUITALK [16].

Un autre exemple d'infrastructure a été AUTOFRACTAL [17] développé par Guillaume Grondin dans sa thèse autour du solveur de contraintes générique BACKTALK [RP96]. Cette infrastructure gère différentes sondes qui renseignent le moteur d'adaptation sur son contexte, c'est à dire son état interne et l'environnement dans lequel il est situé. L'ensemble des sondes est ouvert et peut être enrichi et adapté suivant les besoins de l'application.

Parallèlement à cet effort de développement, nous avons entrepris un travail de recherche sur les infrastructures logicielles. C'est l'objet de notre collaboration formalisée par une convention avec l'équipe RMOD de l'INRIA Lille. Actuellement, je co-encadre deux doctorants dans ce cadre. Le premier est Nick Papoulias [Pap13] qui travaille sur une infrastructure réflexive à

11. http://www.lifl.fr/~duchien/
12. http://fractal.ow2.org/
13. http://vst.mines-douai.fr/FracTalk
14. Laurent Vercouter était enseignant-chercheur à l'Ecole des Mines de St Etienne. Il est actuellement Professeur à l'INSA Rouen (https://sites.google.com/site/lvercouter/home)
15. http://www.emse.fr/~boissier/
16. http://vst.mines-douai.fr/UbiquiTalk
17. http://vst.mines-douai.fr/grondin/14

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

base de miroirs [BU04] pour le débogage à distance. La seconde thèse est celle de Guillermo Polito [Pol15] qui travaille sur une infrastructure supportant l'isolation d'applications réflexives.

La collaboration avec l'INRIA a débuté avec la thèse de Mariano Martinez Peck soutenue en 2012 [MP12]. Cette thèse vise à répondre au besoin de limiter la quantité de mémoire vive (RAM) utilisée pour réduire le coût de revient ou la consommation énergétique problématique dans les applications robotiques. Nos expérimentations ont montré que seuls 20% des objets d'un échantillon de logiciels grandeur nature sont utilisés à l'exécution. Par ailleurs, la mémoire virtuelle des systèmes d'exploitation s'avère inappropriée car elle travaille sur des pages mémoire, donc à gros grain.

En réponse, nous avons proposé MAREA, une solution de mémoire virtuelle au niveau applicatif qui descend au niveau de granularité de l'objet [MPBD+13]. Elle permet de découper le graphe d'objet en RAM en sous-graphes en gérant les intersections, c'est-à-dire les objets appartenant à plusieurs sous-graphes. Ainsi, chaque sous-graphe peut être transféré de la RAM vers le disque et vice-versa en fonction des besoins de l'application. Afin de prendre en compte l'activité de l'application, le découpage en sous-graphes est effectué dynamiquement de manière paresseuse et révisable.

3.3.4 Modèles de coordination décentralisée

Mon ouverture vers les systèmes multi-agents robotiques a été initiée au travers de ma collaboration avec l'équipe MAD du GREYC. Ainsi, j'ai co-encadré avec Serge Stinckwich [18] différents stages de master [Mou06, Gué05, Rob03]. Ces travaux se situent à la frontière entre mes travaux sur les modèles de coordination et ceux sur les langages et modèles de programmation réflexifs. Ainsi, Romain Robbes [Rob03] a intégré le modèle organisationnel AGR [FG97, BBSF07] avec la programmation par aspects [RBS04]. Ludovic Guégan [Gué05] a utilisé la réflexion pour développer une architecture d'agents hybrides adaptables. Enfin, Rémy Mouëza [Mou06] a utilisé le modèle de composants MALEVA [19] [MB01, BMP06] pour développer une architecture à subsomption [Bro85].

La collaboration avec le GREYC recouvre également la thèse de Van Tuan LE [Le10] co-encadrée avec l'équipe MSI de l'UMMISCO de Hanoï (Vietnam). Le point de départ de ces travaux est le besoin de flexibilité à l'échelle des systèmes multi-robots destinés à des applications de sauvetage. Dans un tel contexte, aucune hypothèse ne peut être faite sur l'environnement d'intervention des robots. Plus particulièrement, nous ne pouvons pas faire d'hypothèse sur l'existence d'une infrastructure de communication. Or, les robots doivent communiquer pour coordonner leur exploration du terrain.

Un des problèmes traité par Van Tuan LE est celui de l'exploration avec maintien de la connectivité du réseau *ad hoc* mobile formé par les robots [LBS+09]. Chaque robot planifie localement ses déplacements de manière à rester à proximité d'au moins un voisin qui lui permet de rester connecté au reste de la flotte. Afin d'assurer la cohérence globale des décisions prise

18. Actuellement à l'IRD (http://www.doesnotunderstand.org/).

19. En fait nous avons utilisé notre implémentation Smalltalk MALEVAST (http://vst.mines-douai.fr/MalevaST).

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

individuellement par les robots, nous avons modélisé la planification des déplacements comme un problème de satisfaction de contraintes distribué (*DisCSP*) [DBD+09]. La résolution du problème reste ainsi répartie sur les différents robots.

Au delà de l'exploration, nous nous sommes intéressés au problème plus général de coopération entre robots pour réaliser des tâches arbitrairement complexes nécessitant des compétences diverses. Nous avons proposé une solution d'organisation émergente adaptée aux contextes où le nombre et les capacités précises des robots sont initialement inconnues [LBSM09, LBSD12]. Les concepteurs de l'organisation spécifient les *coalitions*, ensembles de rôles complémentaires nécessaires pour traiter chaque type de tâche. Un protocole d'enchères permet aux robots d'endosser les rôles lors de la découverte des tâches.

La coordination des systèmes multi-robots a été également abordée dans la thèse de Guillaume Grondin en collaboration avec l'équipe SMA de l'Ecole des Mines de St Etienne. Le modèle MADCAR d'architectures adaptable (voir section 3.3.2) a servi de base pour la définition de l'architecture d'agents flexibles : MADCAR-AGENT [GBV09, GBV06a].

Dans MADCAR-AGENT, chaque robot dispose d'un jeu de composants qui ne permet de construire qu'un sous-ensemble des assemblages possibles. Cependant, des robots voisins peuvent échanger des composants. Cet échange permet aux robots de s'influencer mutuellement et constitue donc un moyen indirect de coordonner les adaptations.

3.4 Organisation du document

Ce document est organisé en 4 parties. A la présente introduction succède la partie II (page 45) où je développe mes travaux de recherche en mettant l'accent sur les encadrements de thèse. Ainsi, le chapitre 4 (page 45) présente mes travaux autour de la réflexion et de la méta-programmation. La cible est la flexibilité au niveau du cycle de développement des architectures de contrôle robotiques, en étendant les possibilités de modularisation et d'adaptation. Le chapitre 5 (page 63) traite de la flexibilité au niveau des infrastructures. Il résume nos principaux travaux sur les intergiciels [20] et autres outils de développement modulaires. Nos contributions en termes de modèles de coordination distribués sont l'objet du chapitre 6 (page 81). Elles viennent compléter nos autres travaux toujours avec l'objectif de la flexibilité des systèmes multi-robots. Enfin, le chapitre 7 (page 99) présente quelques unes des pistes que nous comptons poursuivre dans cette direction.

Ce mémoire est complété par l'annexe IV (page 113). Il s'agit de mon CV détaillé qui décrit les différentes facettes de mon activité, de la production scientifique, aux responsabilités pédagogiques, en passant par les encadrements, l'animation scientifique, les projets financés, ainsi que les collaborations.

20. *middlewares*

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

3.5 Références bibliographiques du chapitre

[ADB+99] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing (HUC'99)*, pages 304–307, 1999.

[BBSF07] José-Antonio Báez-Barranco, Tiberiu Stratulat, and Jacques Ferber. *Environments for Multi-Agent Systems III*, volume 4389 of *LNCS*, chapter A Unified Model for Physical and Social Environments, pages 41–50. Springer, 2007.

[BC89] J.-P. Briot and P. Cointe. Programming with explicit metaclasses in smalltalk-80. In *Conference proceedings on Object-oriented programming systems, languages and applications*, OOPSLA '89, pages 419–431, New York, NY, USA, 1989. ACM.

[BCLS04] Eric Bruneton, Thierry Coupaye, Matthieu Leclercq, and Jean-Bernard Stefani. An open component model and its support in java. In Ivica Crankovic, Judith A. Stafford, Heinz W. Schmidt, and Kurt C. Wallnau, editors, *Component-Based Software Engineering, 7th International Symposium, CBSE 2004*, number 3054 in LNCS, Edinburgh, UK, May 2004.

[BCS02] E. Bruneton, T. Coupaye, and J. Stefani. Recursive and dynamic software composition with sharing. In *WCOP'02–Proceedings of the 7th ECOOP International Workshop on Component-Oriented Programming*, Malaga, Spain, Jun 2002.

[BD01] Jean-Pierre Briot and Yves Demazeau, editors. *Systèmes Multi-Agents*. Hermes, 2001.

[BDNW08] Alexandre Bergel, Stéphane Ducasse, Oscar Nierstrasz, and Roel Wuyts. Stateful traits and their formalization. *Comput. Lang. Syst. Struct.*, 34(2-3) :83–108, 2008.

[Bec01] Kent Beck. *Extreme Programming Explained*. Addison-Wesley, 2001.

[BF09] Noury Bouraqadi and Luc Fabresse. Clic : A component model symbiotic with smalltalk. In *Proceedings of the International Workshop on Smalltalk Technologies*, Brest, France, August 2009. ACM.

[BL01] N. Bouraqadi and T. Ledoux. Le point sur la programmation par aspects. *Technique et Science Informatique*, 20(4) :505–528, 2001.

[BL05] Noury Bouraqadi and Thomas Ledoux. *Aspect-Oriented Software Development*, chapter 12 – Supporting AOP using Reflection, pages 261–282. Addison-Welsey, 2005.

[BLR98] N. Bouraqadi, T. Ledoux, and F. Rivard. Safe metaclass programming. In *Proceedings of OOPSLA'98*, Vancouver, British Columbia, Canada, October 1998. ACM.

[BMP06] J. P. Briot, T. Meurisse, and F. Peschanski. Une expérience de conception et de composition de comportements d'agents à l'aide de composants. *L'Objet*, 11 :1–30, 2006.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[Bou99a] N. Bouraqadi. *Un MOP Smalltalk pour l'étude de la composition et de la compatibilité des métaclasses. Application à la programmation par aspects (A Smalltalk MOP for the Study of Metaclass Composition and Compatibility. Application to Aspect-Oriented Programming - In French)*. Thèse de doctorat, Université de Nantes, Nantes, France, July 1999.

[Bou99b] Noury Bouraqadi. Un cadre rélexif pour la programmation par aspects (a reflective framework for aspect-oriented programming - in french). In *Langages et Modèles à Objets (LMO'99)*, Villefranche sur Mer - France, January 1999. Hermes.

[Bou04a] N. Bouraqadi. Safe metaclass composition using mixin-based inheritance. *Journal of Computer Languages and Structures*, 30(1-2) :49–61, April-July 2004. Special issue : Smalltalk Language.

[Bou04b] N. Bouraqadi. Safe metaclass composition using mixin-based inheritance. *Journal of Computer Languages and Structures*, 30(1-2) :49–61, April 2004. Special issue : Smalltalk Language.

[Bro85] Rodeney Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1) :14–23, March 1985.

[Bru07] Davide Brugali, editor. *Software Engineering for Experimental Robotics*, volume 30 of *STAR (Springer Tracts in Advanced Robotics)*. Springer, 2007.

[BU04] Gilad Bracha and David Ungar. Mirrors : design principles for meta-level facilities of object-oriented programming languages. In *Proceedings of the International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'04), ACM SIGPLAN Notices*, pages 331–344, New York, NY, USA, 2004. ACM Press.

[Coi87] Pierre Cointe. Metaclasses are first class : The objvlisp model. In *Conference proceedings on Object-oriented programming systems, languages and applications*, OOPSLA '87, pages 156–162, New York, NY, USA, 1987. ACM.

[Coi88] Pierre Cointe. A Tutorial Introduction to Metaclass Architecture as Provided by Class Oriented Languages. In *Proceedings of International Conference on Fifth Generation Computer Systems, ICOT*, Tokyo, Japan, November 1988.

[Col90] Alain Colmerauer. An introduction to prolog iii. *Communications of the ACM*, 33(7) :69–90, 1990.

[DBD+09] Arnaud Doniec, Noury Bouraqadi, Michael Defoort, Van Tuan Le, and Serge Stinckwich. Distributed constraint reasoning applied to multi-robot exploration. In *Proceedings of ICTAI 2009, 21st IEEE International Conference on Tools with Artificial Intelligence*, pages 159–166, 2009.

[DCL95] Alexis Drogoul, Bruno Corbara, and Steffen Lal. Manta : New experimental results on the emergence of (artificial) ant societies. In *Artificial Societies : The Computer Simulation of Social Life*, pages 190–211. UCL Press, 1995.

[DWBN07] Stéphane Ducasse, Roel Wuyts, Alexandre Bergel, and Oscar Nierstrasz. User-changeable visibility : Resolving unanticipated name clashes in traits. In *Proceedings of OOPSLA'07*, Montréal, Québec, Canada, October 2007.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[Erd08] Hakan Erdogmus. So many languages, so little time. *IEEE Software magazine*, pages 4–6, 2008.

[Fak06] Houssam Fakih. *L'integration des fonctionnalités transversales dans les composants logiciels en utilisant la programmation par aspects*. PhD thesis, Université de Lille 1, December 2006. Thèse co-encadrée et dirigée par Laurence Duchien (LIFL, Université de Lille 1).

[FB05] H. Fakih and N. Bouraqadi. Les aspects et les composants logiciels : Etude de cas avec le modèle de composant fractal. *L'Objet*, 2005.

[FBD04] Houssam Fakih, Noury Bouraqadi, and Laurence Duchien. Towards integrating aspects and components. In *Third AOSD Workshop on Aspects, Components, and Patterns for Infrastructure Software (ACP4IS)*, March 2004.

[FBDH12] Luc Fabresse, Noury Bouraqadi, Christophe Dony, and Marianne Huchard. A language to bridge the gap between component-based design and implementation. *Computer Languages, Systems and Structures*, 2012.

[FDH08] Luc Fabresse, Christophe Dony, and Marianne Huchard. Foundations of a Simple and Unified Component-Oriented Language. *Journal of Computer Languages, Systems & Structures*, 34/2-3(2-3) :130–149, 2008.

[FEAC05] R. Filman, T. Elrad, M. Akşit, and S. Clarke, editors. *Aspect-Oriented Software Development*. Addison-Wesley, 2005.

[Fer95] Jacques Ferber. *Les systèmes multi-agents. Vers une intelligence collective.* InterEditions, Paris, France, 1995.

[FG97] Jacques Ferber and Olivier Gutknecht. Aalaadin : a meta-model for analysis and design of organizations in multi-agent systems. Technical Report R.R.LIRMM97189, LIRMM, December 1997.

[GBV06a] Guillaume Grondin, Noury Bouraqadi, and Laurent Vercouter. Assemblage automatique de composants pour la construction d'agents avec madcar. In *Actes des journées Multi-Agent et Composant (JMAC)*, Nîmes, France, March 2006.

[GBV06b] Guillaume Grondin, Noury Bouraqadi, and Laurent Vercouter. Madcar : an abstract model for dynamic and automatic (re-)assembling of component-based applications. In *Proceedings of the 9th International Symposium on CBSE (Component-Based Software Engineering)*, LNCS, pages 360–367, Sweden, June 2006. Springer.

[GBV08] Guillaume Grondin, Noury Bouraqadi, and Laurent Vercouter. Component reassembling and state transfer in madcar-based self-adaptive software. In *Proceedings of the 46th International Conference TOOLS-EUROPE (Objects, Models, Components, Patterns)*, Switzerland, June 2008.

[GBV09] Guillaume Grondin, Noury Bouraqadi, and Laurent Vercouter. Un modèle pour le développement d'agents auto-adaptables. In *Actes des JFSMA (Journées Francophones des Systèmes Multi-Agents)*, Lyon, France, October 2009.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[Gro08] Guillaume Grondin. *Un modèle d'agents auto-adaptables à base de composants*. PhD thesis, Ecole des Mines de Saint Etienne, November 2008. Thèse co-encadrée avec Laurent Vercouter (Ecole des Mines de St Etienne) sous la direction d'Olivier Boissier (Ecole des Mines de St Etienne).

[Gué05] Ludovic Guégan. Hybridation paramètrable d'agents pour systèmes embarqués. Master's thesis, Université de Caen, 2005.

[Ham04] Ali Hamadi. Une implémentation du modèle de composants fractal en smalltalk. Master's thesis, Université de Nantes, September 2004.

[KLM+97] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-oriented programming. In M. Akşit and S. Matsuoka, editors, *Proceedings of ECOOP'97*, number 1241 in LNCS, pages 220–242. Springer-Verlag, June 1997.

[KS08] David Kortenkamp and Reid Simmons. *Handbook of Robotics*, chapter 8- Robotic Systems Architectures and Programming, pages 187–206. Springer, 2008.

[LBS+09] Van Tuan Le, Noury Bouraqadi, Serge Stinckwich, Victor Moraru, and Arnaud Doniec. Making networked robot connectivity-aware. In *Proceedings of ICRA (International Conference on Robotics and Automation)*, Kobe, Japan, May 2009.

[LBSD12] V. T. Le, N. Bouraqadi, S. Stinckwich, and A. Doniec. Role-based dynamic coalitions of multi-tasked rescue robots. In *Proceedings of the 9th International ISCRAM Conference*, Vancouver, Canada, apr 2012.

[LBSM09] Van Tuan Le, Noury Bouraqadi, Serge Stinckwich, and Victor Moraru. Connectivity awareness in networked robotic systems. In *Proceedings of IEEE-RIVF International Conference on Computing and Communication Technologies*, Da Nang City, Vietnam, July 2009.

[Le10] Van Tuan Le. *Coopération dans les systèmes multi-robots : Contribution au maintien de la connectivité et à l'allocation dynamique de rôles*. PhD thesis, Université de Caen, October 2010. Thèse co-encadrée avec Serge Stinckwich (GREYC Univ. de Caen/MSI Hanoï) et Victor Moraru (MSI Hanoï) sous la direction de François Bourdon ((GREYC Univ. de Caen).

[LW05] Kung-Kiu Lau and Zheng Wang. A taxonomy of software component models. In *EUROMICRO-SEAA*, pages 88–95. IEEE Computer Society, 2005.

[Mal97] Jacques Malenfant. *Abstraction, encapsulation et réfleion dans les langages à prototypes*. Thèse d'habilitation à diriger des recherches, Université de Nantes - Faculté des Sciences et des Techniques, April 1997.

[Mal11] J. Malenfant. De la robotique communicante aux « cyber-physical systems ». Exposé invité aux Journées Nationales de Recherche en Robotique (http://jnrr2011.irccyn.ec-nantes.fr/, 2011.

[MB01] Thomas Meurisse and Jean-Pierre Briot. Une approche à base de composants pour la conception d'agents. *Technique et science informatiques (TSI)*, 20(4) :583–602, 2001.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[MNC+89] G. Masini, A. Napoli, D. Colnet, D. Léonard, and K. Tombre. *Les langages à objets*. InterEditions, 1989.

[Mou06] Rémy Mouëza. Architecture réactive à subsomption à l'aide des composants maleva. Master's thesis, Université de Caen, June 2006.

[MP12] Mariano Martinez-Peck. *Application-Level Virtual Memory for Object-Oriented Systems*. PhD thesis, Université de Lille, October 2012. Thèse co-encadrée avec Luc Fabresse (Ecole des Mines de Douai) et Marcus Denker (INRIA Lille Nord Europe) sous la direction de Stéphane Ducasse (INRIA Lille Nord Europe).

[MPBD+13] Mariano Martinez-Peck, Noury Bouraqadi, Marcus Denker, Stéphane Ducasse, and Luc Fabresse. Object-based virtual memory brought to the application level. *Journal Of Object Technology*, 12(1), jan 2013.

[OPP96] Lars Overgaard, Henrik Petersen, and John Perram. Motion planning for an articulated robot : A multi-agent approach. In *Distributed Software Agents and Applications*, Lecture Notes in Computer Science. Springer, 1996.

[Pap13] Nick Papoulias. Refléxion et débogage à distance d'applications contraintes en ressources. Thèse co-encadrée avec Luc Fabresse (Ecole des Mines de Douai) et Marcus Denker (INRIA Lille Nord Europe) sous la direction de Stéphane Ducasse (INRIA Lille Nord Europe), décembre 2013.

[Par72] David L. Parnas. On the criteria to be used in decomposing systems int modules. *Communications of the ACM*, 15(12), 1972.

[Par08] Lynne E. Parker. *Handbook of Robotics*, chapter 40. Multiple Mobile Robot Systems, pages 921–941. Springer, 2008.

[Pau07] Linda Dailey Paulson. Developers shift to dynamic programming languages. *IEEE Computer*, pages 12–15, 2007.

[Pol15] Guillermo Polito. Isolation et modularisation de systèmes réflexifs. Thèse co-encadrée avec Luc Fabresse (Ecole des Mines de Douai) et Marcus Denker (INRIA Lille Nord Europe) sous la direction de Stéphane Ducasse (INRIA Lille Nord Europe), 2015.

[RBS04] R. Robbes, N. Bouraqadi, and S. Stinckwich. Un modèle multi-agent unifiant les notions de groupe et d'aspect. In *Systèmes multi-agents défis scientifiques et nouveaux usages - Actes des JFSMA 2004*, Paris, France, November 2004.

[Rob03] Romain Robbes. Mise en oeuvre de la programmation par aspects dans le cadre des systèmes multi-agents. Master's thesis, Université de Caen, 2003.

[RP96] Pierre Roy and François Pachet. Reifying constraint satisfaction in smalltalk. *Journal of Object-Oriented Programming,*, 1996.

[SDNB03] Nathanael Schärli, Stéphane Ducasse, Oscar Nierstrasz, and Andrew Black. Traits : Composable units of behaviour. In *Proceedings of European Conference on Object-Oriented Programming (ECOOP'03)*, pages 248—274, july 2003.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[SGM02] C. Szyperski, D. Gruntz, and S. Murer. *Component Software : Beyond Object-Oriented Programming*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, second edition, 2002.

[Smi84] B. C. Smith. Reflection and semantics in lisp. In *Proceedings of the 14th Annual ACM Symposium on Principles of Programming Languages, POPL'84*, pages 23–35, Salt Lake City, Utah, USA, January 1984.

[Wik12] Wikipedia. Dynamic programming language. http://en.wikipedia.org/wiki/Dynamic_programming_language, mar 2012.

[WJ95] Michael Wooldridge and Nicholas R. Jennings. Intelligent agents : Theory and practice. *The Knowledge Engineering Review*, 10(2) :115–152, 1995.

[Zim96] Chris Zimmermann, editor. *Advances in Object-Oriented Metalevel Architectures and Reflection*. CRC Press, 1996.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

Deuxième partie

Directions de recherche

© 2014 Tous droits réservés.

doc.univ-lille1.fr

4. Modularité pour et par la réflexion

Une facette de mes travaux cible la flexibilité au niveau du cycle de développement des architectures de contrôle robotiques. Cette flexibilité est introduite par l'intermédiaire de la réflexion. Je me suis reposé pour cela sur l'expertise développée durant mon doctorat sur la modularité et les langages réflexifs [Bou99a]. J'avais étudié entre autres l'utilisation de la réflexion comme support au paradigme de programmation par aspects [Bou99b].

Mon intérêt pour la modularité m'a conduit à explorer l'approche complémentaire que constituent les composants logiciels [Szy98]. Cette ouverture vers les composants s'est concrétisée notamment par des co-encadrements de thèses. Celle de Houssam Fakih [Fak06] s'est focalisée sur la flexibilité par la modularisation, puisqu'elle traite de l'intégration de la programmation par aspects et des composants logiciels. Financée dans le cadre du projet MAAC (voir la section D.8 page 147), cette thèse a débouché sur la représentation des aspects sous forme de composants logiciels. Le tissage (*weaving*) revient alors à assembler les composants-aspects avec les composants applicatifs.

La thèse Guillaume Grondin [Gro08] qui est à l'intersection de mes trois directions de recherche, a également traité des composants. Ce travail, financé dans le cadre du projet MOSAÏQUES (voir la section D.7, page 145), aborde la problématique d'adaptation dynamique d'architecture d'agents à base de composants logiciels dans des applications multi-robots. La solution proposée permet de représenter le logiciel de contrôle d'un robot sous la forme d'un jeu de composants qui peuvent être ré-assemblés suivant différentes architectures. Le choix de l'architecture et du sous-ensemble de composants à utiliser dépend du contexte et peut donc varier dans le temps.

Enfin, je co-encadre actuellement deux doctorants : Nick Papoulias [Pap13] et Guillermo Polito [Pol15] dans le cadre d'une collaboration avec l'équipe RMoD de l'INRIA Lille Nord Europe (voir section D.3 page 142). Ces deux thèses définissent des modèles réflexifs basés sur le concept de *miroirs* [BU04], qui sont des méta-objets qui encapsulent toutes les opérations réflexives d'un système. Ils permettent ainsi de découpler les niveaux de base et méta, ce qui permet de disposer de différentes implémentations du niveau méta ou de déporter celui-ci. Nous utilisons cette propriété pour définir des infrastructures destinées au développement et à l'exécution de logiciels de contrôle de robots. Ces travaux se situent donc à la frontière entre mes travaux sur la réflexion et ceux sur les infrastructures présentées en chapitre 5 (page 63).

Dans le cadre de mes travaux sur les langages et modèles réflexifs, j'ai également encadré un certain nombre d'étudiants de master [Cas00, Nas03, Rob03, Ham04, Leb04, Mou06], ainsi

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

qu'un ingénieur de recherche. Ce dernier encadrement était dans le cadre d'un contrat de transfert de technologie avec une *statup* (voir la section D.5 page 143). Le travail réalisé a nécessité la mise en œuvre de la réflexion combinée à une démarche de développement agile du type *Extreme Programming* (XP) [Bec01].

Dans le reste de ce chapitre, nous nous limitons à nos contributions qui se positionnent au cœur de l'utilisation de la réflexion *pour* la modularité. En section 4.1, nous présentons le travail de la thèse de Houssam Fakih [Fak06] sur l'intégration des deux paradigmes Aspect et Composant. Puis, nous décrivons en section 4.2 (page 51), la solution de ré-assemblage de composants issue de la thèse de Guillaume Grondin [Gro08]. Ces deux travaux partagent en commun une validation avec le modèle de composants FRACTAL [BCS02, BCLS04] défini conjointement par France Télécom R&D et l'INRIA Rhône-Alpes. En effet, les expérimentations ont exploité FRACTALK, la projection en SMALLTALK de ce modèle dont la première version est issue du master d'Ali Hamadi [Ham04].

4.1 Intégration des paradigmes Aspect et Composant

4.1.1 Motivation

La *programmation par aspects (Aspect Oriented Programming, AOP)* [KLM+97] est un paradigme de programmation qui propose de structurer les applications en modules appelés *aspects*, qui sont orthogonaux aux modules d'un langage hôte. Dans le cas d'un langage hôte basé sur les classes, un même aspect qui décrit une seule facette de l'application (distribution, mobilité, synchronisation, ...) peut être *tissé* avec plusieurs classes. La connexion que représente l'opération de *tissage* est effectuée en des points du code ou du flot d'exécution appelés *points de jonction*.

Les composants logiciels [Szy98] se situent dans le prolongement de la programmation par objets. Ce modèle de programmation rend obligatoires certaines bonnes pratiques de la programmation par objets. En effet, un *composant* est un fragment logiciel dont on a explicité à la fois les *ports* de connexion avec les autres composants et les dépendances vis à vis du contexte d'exécution (OS, bibliothèques, ...). Ainsi, le paradigme composant promeut l'inversion de contrôle. Les connexions entre composants sont réalisées par une entité tierce, sur la base d'une *architecture*. L'architecture définie lors de la conception est ainsi directement matérialisée lors du développement sous forme d'un assemblage. Par ailleurs, les composants renforcent l'encapsulation via les *contrats* [BJPW99a]. Ceux-ci constituent une documentation censée décrire le fonctionnement des composants. Le but est d'éviter aux utilisateurs l'analyse de l'implémentation. Ainsi, il y a moins de risque de couplage entre composants, ce qui favorise la flexibilité en rendant les composants interchangeables.

Les aspects et les composants sont deux paradigmes *post-objet* complémentaires. La programmation par aspects requière un autre paradigme pour définir la "base" de l'application. Par ailleurs, les composants ne permettent pas de gérer de manière modulaire le code correspondant à des fonctionnalités transversales. Il est donc légitime de vouloir combiner ces deux paradigmes afin de bénéficier de leurs avantages respectifs.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

4.1.2 L'existant

Cette section évalue l'état de l'art au moment de la thèse de Houssam Fakih soutenue en 2006. Le modèle de composants industriel dominant de l'époque était EJB [MH99]. Une des forces de ce modèle est la notion de conteneur qui définit les aspects : persistance, transaction, distribution et sécurité. Il présente l'intérêt d'éviter au développeur de se préoccuper des conflits entre aspects tissés aux mêmes points de jonction. Ceux-ci sont résolus une fois pour toutes dans le conteneur. Mais, cette approche offre un petit ensemble d'aspects fermé, avec une implémentation certes paramétrables mais figée. Le conteneur est monolithique. On ne peut pas ajouter de nouveaux aspects, ni supprimer ceux dont on n'a pas besoin. De même, il n'est pas possible de remplacer l'implantation d'un aspect. Il faut remplacer le conteneur dans sa globalité.

Dans ce qui suit nous comparons les travaux les plus représentatifs parmi ceux qui offrent un ensemble d'aspects ouvert. Pour cela, nous utilisons les critères suivants :

- *Composite*. Le choix du modèle de composant est important car il conditionne les possibilités de réutilisation des composants, mais également les points de tissage avec les aspects. Les modèles hiérarchiques sont les plus intéressants car ils permettent de construire des composants *composites* à partir de composants pré-existants.
- *Encapsulation*. Il s'agit d'évaluer la préservation de l'encapsulation des composants. La violation de l'encapsulation conduit à l'introduction d'un couplage entre aspects et composants, ce qui est défavorable à l'évolution de l'application par remplacement des composants.
- *Couverture*. Ce critère correspond à la latitude offerte aux développeurs d'aspects. Idéalement, les développeurs doivent avoir accès à n'importe quel point du flot de l'exécution des composants, comme par exemple tous les messages, qu'ils proviennent du composant lui même ou d'autres composants.
- *Réutilisabilité*. L'objet de ce critère est d'évaluer le potentiel de réutilisation des aspects. Ce potentiel dépend des constructions utilisées pour représenter les aspects.
- *Dynamicité*. Il est question ici de la possibilité d'adapter les applications à chaud par la modification de leurs aspects et des composants auxquels ils sont tissés. Cela requiert un support pour le tissage/*dé-tissage* dynamique d'aspects.

	Composants	**Composite**	**Encapsulation**	**Couverture**	**Réutilisabilité**	**Dynamicité**
AspectJ2EE	EJB	—	+++	—	++	—
JBoss-AOP	EJB	—	—	+++	++	++
JAsCo	Java Beans	—	+++	—	++	+++
FuseJ	Java Beans	—	+++	—	+++	+++
CAM-DAOP	CAM	—	+++	—	+++	+++

TABLE 4.1 – Evaluation des travaux qui intègrent les aspects et les composants

Le tableau 4.1 résume notre évaluation de l'état de l'art. Notre sélection de travaux regroupe : JBoss-AOP [BCF+04] et AspectJ2EE [CG04] qui étendent le modèle de composants EJB, ainsi que JAsCo [SVJ03] et FuseJ [SFV05] qui reposent tous les deux sur le modèle des Java Beans [Ham97]. Nous évaluons également CAM-DAOP [MPT05] qui introduit son propre modèle de composants. Ces différents travaux se caractérisent par des modèles de composants

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

plats. Ils n'offrent donc pas la possibilité de construire des composites et n'abordent donc pas le problème de tissage d'aspects avec les *sous-composants*.

A l'exception de JBoss-AOP, toutes les propositions préservent l'encapsulation en introduisant des points de jonctions indépendants de l'implémentation des composants. JBoss-AOP quant à lui donne accès à des données privées des composants EJB comme l'accès aux champs déclarés privés. Il ne permet donc pas de considérer les composants comme des boîtes noires, mais présente le bénéfice d'être plus couvrant que les autres propositions. AspectJ2EE, par contre est moins couvrant puisqu'il ne donne accès qu'aux appels à distance de méthodes et aux attributs des *beans* accessibles via des accesseurs en lecture et/ou en écriture. L'activité interne d'un bean, comme par exemple l'envoi de message à this n'est pas capturée. Dans JAsCO et FuseJ, aucune information reliée à l'état interne des composants n'est exposée, puisque les points de jonctions ne concernent que les envois de messages et les émissions d'événements qui passent par les ports des composants. De même CAM-DAOP, préserve l'aspect boîte noire des composants, puisque ses points de jonction ne concernent que l'activité "externe" des composants. En effet, il couvre la réception et l'émission d'événements, l'échange de messages entre composants, ainsi que la création et la destruction des composants.

Afin de discuter de la réutilisabilité des aspects, nous devons considérer différents niveaux introduits par Suvée et al. [SVJ03].

- *Où*. Désigne les points de jonction dans le code de base ou son flot d'exécution où a lieu l'intégration avec les aspects. Cette partie est totalement dépendante du niveau de base cible.
- *Quand*. Correspond au moment d'intervention de l'aspect par rapport aux points de jonction, définis par des constructions telles que *before* ou *around* dans AspectJ [Lad03]. Ce niveau peut varier d'une application à une autre.
- *Quoi*. Représente les traitements réalisés par l'aspect. Elle est indépendante de tout domaine applicatif.

Le *Quoi* est la partie la plus générique de l'aspect. Elle doit donc être découplée du *Quand*, mais surtout du *Où*, afin de maximiser les possibilités de réutilisation. Ce découplage est réalisé dans CAM-DAOP et dans FuseJ. AspectJ2EE, JBoss-AOP et JAsCO se limitent quant à eux à isoler la spécification des points de jonction (le *Où*).

Le dernier critère de notre évaluation est la dynamicité du tissage. L'approche de AspectJ2EE est la moins intéressante. Elle repose sur la génération statique de sous-classes du code des *beans* qui intégrent directement le code des aspects. Ainsi, une fois l'application chargée, il n'est plus possible de dissocier les aspects des composants ou de tisser de nouveaux aspects.

JBoss-AOP propose une approche plus intéressante. Les classes des composants sont enrichies avec une interface qui expose les points de jonction et permet donc le tissage à l'exécution. Cependant, JBoss-AOP est partiellement fermé et ne permet pas de redéfinir certains aspects fournis par défaut comme la sécurité par exemple. Il impose que la définition de la sécurité fournie par le serveur d'applications JBoss sous-jacent à JBoss-AOP soit utilisée.

CAM-DAOP, JAsCo et FuseJ permettent tous les trois un support du tissage à l'exécution pour tous les aspects. En effet, CAM-DAOP repose sur un *middleware* qui redirige l'exécution des points de jonction vers les aspects appropriés. JAsCo et FuseJ reposent quant à eux sur des connecteurs qui interceptent les communications. Ils requièrent cependant que les composants

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

soient préalablement instrumentés pour supporter le tissage.

4.1.3 Nos travaux

Ce travail a été effectué dans cadre de la thèse de Houssam Fakih [Fak06] que j'ai co-encadrée avec Laurence Duchien du LIFL. Notre contribution a été de proposer le modèle FRACTAL-AOP [FBD04, FB05] qui introduit les aspects dans le modèle de composants FRACTAL de l'INRIA-France Télécom R&D [BCS02]. Ceci constitue une première originalité de ce travail car il s'attaque à modèle de composants hiérarchique.

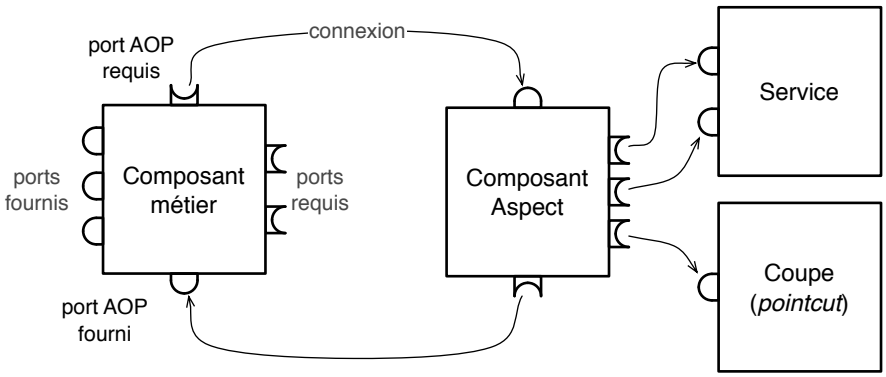

FIGURE 4.1 – Modèle de composants et d'aspects de FRACTAL-AOP

Afin d'assurer la couverture de tous les points du flot d'exécution des composants, tout en préservant l'encapsulation, FRACTAL-AOP étend les composants FRACTAL avec deux ports dits *AOP*. Ils permettent la connexion avec un éventuel aspect également représenté par un composant (voir Figure figure 4.1). L'un de ces ports est requis, tandis que l'autre est fourni. Quand le composant doit effectuer un traitement (par exemple exécuter une opération), il active l'aspect via le port requis. Ce dernier peut réaliser les traitements transverses adéquats. L'aspect choisit le moment d'exécuter le traitement de base du composant, et l'active en retour via le port AOP fourni.

Les deux ports AOP permettent aux aspects de suivre l'activité des composants et éventuellement de l'altérer. Ainsi, les aspects sont analogues à des méta-objets et les composants métier correspondent à des objets de base. Cependant, l'encapsulation est préservée dans le sens où seules les opérations et les attributs publiques sont exposées par le composant.

Outre la possibilité de définir des aspects transverses aux composants sans violer l'encapsulation, notre modèle permet de définir les aspects eux-mêmes sous forme de composants. L'opération de *tissage* des aspects est matérialisée sous la forme d'une opération d'assemblage qui lie les *composants aspects* aux *composants métier* via des ports dédiés.

En termes de couverture, FRACTAL-AOP va au delà que la plupart des propositions de l'état de l'art. Le modèle capture tous les traitements publiques qu'ils soient appelés depuis l'extérieur du composant ou par des appels internes. Il capture également les appels à des opérations

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

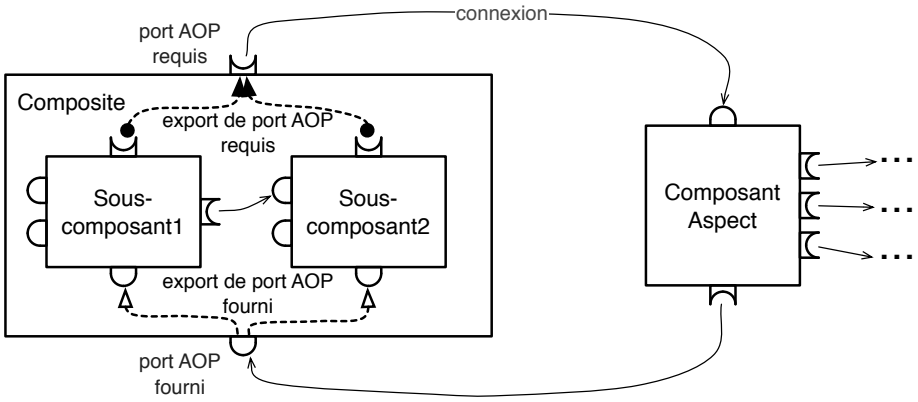

FIGURE 4.2 – Export des ports AOP des sous-composants

publiques ou des accès aux attributs publiques des sous-composants des composites. Comme le montre la figure 4.2, cette exposition de la partie publique des sous-composants se fait toujours par l'intermédiaire du composite de manière à préserver l'encapsulation. Les ports AOP des sous-composants sont exportés sur ceux du composite parent. A cet effet, nous avons introduit deux nouveaux types de connexions, l'un pour l'export de ports requis et l'autre pour l'export des ports fournis.

La réutilisation était également une de nos préoccupations lors de la définition de FRACTAL-AOP. Nous avons séparé les trois niveaux que sont le : *Quoi* (traitements réalisés par l'aspect), le *Quand* (moment d'intervention de l'aspect tel que *before* et *around*) et le *Où* (points de jonction). Ces trois niveaux ont été réifiés en trois composants comme nous pouvons le voir sur la figure 4.1 (page 49).

Le composant service représente le *Quoi*. Il est totalement découplé des composants métier et décrit exclusivement les traitements relatifs à un service (par exemple la persistance). Le *Quand* et le *Où* sont représentés respectivement par le composant Aspect et le composant Coupe. Le composant Aspect reçoit des composants métier auxquels il est connecté, les requêtes d'exécution des traitements de base. Il interroge la coupe pour évaluer si ces points de jonction *potentiels* doivent être capturés (le *Où*). Si c'est le cas, il décide du moment (le *Quand*) d'appeler les traitements du composant service.

Grace à la séparation des trois niveaux *Quoi*, *Quand*, *Où* nous favorisons non seulement la réutilisation, mais également l'adaptation dynamique. Ces trois niveaux sont représentés par trois composants et sont liés par des connexions, qui comme toutes les connexions du modèle FRACTAL peuvent être révisées à l'exécution. Il est donc possible de remplacer par exemple le composant Service pour introduire une nouvelle implémentation ou bien le composant Coupe pour modifier les points de jonction d'intervention de l'aspect. Par ailleurs, la connexion entre le composant Aspect et les composants métier est également révisable à l'exécution. Il est donc possible de tisser/*dé-tisser* les aspects suivant les besoins applicatifs.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

4.2 Adaptation d'assemblages de composants

4.2.1 Motivation

L'objet de cette section est la flexibilité d'assemblages de composants par l'adaptation. Il s'agit de réviser l'architecture et éventuellement replacer les composants qui constituent un logiciel pour le conformer à des conditions nouvelles.

Ce besoin d'adaptation se retrouve dans tous les logiciels sensibles au contexte (*context-aware*). C'est typiquement le cas des logiciels de contrôle de robots. Ils sont alimentés par des capteurs qui détectent les changements de l'état interne du robot (par exemple le niveau de charge de la batterie) ou de son environnement (par exemple la force ou l'orientation du vent pour un drone).

L'adaptation à de tels changements vise à rendre l'activité du robot plus pertinente et plus performante étant donné son nouveau contexte. Mais, l'adaptation peut avoir d'autres motivations comme indiqué par Ketfi et al. [KBY02] :

- *Correction des bogues :* Pour corriger les erreurs de fonctionnement d'une entité au sein de l'application, il est nécessaire de la modifier ou de la remplacer par une nouvelle version.
- *Évolution des fonctionnalités :* L'adaptation peut être requise pour ajuster les fonctionnalités de l'application. C'est nécessaire pour suivre l'évolution des besoins du client ou pour prendre en compte l'apparition ou la disparition de certains équipements comme par exemple un capteur GPS.
- *Amélioration des performances :* L'objectif est d'optimiser les performances de l'application. Pour cela, on peut modifier ou même remplacer certaines parties de l'application pour résoudre une tâche plus efficacement. C'est typiquement le cas des applications qui basculent de la 3G vers le Wifi.

4.2.2 L'existant

La problématique d'adaptation des logiciels n'est pas simple à étudier du fait des différentes dimensions de l'adaptation [LBB+01, Sen03]. Une adaptation peut survenir à différents moments, et pour plusieurs raisons. Elle peut être initiée de différentes façons et peut porter sur des éléments variés d'une application. Ainsi, les solutions d'adaptation logicielle peuvent être étudiées suivant quatre dimensions différentes :

- *La portée :* Sur quels éléments de l'application peut porter une adaptation ?
- *Le moment :* À quels moments peut-on adapter une application ?
- *L'acteur :* À qui incombe la décision d'adapter une application ?
- *La mise en œuvre :* Comment réalise-t-on l'adaptation d'une application ?

Nous nous intéressons exclusivement aux applications à base de composants. Ainsi, suivant la première dimension, la portée des adaptations s'étend aux composants, à leurs liaisons, ainsi qu'aux architectures logiques ou physiques des applications.

Le moment de l'adaptation peut varier suivant que les modifications ont lieu au développement, au déploiement ou à l'exécution. L'objet de notre étude étant des applications sensibles au

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

contexte, nous ne retenons dans cet état de l'art que les approches qui proposent des adaptations dynamiques, donc à l'exécution.

Avec des adaptations dynamiques, l'intervention humaine est difficile voire impossible. En effet, la machine peut être distante comme par exemple le cas d'un robot d'exploration spatiale ou tout simplement un robot qui est dans un environnement qui empêche les communications radio (cas des robots sous-marins).

Nous restreignons donc notre étude aux travaux sur l'adaptation dynamique, réalisée de manière autonome par des applications à base de composants. Nous comparons les travaux sur la base de la dernière dimension qui est la mise en œuvre de l'adaptation. Pour cela, nous nous appuyons sur les critères suivants :

Cohérence : C'est la capacité d'éviter les erreurs de fonctionnement à cause du processus d'adaptation et des problèmes qu'il peut engendrer (instabilité, transfert d'état, dépendances, versions, nommage).

Performance : Ce critère fait référence à la quantité de ressources matérielles utilisées par le processus d'adaptation.

Disponibilité : Il s'agit de la capacité de garantir que certaines fonctionnalités prioritaires pourront être utilisées durant le processus d'adaptation.

Simultanéité : C'est la capacité de coordonner des adaptations qui sont déclenchées ou réalisées "simultanément" dans différentes parties de l'application.

Ouverture : Représente la possibilité d'adapter le processus d'adaptation lui même. Cette ouverture se fait idéalement en donnant accès aux mécanismes sous-jacents à l'adaptation suivant le principe de l'« *Open Implementation* » [Kic96].

Généricité : Il s'agit de l'indépendance entre les descriptions des adaptations et l'application adaptée. Idéalement, cette indépendance doit être la plus grande pour favoriser la réutilisation de stratégies récurrentes d'adaptations dans des applications différentes.

	Cohérence	Performance	Disponibilité	Simultanéité	Ouverture	Généricité
C2	++	+	+	+	+	+
Darwin	++	+	+	++	+	+
SAFRAN	+++	+	+	++	+++	++
Think	++	++	+	++	++	+
SOFA	++	+	+	+	++	+
CASA	+++	++	+++	+++	+++	+

TABLE 4.2 – Comparatif des travaux sur l'adaptation d'applications à base de composants

Le tableau 4.2 récapitule les résultats de notre étude sur l'adaptation dynamique des applications à base de composants. Nous décrivons ci-dessous les caractéristiques des solutions les plus significatives pour chaque critère.

Cohérence

La cohérence des adaptations a été abordée par les différents travaux. Cependant, seuls CASA [MG05, MG03] et SAFRAN [Dav05, DL05, DL03] fournissent des mécanismes qui

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

aident à garantir la cohérence de l'état des composants et des assemblages produits par les adaptations.

CASA qui est basé sur un modèle de composants *ad hoc*, repose sur une description de l'ensemble des architectures valides. Les adaptations consistent à passer d'une architecture à une autre en supprimant, ou en ajoutant ou encore en remplaçant des connexions et des composants. Ainsi, les architectures produites par les adaptations sont cohérentes par construction. De plus, les composants interchangeables sont identifiés par les développeurs qui fournissent les fonctions de transfère de leurs états. Ainsi, CASA assure la cohérence de l'état des composants lors du remplacement d'un composant par un autre.

SAFRAN étend le modèle de composants FRACTAL avec un support de l'adaptation. Chaque composant est piloté à l'aide d'une politique d'adaptation réactive qui lui est propre. Cette approche peut conduire à des incohérences puisque chaque composant possède son propre ensemble de règles et s'adapte indépendamment des autres composants. Cependant, SAFRAN intègre un mécanisme pour détecter de telles incohérences *a posteriori*, que ce soit au niveau architectural ou au niveau état des composants. Dans un tel cas, les modifications sont annulées et les composants en cause sont ramenés à leur états antérieurs à l'adaptation.

Performance et Disponibilité

Le critère de performance est en général un point faible des travaux. Il n'est pas pris en compte dans C2 [MORT96, Med96, OT98], Darwin [MDK94], SOFA [PBJ98, BHP06] et SAFRAN. Seuls Think [Fas01, Sen03, SCS02] et CASA l'abordent partiellement. Les auteurs de Think ont porté leur attention au surcoût en termes de ressources mémoire et CPU, introduit par l'infrastructure d'adaptation dynamique. Ainsi, Think, qui est basé sur le modèle de composants FRACTAL, a été implémenté en langage C. Les mesures effectués ont montré que les indirections introduites pour permettre les reconfigurations des assemblages en Think introduisent un surcoût limité sur le fonctionnement normal des applications (hors adaptation).

CASA permet aux développeurs d'influer indirectement sur les ressources utilisées pour effectuer les adaptations. En effet, il offre deux stratégies de remplacement de composants. La première, dite gloutonne, privilégie la vitesse de l'adaptation en arrêtant l'application même si une opération est cours. L'autre, dite paresseuse, défère l'adaptation jusqu'à la fin des traitements en cours, de manière perturber le moins possible l'application. Il s'agit donc plus d'une manière de contrôler la disponibilité des services de l'application que d'une gestion des performances des adaptations à proprement parlé.

Simultanéité

CASA se distingue également sur le plan de la simultanéité des adaptations. En effet, une application dans CASA passe d'une architecture à une autre choisie parmi l'ensemble des architectures valides spécifiées par les développeurs. Ainsi, chaque adaptation peut toucher potentiellement plusieurs composants de l'application.

SAFRAN et Think permettent une simultanéité partielle des adaptations car ils reposent sur le modèle FRACTAL qui est hiérarchique. En effet, l'adaptation d'un composite peut se traduire

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

par la modification de ses sous-composants, de leurs attributs et éventuellement de leurs sous-composants respectifs.

Le langage de description d'architectures (*ADL*) DARWIN offre également un certain niveau de simultanéité des adaptations. En effet, il dispose d'un opérateur d'itération qui permet de créer ou supprimer un ensemble de composants et de connexions.

Ouverture

L'ouverture de SAFRAN se décompose en deux facettes. La première est la possibilité de définir des *politiques d'adaptation*. Une telle politique est un ensemble de règles du type ECA : Evénement, Condition, Action. L'événement est typiquement un message émis par une sonde. C'est lui qui déclenche l'évaluation de la règle et plus précisément le test de la condition. Quand la condition est vérifiée, l'action de la règle est exécutée. Cette action est une opération d'adaptation qui peut être arbitrairement complexe.

SAFRAN offre également la possibilité de *méta-adaptations* de modifier les politiques dynamiquement. Ainsi, il est possible d'ajouter ou de supprimer des règles ECA attachées à un composant.

CASA adopte une approche plus déclarative. Les architectures valides sont référencées dans le *contrat de l'application*. Ce contrat est découpé en *contextes*. Chaque contexte correspond à un état du monde tel qu'il est perçu au travers des sondes de l'application. A chaque contexte correspond un ensemble architectures alternatives. Ces architectures sont listées suivant l'ordre de préférence des concepteurs.

Le choix d'une architecture dans CASA se fait en deux temps. D'abord le contexte qui correspond aux informations collectées par les sondes est identifié. Ensuite, la liste d'architectures valides est parcourue de haut en bas. L'architecture retenue est la première parmi celles dont les besoins en ressources peuvent être satisfaits.

Enfin, rappelons qu'avec CASA le développeur dispose de deux stratégies pour réaliser les adaptations. Comme nous l'avons indiqué plus haut, il peut choisir de remplacer les composants d'une manière paresseuse ou gloutonne.

Notons que Think et SOFA répondent partiellement au besoin d'ouverture. Ils disposent d'entités chargées des adaptations modifiables par les développeurs. Cependant, aucun support n'est fourni pour aider à la réalisation ces changements qui peuvent être délicats.

Généricité

Le manque de généricité est un défaut général aux approches étudiées. Par exemple CASA qui est intéressant pour les autres critères souffre à deux niveaux du problème de couplage fort. Les composants interchangeables doivent être désignés comme tels explicitement lors du développement. Par ailleurs, les architectures alternatives référencent directement les composants.

Néanmoins, SAFRAN se distingue par la possibilité de définir des politiques d'adaptations génériques. Le langage de définition de ces politiques permet de définir des règles ECA génériques et réutilisables pour différents composants d'une même application ou dans différentes applications. Le développeur disposent à cet effet de *wildcards* et de liens relatifs pour par exemple

© 2014 Tous droits réservés.

doc.univ-lille1.fr

accéder à tous les sous-composants d'un composite donné. Ils peuvent également paramétrer une règle avec le composite qui l'exécute, la rendant ainsi réutilisable pour différents composants. Cependant, l'utilisation de ces élément reste à la charge du développeur qui peut écrire des politiques fortement couplées aux composants de l'application.

4.2.3 Nos travaux

Le modèle MaDcAr

Nos travaux sur les adaptations dynamiques d'applications à base de composants ont débouché sur le modèle MADCAR [1] [GBV06, GBV08a, GBV08b] représenté sur la figure 4.3. Selon ce modèle, une application peut être décomposée en deux niveaux :

- Un *niveau de base* (côté droit de la figure) constitué d'un ensemble de composants.
- Un *niveau méta* (côté gauche de la figure) qui raisonne et modifie le niveau de base en ré-assemblant les composants. Il réalise donc l'adaptation dynamique.

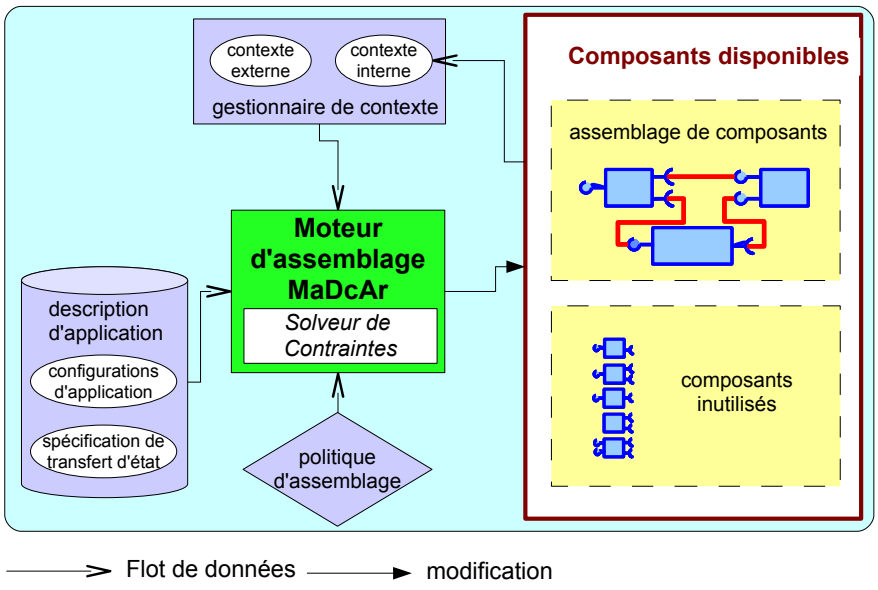

FIGURE 4.3 – Le modèle MADCAR

Au cœur de notre modèle MADCAR se trouve une entité que nous appelons *moteur d'adaptation*. Ce moteur assemble les *composants disponibles* suivant l'une des architectures spécifiées par les développeurs dans la *description de l'application*. Le choix des composants à assembler et de l'architecture est guidé par une *politique d'assemblage* et par le *contexte* courant de l'application.

Un gestionnaire de contexte fournit le contexte interne (état de l'application) et externe (état de l'environnement) de l'application [CCDG05]. La politique d'assemblage spécifie le moment

1. Model for Automatic and Dynamic Component Assembly Reconfiguration

© 2014 Tous droits réservés.

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

de déclencher les adaptations et la règle pour décider de l'assemblage à réaliser. L'ensemble des assemblages valides est spécifié dans des *configurations* qui font partie de la description de l'application. Chaque configuration comporte une architecture abstraite qui spécifie les composants attendus à l'aide de contrats (voir la section 4.2.3, page 56). Une configuration comporte également des fonctions dites de *caractérisation* qui permettent de calculer les coûts et la pertinence des assemblages. La moteur d'adaptation utilise ces fonctions pour comparer les différents assemblages et guider ainsi la décision. Cette décision se traduit par le choix d'une configuration et d'un sous-ensemble parmi le jeu de composants disponibles. Ce processus repose sur un solveur de contraintes [2]. Le moteur construit alors l'assemblage et transfère l'état de l'application depuis les composants qui formaient l'ancien assemblage.

Configurations

L'adaptation d'un assemblage peut donc transformer l'architecture même du logiciel. L'explicitation des différentes architectures valides pour une application indépendamment des composants assemblés constitue une des originalités de notre travail comparé à l'état de l'art. Selon le modèle MADCAR chaque application est décrite à l'aide d'un ensemble de *configurations*. Chaque configuration décrit un ensemble d'assemblages valides.

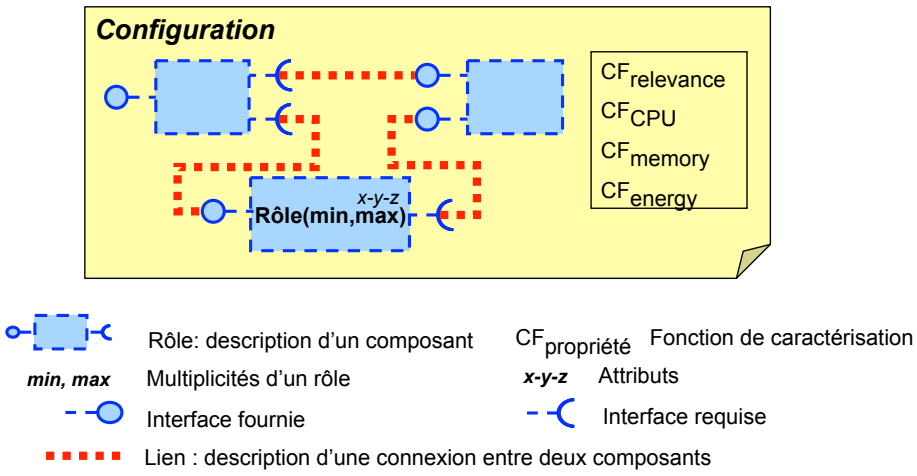

FIGURE 4.4 – Une configuration d'une application selon le modèle MADCAR

Comme le montre la figure 4.4, une configuration englobe une architecture *abstraite*. Il s'agit des spécifications de composants appelées *rôles* et des descriptions des connexions entre les composants. Un rôle est un ensemble de contrats [Mey92, BJPW99b] qui spécifient notamment les ports fournis et requis par les composants compatibles, leurs attributs et les valeurs initiales de ces derniers. Ce sont ces contrats qui sont utilisés par le moteur d'adaptation pour définir les contraintes qui permettent de retrouver les composants adéquats. Pour chaque rôle, le solveur

2. Notre implémentation repose sur le framework de solveurs BackTalk [RP96] que nous avons porté sur le Smalltalk libre Squeak. Le code sous licence MIT est disponible sur http://vst.mines-douai.fr/BackTalk.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

doit trouver un nombre minimum de composants. En effet, un rôle définit également deux *multiplicités* qui définissent les nombres minimum et maximum de composants qui peuvent remplir simultanément ce rôle dans un assemblage.

En plus de définir l'architecture de l'application, les développeurs doivent compléter chaque configuration avec des *fonctions de caractérisation*. Il s'agit de fonctions qui retournent, chacune un entier dans l'intervalle [0 100] utilisé pour le choix de la configuration.

Nous avons identifié 4 fonctions de caractérisation différentes. Il y a d'une part la fonction $CF_{relevance}$ qui donne la pertinence de la configuration étant donné le contexte. Par exemple, une configuration peut être plus ou moins pertinente suivant le débit réseau disponible à un instant donné. Les trois autres fonctions (CF_{energy}, CF_{memory} et CF_{CPU}) permettent d'estimer le coût en termes de consommation des ressources énergie, mémoire et CPU de l'assemblage construit sur la base de la configuration. Elles prennent deux paramètres : un ensemble de composants compatibles avec la configuration et le contexte courant.

Transfert d'état

Les configurations d'une même application peuvent spécifier des architectures très différentes. Néanmoins, il faut s'assurer que l'état de l'application reste cohérent [SF93]. Les informations commune à deux assemblages doivent être transférées, même si elles sont représentées par des structures de données différentes. Par exemple, une machine distante peut être désignée à l'aide d'un seul attribut uri dans une configuration et à l'aide de trois attributs : protocole, addresseIP et port dans une autre configuration.

Afin de conserver l'information lors du passage d'une configuration à une autre, les développeurs fournissent *un réseau de fonctions de transfert* qui permettent de passer d'une configuration à une autre. Après chaque révision de l'assemblage, le moteur d'adaptation utilise le réseau pour calculer les valeurs à utiliser pour initialiser les attributs des composants formant le nouvel assemblage. Les nœuds du réseau sont les attributs des configurations. Les arcs sont des *fonctions de transfert* qui permettent d'initialiser les attributs d'une configuration avec les valeurs obtenues des attributs d'une autre configuration.

Une fonction de transfert permet de calculer un sous-ensemble des attributs d'une configuration à partir d'un sous-ensemble des attributs d'une autre configuration. Un arc entre les attributs de deux configurations A et B correspond à deux *fonctions de transfert* $f_{A\to B}$ et $f_{B\to A}$. La fonction $f_{A\to B}$ permet de calculer les valeurs de certains attributs $b_{1..m}$ de la configuration B à partir des valeurs de certains attributs $a_{1..n}$ de la configuration A. Symétriquement, la fonction $f_{B\to A}$ permet de calculer les valeurs de certains attributs $a_{1..n}$ de la configuration A à partir des valeurs de certains attributs $b_{1..m}$ de la configuration B.

Il est important de noter que pour assurer la cohérence de l'état de l'application, il est nécessaire qu'on puisse transférer l'état entre deux configurations choisies arbitrairement. Pour cela, il est nécessaire que les attributs définis dans chaque configuration soient connectés aux attributs d'au moins une autre configuration à l'aide du réseau de transfert. Ainsi, entre deux configurations prises arbitrairement, il y a toujours un chemin avec une succession plus ou moins longue de fonctions de transfert.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

4.3 Bilan du chapitre

Dans ce chapitre j'ai présenté nos travaux sur la modularisation des logiciels de contrôle de robots. Il s'agit d'introduire de la flexibilité aussi bien durant le développement qu'à l'exécution en ayant une vision centrée sur les robots pris individuellement.

Le cœur du chapitre a porté sur les travaux des doctorants que j'ai co-encadrés sur le sujet. Ces contributions ont en commun d'avoir recours au concept de composant logiciel et de supporter la révision des assemblages à l'exécution. Les composants sont utilisés comme vecteur de la modularité ce qui a pour conséquence de rendre les logiciels plus flexibles. Le recours aux composants facilite la répartition sur une équipe de taille variable la charge de développement. De plus, il réduit l'effort pour maintenir et faire évoluer les logiciels d'autant plus que nous représentons les propriétés transversales (les aspects) sous forme de composants. Nous avons également montré qu'il est possible de réaliser des assemblages flexibles, qui sont révisés à l'exécution afin d'ajuster le comportement de chaque robot à son environnement et aux ressources disponibles. Cette capacité peut être étendue pour intégrer des stratégies de coordination de flottes robotiques comme je le décris dans le chapitre 6 (page 81).

Les travaux du présent chapitre reposent sur la réflexion pour introduire les concepts dans le langage sous forme d'entités de première classe et pour réviser les programmes à l'exécution. Le recours à la réflexion constitue un trait partagé avec les contributions du chapitre 5 (page 63) où j'aborde la facette complémentaire que sont les infrastructures pour les logiciels de contrôle de robots.

4.4 Références bibliographiques du chapitre

[BCF+04] Bill Burke, Austin Chau, Marc Fleury, Adrian Brock, Andy Godwin, and Harald Gliebe. Jboss aspect oriented programming. http://www.jboss.org/, 2004.

[BCLS04] Eric Bruneton, Thierry Coupaye, Matthieu Leclercq, and Jean-Bernard Stefani. An open component model and its support in java. In Ivica Crankovic, Judith A. Stafford, Heinz W. Schmidt, and Kurt C. Wallnau, editors, *Component-Based Software Engineering, 7th International Symposium, CBSE 2004*, number 3054 in LNCS, Edinburgh, UK, May 2004.

[BCS02] E. Bruneton, T. Coupaye, and J. Stefani. Recursive and dynamic software composition with sharing. In *WCOP'02–Proceedings of the 7th ECOOP International Workshop on Component-Oriented Programming*, Malaga, Spain, Jun 2002.

[Bec01] Kent Beck. *Extreme Programming Explained*. Addison-Wesley, 2001.

[BHP06] T. Bures, P. Hnetynka, and F. Plasil. Sofa 2.0 : Balancing advanced features in a hierarchical component model. In *SERA'06 : Proceedings of the 4th International Conference on Software Engineering Research, Management and Applications*, pages 40–48, Washington, DC, USA, 2006. IEEE Computer Society.

[BJPW99a] Antoine Beugnard, Jean-Marc Jézéquel, Noël Plouzeau, and Damien Watkins. Making components contract aware. *IEEE Computer*, 32(7) :38–45, 1999.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[BJPW99b] Antoine Beugnard, Jean-Marc Jezequel, Noel Plouzeau, and Damien Watkins. Making components contract aware. *Computer*, 32(7) :38–45, 1999.

[Bou99a] N. Bouraqadi. *Un MOP Smalltalk pour l'étude de la composition et de la compatibilité des métaclasses. Application à la programmation par aspects (A Smalltalk MOP for the Study of Metaclass Composition and Compatibility. Application to Aspect-Oriented Programming - In French)*. Thèse de doctorat, Université de Nantes, Nantes, France, July 1999.

[Bou99b] Noury Bouraqadi. Un cadre rélexif pour la programmation par aspects (a reflective framework for aspect-oriented programming - in french). In *Langages et Modèles à Objets (LMO'99)*, Villefranche sur Mer - France, January 1999. Hermes.

[BU04] Gilad Bracha and David Ungar. Mirrors : design principles for meta-level facilities of object-oriented programming languages. In *Proceedings of the International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'04), ACM SIGPLAN Notices*, pages 331–344, New York, NY, USA, 2004. ACM Press.

[Cas00] Gabriel Casarini. Towards transparent strong mobility using a reflective smalltalk. Master's thesis, Vrije Universiteit Brussel (Belgium) In Collaboration with Ecole des Mines de Nantes (France)., 2000.

[CCDG05] J. Coutaz, J. L. Crowley, S. Dobson, and D. Garlan. Context is key. *Commununication of the ACM*, 48(3) :49–53, 2005.

[CG04] Tal Cohen and Joseph Gil. Aspectj2ee = aop + j2ee : Towards an aspect-based, programmable and extensible middleware framework. In *Proceedings of ECOOP*, 2004.

[Dav05] Pierre-Charles David. *DÈveloppement de composants Fractal adaptatifs : un langage dÈdiÈ ‡ l'aspect d'adaptation*. Phd thesis, UniversitÈ de Nantes / . . .cole des Mines de Nantes, 2005.

[DL03] Pierre-Charles David and Thomas Ledoux. Towards a framework for self-adaptive component-based applications. In *DAIS'03 : Proceedings of the 4th IFIP International Conference on Distributed Applications and Interoperable Systems*, volume 2893 of *LNCS*, pages 1–14. Springer-Verlag, 2003.

[DL05] Pierre-Charles David and Thomas Ledoux. WildCAT : a generic framework for context-aware applications. In *MPAC'05 : Proceeding of the 3rd International Workshop on Middleware for Pervasive and Ad-Hoc Computing*, Grenoble, France, 2005.

[Fak06] Houssam Fakih. *L'integration des fonctionnalités transversales dans les composants logiciels en utilisant la programmation par aspects*. PhD thesis, Université de Lille 1, December 2006. Thèse co-encadrée et dirigée par Laurence Duchien (LIFL, Université de Lille 1).

[Fas01] Jean-Philippe Fassino. *THINK : vers une architecture de systËmes flexibles*. PhD thesis, Ecole Nationale SupÈrieure des TÈlÈcommunications, 2001.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[FB05] H. Fakih and N. Bouraqadi. Les aspects et les composants logiciels : Etude de cas avec le modèle de composant fractal. *L'Objet*, 2005.

[FBD04] Houssam Fakih, Noury Bouraqadi, and Laurence Duchien. Towards integrating aspects and components. In *Third AOSD Workshop on Aspects, Components, and Patterns for Infrastructure Software (ACP4IS)*, March 2004.

[GBV06] Guillaume Grondin, Noury Bouraqadi, and Laurent Vercouter. Madcar : an abstract model for dynamic and automatic (re-)assembling of component-based applications. In *Proceedings of the 9th International Symposium on CBSE (Component-Based Software Engineering)*, LNCS, pages 360–367, Sweden, June 2006. Springer.

[GBV08a] Guillaume Grondin, Noury Bouraqadi, and Laurent Vercouter. Assemblage automatique et adaptation d'applications à base de composants. In *Actes de LMO (Langage et Modèles à Objets)*, Monréal, Canada, March 2008.

[GBV08b] Guillaume Grondin, Noury Bouraqadi, and Laurent Vercouter. Component reassembling and state transfer in madcar-based self-adaptive software. In *Proceedings of the 46th International Conference TOOLS-EUROPE (Objects, Models, Components, Patterns)*, Switzerland, June 2008.

[Gro08] Guillaume Grondin. *Un modèle d'agents auto-adaptables à base de composants*. PhD thesis, Ecole des Mines de Saint Etienne, November 2008. Thèse co-encadrée avec Laurent Vercouter (Ecole des Mines de St Etienne) sous la direction d'Olivier Boissier (Ecole des Mines de St Etienne).

[Ham97] Graham Hamilton. JavaBeans. API Specification, Sun Microsystems, July 1997. Version 1.01.

[Ham04] Ali Hamadi. Une implémentation du modèle de composants fractal en smalltalk. Master's thesis, Université de Nantes, September 2004.

[KBY02] A. Ketfi, N. Belkhatir, and P. Y.Cunin. Adaptation dynamique, concepts et experimentations. In *ICSSEA'02 : Proceedings of the International Conference "Software and Systems Engineering and their Applications"*, Paris, France, 2002.

[Kic96] Gregor Kiczales. Beyond the black box : Open implementation. *IEEE Software*, 13(1) :8–11, 1996.

[KLM+97] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-oriented programming. In M. Akşit and S. Matsuoka, editors, *Proceedings of ECOOP'97*, number 1241 in LNCS, pages 220–242. Springer-Verlag, June 1997.

[Lad03] Ramnivas Laddad. *AspectJ in Action*. Manning, 2003.

[LBB+01] T. Ledoux, M. Blay, E. Bruneton, D. Caromel, T. Coupaye, D. Hagimont, J-M. Menaud, J. Noyé, and M. Riveill. Etat de l'art sur l'adaptabilité. Livrable D1.1, RNTL ARCAD, Ecole des Mines de Nantes, December 2001.

[Leb04] Gabriel Leblanc. Vers une généralisation du concept d'aspect. Master's thesis, Université de Lille 1, July 2004.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[MDK94] Jeff Magee, Naranker Dulay, and Jeff Kramer. A Constructive Development Environment for Parallel and Distributed Programs. In *Proceedings 2nd IEEE International Workshop on Configurable Distributed Systems (IWCDS-2)*, 1994.

[Med96] Nenad Medvidovic. Adls and dynamic architecture changes. In *Joint proceedings of the second international software architecture workshop (ISAW-2) and international workshop on multiple perspectives in software development (Viewpoints'96) on SIGSOFT'96 workshops*, pages 24–27, 1996.

[Mey92] B. Meyer. Applying "design by contract". *Computer*, 25(10) :40–51, 1992.

[MG03] Arun Mukhija and Martin Glinz. CASA - a contract-based adaptive software architecture framework. In *ASWN'03 : Proceedings of the 3rd IEEE Workshop on Applications and Services in Wireless Networks*, pages 275–286, Berne, Switzerland,, July 2003. IEEE Computer Society.

[MG05] Arun Mukhija and Martin Glinz. Runtime adaptation of applications through dynamic recomposition of components. In *ARCS'05 : Proceedings of the 18th International Conference on Architecture of Computing Systems*, pages 124–138, Innsbruck, Austria, March 2005.

[MH99] Richard Monson-Haefel. *Enterprise JavaBeans*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1999.

[MORT96] Nenad Medvidovic, Peyman Oreizy, Jason E. Robbins, and Richard N. Taylor. Using object-oriented typing to support architectural design in the c2 style. In *SIGSOFT'96 : Proceedings of the 4th ACM SIGSOFT symposium on Foundations of software engineering*, pages 24–32, New York, NY, USA, 1996. ACM Press.

[Mou06] Rémy Mouëza. Architecture réactive à subsomption à l'aide des composants maleva. Master's thesis, Université de Caen, June 2006.

[MPT05] Lidia Fuentes Mónica Pinto and José Maria Troya. A dynamic component and aspect- oriented platform. *The Computer Journal*, 48(4) :401–420, 2005.

[Nas03] Rabi Nasrallah. Programmation par aspects et services web. Master's thesis, Université de Technologie de Troie, July 2003.

[OT98] P. Oreizy and R. Taylor. On the role of software architectures in runtime system reconfiguration. In *CDS'98 : Proceedings of the International Conference on Configurable Distributed Systems*, page 61, Washington, DC, USA, 1998. IEEE Computer Society.

[Pap13] Nick Papoulias. Refléxion et débogage à distance d'applications contraintes en ressources. Thèse co-encadrée avec Luc Fabresse (Ecole des Mines de Douai) et Marcus Denker (INRIA Lille Nord Europe) sous la direction de Stéphane Ducasse (INRIA Lille Nord Europe), décembre 2013.

[PBJ98] F. Plasil, D. Balek, and R. Janecek. SOFA/DCUP : Architecture for component trading and dynamic update. In *ICCDS'98 : Proceedings of the 4th IEEE International Conference on Configurable Distributed Systems*, pages 35–42, 1998.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[Pol15] Guillermo Polito. Isolation et modularisation de systèmes réflexifs. Thèse co-encadrée avec Luc Fabresse (Ecole des Mines de Douai) et Marcus Denker (INRIA Lille Nord Europe) sous la direction de Stéphane Ducasse (INRIA Lille Nord Europe), 2015.

[Rob03] Romain Robbes. Mise en oeuvre de la programmation par aspects dans le cadre des systèmes multi-agents. Master's thesis, Université de Caen, 2003.

[RP96] Pierre Roy and François Pachet. Reifying constraint satisfaction in smalltalk. *Journal of Object-Oriented Programming,*, 1996.

[SCS02] A. Senart, O. Charra, and J.-B. Stefani. Developing dynamically reconfigurable operating system kernels with the think component architecture. In *ECOOSE' 02 : Proceedings of the Workshop on Engineering Context-aware Object-Oriented Systems and Environments, in association with OOPSLA'02*, Seattle, USA, November 2002.

[Sen03] Aline Senart. *Canevas logiciel pour la construction d'infrastructures logicielles dynamiquement adaptables*. PhD thesis, Institut National Polytechnique de Grenoble, November 2003.

[SF93] Mark E. Segal and Ophir Frieder. On-the-fly program modification : Systems for dynamic updating. *IEEE Software*, 10(2) :53–65, 1993.

[SFV05] Davy Suvée, Bruno De Fraine, and Wim Vanderperren. Fusej : An architectural description language for unifying aspects and components. In *Proceedings the AOSD SPLAT Workshop : Software engineering Properties of Languages for Aspect*, mar 2005.

[SVJ03] Davy Suvée, Wim Vanderperren, , and Viviane Jonckers. Jasco : an aspect-oriented approach tailored for component based software development. In *Proceedings of AOSD*, 2003.

[Szy98] C. Szyperski. *Component software : beyond object-oriented programming*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1998.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

5. Infrastructures logicielles modulaires

Une bonne pratique dans le développement logiciel en général et celui des applications robotiques en particulier consiste à recourir à des infrastructures logicielles. Par infrastructures logicielles nous entendons des *middlewares*, des *frameworks* ou encore des moteurs d'exécution (*run-time*). Il s'agit d'artefacts logiciels qui aident à résoudre des problèmes récurrents, en encadrant le processus de développement ou en offrant des services durant le cycle de vie d'une application. Plus précisément, il s'agit de faciliter la mise au point de ces logiciels, leur packaging, leur déploiement ainsi que la gestion des ressources à l'exécution en utilisant la *modularisation* et les langages *dynamiques*.

Notre travail sur les infrastructures trouve son origine dans notre besoin d'aller plus loin que la simple validation expérimentale des modèles théoriques. Le but est de constituer une bibliothèque d'outils réutilisables, qui peuvent servir à la fois comme démonstrateurs et comme support à d'autres travaux. L'exemple le plus récent en la matière est en rapport avec le *middleware* robotique ROS (Robot Operating System) [1]. Nous développons dans le cadre du projet ROBOSHOP (voir section D.2 page 140) PHAROS [2] qui est un client pour ROS réalisé en PHARO. Il sera utilisé notamment dans un démonstrateur de robot guide en galerie marchande, qui sera présenté au salon VAD Conext en octobre 2013. C'est dans ce cadre que je co-encadre Santiago Bragagnolo ingénieur de recherche (18 mois) qui participe au projet depuis septembre 2012.

Un autre exemple d'infrastructure qui cible la répartition est celui de l'intergiciel orienté-service UBIQUITALK [3]. La plate-forme UBIQUITALK a été initiée dans le cadre du projet MOSAÏQUES (voir section D.7 page 145) par Michaël Piel, ingénieur de recherche (12 mois) de 2005 à 2006. Par la suite, Gautier Dhordain a continué pendant 6 mois (2008-2009) à développer UBIQUITALK pour finaliser un démonstrateur sur le thème du commerce ubiquitaire, qui a été présenté à différentes reprises, notamment dans le cadre du pôle de compétitivité Industries du Commerce (PICOM). Enfin, nous avons exploré un autre domaine d'application qui est le calcul réparti pour des simulations physiques [GABD10] dans le cadre du projet ODICE (voir section D.6 page 144). Ce dernier travail a été mené par 2 post-doctorants Gautier Bastide et Guillaume Grondin qui ont successivement travaillé sur le sujet pendant 6 mois chacun.

Les infrastructures logicielles constituent également l'objet de notre collaboration avec l'équipe RMOD de l'INRIA Lille (voir D.3 page 142). C'est dans le cadre de ce partenariat que je co-

1. http://ros.org
2. http://car.mines-douai.fr/category/pharos
3. http://vst.mines-douai.fr/UbiquiTalk

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

encadre actuellement deux doctorants. Le premier est Nick Papoulias [Pap13] qui travaille sur une infrastructure réflexive à base de miroirs (*mirrors*) [BU04] pour le débogage à distance. La seconde thèse est celle de Guillermo Polito [Pol15] qui travaille sur une infrastructure qui permet de réduire l'empreinte mémoire des applications réflexives tout en assurant leur isolation.

La collaboration avec l'INRIA a débuté avec la thèse de Mariano Martinez Peck soutenue en 2012 [MP12]. Elle a débouché sur MAREA, une solution de mémoire virtuelle au niveau applicatif qui descend au niveau de granularité de l'objet [MPBD+13]. Cette infrastructure se substitue à la mémoire virtuelle des systèmes d'exploitation qui s'avère inappropriée car déconnectée du contexte applicatif et travaillant à l'échelle de pages mémoire. A contrario, MAREA permet au développeurs de l'application de définir une politique de gestion de la mémoire en travaillant sur des graphes d'objets.

Enfin, notons que la thèse de Guillaume Grondin [Gro08] avait une facette infrastructure. L'implémentation AUTOFRACTAL [4] du modèle MADCAR dispose de différentes sondes qui renseignent le moteur d'adaptation sur l'état interne de l'application et l'environnement dans lequel elle est située. Cet ensemble de sondes est ouvert et peut être enrichi suivant les besoins applicatifs. Par ailleurs, un sous-produit intéressant d'AUTOFRACTAL est le portage du *framework* BACKTALK du Smalltalk commercial VisualWorks vers le Smalltalk libre Squeak [NDPB09].

Dans la suite de ce chapitre, je décris les travaux effectués par deux doctorants que j'ai co-encadrés. La section 5.1 résume les contributions de Mariano Martinez-Peck sur une infrastructure d'exécution. Elle permet de prendre en compte dans les applications la quantité de mémoire disponible sur les robots. La seconde thèse décrite en section 5.2 (page 70) est celle de Nick Papoulias. Elle traite d'une infrastructure pour le débogage *in vivo* de logiciels de contrôle de robots.

5.1 Une mémoire virtuelle au niveau applicatif

5.1.1 Motivation

Ce travail a été mené dans le cadre de la thèse de Mariano Martinez-Peck soutenue en 2012 [MP12]. Le constat de départ est que la mémoire est toujours une ressource critique pour les logiciels. C'est le cas dans les systèmes embarqués et en particulier les robots de petite taille, la quantité de mémoire disponible est limitée.

Dans le cas de robot de grande taille, ceux-ci embarquent des ordinateurs portables complets avec plusieurs gigaoctets de mémoire vive. Généralement, cette quantité de mémoire est largement suffisant pour exécuter la plupart des logiciels. Cependant, qui dit plus de mémoire dit une plus grande consommation électrique, une autre ressource critique sur tous les robots fonctionnant sur batterie. Par ailleurs, les 4 ou même 8Go dont disposent la plupart des machines de nos jours restent largement inférieurs aux centaines de gigaoctets des disques durs. L'essentiel des données résident donc sur des supports de mémoire secondaire qui restent lent d'accès.

4. http://vst.mines-douai.fr/grondin/14

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

5.1.2 L'existant

La gestion de la ressource mémoire est une problématique qui a suscité de multiples travaux. Nous les avons classés en familles que nous décrivons ci-dessous.

- **Utilisation de *run-time a priori* compact** Cette famille de solutions se base sur l'utilisation d'une infrastructure d'exécution et des bibliothèques (*run-time*) qui offrent un ensemble de fonctionnalités minimal. C'est le cas par exemple de J2ME qui est une version de Java auquel bon nombre de fonctionnalités et autres classes de bases ont été retirés. Le problème de cette approche est que les choix faits par le concepteur de J2ME ne conviennent pas à toutes les applications. Par exemple, le retrait des packages java.lang.reflect et de leur support au niveau machine virtuelle exclu *de facto* les applications qui requièrent de l'introspection. A contrario, il est aisé d'imaginer que toutes les fonctionnalités offertes par le *run-time* (par exemple JDBC pour J2ME) ne sont pas utilisées dans toutes les applications. Il en résulte donc une occupation inutile d'une partie de la mémoire.
- **Compactage *a posteriori* du *run-time*** L'idée intéressante portée par les travaux de cette famille consiste à laisser les développeurs de l'application décider des parties du *run-time* à supprimer. Cette approche a été adoptée dans JITS [CGV10] où les développeurs d'applications pour mobiles codent avec un *run-time* Java complet, destiné aux applications pour machines de bureau (Java 2 SE). Le processus de développement est étendu avec une phase d'élagage du *run-time* où la machine virtuelle java et les bibliothèques de base sont compactées par les développeurs de l'application. Cependant cette tâche est fastidieuse et peut être source d'erreurs. D'où l'idée d'automatiser le processus sur la base d'une analyse statique pour identifier les parties utilisées dans les applications et retirer.
  Les travaux qui ont recours à l'analyse statique requièrent de disposer d'un typage de l'application à compacter. C'est à ce niveau que se situe le défi majeur de ces approches. Outre le problème d'inférence de type dans les langages dynamiques, l'analyse pose problème dans les langages réflexifs, qu'il soient dynamiques ou statiques [BSS+11].
- **Utilisation d'une Mémoire Virtuelle** Une troisième famille de solutions consiste à recourir à un gestionnaire de mémoire virtuelle. Ce dernier utilise un support mémoire secondaire -typiquement une partition de *swap* d'un disque dur- pour stocker les pages mémoires de la RAM qui ne sont pas utilisées à un instant donné. Quand ces dernières sont nécessaires, elles sont transférées vers la RAM en remplacement d'autres parties qui sont transférées vers le disque dur. Dans le cas d'une application qui requiert plus de mémoire que de RAM disponible, seules les pages mémoire utilisées à un instant donné seront en RAM. Le reste sera stocké provisoirement sur une mémoire secondaire.
  Cette idée largement utilisée dans les systèmes d'exploitation souffre d'une limitation majeure. Le gestionnaire de la mémoire virtuelle traite les applications comme des boîtes noires. L'allocation de la mémoire et le transfert de son contenu (*swap*) sont découplés de l'activité des applications. Dès lors, il est très difficile de gérer des situations où l'application manipule de manière répétée des pages différentes de la mémoire. Par exemple, des objets créés à des moments différents par l'application et donc situés sur des pages mémoire différentes, peuvent se référencer. Quand un traitement manipule de tels objets de manière répétée (typiquement le ramasse-miette), les différentes pages mémoire sont

© 2014 Tous droits réservés.
doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

transférées à tour de rôle de la RAM vers le disque dur et vice-versa [YBK+06, HFB05]. C'est le phénomène du *thrashing* où le système d'exploitation accapare les ressources de la machine.
Ce problème est rencontré dans certains systèmes d'exploitation tels que Mach [You87], V++ [Che88] et Apertos [Yok92, Yok93]. Les développeurs des applications ont la possibilité de fournir leur propre gestionnaire de mémoire virtuelle. Cependant, l'effort de développement d'un tel gestionnaire et la compétence technique requise réduisent l'intérêt de cette ouverture.
Les travaux Krueger et al. [KLVA93] permettent de simplifier l'intervention des développeurs. Ils fournissent un canevas outillé pour aider au développement et au test de différentes stratégies de gestion de la mémoire.
L'approche d'Engler et al. [EKO95] s'attaque au même verrou avec un micro-noyau qui permet la gestion des ressources au niveau applicatif. Les développeurs d'application peuvent adapter la politique qui régit le fonctionnement du gestionnaire de mémoire virtuelle.
L'idée d'une mémoire virtuelle a également été reprise dans le contexte d'infrastructures objet. C'est le cas de Melt [BM08] qui détecte les objets Java inutilisés dans les applications et les transfère de la RAM vers le disque. Le but de ses concepteurs était d'éviter l'arrêt intempestif des applications sujettes à des fuites mémoire qui saturent la RAM.
LOOM [Kae86] développé pour Smalltalk-80, gérait une mémoire virtuelle avec également une granularité objet. L'objectif était de s'accommoder de très faibles quantités de RAM caractéristiques des machines de l'époque. Il traite aussi des problèmes qui ne sont plus d'actualité du fait des progrès réalisés par exemple sur les ramasse-miettes ou sur la vitesse des disques durs.

Nous nous intéressons plus spécifiquement à la famille de solutions basées sur une mémoire virtuelle. Afin de comparer les travaux nous avons identifié quatre critères que nous listons ci-dessous.

- **Granularité objet.** Est-ce que la mémoire virtuelle manipule des objets ou des pages d'octets ?
- **Couverture.** Est-ce que le gestionnaire de la mémoire virtuelle manipule aussi bien les objets applicatifs que le code ?
- **Contrôlabilité au niveau applicatif.** Est-ce que les développeurs peuvent adapter la mémoire virtuelle aux besoins applicatifs ?
- **Portabilité.** Est-ce que la mémoire virtuelle est portable entre différents systèmes d'exploitation ou architectures matérielles ?

	Granularité Objet	**Couverture**	**Contrôlabilité**	**Portabilité**
OS Mach, V++ et Apertos	-	+++	+	-
OS Krueger et al.	-	+++	+++	-
OS Micronoyau	-	+++	+++	-
Melt	+++	-	-	+++
LOOM	+++	+++	-	-

TABLE 5.1 – Evaluation des travaux utilisant une mémoire virtuelle

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

Le tableau 5.1 résume l'évaluation de l'état de l'art. Les solutions autour de systèmes d'exploitation souffrent essentiellement de deux problèmes. Le premier est le fait que la mémoire virtuelle travaille au niveau des octets et non au niveau des objets. Dès lors, le contrôle mis à disposition des applications objet est délicat. Les développeurs doivent gérer le délicat problème de la correspondance entre objets et octets en mémoire. La portabilité est également un problème. Une stratégie de gestion de la mémoire développée pour un système requiert un effort de la part des développeurs pour le portage, quand cela est possible.

Si on regarde les approches qui travaillent au niveau des objets, la situation n'est pas satisfaisante non-plus. Melt qui bénéficie de la portabilité de sa machine virtuelle Java modifiée ne couvre que les objets. Quant à LOOM, s'il traite aussi bien les objets que les classes, il n'est pas portable car il a été développé pour une architecture matérielle bien spécifique. Enfin, les stratégies de gestion de la mémoire virtuelle de Melt et LOOM ne sont pas contrôlables par les applications.

5.1.3 Nos travaux

Les travaux que nous avons réalisés appartiennent à la famille de solutions basées sur une mémoire virtuelle. Ils relèvent plus spécifiquement des mémoires virtuelles dirigées par des applications pour les langages dynamiques à objets. L'originalité de notre proposition est qu'elle repose sur une mémoire virtuelle qui est à la fois : avec une granularité objet et contrôlable au niveau applicatif. Les développeurs d'une application désignent les graphes d'objets à transférer et définissent le moment de les déplacer de la RAM vers le disque dur. Quand les objets sont à nouveaux nécessaires, ils sont rechargés automatiquement dans la RAM. Le choix des objets à transférer vers la mémoire secondaire est dirigé par la connaissance des développeurs de l'application, ainsi que de l'usage fait des objets. Ainsi, la mémoire est libérée à bon escient.

Notre solution, appelée MAREA [MPBD+13] cumule plusieurs propriétés intéressantes. Nous décrivons ci-dessous les plus importantes d'entre elles, à savoir : la généralité, la transparence et la célérité.

Généralité

Les langages dynamiques réifient sous forme d'objets différents éléments du programme et du support d'exécution. C'est le cas par exemple de la pile d'exécution, des méthodes et des classes dans Smalltalk. La propriété de généralité correspond à la capacité du gestionnaire de mémoire virtuelle de transférer n'importe quel objet, que ce soit un objet de base ou bien une réification.

Cette propriété est intéressante car elle permet de maximiser la quantité de mémoire libérée. Elle pose en revanche un double défi. D'une part il faut pouvoir sauvegarder et restituer n'importe quel objet, en particulier les réifications. D'autre part, il faut pouvoir remplacer n'importe quel objet par un *proxy* qui plus est capable de capturer toutes les tentatives de manipulation pour déclencher le rechargement dans la RAM du graphe d'objets.

Afin de répondre à ces défis, le langage support doit permettre de créer et de manipuler les réifications à l'exécution. Par manipuler, nous entendons donner accès en lecture (*introspection*)

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

à leurs structure pour la sauvegarde sur disque. De même, il faut disposer d'un accès en écriture (*intercession*) afin de les initialiser lors de la restitution sur la RAM. Ce support à la manipulation doit s'étendre également aux liens -typiquement des références- que peuvent disposer d'autres objets vers les réifications. Si nous considérons l'exemple d'une classe C réifiée, le langage doit permettre de lire et modifier sa structure, par exemple le lien vers la superclasse ou la collection de méthodes. Il doit aussi permettre de retrouver les sous-classes et les instances de C, pour remplacer leurs références vers cette classe par un *proxy* si celle-ci doit être transférée vers le disque.

L'implémentation de MAREA repose sur le langage PHARO qui offre ces différentes capacités réflexives. Mais, cela est insuffisant car le remplacement des réifications par des *proxies* simples provoque un arrêt intempestif de la machine virtuelle. En effet, celle-ci fait des hypothèses sur la structure de certains objets tels que les classes ou les méthodes. Nous avons dû donc concevoir et implémenter GHOST une nouvelle bibliothèque de *proxies* qui prend en compte les spécificités des réifications [MPBD+ed]. GHOST permet de construire des *proxies* qui ont une structure compatible avec les réifications qu'ils remplacent. Par ailleurs, ces *proxies* capturent les différentes façons d'utiliser les réifications et déclenchent leur restitution en mémoire principale. Par exemple, une classe réifiée peut non seulement recevoir des messages, mais elle est également utilisée par la machine virtuelle pour la recherche de méthode lorsqu'une de ses instances ou une instance d'une sous-classe reçoit un message. Un autre exemple est celui d'une méthode réifiée qui est exécutée.

Transparence

La propriété de transparence d'une mémoire virtuelle peut être abordée suivant différentes facettes. Un premier point de vue est celui des traitements effectués par les applications. Ces traitements doivent être identiques en présence et en l'absence de la mémoire virtuelle.

Le second point de vue est celui du code source de l'application. Il doit rester inchangé même si nous avons recours à une mémoire virtuelle. Même si les développeurs peuvent contrôler la gestion de la mémoire virtuelle, le code pour transférer les objets entre la mémoire principale et secondaire doit être totalement découplé du code applicatif.

Dans MAREA, les développeurs ont la possibilité de transférer un graphe d'objets vers le disque dur. Ce traitement peut être effectué avant le déploiement des applications. Dans ce cas MAREA est utilisé comme un outil de développement, indépendant de toute application. Les objets déchargés qui sont référencés par d'autres objets applicatifs sont remplacés par des *proxies*. Le code des *proxies* qui gère le rechargement dans la RAM est générique car indépendant de toute application. Nous avons montré que cette utilisation de MAREA permet de réduire l'empreinte mémoire de 25% à 40% pour différentes applications réelles.

Nous travaillons actuellement sur une automatisation du processus de transfert de la RAM vers le disque. Dans ce cas, les développeurs expriment une politique qui permet à MAREA de choisir les graphes d'objets à transférer et le moment de les décharger de la RAM. Le rechargement reste automatique quand le besoin se présente. Notre approche est de considérer la gestion de la mémoire virtuelle comme un aspect au sens de la programmation par aspects. Ce code est transversal à l'application et de ce fait garantit la transparence.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

Célérité

Afin qu'une solution soit réellement utilisable, il faut qu'elle opère le plus rapidement possible afin que les temps de réponse des applications restent sensiblement inchangés. C'est la propriété de célérité. Il s'agit de minimiser le surcoût temporel introduit par une mémoire virtuelle.

Une manière de réduire ce surcoût est de minimiser le nombre de transferts d'objets entre la RAM et le disque. Pour cela, nous groupons les objets par graphes qui sont transférés ensemble de la RAM vers le disque dur et vice-versa. Etant donné un ou plusieurs objets racines, tous les objets accessibles directement ou indirectement (fermeture transitive) font partie du même graphe. Lorsqu'un graphe est retiré de la RAM, tous les objets racines sont remplacés par des *proxies*. C'est le cas également des quelques autres objets dits *partagés*, car faisant partie de plus d'un graphe (par exemple les objets C et D de la figure 5.1). Tous les autres objets sont simplement détruits. L'envoi d'un message à un *proxy* déclenche le rechargement en RAM de la totalité du graphe.

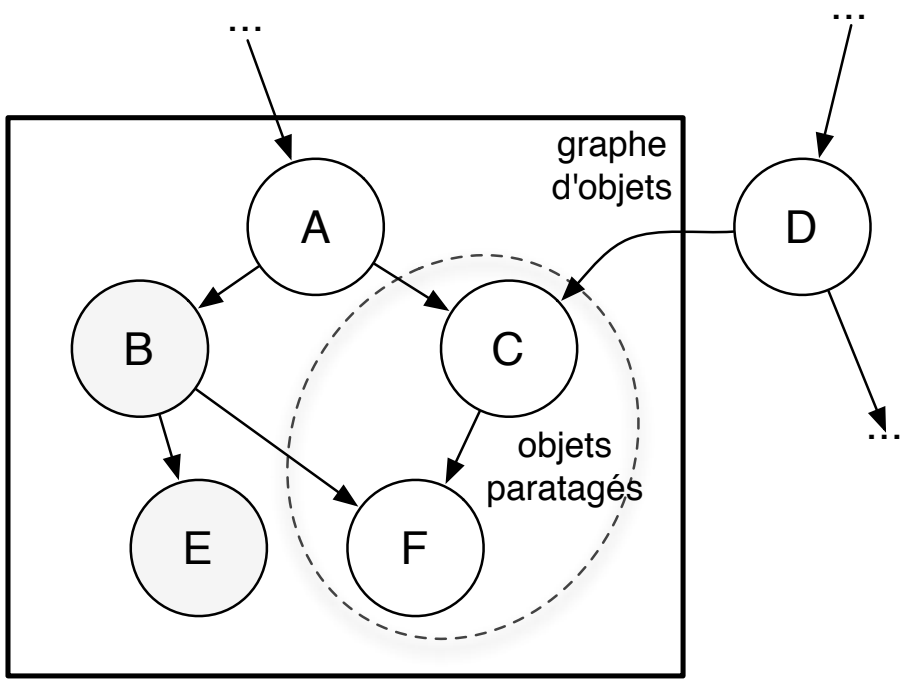

FIGURE 5.1 – Objets partagés entre différents graphes

La détection des objets *partagés* est un point délicat qui peut sérieusement ralentir les transferts. Notre solution permet d'avantageusement éviter de parcourir la totalité de la mémoire. L'idée est de remplacer tous les objets du graphe par des *proxies*. Comme les *proxies* ne se référencent pas, le ramasse-miettes détruit les *proxies* superflus et ne conserve que ceux qui sont

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

référencés par ailleurs.

Une troisième source de ralentissements potentiels est le remplacement d'objets par des *proxies* et vice-versa. Le replacement d'un objet o_1 par un autre objet o_2 nécessite de pouvoir retrouver toutes les références vers o_1 pour les faire "pointer" vers o_2. Notre implémentation du remplacement repose sur la variante de la primitive become: offerte par la machine virtuelle PHARO qui permet de faire un remplacement de plusieurs objets en une seule opération. Le gain obtenu par le traitement en code natif d'une masse d'objets est contre balancé par le fait que cette primitive réalise dans PHARO un parcours de la totalité de la mémoire. Cet inconvénient doit être néanmoins nuancé. Il est spécifique à l'implémentation du become: dans PHARO. Des implémentations plus performantes en évitant le parcours de toute la mémoire sont disponibles dans d'autres dialectes SMALLTALK. Enfin, le temps d'exécution de notre implémentation de MAREA reste acceptable.

Nombre d'objets	**Durée du transfère RAM → disque (ms)**	**Durée du transfère disque → RAM (ms)**
51	40	37
236	47	44
777	39	41
5758	130	50
21753	256	122

TABLE 5.2 – Exemples de durées de transfères de graphes d'objets avec MAREA

Le tableau 5.2 donne des exemples de temps de transfert pour des graphes de différentes tailles avec notre implémentation de MAREA. Outre le fait que ces durées restent relativement faible, soulignons leur asymétrie. Cette asymétrie est un choix de conception dans MAREA. Le point de départ est que les transferts de la RAM vers le disque se font en tâche de fond, quand les objets ne sont plus utilisés. En revanche, les transferts inverses (disque vers RAM) se font au moment même où l'application requiert les objets. C'est cette partie qui est la plus critique. Nous avons donc adopté un format de sérialisation qui minimise la reconstruction d'objets [DMPDA12]. Cette optimisation requiert une préparation qui est effectuée au moment de transférer les graphes d'objets de la RAM vers le disque.

5.2 Débogage à distance

5.2.1 Motivation

Le travail mené par Nick Papoulias dans sa thèse part du constat que le développement de logiciels pour des systèmes contraints en ressources tels que les robots requiert deux machines : la cible du logiciel et la machine de développement. En effet, les contraintes sur les ressources de calcul (mémoire, CPU) interdisent souvent l'installation de l'environnement de développement sur la machine cible, ou tout au moins en rendent l'usage difficile. Par ailleurs, d'autres

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

contraintes sur les entrées/sorties (absence des ports pour le clavier ou l'écran) peuvent restreindre les possibilités d'intervention des développeurs sur de tels machines.

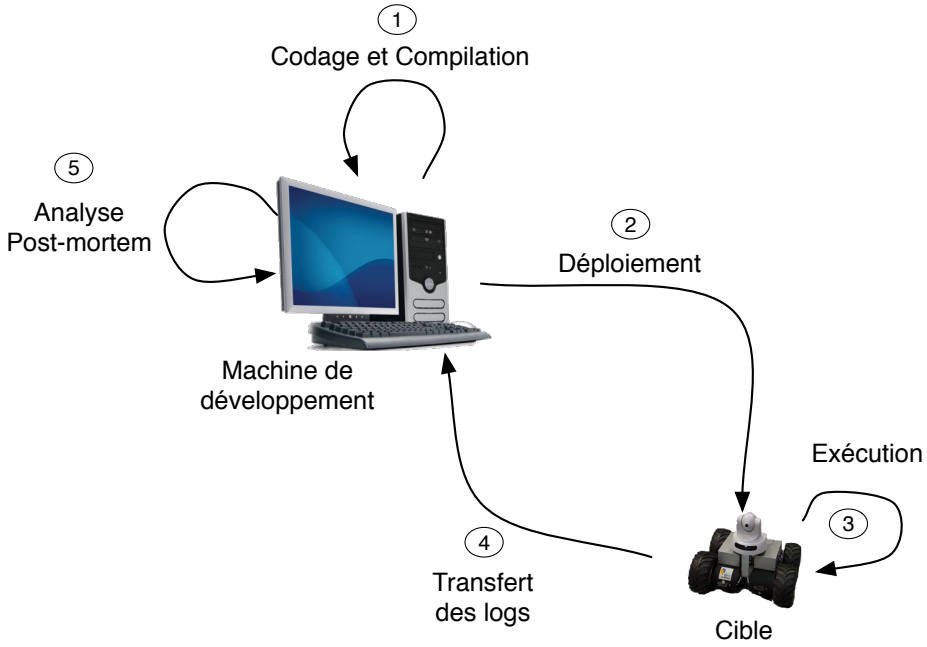

FIGURE 5.2 – Débogage d'un logiciel pour un robot en l'absence d'outils appropriés

Dans ce cadre, le processus de développement et plus spécifiquement le débogage [5] devient particulièrement fastidieux. La figure 5.2 montre les différentes étapes nécessaire pour déboguer une application destinée à un robot, en l'absence d'outils appropriés. Après codage et compilation sur la machine du développeur (étape 1 de figure 5.2), le code est déployé et exécuté sur le robot (étapes 2 et 3). Les traces (*logs*) de l'exécution sont ensuite transférées vers la machine du développeur (étape 4) pour une analyse *post mortem* des traitements réalisés (étape 5).

Souvent, les traces récoltées ne suffisent pas à déterminer la source d'un dysfonctionnement. Il faut alors modifier le programme ou ses paramètres d'exécution et réaliser à nouveau la séquence déploiement, exécution, recueil des *logs* en espérant avoir toutes les informations pour comprendre le problème. Une fois le programme modifié pour palier une erreur, le développeur doit réaliser un nouveau cycle pour s'assurer que la modification corrige bien le dysfonctionnement initialement constaté et n'introduit pas de nouvelles bogues.

5.2.2 L'existant

Emulateurs et Simulateurs

Une approche possible est d'utiliser des émulateurs, comme c'est le cas pour certains *smartphones* ou des simulateurs robotiques comme Morse [ELD+12], Gazebo [KH04] ou Webots [Mic04].

5. ou déverminage, en anglais *debug*

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

Dans ce cas, l'exécution se fait sur une machine accessible aux développeurs.

Cependant, émulateurs et simulateurs ne reprennent qu'une partie des caractéristiques des machines réelles et simplifient la réalité du monde physique. Les capteurs, actionneurs ainsi que les contraintes en ressources sont souvent mal simulés quand ils ne sont pas simplement ignorés. Notons enfin que le recours à un émulateur n'est pas toujours possible. Pour beaucoup d'équipements, un émulateur n'existe pas et serait coûteux à développer.

JPDA de Java

JPDA (*Java Platform Debugger Architecture*) est la solution JAVA pour le débogage à distance [JPD]. Elle repose sur une extension de la machine virtuelle JAVA, donc du code natif, accessible via JVMTI (*Java Virtual Machine Tool Interface*). Le débogueur peut être développé en n'importe quelle technologie, du moment qu'il soit en mesure de communiquer suivant le protocole réseau JDWP (*Java Debug Wire Protocol*) avec l'interface JVMTI de la machine virtuelle de l'application déboguée. Dans le cas des implémentations développées en JAVA, celles-ci doivent être conformes à l'API JDI (Java Debug Interface) [JDI].

Grâce à JPDA infrastructure, il est possible en JAVA d'introspecter l'état d'une machine virtuelle en cours d'exécution, suspendre ou activer les *threads*, placer des points d'arrêt, ou activer des notifications lors par exemple des exceptions et des chargements de classe. Il est également possible d'obtenir la pile d'appel des méthodes, modifier une variable temporaire ou un paramètre. De même, JPDA permet d'envoyer des messages aux objets et d'accéder aux champs en lecture et en écriture. La réflexion structurelle est par contre limitée à l'introspection. Il est possible par exemple d'accéder en lecture à une classe obtenir ses champs, ses méthodes, sa superclasse et éventuellement l'instancier. Mais, il n'est pas possible de modifier les classes déjà chargées.

JRebel et DCE

DCE [WWS10] et JRebel [Zer12] proposent tous les deux des extensions de la machine virtuelle JAVA afin d'autoriser le remplacement à l'exécution de classes déjà chargées. Bien que ces solutions ne traitent pas directement du débogage, elles peuvent être utilisées pour lever une partie des limitations de l'infrastructure de débogage JPDA (voir section 5.2.2).

.Net

La solution de débogage à distance de .Net est intégrée à Microsoft Visual Studio [Mic12b]. Comme pour JPDA de JAVA, la solution de .Net permet l'introspection de l'application en cours d'exécution. De plus, elle permet de modifier le programme en cours du débogage, mais avec bon nombre de restrictions [Mic12c]. Enfin, notons que .Net intègre une solution pour sécuriser le débogage à distance [Mic12a]. C'est notamment utile pour préserver la machine déboguée d'attaques qui exploiteraient les points d'entrée offerts au débogueur.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

GDB

GDB (GNU Debugger) [RS03] est utilisé pour le débogage du langage C et ses sur-couches comme C++ et Objective-C. Afin de déboguer des machines distantes, GDB requiert le lancement d'un serveur *gdb-server* sur la machine cible. Ce serveur vient s'attacher au processus de l'application à déboguer qui a été compilée en activant l'intégration des méta-informations de débogage. En plus d'insérer des points d'arrêt et de lire les valeurs des variables, GDB permet une forme limitée de modification à chaud du code. Il s'agit de changements bas-niveau (niveau des octets) de l'exécutable déjà chargé dans la RAM.

Smalltalk

SMALLTALK se caractérise par un puissant débogueur [LP90]. Il permet de modifier n'importe quelle partie du logiciel débogué du fait des capacités réflexives du SMALLTALK [Riv96]. C'est d'ailleurs cette possibilité qui a facilité la mise en place des approches agiles et plus spécifiquement du développement dirigé par les tests [ABF05]. Cependant, le débogueur SMALLTALK ne permet de travailler que sur du code qui est localisé dans la même image.

Bifrost

Bifrost [Res12] est une extension des capacités réflexives de Smalltalk qui a été utilisée notamment pour le débogage *centré-objet* [RBN12]. Dans les débogueurs traditionnels, le développeur se retrouve typiquement trop près du code. Celui-ci peut mêler des traitements qui impliquent différents objets et variables alors que le développeur peut n'être intéressé que par un objet particulier. La solution de débogage construite autour de Bifrost permet de traiter ce problème. Elle offre par exemple un mode pas à pas au niveau objet : le développeur reprend la main, chaque fois que l'objet reçoit un message ou est passé comme paramètre. Si cette solution améliore considérable l'efficacité lors du débogage des applications à objets, elle n'offre cependant pas de support pour le débogage à distance.

Comparaison des travaux

Afin de comparer les différents travaux listés ci-dessus, nous avons défini les quatre critères ci-dessous :

- **Interactivité :** C'est la capacité d'inspecter et modifier pendant l'exécution les objets de l'application déboguée, ainsi que son code.
- **Couverture :** Il s'agit de l'étendue, la granularité et le niveau d'abstraction des points où peut intervenir le développeur durant la mise au point d'un programme. Dans une application à objet, le développeur doit pouvoir par exemple insérer un point d'arrêt conditionné par le changement d'un champ particulier, d'un objet particulier.
- **Distribution :** Ce critère correspond au niveau de flexibilité du *middleware* utilisé pour le débogage à distance. Idéalement, il doit pouvoir être adapté aux besoins de chaque architecture rencontrée par les développeurs en fournissant par exemple différents protocoles et différentes stratégies de déploiement.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

– **Sécurité :** Il s'agit de sécuriser les points d'accès offerts par l'infrastructure pour le débogage, afin d'éviter un détournement par un tiers indésirable. Ce problème se pose quand le débogage est effectué au travers un réseau non sécurisé de logiciels embarqués sur des équipements sensibles.

	Interactivité	Couverture	Distribution	Sécurité
JPDA	+	+	+	+++
.NET	+	+	++	+++
GDB	+	+	+	++
DCE	+++	+	+	+++
JRebel	+++	+	+	+++
Smalltalk	+++	+	-	-
Bifrost	+++	+++	-	-

TABLE 5.3 – Evaluation des travaux en lien avec le débogage à distance

Le tableau 5.3 résume l'évaluation des différents travaux. La solution la plus intéressante en termes d'interactivité et de couverture est Bifrost. Les capacités d'intercessions poussées combinées avec un niveau de granularité centré objet rend l'approche de Bifrost particulièrement séduisante pour le débogage. Cependant, il n'offre aucun support à la distribution tout comme Smalltalk. Les autres travaux permettent certes le débogage à distance avec un bon niveau de sécurité, mais les *middlewares* sont rigides. La seule exception est .Net qui bénéficie d'un certain niveau de flexibilité avec les paramètrages et l'extensibilité des services offerts par son *middleware* COM+.

5.2.3 Nos travaux

L'idée : débogage interactif à distance

Le travail réalisé dans le cadre de la thèse de Nick Papoulias repose sur l'utilisation de la réflexion pour déboguer à distance. L'environnement de développement est situé sur la machine du développeur et permet de déboguer de manière *interactive* le logiciel en cours d'exécution sur la machine cible.

La figure 5.3 donne les différentes étapes de ce processus de débogage à distance. L'interactivité qui caractérise notre proposition apparaît à partir de l'étape de 3. En effet, l'exécution est observée à distance par le débogueur qui est mis à jour avec l'état de l'application (étape 4). Par ailleurs, le développeur peut être actif (étape 5). Il peut ainsi modifier *à la volée* les informations observées pour analyser le fonctionnement de l'application, interrompant l'exécution en insérant des points d'arrêt. Il peut également être plus intrusif en modifiant l'état de l'application (par exemple la valeur d'un champ) ou même son code (par exemple le remplacement d'une méthode ou l'ajout d'une classe).

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

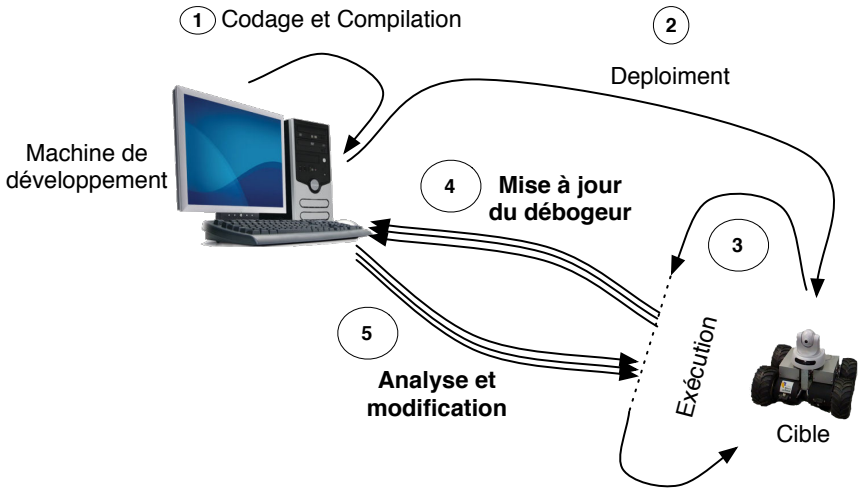

FIGURE 5.3 – Débogage à distance avec notre solution

La solution : utilisation de méta-objets miroirs

La solution proposée dans la thèse de Nick Papoulias repose sur l'utilisation de la réflexion et plus spécifique des *miroirs* [BU04]. Initialement introduits dans le langage à prototypes SELF [SU95], les miroirs sont des méta-objets découplés du niveau de base. Ainsi, il est possible d'avoir différentes implémentations du méta-niveau. Différents méta-niveaux peuvent se distinguer par la réification de plus ou moins d'entités ou par des capacités d'intercession plus ou moins limitées à certaines entités. De même, différents méta-niveaux peuvent fournir des implémentations différentes d'un même MOP.

Avec MERCURY, nous nous avons exploré l'idée d'un méta-niveau qui permettrait le débogage du niveau de base, à distance. La figure 5.4 donne une représentation abstraite des différentes couches de notre solution appelée MERCURY. L'application située sur la machine cible correspond en fait au niveau de base. Le méta-niveau déporté sur la machine du développeur est exploité par le débogueur pour analyser et corriger le niveau de base. Les opérations d'introspection et d'intercession nécessitent de disposer sur machine cible d'une infrastructure qui permet au méta-niveau de réaliser toutes les opérations réflexives. Il s'agit d'un ensemble de primitives bas niveau qui donnent accès par exemple en lecture et en écriture à la structure en mémoire des objets applicatifs et des éléments de l'infrastructure d'exécution (par exemple la pile d'exécution).

Le déport du méta-niveau permet de gagner sur deux plans : l'empreinte mémoire sur la machine cible et le trafic réseau. Du point de vue de la mémoire, les méta-objets se trouvent sur la mémoire de la machine du développeur. Ils manipulent le niveau de base via le réseau. Cette communication reste néanmoins à un niveau modéré. Elle intervient lorsque l'utilisateur souhaite obtenir une information d'exécution (par exemple : pile d'appels de méthode, valeur d'une variable) ou modifier la cible (par exemple : modification d'une référence, remplacement d'une méthode). Les communications peuvent être également initiées par l'infrastructure sur

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

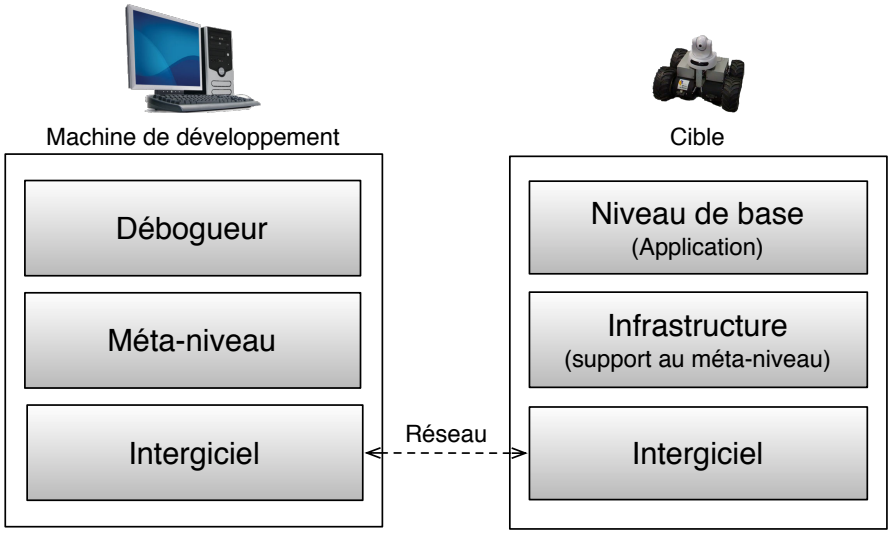

FIGURE 5.4 – Notre solution repose sur un méta-niveau déporté pour le débogage à distance

la cible, pour notifier un événement tel que la modification d'un champ ou la réception d'un message par objet observé par le débogueur.

Les communications entre les deux parties permettent de maintenir le lien causal entre le niveau de base et le niveau méta. Néanmoins, les développeurs peuvent couper provisoirement ce lien dans le but de tester l'application sans perturbations et s'approcher ainsi des conditions de production. En plus de déconnecter le méta-niveau, notre proposition permet également de décharger l'essentiel de l'infrastructure et libérer ainsi le maximum d'espace mémoire pour l'application. Seul reste un noyau qui permet de capturer les exceptions et leurs contextes pour permettre aux développeurs de traquer les erreurs qui se sont produites à l'issue du test. Ce noyau permet également de recharger l'infrastructure suite à une requête débogueur (mode *pull*) ou quand un événement défini par les développeurs se produit (par exemple des exceptions). Le méta-niveau est alors re-connecté au niveau de base, puis actualisé afin de refléter l'état final de l'application.

5.3 Bilan du chapitre

Dans ce chapitre, j'ai présenté un résumé de mes travaux sur les infrastructures logicielles pour la robotique. J'ai particulièrement mis l'accent sur deux thèses que j'ai co-encadrées dans le cadre d'un partenariat avec l'équipe RMoD de l'INRIA Lille.

La première thèse a débouché sur MAREA, une infrastructure destinée à la gestion de la ressource mémoire. En effet, MAREA est une mémoire virtuelle, à la granularité des objets, pilotable au niveau applicatif. Cette solution permet d'ajuster la stratégie de gestion de la mémoire en fonction du logiciel de contrôle des robots. Ainsi, nous introduisons de la flexibilité dans la gestion de la mémoire non seulement entre des logiciels différents, mais également pour un même

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

logiciel qui peut modifier à chaud la manière de gérer la mémoire en fonction du contexte.

La deuxième infrastructure décrite dans ce chapitre est MERCURY. Elle concerne cette fois la phase de développement et plus particulièrement la mise au point. En effet, MERCURY est une solution pour le débogage à distance de logiciels pendant leur exécution sur les robots. Cette solution permet aux développeurs d'être plus ou moins intrusifs dans l'analyse de l'exécution des programmes. Ils peuvent suivre l'exécution pas à pas ou en raisonnant au niveau activité d'un objet pris isolément, ou encore laisser le logiciel fonctionner en minimisant les perturbations. Dans ce dernier cas, l'infrastructure de débogage est déchargée de la mémoire du robot. Seule subsiste un noyau minimal qui permet de capturer le contexte des éventuelles exceptions et recharger l'infrastructure pour analyser les éventuels problèmes.

5.4 Références bibliographiques du chapitre

[ABF05] Alex Abacus, Mike Barker, and Paul Freedman. Using test-driven software development tools. *IEEE Software*, 22(2) :88–91, 2005.

[BM08] Michael D. Bond and Kathryn S. McKinley. Tolerating memory leaks. In Gail E. Harris, editor, *OOPSLA : Proceedings of the 23rd Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, October 19-23, 2008, Nashville, TN, USA*, pages 109–126. ACM, 2008.

[BSS+11] Eric Bodden, Andreas Sewe, Jan Sinschek, Hela Oueslati, and Mira Mezini. Taming reflection. In *International Conference on Software Engineering*, 2011.

[BU04] Gilad Bracha and David Ungar. Mirrors : design principles for meta-level facilities of object-oriented programming languages. In *Proceedings of the International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'04), ACM SIGPLAN Notices*, pages 331–344, New York, NY, USA, 2004. ACM Press.

[CGV10] Alexandre Courbot, Gilles Grimaud, and Jean-Jacques Vandewalle. Efficient off-board deployment and customization of virtual machine-based embedded systems. *ACM Transaction on Embedded Computer Systems*, 9 :21 :1–21 :53, mar 2010.

[Che88] David Cheriton. The v distributed system. *Commun. ACM*, 31(3) :314–333, March 1988.

[DMPDA12] Martin Dias, Mariano Martinez Peck, Stéphane Ducasse, and Gabriela Arévalo. Fuel : A fast general purpose object graph serializer. *Software : Practice and Experience*, 2012.

[EKO95] Dawson R. Engler, M. Frans Kaashoek, and James O'Tool. Exokernel : An operating system architecture for application-level resource management. In *SOSP*, pages 251–266, 1995.

[ELD+12] Gilberto Echeverria, Séverin Lemaignan, Arnaud Degroote, Simon Lacroix, Michael Karg, Pierrick Koch, Charles Lesire, and Serge Stinckwich. Simulating complex robotic scenarios with morse. In *SIMPAR*, pages 197–208, 2012.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[GABD10] Guillaume Grondin, George Aouad, Noury Bouraqadi, and Denis Damidot. Odice : an open distributed platform for computational simulation applied to concrete, concrete modeling. In *Proceedings of ConMod 2010 Symposium on Concrete Modelling*, Lausanne, Switzerland, June 2010. RILEM Publications.

[Gro08] Guillaume Grondin. *Un modèle d'agents auto-adaptables à base de composants*. PhD thesis, Ecole des Mines de Saint Etienne, November 2008. Thèse co-encadrée avec Laurent Vercouter (Ecole des Mines de St Etienne) sous la direction d'Olivier Boissier (Ecole des Mines de St Etienne).

[HFB05] Matthew Hertz, Yi Feng, and Emery D. Berger. Garbage collection without paging. In *Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation*, PLDI '05, pages 143–153, New York, NY, USA, 2005. ACM.

[JDI] Java debug interface (jdi). http ://java.sun.com/j2se/1.4.2/docs/jguide/jpda/-jarchitecture.html.

[JPD] Sun microsystems, java platform debugger architecture. http ://-java.sun.com/products/jpda/.

[Kae86] Ted Kaehler. Virtual memory on a narrow machine for an object-oriented language. *Proceedings OOPSLA '86, ACM SIGPLAN Notices*, 21(11) :87–106, November 1986.

[KH04] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *International Conference on Intelligent Robots and Systems*, pages 2149–2154, Sendai, Japan, Sep 2004.

[KLVA93] Keith Krueger, David Loftesness, Amin Vahdat, and Thomas Anderson. Tools for the development of application-specific virtual memory management. In *Proceedings OOPSLA '93, ACM SIGPLAN Notices*, volume 28, pages 48–64, October 1993.

[LP90] Wilf R. LaLonde and John R. Pugh. *Inside Smalltalk : vol. 1*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990.

[Mic04] Olivier Michel. Webots : Professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1) :39–42, 2004.

[Mic12a] Microsoft. Debugger security, visual studio 2012. http ://msdn.microsoft.com/en-us/library/vstudio/ms242231.aspx, 2012.

[Mic12b] Microsoft. How to : Set up remote debugging, visual studio 2012. http ://msdn.microsoft.com/en-us/library/bt727f1t.aspx, 2012.

[Mic12c] Microsoft. Supported code changes (c#), visual studio 2012. http ://msdn.microsoft.com/en-us/library/ms164927.aspx, 2012.

[MP12] Mariano Martinez-Peck. *Application-Level Virtual Memory for Object-Oriented Systems*. PhD thesis, Université de Lille, October 2012. Thèse co-encadrée avec Luc Fabresse (Ecole des Mines de Douai) et Marcus Denker (INRIA Lille Nord Europe) sous la direction de Stéphane Ducasse (INRIA Lille Nord Europe).

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[MPBD+13] Mariano Martinez-Peck, Noury Bouraqadi, Marcus Denker, Stéphane Ducasse, and Luc Fabresse. Object-based virtual memory brought to the application level. *Journal Of Object Technology*, 12(1), jan 2013.

[MPBD+ed] Mariano Martinez-Peck, Noury Bouraqadi, Stéphane Ducasse, Luc Fabresse, and Marcus Denker. Ghost : A uniform and lightweight proxy implementation. Submitted.

[NDPB09] Oscar Nierstrasz, Stephane Ducasse, Damien Pollet, and Andrew P. Black. *Squeak by Example*. Square Bracket Associates, 2009.

[Pap13] Nick Papoulias. Refléxion et débogage à distance d'applications contraintes en ressources. Thèse co-encadrée avec Luc Fabresse (Ecole des Mines de Douai) et Marcus Denker (INRIA Lille Nord Europe) sous la direction de Stéphane Ducasse (INRIA Lille Nord Europe), décembre 2013.

[Pol15] Guillermo Polito. Isolation et modularisation de systèmes réflexifs. Thèse co-encadrée avec Luc Fabresse (Ecole des Mines de Douai) et Marcus Denker (INRIA Lille Nord Europe) sous la direction de Stéphane Ducasse (INRIA Lille Nord Europe), 2015.

[RBN12] Jorge Ressia, Alexandre Bergel, and Oscar Nierstrasz. Object-centric debugging. In *Proceeding of the 34rd international conference on Software engineering*, ICSE '12, 2012.

[Res12] Jorge Ressia. *Object-Centric Reflection*. PhD thesis, Institut fur Informatik und angewandte Mathematik, 2012.

[Riv96] Fred Rivard. Smalltalk : a reflective language. In *Proceedings of REFLECTION '96*, pages 21–38, April 1996.

[RS03] Stan Shebs Richard Stallman, Roland Pesch. *Debugging with GDB*. Gnu Press, 2003.

[SU95] Randall B. Smith and David Ungar. Programming as an experience : The inspiration for self. In W. Olthoff, editor, *Proceedings ECOOP '95*, volume 952 of *LNCS*, pages 303–330, Aarhus, Denmark, August 1995. Springer-Verlag.

[WWS10] Thomas Würthinger, Christian Wimmer, and Lukas Stadler. Dynamic code evolution for java. In *Proceedings of the 8th International Conference on the Principles and Practice of Programming in Java*, PPPJ '10. ACM, 2010.

[YBK+06] Ting Yang, Emery D. Berger, Scott F. Kaplan, J. Eliot, and B. Moss. Cramm : Virtual memory support for garbage-collected applications. In *In USENIX Symposium on Operating Systems Design and Implementation*, pages 103–116, 2006.

[Yok92] Yasuhiko Yokote. The apertos reflective operating system : The concept and its implementation. In *Proceedings OOPSLA '92, ACM SIGPLAN Notices*, volume 27, pages 414–434, October 1992.

[Yok93] Yasuhiko Yokote. Kernel structuring for object-oriented operating systems : The apertos approach. In *Object Technologies for Advanced Software, First JSSST International Symposium*, volume 742 of *Lecture Notes in Computer Science*, pages 145–162. Springer-Verlag, November 1993.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[You87] Robert L. Young. An object-oriented framework for interactive data graphics. In *Proceedings OOPSLA '87, ACM SIGPLAN Notices*, volume 22, pages 78–90, December 1987.

[Zer12] ZeroTurnAround. What developers want : The end of application redeployes. http ://files.zeroturnaround.com/pdf/JRebelWhitePaper2012-1.pdf, 2012.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

6. Modèles de coordination décentralisée

Quel que soit le domaine d'application, une des questions centrales qu'on doit traiter dans un système multi-robots est comment coordonner efficacement les robots pendant une mission [Par03]. La distribution inhérente aux systèmes multi-robots, combinée au facteur d'échelle dans le cas de flottes de grandes taille m'ont poussé à adopter une approche de coordination décentralisée. Chaque robot doit, de manière autonome prendre des décisions localement. Le challenge dans ce cas est d'assurer la cohérence globale du système. Cette difficulté est accrue du fait des environnements changeants et de l'hétérogénéité des robots.

Dans le cadre du doctorat de Guillaume Grondin [Gro08] (soutenue en 2008), nous avons définit un modèle d'architecture de contrôle flexible qui permet aux robots de faire face aux évolutions de leur environnement. Cette thèse a été co-encadrée avec Laurent Vercouter [1] et Olivier Boissier de l'Ecole des Mines de St Etienne. Le modèle MADCAR-AGENT qui en est le résultat permet à un robot de réviser son architecture de contrôle qui est matérialisée par un assemblage de composants. Le choix d'une architecture dépend du contexte et des composants disponibles dans le robot au moment de l'adaptation. De plus, MADCAR-AGENT permet aux robots de coordonner leurs adaptations en échangeants des composants logiciels. Ces adaptations peuvent conduire l'agent à changer de modèle architectural, comme initié par Ludovic Guégan dans son master [Gué05].

Comme notre approche de la coordination de flottes de robots repose sur la communication, il est important de disposer d'une infrastructure réseau robuste. Cette question a été traitée dans la thèse de Van Tuan Le [Le10] (soutenue en 2010), co-encadrée avec Serge Stinckwich [2] (IRD + GREYC, Université de Caen) et Victor Moraru (MSI, UMMISCO), sous la direction de François Bourdon (GREYC, Université de Caen). Ce travail a été effectué dans le cadre de la collaboration avec le GREY de l'Université de Caen et MSI de l'UMMISCO (IRD + Université Pierre et Marie Curie - Paris) à Hanoï, Vietnam (voir la section D.4 page 143). La solution proposée est totalement distribuée [LBS+09]. Chaque robot planifie localement ses déplacements de manière à rester à proximité d'au moins un robot connecté au reste de la flotte. La direction à suivre doit respecter la contrainte de connectivité. La cohérence globale des déplacements des robots est assurée en résolvant l'ensemble des contraintes comme un problème de satisfaction de contraintes distribué (*DisCSP*) [DBD+09]. Cette résolution est ainsi répartie sur les différents robots constituant la flotte.

1. Actuellement à l'INSA Rouen
2. http://www.doesnotunderstand.org/

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

Toujours dans le cadre de la thèse de Van Tuan Le, nous nous sommes également intéressés au problème plus général de coopération entre robots pour réaliser des tâches arbitrairement complexes. Notre logique de flexibilité, nous a conduit à choisir des organisations émergentes [LBSM09, LBSD12]. Celles-ci sont adaptées pour des situations où le nombre et les capacités précises des robots sont inconnues des développeurs. Les développeurs spécifient les coalitions qui doivent être formées par les robots afin de réaliser chaque type de tâche. Une spécification de coalition est un ensemble de rôles complémentaires. A la découverte d'une tâche la coalition appropriée est formée. Un système d'enchère permet de déterminer les robots participants et le ou les rôles endossés par chacun d'eux.

Le thème de la coordination décentralisée a également été au cœur des 12 mois (2008-2009) de post-doc de Michaël Defoort. Ce travail a débouché sur une solution permettant à différents robots de se coordonner pour éviter les collisions [DDB09, DDB11]. La coordination se limite à un échange de la position et de la vitesse entre voisins. Chaque robot ensuite estime les positions et vitesses futures de ses voisins et planifie une trajectoire qui les évite tout en le rapprochant de sa destination. Cette trajectoire est ensuite ajustée de manière réactive sur la base de champs de potentiels, pour éviter les obstacles inconnus au moment de la planification.

Le reste de mes travaux sur la coordination de systèmes multi-robots se situe à la frontière avec mon activité sur les langages et modèles réflexifs (cf. chapitre 4 page 45). C'est notamment le cas du master de Romain Robbes [Rob03] qui a exploré l'utilisation de la programmation par aspects dans la définition des organisations [RBS04b, RBS04a]. Les aspects sont introduits sous la forme de groupes d'ordre supérieur qui modifient la sémantique des rôles dans les groupes applicatifs. Le master de Rémy Moüeza [Mou06], également sur la même frontière rejoint la thèse de Guillaume Grondin dans le sens où il a eu recourt aux composants logiciels pour définir des architectures à subsomption [Bro85] sensibles aux ressources. Ce travail a été effectué avec MALEVAST qui est notre implémentation [3] en Smalltalk du modèle de composants MALEVA [MB01a, BMP06a].

Dans le reste de ce chapitre, j'illustre les travaux que j'ai encadrés sur la coordination des systèmes multi-robots. Dans la section 6.1, je décris la partie de la thèse de Guillaume Grondin qui traite de la coordination des adaptations dans des architectures d'agents robotiques à base de composants logiciels. Puis, dans la section 6.2, je résume la partie des travaux de thèse de Van Tuan LE qui avaient pour objet le maintien d'un réseau de communication robuste dans une flotte robotique.

6.1 Architecture d'agents robotiques auto-adaptables

6.1.1 Motivation

Le besoin d'adaptation se fait sentir dans les applications où les changement de l'environnement est susceptible d'avoir de lourdes conséquences sur les performances des robots. Considérons une flotte de robots chargés de la cartographie d'une zone sinistrée, à la suite d'un accident

3. http://vst.mines-douai.fr/MalevaST

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

naturel ou industriel. Afin d'exploiter au mieux le nombre de robots pour accélérer la cartographie, ceux-ci doivent collaborer. Ainsi, différents robots vont explorer différentes partie du terrain.

Supposons que les informations recueillies doivent être transmises au fur et à mesure de la cartographie à un centre de commandement des secours. Les communications se faisant par ondes radio, les robots doivent former un réseau *ad hoc* pour acheminer les messages.

Chaque robot doit donc assurer deux objectifs qui peuvent être contradictoires. D'une part, il doit s'éloigner des autres robots pour explorer des zones différentes et minimiser les recouvrements. D'autre part, il doit rester suffisamment près des autres robots pour maintenir le réseau connecté. Cette tension combinée aux aléas du terrain peut conduire à une déconnexion. Le comportement d'un robot doit donc s'adapter suivant différent paramètres :

- Proximité des autres robots ;
- Densité de robots dans le voisinage ;
- Existence de chemin de communication avec le centre de commandement des secours.

6.1.2 L'existant

L'état de l'art nous a montré l'existence de deux principales familles d'architectures d'agents adaptables :

Les architectures à couches : Ces architectures sont organisées en couches hiérarchiques. Les couches sont monolithiques. Mais, chacune d'elle est modifiable via des points d'entrées qui permettent à la couche supérieure d'intervenir.

Les architectures à composants : Ces architectures sont plus modulaires que les précédentes dans la mesure où elles décrivent des assemblages de composants logiciels. Cette modularité offre la possibilité de réaliser des adaptations à une granularité plus fine.

Ce premier découpage laisse entrevoir que le recours aux composants logiciels est plus approprié pour réaliser des agents flexibles. Afin de confirmer cette intuition et comparer plus finement les différents travaux, nous avons recours aux 6 critères suivants :

Initiateur (qui ?) : Est-ce que l'adaptation est déclenchée par l'agent (+++) ou non (+) ? Si oui, quelle partie de l'agent est responsable de ce déclenchement ?

Moment (quand ?) : Quand est-ce qu'une adaptation peut être déclenchée ? L'adaptation est-elle dynamique (+++) ou non (+) ?

Portée (quoi ?) : Est-ce que l'adaptation porte sur une partie du comportement (+), sur le comportement dans son ensemble (++) ou sur le comportement et l'architecture de l'agent (+++) ?

Mise en œuvre (comment ?) : Par quel mécanisme les adaptations sont-elles réalisées ? Est-ce que les mécanismes utilisés sont complètement *ad hoc* (+) ou plutôt élaborés (++) ? La réalisation des adaptations est-elle contrôlée (+++), *i.e.* conforme à la volonté du concepteur de l'agent ?

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

Reconfigurabilité (méta-adaptation ?) : Est-ce que le modèle offre un cadre pour reconfigurer le mécanisme d'adaptation lui-même (+++) ? Ou bien permet-il seulement de réaliser de telles reconfigurations (++) ou non (+) ?

Coordination (avec qui ?) : Y a-t-il un support pour la coordination des adaptations réalisés par différents agents (+++) ou non (+) ?

	Initiateur	Moment	Portée	Mise en œuvre	Reconfigurabilité	Coordination
Architectures à couches						
TOURINGMACHINE	+++	+++	+	+	+	+
INTERRAP	+++	+++	+	+	+	+
META-CONTROL	+++	+++	+	+	+	+
Architectures à base de composants						
MAST	+	+++	+++	++	+	+
MAGIQUE	+++	+++	+++	++	+	+
MALEVA	+++	+++	+++	++	+	+
BOND	+++	+++	+++	++	++	++
JAVACT$^{\delta}$	+++	++	++	+++	++	+

TABLE 6.1 – Comparatif d'architecture d'agents adaptables

La table 6.1 résume notre évaluation de quelques architectures existantes sur la base des précédents critères. A l'exception de MAST [Ver04] et JAVACT$^{\delta}$ [LA07] toutes les architectures étudiées permettent de développer des agents capables de s'auto-adapter à l'exécution. MAST permet l'adaptation à l'exécution, mais c'est à l'initiative d'un développeur qui ajoute ou retire un composant. La révision de l'assemblage (connexions/déconnexions) est quant à elle faite par l'agent lui même. A l'opposé JAVACT$^{\delta}$ permet de développer des agents capables d'initier leurs adaptations. Il offre la possibilité de changer dynamiquement de comportement. Cependant, certains composants clés, tels que ceux chargés de la communication sont figés à la conception et ne peuvent être remplacés dynamiquement.

La portée des adaptations montre clairement le bénéfice des composants. En effet, les architectures à couches tels que TOURING MACHINE [Fer92b, Fer92a], INTERRAP [Mül96], ou encore META-CONTROL [RW91] ne permettent de modifier que les comportements des agents. Leurs structures sont quant à elles fixes du fait de l'aspect monolithique des couches. Notons que JAVACT$^{\delta}$ est légèrement en retrait par rapport aux autres architectures à base de composants. En effet, un agent JAVACT$^{\delta}$ est un acteur [Hew77] contrôlé par un méta-niveau à base de composants. Le niveau de base est constitué de comportements monolithiques, dont l'adaptation se résume à un ajout ou une suppression.

Un second bénéfice du recours aux composants concerne la facilité de mise en œuvre des adaptations. Les approches basées sur les composants présentent l'avantage de reposer sur les mécanismes génériques de connexion/déconnexion ainsi que le paramétrage des attributs des composants. Sur ce plan JAVACT$^{\delta}$, BOND [BM00, BM04] et MALEVA [MB01b, BMP06b] se démarquent par le fait qu'ils offrent certaines garanties sur la cohérence de l'architecture résultant des adaptations. JAVACT$^{\delta}$ dispose de mécanismes de vérification statique. Quant à BOND et MALEVA, chaque adaptation est définie par les développeurs sous forme d'un ensemble d'opérations à réaliser. Ainsi, la cohérence des adaptations est à la charge du développeur. Néanmoins, aucun cadre n'est fourni pour aider à la description de ces adaptations et à en vérifier la cohérence.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

En termes de reconfigurabilité, seuls BOND et JAVACT$^{\delta}$ permettent de modifier le mécanisme d'adaptation. Cependant, ils n'offrent pas de cadre particulier pour assister les développeurs. BOND ne fait pas de distinction entre l'adaptation du comportement applicatif de l'agent et la reconfiguration de son mécanisme d'adaptation. Quant à JAVACT$^{\delta}$, le seul support à la reconfiguration est la possibilité de remplacer le composant chargé de la politique d'adaptation.

Enfin, aucune des différentes architectures étudiées n'offre de support pour coordonner des adaptations réalisées par différents agents d'un même système. La seule exception est BOND qui permet de fusionner deux agents, pour les remplacer par un troisième. Mais, il ne dispose pas de mécanismes pour coordonner les adaptations de différents agents, tout en préservant leurs identités.

6.1.3 Nos travaux

Ce travail essentiellement porté par la thèse de Guillaume Grondin [Gro08] propose l'architecture auto-adaptable MADCAR-AGENT [GBV09, GBV06]. Cette architecture repose sur nos travaux en matière de *composants logiciels* présentés dans le chapitre 4. Elle a été validée par une simulation de sauvetage robotique. Ainsi, nous avons montré comment un agent robotique peut réviser son propre assemblage afin d'assurer au mieux la réalisation de la mission qui lui incombe. La dimension système multi-agents est exploitée pour mutualiser les composants qui sont échangés en fonction des besoins.

L'architecture MADCAR-AGENT

L'architecture MADCAR-AGENT peut être décomposée en trois niveaux comme le montre la figure 6.1. Le *niveau infrastructure* représente le corps du robot. Il s'agit des éléments physiques et des interfaces logicielles qui permettent d'y accéder. Ce niveau donne ainsi accès aux différents équipements embarqués sur le robot, à savoir les interfaces de communication, les capteurs et les effecteurs.

Le *niveau de base* correspond à la partie de l'architecture chargée du comportement applicatif du robot. Il est constitué d'un jeu de composants, dont un sous-ensemble peut former un assemblage arbitrairement complexe. Par exemple, dans le cas d'un robot de sauvetage, nous pouvons avoir des assemblages pour effectuer des tâches de haut niveau telle que la planification de la mission. D'autres assemblage permettrons la réalisation de tâches plus concrètes comme la cartographie du terrain ou la reconnaissance d'humain.

Les composants du niveau de base peuvent être directement connectés aux capteurs et effecteurs du niveau infrastructure. En revanche, les communications nécessitent de passer par le niveau méta. En effet, celui-ci utilise également l'interface de communication pour cordonner les adaptations avec les autres agents. Il est donc nécessaire de filtrer les communications pour ne transmettre au niveau de base que celles qui lui sont spécifiques.

Le *niveau méta* supervise l'activité du niveau de base et pilote les adaptations. Il gère le "stock" de composants et modifie les assemblages du niveau de base en fonction du contexte et des informations échangées avec les autres robots. A cet effet, il est constitué de quatre briques, dont deux sont le gestionnaire de contexte et le moteur d'assemblage définis dans le modèle

© 2014 Tous droits réservés.
doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

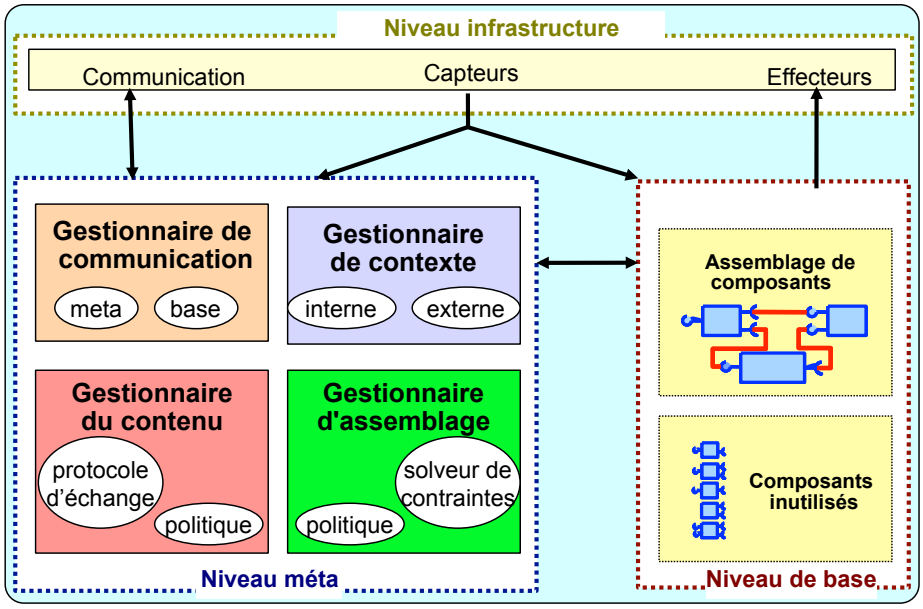

FIGURE 6.1 – MADCAR-AGENT : Architecture d'agents robotiques auto-adaptables

MADCAR (voir 4.2, page 51). Les deux briques restantes sont le gestionnaire de communication et le gestionnaire de contenu.

Le gestionnaire de communication est responsable de l'acheminement des messages. En effet, les agents peuvent communiquer pour réaliser une tâche applicative, ou bien pour leurs besoins d'adaptation. Il faut donc distinguer les messages applicatifs, échangés par les niveaux de base des agents, et les messages liés aux adaptations, échangés par les niveaux méta. Le gestionnaire de contenu effectue l'aiguillage des messages vers le bon niveau. Par ailleurs, il bloque les messages destinés au niveau de base pendant les adaptations.

Coordination par change de composants

Le modèle MADCAR-AGENT distingue le niveau applicatif (ou niveau de base) du niveau méta. En conséquence, la société d'agents dispose de deux niveaux organisationnels entre agents. Les comportements définis dans le niveau de base peuvent régir les interactions et les communications entre les agents sur un plan applicatif. MADCAR-AGENT laisse les développeurs décider de ces règles, puisqu'elles sont spécifiques à l'application. En revanche, il fournit un cadre pour les interactions entre les niveaux méta des agents. Ce cadre permet aux agents de se coordonner par échange de composants.

Lors des adaptations, le moteur d'assemblage détermine les configurations les plus pertinentes mais pour lesquelles il ne dispose pas de tous les composants requis. Par ailleurs, il maintien une table avec la fréquence d'utilisation des composants et des configurations. Le gestionnaire du contenu de l'agent exploite ces deux ensembles d'informations. Guidé par la politique de gestion de contenu fournie par les développeurs, il décide des composants à supprimer pour

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

libérer la mémoire et ceux à obtenir auprès d'autres agents.

La politique de gestion de contenu fourni différentes constantes et fonctions qui pilotent l'activité du gestionnaire de contenu. Certaines concernent la suppression de composants comme par exemple : l'utilité des composants, la valeur seuil de cette utilité au delà de laquelle la suppression est interdite, le pourcentage de l'occupation de la mémoire au delà duquel les composants les moins utiles sont supprimés. D'autres concernent l'ajout de composants comme par exemple : déterminer les rôles les plus prioritaires pour lesquels il faut trouver des composants.

Tout comme pour la politique d'adaptation, nous avons choisi de représenter la politique de gestion de contenu à l'aide de fonctions. Cette approche présente l'avantage d'offrir plus de souplesse aux développeurs pour définir des règles qui varient dans le temps avec le contexte.

La politique de gestion de contenu permet également de définir le moment de se coordonner avec les autres agents et échanger des composants. Cela permet de gérer la consommation des ressources (notamment CPU et réseau) afin gérer le contenu sans perturber le fonctionnement ni du moteur d'adaptation ni du niveau de base de l'agent. Cela influe sur le déclenchement des ajouts et suppressions des composants, mais également sur la réponse aux sollicitations des autres composants. Un agent qui est sollicité pour fournir des composants peut rejeter la demande s'il ne dispose pas de ressources pour gérer la demande. En effet, répondre à une demande de composants passe par la résolution d'un problème de satisfaction de contraintes. Le demandeur transmet un rôle qui est utilisé par l'agent fournisseur afin de trouver un composant adéquat. Les contraintes correspondent aux contrats définis par le rôle (voir section 4.2.3 page 56).

6.2 Collaboration pour le maintien de la connectivité d'un système multi-robots

6.2.1 Motivation

Nous avons vu dans la section 6.1 (page 82) que les robots ont besoin de communiquer afin de coordonner leurs adaptations. En fait, la communication est une condition *préalable* à bon nombre d'algorithme de coordination. La présence d'un canal de communication sûr et de débit suffisant est même indispensable pour mettre en place un mécanisme de coordination sophistiqué et efficace.

La communication entre les robots dans une équipe peut être réalisée implicitement ou explicitement. La communication implicite, typiquement via l'environnement, est généralement accomplie par les actionneurs et les capteurs des robots. Cela limite à la fois la quantité de données transmises et le degré d'abstraction des informations ainsi échangées. Par conséquent, la communication implicite ne convient pas pour des mécanismes sophistiqués de coordination. Il faut recourir à la communication explicite. D'autre part, avec l'avancement des technologies de communication, les robots d'aujourd'hui sont équipés d'interfaces de communication sans fil haut-débit qui leur fournissent un moyen de communication relativement sûr.

Afin d'atteindre un degré élevé de flexibilité et d'autonomie, la solution de communication doit permettre aux robots de s'organiser automatiquement en réseau, sans aucune administration centralisée. En plus de l'auto-configuration du réseau, la solution de communication doit être

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

capable de s'adapter aux mobilités des robots pendant leur mission. C'est notamment le cas dans les applications de sauvetage robotique, où on ne peut pas raisonnablement disposer d'une infrastructure de communication.

Une telle solution doit donc assurer le *maintien de la connectivitié* du réseau de robots. Elle permet donc de :

1. vérifier l'ensemble des robots forment bien un réseau ;
2. planifier les déplacements de manière à éviter les déconnexions dans le réseau.

6.2.2 L'existant

Travaux sur les réseaux *ad hoc* mobiles (MANET)

Dans le domaine des réseaux mobiles, un réseau sans infrastructure dont les nœuds peuvent se déplacer est connu sous le nom réseau *ad hoc* mobile (MANET – *Mobile Ad hoc NETwork*) [Per01]. Chaque nœud joue également le rôle de routeur et fait suivre les communications depuis et vers ses voisins.

Il existe différents protocoles de routage qui correspondent à différentes stratégies pour choisir les nœuds routeurs et la manière de relayer les communications. Les auteurs dans [AWD03, RT99] fournissent une taxonomie des principaux protocoles. En résumé, nous pouvons dire qu'un protocole de routage dans un MANET est en charge de trouver, de manière réactive (lorsque l'acheminement des données est nécessaire) ou pro-active (sans d'avoir besoin de relayer des données), une route pour acheminer les données d'un nœud source vers un nœud cible. Dans certains protocoles dits *non-structurés* tous les nœuds jouent le même rôle, alors que dans les protocoles *structurés* différents nœuds peuvent assurer des fonctions différentes pour l'acheminement des données. C'est le cas des protocoles *hiérarchiques* où certains nœuds jouent le rôle de maîtres qui gèrent et mettent à jour un cluster auquel appartiennent d'autres nœuds. Cependant, tous ces protocoles n'ont aucun contrôle sur la mobilité des nœuds. Ils n'offrent donc aucune solution pour maintenir la connectivité d'un réseau en influençant les déplacements des nœuds.

Travaux sur les réseaux de capteurs fixes

Dans le contexte des réseaux de capteurs, il est nécessaire d'économiser le maximum de l'énergie pour prolonger la durée d'exploitation des capteurs. A cet effet, la topologie de ces réseaux est contrôlée en allumant certains nœuds et en éteignant d'autres, le tout sans compromettre la connectivité globale [Raj02, XHE01, KD01].

Etant donné une configuration initiale d'un réseau de capteur, le problème du contrôle de la topologie consiste à trouver l'ensemble minimal de connexions à garder. Les connexions qui n'appartiennent pas à cet ensemble peuvent être provisoirement supprimées (arrêt de certains nœuds), de sorte que la connectivité globale du réseau ne soit pas affectée. Partant de là, les algorithmes de contrôle de la topologie vérifient la connectivité d'un réseau et le cas échéant permettent d'identifier les nœuds ou les connexions critiques. Cependant, ils ne traitent pas de la question des déplacements des nœuds, étant donné que les capteurs sont fixes.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

Travaux sur les systèmes multi-robots

Nous classons les travaux qui assurent le maintien de la connectivité dans les réseaux de robots suivant les critères suivants :

- *Généralité* : Ce critère permet de distinguer les solutions *ad hoc* (—) à un domaine d'application donné de celles qui peuvent être utilisées dans différentes applications (+++).
- *Distribution* : Certaines solutions se basent sur une machine centrale qui fait tous les calculs requis pour maintenir la connectivité (—). A l'opposé, les solutions distribuent la charge de calcul sur les différents robots (+++).
- *Passage à l'échelle* : Ce critère permet d'évaluer si une la solution convient pour des réseaux de très grande taille (+++) ou bien si le modèle de communication ne convient qu'à de petit réseaux (—).
- *Robustesse*. Une solution robuste (+++) dispose d'un mécanisme qui limite les risques des déconnexions suite à des événements extérieurs. Par opposition, une solution fragile (—) conduit à la partition du réseau suite à des aléas comme la panne d'un robot.

Travaux	Généralité	Distribution	Passage à l'échelle	Robustesse
[RB07, SJK08]	—	—	—	—
[VM04, NSBJ06, SYTX06]	—	+++	—	—
[ZP05, SS08]	+++	—	—	—
[GJ06, ZP07, ZP08, MZKP09, Sch09]	+++	+++	—	—
[VD06, Sch09]	+++	+++	+++	—

TABLE 6.2 – Travaux sur le maintien de la connectivité dans les systèmes multi-robots.

Le tableau 6.2.2 résume notre analyse de l'état de l'art sur la base de nos 4 critères. Certains travaux assurent le maintien de connectivité dans des contextes applicatifs spécifiques [RB07, SJK08, VM04, NSBJ06, SYTX06]. Les solutions proposées ne sont donc pas directement transposables à d'autres types de missions robotiques. Par ailleurs, elles souffrent d'une quantité importante de communications. Dans le cas de solutions centralisées [RB07, SJK08, ZP05, SS08] il faut acheminer la position de chaque robot au serveur centralisé. Celui-ci calcule les nouvelles positions qui assurent le maintien de la connectivité et les transmet en retour à chaque robot.

Dans le cas des solutions distribuées, une majorité [VM04, NSBJ06, SYTX06, GJ06, ZP07, ZP08, MZKP09, Sch09] requière que chaque robot connaisse les positions de tous les autres robots. D'où un très grand nombre de messages échangés, ce qui limite les solutions à des réseaux de petites échelle.

Cette limitation est levée dans les travaux [VD06, Sch09] qui reposent sur la notion *d'arbre couvrant*. Un tel arbre a pour nœuds l'ensemble des robots et pour branches des connexions entre robots. L'existence d'un tel arbre prouve que le réseau de robots est connexe. Le maintien de la connectivité consiste à maintenir les branches de l'arbres. Comme chaque branche correspond à une connexion entre deux nœuds, le maintien d'une branche ne requiert que la communication entre deux robots seulement.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

Enfin, aucun des travaux n'offre de solution robuste face aux aléas du terrain. La perte d'un robot que ce soit suite à une panne ou à un glissement de terrain par exemple peut conduire à la cission du réseau.

6.2.3 Nos travaux

La proposition que nous avons faite dans le cadre de la thèse de VAN TUAN LE [LBS+09, LBSM09, LBS+10] vise à satisfaire le besoin de connectivité pour toute application (généralité), de manière distribuée, pour des réseaux de grande taille et de manière robuste. Le point de départ est que chaque robot doit avoir un niveau de connaissance partiel, mais suffisamment riche de la connectivité du réseau.

Le système multi-robots forme un réseau *ad hoc* que nous supposons initialement connecté. La connaissance de la connectivité s'exprime en termes de chemins (*routes*) de communication vers des robots références. Un robot référence est un robot qui a typiquement un grand nombre de voisins, mais ce choix peut être guidé par les besoins applicatifs. Tant que tous les robots du réseau maintiennent au moins un chemin vers le robot de référence, le réseau reste connexe.

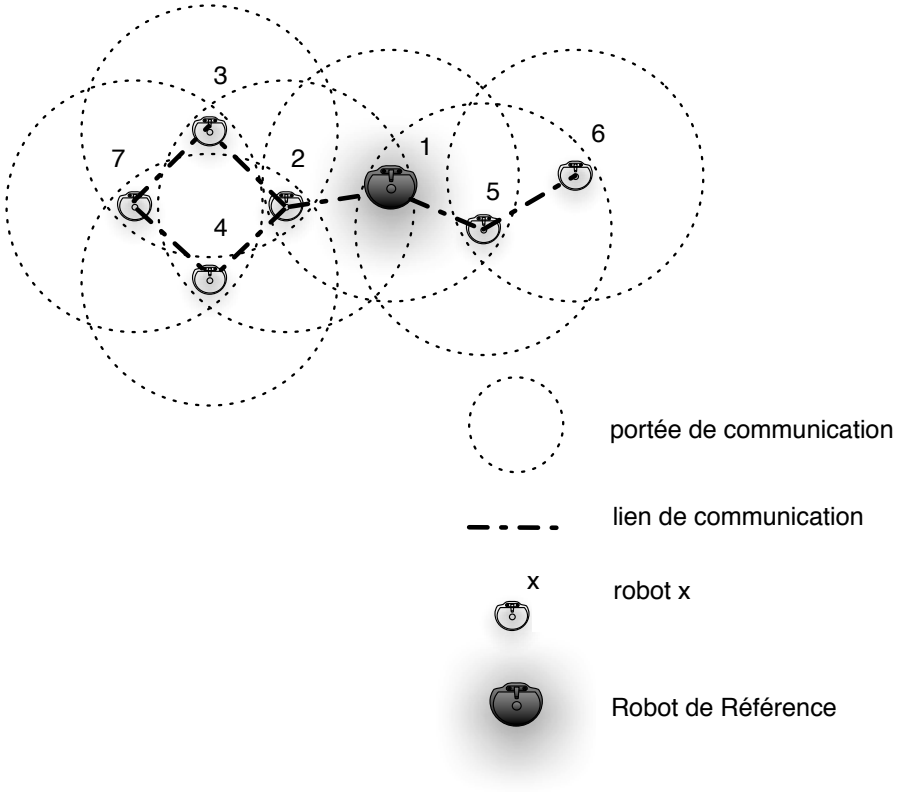

FIGURE 6.2 – Exemple de robots en réseau où le robot 1 est le noeud de référence

L'acquisition par un robot de la connaissance de la connectivité et sa mise à jour est effectuée

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

Robot	Chemins mémorisés
R_1	—
R_2	$\{R_1\}$
R_3	$\{R_2, R_1\}$
R_4	$\{R_2, R_1\}$
R_5	$\{R_1\}$
R_6	$\{R_5, R_1\}$
R_7	$\{R_3, R_2, R_1\}$, $\{R_4, R_2, R_1\}$

TABLE 6.3 – Chemins vers le nœud de référence mémorisés par les robots de la figure 6.2

de manière totalement distribuée. Prenons l'exemple de la figure 6.2. Nous supposons qu'il y a eu des enchères à l'issue desquelles, le robot R_1 est désigné comme référence. A partir de ce moment, R_1 *broadcast* de manière régulière un chemin comportant uniquement son identifiant $\{R_1\}$. Les robots voisins R_2 et R_5 qui reçoivent un tel message, mémorisent le chemin $\{R_1\}$ et *broadcastent* à leur tour les chemins qui donnent accès à R_1. Ainsi, R_2 va émettre la route $\{R_2, R_1\}$, tandis que R_5 va émettre la route $\{R_5, R_1\}$. Ce processus est répété de proche en proche et chaque robot va ne propager que les chemins sans cycles. Le tableau 6.3 donne les chemins mémorisés par chaque robot à l'issu de ce processus.

Ainsi, chaque robot reçoit uniquement les chemins qui lui sont accessibles pour communiquer avec le robot de référence. Il décide localement des chemins qu'il mémorise pour le *routage* des communications. Ce sont généralement les chemins disjoints [4] les plus courts. Ce filtrage des informations mémorisées et rediffusées limite la quantité de données échangées. Couplé à la distribution il permet le passage à l'échelle de notre solution.

Pendant la mission, le choix des déplacements effectués par un robot peut impliquer des communications avec ses voisins. Nous avons illustré cette idée dans une application d'exploration multi-robots où la connectivité est maintenue à l'aide d'une approche à base de satisfaction de contraintes distribuées (*distCSP*) [DBD+09].

Dans le cas général, bien qu'il y ait communication, chaque robot décide localement des déplacements qu'il doit effectuer de manière à maintenir au moins un chemin vers chaque robot de référence. Un robot peut donc décider de rompre une connexion avec un voisin, sachant qu'il dispose de connexions alternatives. Ainsi, la structure du réseau peut évoluer au fur et à mesure de l'avancement de la mission. En plus des déconnexions, de nouvelles connexions peuvent être établies. Ces différents changements sont notifiés par les robots qui les constatent afin que leurs voisins mettent à jour leurs connaissance de la connectivité du réseau.

Cette mise à jour des connaissances de la connectivité contribue à la robustesse de notre solution. Les déconnexions inattendues sont gérées comme celles prévisibles suite aux déplacements. Les robots qui les constatent signalent les changements. Un autre élément de robustesse est la possibilité d'avoir deux références ou même plus, dans une même flotte. Comme chaque robot maintient au moins un chemin vers chaque robot de référence, la rupture intempestive d'un

4. pas de nœud intermédiaire en commun

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

chemin vers un robot de référence peut souvent être compensée en passant par un autre robot de référence.

Dans des cas plus rares, comme par exemple l'effondrement d'un bâtiment dans lequel évoluent les robots, le réseau peut être fractionné en deux ou plusieurs sous-réseaux déconnecté. Dans le cas d'un sous-réseau sans robot de référence, la procédure du choix de robots de référence est réitérée. Et dans tous les cas, la rencontre de deux sous-réseaux conduit à leur fusion par le partage des connaissances des chemins respectifs.

6.3 Bilan du chapitre

Dans ce chapitre, j'ai résumé les travaux que j'ai co-encadrés sur la coordination des systèmes multi-robots. Au travers d'extraits des thèses de Guillaume Grondin et de Van Tuan LE, j'ai montré des solutions de coordinations qui permettent la construction de systèmes multi-robots flexibles.

Guillaume Grondin a notamment montré l'intérêt d'utiliser les composants logiciels et les solutions issus de nos travaux en lien avec l'utilisation de la réflexion pour la modularité (voir Chapitre 4, page 45). Le modèle MADCAR-AGENT, issu de ces travaux permet de construire des architectures d'agents adaptables, car capables de ré-assembler les composants qui les constituent en fonction du contexte. Les robots sont ainsi en mesure de réviser leurs comportements suivant l'environnement dans lequel ils sont situés. Les adaptations de robots voisins sont coordonnées par des échanges de composants.

La seconde illustration de mes travaux sur la coordination des systèmes multi-robots aborde le problème de communication. En effet, un moyen de communication directe est indispensable pour mettre en œuvre des stratégies de coordination élaborées. Cependant, une infrastructure de communication n'est pas toujours disponible comme par exemple lors de l'exploration spatiale. Les robots doivent donc former et maintenir un réseau *ad hoc* mobile, en plus de la réalisation leur mission. La construction et le maintien d'un tel réseau fût traité dans une partie de la thèse Van Tuan LE. Nous avons proposé une solution robuste qui passe à l'échelle en utilisant les problèmes satisfaction de contraintes distribuées. Cette solution est en plus flexible, dans le sens où elle est indépendante des caractéristiques des robots, de leur nombre ou même de la mission qu'ils doivent réaliser.

6.4 Références bibliographiques du chapitre

[AWD03] Mehran Abolhasan, Tadeuz Wysocki, and Eryk Dutkiewicz. A review of routing protocols for mobile ad hoc networks. *Ad Hoc Networks 2*, June 2003.

[BM00] Ladislau Bölöni and Dan C. Marinescu. Agent surgery : The case for mutable agents. In *Proceedings of the Third Workshop on Bio-Inspired Solutions to Parallel Processing Problems (BioSP3)*, Cancun, Mexico, May 2000.

[BM04] Ladislau Bölöni and Dan C. Marinescu. *Design of Intelligent Multi-Agent Systems*, chapter Adaptation and mutation in multi-agent systems and beyond. Springer,

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

2004.

[BMP06a] J. P. Briot, T. Meurisse, and F. Peschanski. Une expérience de conception et de composition de comportements d'agents à l'aide de composants. *L'Objet*, 11 :1–30, 2006.

[BMP06b] Jean-Pierre Briot, Thomas Meurisse, and Frédéric Peschanski. Une expérience de conception et de composition de comportements d'agents à l'aide de composants. *L'Objet*, 11(3), 2006.

[Bro85] Rodeney Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1) :14–23, March 1985.

[DBD+09] Arnaud Doniec, Noury Bouraqadi, Michael Defoort, Van Tuan Le, and Serge Stinckwich. Distributed constraint reasoning applied to multi-robot exploration. In *Proceedings of ICTAI 2009, 21st IEEE International Conference on Tools with Artificial Intelligence*, pages 159–166, 2009.

[DDB09] Michaël Defoort, Arnaud Doniec, and Noury Bouraqadi. A decentralized collision avoidance algorithm for multi-robots navigation. In *Proceedings of ICINCO (International Conference on Informatics in control, Automation and Robotics)*, Milano, Italy, July 2009.

[DDB11] Michaël Defoort, Arnaud Doniec, and Noury Bouraqadi. *Informatics in Control Automation and Robotics*, chapter Decentralized Robust Collision Avoidance Based on Receding Horizon Planning and Potential Field for Multi-Robots Systems, pages 201–215. Number 85 in LNEE. Springer, 2011.

[Fer92a] Innes A. Ferguson. Touring machines : Autonomous agents with attitudes. *Computer*, 25(5) :51–55, 1992.

[Fer92b] Innes A. Ferguson. *TouringMachines : An architecture for dynamic, rational, mobile agents*. PhD thesis, University of Cambridge, 1992.

[GBV06] Guillaume Grondin, Noury Bouraqadi, and Laurent Vercouter. Assemblage automatique de composants pour la construction d'agents avec madcar. In *Actes des journées Multi-Agent et Composant (JMAC)*, Nîmes, France, March 2006.

[GBV09] Guillaume Grondin, Noury Bouraqadi, and Laurent Vercouter. Un modèle pour le développement d'agents auto-adaptables. In *Actes des JFSMA (Journées Francophones des Systèmes Multi-Agents)*, Lyon, France, October 2009.

[GJ06] Maria Carmela De Gennaro and Ali Jadbabaie. Decentralized control of connectivity for multi-agent systems. In *Proceedings of the 45th IEEE Conference on Decision and Control*, 2006.

[Gro08] Guillaume Grondin. *Un modèle d'agents auto-adaptables à base de composants*. PhD thesis, Ecole des Mines de Saint Etienne, November 2008. Thèse co-encadrée avec Laurent Vercouter (Ecole des Mines de St Etienne) sous la direction d'Olivier Boissier (Ecole des Mines de St Etienne).

[Gué05] Ludovic Guégan. Hybridation paramètrable d'agents pour systèmes embarqués. Master's thesis, Université de Caen, 2005.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[Hew77] Carl Hewitt. Viewing control structures as patterns of passing messages. *Articial Intelligence*, 8(3) :323–364, 1977.

[KD01] S. Agarwal S. Krishnamurthy R. Katz and S. Dao. Distributed power control in ad-hoc wireless networks. In *Proceedings of PIMRC*, 2001.

[LA07] Sébastien Leriche and Jean-Paul Arcangeli. Adaptive Autonomous Agent Models for Open Distributed Systems. In *International Multi-Conference on Computing in the Global Information Technology (ICCGI), Guadeloupe, 04/03/2007-09/03/2007*, pages 19–24, http ://www.computer.org, march 2007. IEEE Computer Society.

[LBS+09] Van Tuan Le, Noury Bouraqadi, Serge Stinckwich, Victor Moraru, and Arnaud Doniec. Making networked robot connectivity-aware. In *Proceedings of ICRA (International Conference on Robotics and Automation)*, Kobe, Japan, May 2009.

[LBS+10] Van Tuan Le, Noury Bouraqadi, Serge Stinckwich, Victor Moraru, and Arnaud Doniec. Maintaining connectivity in multi-robot systems through connectivity awareness. In *Proceedings of the 5th National Conference on"Control Architecture of Robots" (CAR)*, Douai, France, May 2010.

[LBSD12] V. T. Le, N. Bouraqadi, S. Stinckwich, and A. Doniec. Role-based dynamic coalitions of multi-tasked rescue robots. In *Proceedings of the 9th International ISCRAM Conference*, Vancouver, Canada, apr 2012.

[LBSM09] Van Tuan Le, Noury Bouraqadi, Serge Stinckwich, and Victor Moraru. Connectivity awareness in networked robotic systems. In *Proceedings of IEEE-RIVF International Conference on Computing and Communication Technologies*, Da Nang City, Vietnam, July 2009.

[Le10] Van Tuan Le. *Coopération dans les systèmes multi-robots : Contribution au maintien de la connectivité et à l'allocation dynamique de rôles*. PhD thesis, Université de Caen, October 2010. Thèse co-encadrée avec Serge Stinckwich (GREYC Univ. de Caen/MSI Hanoï) et Victor Moraru (MSI Hanoï) sous la direction de François Bourdon ((GREYC Univ. de Caen).

[MB01a] Thomas Meurisse and Jean-Pierre Briot. Une approche à base de composants pour la conception d'agents. *Technique et science informatiques (TSI)*, 20(4) :583–602, 2001.

[MB01b] Thomas Meurisse and Jean-Pierre Briot. Une approche à base de composants pour la conception d'agents. *Technique et Science Informatiques (TSI)*, 20(4) :583–602, 2001.

[Mou06] Rémy Mouëza. Architecture réactive à subsomption à l'aide des composants maleva. Master's thesis, Université de Caen, June 2006.

[Mül96] J. P. Müller. *The Design of Intelligent Agents : A Layered Approach*. Number 1177 in Lecture Notes in Artificial Intelligence (LNAI). Springer-Verlag, 1996.

[MZKP09] Nathan Micheal, Michael M. Zavlanos, Vijay Kumar, and George J. Pappas. Maintaining connectivity in mobile robot networks. In *Experimental Robotics, Springer*

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

Tracts in Advanced Robotics Book Series, volume 54, pages 117–126. Springer Berlin/Heidelberg, 2009.

[NSBJ06] Giuseppe Notarstefano, Ketan Savla, Francesco Bullo, and Ali Jadbabaie. Maintaining limited-range connectivity among second-order agents. In *Proceedings of the 2006 American Control Conference*, 2006.

[Par03] Lynne E. Parker. *Multi-Robot Systems From Swarms to Intelligent Automata : Volume II*, volume II, chapter The effect of heterogeneity in teams of 100+ mobile robots, pages 205–215. Kluwer 2003, 2003.

[Per01] C. Perkins. *Ad Hoc Networking*. Addison-Wesley, 2001.

[Raj02] Rajmohan Rajaraman. Topology control and routing in ad hoc networks : a survey. *ACM SIGACT News*, 33(2) :60–73, 2002.

[RB07] Martijn N. Rooker and Andreas Birk. Multi-robot exploration under the constraints of wireless networking. *Control Engineering Practice*, 15(4) :435–445, 2007.

[RBS04a] R. Robbes, N. Bouraqadi, and S. Stinckwich. An aspect-based multi-agent system. In *Research Track of the ESUG 2004 Smalltalk Conference*, Köthen (Anhalt), Germany, September 2004.

[RBS04b] R. Robbes, N. Bouraqadi, and S. Stinckwich. Un modèle multi-agent unifiant les notions de groupe et d'aspect. In *Systèmes multi-agents défis scientifiques et nouveaux usages - Actes des JFSMA 2004*, Paris, France, November 2004.

[Rob03] Romain Robbes. Mise en oeuvre de la programmation par aspects dans le cadre des systèmes multi-agents. Master's thesis, Université de Caen, 2003.

[RT99] Elizabeth M. Royer and Chai-Keong Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, April 1999.

[RW91] S. Russell and E. Wefald. *Do the right Thing*. MIT Press, 1991.

[Sch09] Michael D. Schuresko. *Controlling Global Network Connectivity of Robot Swarms with Local Interactions*. PhD thesis, University of California, March 2009.

[SJK08] Ethan Stump, Ali Jadbabaie, and Vijay Kumar. Connectivity management in mobile robot teams. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, pages 1525–1530, May 2008.

[SS08] Kunal Srivastava and Mark W. Spong. Multi-agent coordination under connectivity constraints. In *Proceedings of the 2008 American Control Conferences*, pages 2648–2653, 2008.

[SYTX06] Weihua Sheng, Qingyan Yang, Jindong Tan, and Ning Xi. Distributed multi-robot coordination in area exploration. *Robotics and Autonomous Systems*, 54 :945–955, 2006.

[VD06] Jose Manuel Vazquez-Diosdado. *Behaviour Based Simulated Low-Cost Multi-Robot Exploration*. PhD thesis, Institute of Perception, Action and Behaviour School of Informatics University of Edinburgh, 2006.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[Ver04] L. Vercouter. MAST : Un modèle de composants pour la conception de SMA . In *JournÈes Multi-Agents et Composants (JMAC 2004)*, pages 18–31, Paris, France, Novembre 2004. École des Mines de Paris.

[VM04] Jose Vazquez and Chris Malcolm. Distributed multirobot exploration maintaining a mobile network. In *Proceedings of 2nd International IEEE Conference Intelligent Systems, 2004.*, volume 3, pages 113–118, 2004.

[XHE01] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'01)*, Rome, Italy, July 2001.

[ZP05] Michael M. Zavlanos and George J. Pappas. Controlling connectivity of dynamic graphs. In *Proceedings of 44th IEEE Conference on Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05.*, pages 6388–6393, 2005.

[ZP07] Michael M. Zavlanos and George J. Pappas. Potential fields for maintaining connectivity of mobile networks. *IEEE Transactions on Robotics*, 23(4) :812–816, August 2007.

[ZP08] Michael M. Zavlanos and George J. Pappas. Distributed connectivity control of mobile networks. *IEEE Transactions on Robotics*, 24(6) :1416–1428, Dec 2008.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

Troisième partie

Conclusion

© 2014 Tous droits réservés.

doc.univ-lille1.fr

7. Bilan et Perspectives des Travaux

- *Loi Zéro : Un robot ne peut pas faire de mal à l'humanité, ni, par son inaction, permettre que l'humanité soit blessée.*
- *Première Loi : Un robot ne peut porter atteinte à un être humain, ni, restant passif, permettre qu'un être humain soit exposé au danger, sauf contradiction avec la Loi Zéro.*
- *Deuxième Loi : Un robot doit obéir aux ordres que lui donne un être humain, sauf si de tels ordres entrent en conflit avec la Première Loi ou la Loi Zéro.*
- *Troisième Loi : Un robot doit protéger son existence tant que cette protection n'entre pas en conflit avec la Première ou la Deuxième Loi ou la Loi Zéro.*

Lois de la robotique - Isaac Asimov

La robotique, aujourd'hui balbutiante, est vouée à jouer un rôle important dans tous les secteurs de notre société. Les transports et l'aide à domicile en sont deux exemples. Les véhicules actuels vont progressivement laisser la place à des versions robotiques, totalement autonomes et capables de réduire les émissions de polluants. A la maison, les robots seront utilisés pour l'aide à domicile d'une population vieillissante. Dans ces deux exemples, les robots auront besoin de communiquer et de coordonner leurs actions. Ainsi, sur la route les voitures autonomes devront coopérer pour éviter les bouchons et mieux exploiter les infrastructures routières. Au domicile, le robot aspirateur et le robot laveur de sol devront coordonner leurs passages avec ceux d'un robot doté d'un bras manipulateur chargé du rangement et du transport du robot laveur de vitres. Ces exemples et bien d'autres révèlent le besoin de robots autonomes capables de réaliser des missions collectivement, dans des environnements variables. C'est ce besoin qui motive mes travaux sur les systèmes multi-robots flexibles.

7.1 Synthèse des contributions

Les différents travaux que j'ai co-encadrés à ce jour concourent vers un objectif commun : assister les développeurs logiciels dans les différentes phases du processus de production de systèmes multi-robots flexibles. Pour cela je me suis placé dans un cadre de pensée qui repose sur quatre piliers que sont : la programmation par composants, la programmation par aspects, les langages dynamiques et les systèmes multi-agents. Je partage ce cadre aussi bien avec les membres permanents de mon équipe, qu'avec les doctorants, post-doctorants, ingénieurs de recherche et masters. De même, les partenariats que j'ai pu développer ont permis de renforcer ce cadre avec des compétences scientifiques et techniques complémentaires.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

Un système multi-robots est d'abord un ensemble de robots ayant chacun un logiciel de contrôle. Une partie de mes contributions vise l'échelle *microscopique* de ces systèmes, où on raisonne sur un robot pris individuellement. C'est le cas dans les thèses de Houssam Fakih [Fak06], de Mariano Martinez-Peck [MP12] et de Nick Papoulias [Pap13].

Houssam Fakih a proposé de fusionner dans un même modèle de programmation les composants et les aspects. Le but était de combiner les avantages respectifs de ces deux paradigmes en termes de modularité des logiciels [FB05]. La validation a reposé sur une extension de FRACTALK qui est la projection en SMALLTALK du modèle de composants FRACTAL. Cette extension a exploité la dynamicité de SMALLTALK et plus spécifiquement ses capacités réflexives.

La réflexion est l'un des points en commun avec le travail de thèse de Nick Papoulias sur une infrastructure pour le débogage à distance. Mais, alors qu'Houssam Fakih a travaillé sur la modularité *par* la réflexion, Nick Papoulias étudie la modularité *pour* la réflexion. En effet, Nick a recours à des méta-objets *miroirs* [BU04] qu'il a étendus pour minimiser les méta-données distantes [PBD+11]. Le modèle résultat de ce travail permet de déboguer à distance les logiciels de contrôle de robots tout en minimisant les ressources mémoire du robot mobilisées pour le débogage.

La réduction de l'empreinte mémoire a été également la préoccupation centrale de la thèse de Mariano Martinez-Peck. Mariano a proposé MAREA, une infrastructure d'exécution qui constitue une mémoire virtuelle au niveau applicatif [MPBD+13]. MAREA travaille à la granularité de l'objet et permet aux développeurs de décider des objets à transférer sur disque dur et du moment d'effectuer ce transfert. Notons que cette infrastructure est transversale au code applicatif et peut être considérée à ce titre comme un aspect.

L'autre partie de mes travaux relève du niveau *macroscopique*, où on raisonne sur un système multi-robots dans sa globalité. C'est le cas de la thèse de Van Tuan Le [Le10] et du post-doc de Michaël Defoort [Def09].

Van Tuan Le s'est attaqué au problème de disponibilité d'un réseau de communication robuste, pré-requis de la majorité des solutions de coordination. Il a proposé une solution où la flotte robotique est vue comme un système multi-agents. Les robots se chargent de former un réseau *ad hoc* et d'en maintenir la connectivité, tout en réalisant les déplacements requis par leur mission [LBS+09, DBD+09].

Les déplacements des robots ont également été l'objet du post-doc de Michaël Defoort. Celui-ci a proposé une stratégie collaborative de navigation et d'évitement d'obstacles [DDB11, DDB09].

La thèse de Guillaume Grondin [Gro08] relève des deux niveaux *microscopique* et *macroscopique* à la fois. Elle montre une articulation possible entre l'architecture de contrôle d'un robot et la coordination entre les membres d'une flotte robotique. Guillaume a défini une architecture adaptable à base de composants logiciels FRACTALK pour contrôler chaque agent robotique [GBV06, GBV08, GBV09]. Les assemblages de composants sont révisés en fonction de l'évolution des ressources internes au robot ou disponibles dans son environnement. Ces adaptations sont coordonnées par les interactions entre les robots qui peuvent échanger des composants.

Les travaux résumés ci-dessus constituent une sélection représentative des différents co-encadrements que j'ai pu effectuer depuis ma prise de fonction à l'ÉCOLE DES MINES DE DOUAI en 2001. Parallèlement, j'ai mis en place la thématique de recherche Composants, Agents et

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

Robotique (CAR [1]) qui compte actuellement 4 membres permanents. Cette équipe participe actuellement à différents projets avec différents partenaires (tels que l'INRIA Lille, le LIRMM ou l'IRD Hanoï, Vietnam). Nous sommes également actifs au niveau de la communauté scientifique. Du fait de notre positionnement thématique, nous sommes présents dans 2 groupes de recherche :

- Le GdR Génie de la Programmation et du Logiciel (GPL), groupe COSMAL [2]
- Le GdR Robotique, GT CAR [3] où je co-anime les journées nationales du même nom depuis 2010.

Je participe à différents comités de programmes et de relecture internationaux comme ICRA [4], SCP [5] et SSRR [6]. J'ai également été membre du bureau de l'association ESUG [7] (2003-2010) qui anime la communauté du langage Smalltalk au niveau mondial et organise tous les ans une conférence internationale. L'équipe continue à être active au niveau de la communauté Smalltalk, que ce soit au niveau d'ESUG, qu'au niveau du récent consortium de PHARO [8]. Cette activité se traduit par la co-organisation de manifestations ainsi que par la production de logiciels sous licence libre (MIT). Le dernier en date est une implémentation en PHARO d'un client du *middleware* robotique ROS [9].

Enfin, en tant qu'enseignant-chercheur, j'ai endossé différentes responsabilités pédagogiques en plus de mon activité de recherche. Tout d'abord, j'effectue des enseignements d'informatique à différents niveaux du cycle d'ingénieur et en particulier dans la spécialité informatique. De plus, j'ai assuré pendant 4 années (2007-2011) la charge de responsable de l'enseignement informatique à l'ÉCOLE DES MINES DE DOUAI.

7.2 Pistes pour des travaux futurs

De la même manière que chaque personne dispose aujourd'hui de différents équipements électroniques (smartphone, tablette, ordinateur portable, serveur), nul doute qu'elle aura à sa disposition plusieurs robots, dans un futur assez proche. Et même si ces robots seraient destinés à des usages différents, ils auront besoin de coordonner leurs activités et éventuellement de coopérer. Ce besoin sera grandissant avec la miniaturisation et le recours à des micro-robots voire à des nano-robots. Se pose alors la question de comment développer et exploiter ces flottes de robots.

Je m'intéresse plus particulièrement aux systèmes multi-robots ouverts, où les robots peuvent se joindre à la flotte ou la quitter pendant la réalisation de la mission. C'est le cas par exemple d'un système multi-robots destiné à la recherche de survivants suite à un séisme. La mission peut

1. http://car.mines-douai.fr
2. Composants Objets Services : Modèles, Architectures et Langages, http://gdr-gpl.cnrs.fr/Groupes/COSMAL/Description
3. Control Architecture for Robots, http://www.gdr-robotique.org/groupes_de_travail/?id=4
4. International Conference on Robotics and Automation, http://www.icra2013.org/
5. Science of Computer Programmaing (journal), http://www.journals.elsevier.com/science-of-computer-programming/
6. International Symposium on Safety Security and Robotics, http://www.ssrr-conference.org/2013/
7. European Smalltalk Users Group, http://www.esug.org
8. http://consortium.pharo.org/
9. http://www.ros.org

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

commencer avec un petit ensemble de robots qui est enrichi avec l'arrivée de renforts. Au fur et à mesure de la mission, les robots aux batteries déchargées seront retirés de la flotte, puis réintroduits une fois la charge effectuée. Les éventuelles pannes et les réparations sont également des causes possibles de la variation du nombre de robots.

Une solution idéale pour de tels systèmes doit être flexible en alliant séparation des préoccupations et automatisation. La séparation des préoccupations vise à réduire la complexité du développement des systèmes multi-robots en subdivisant la tâche des développeurs en sous-tâches découplées. Ainsi, le développement d'un robot doit pouvoir être effectué en faisant abstraction des flottes cibles, c'est-à-dire sans faire d'hypothèses sur l'organisation sociale et les caractéristiques des autres robots. De manière symétrique, une mission doit pouvoir être définie sans imposer de contraintes ni sur les compétences individuelles des robots ni sur le nombre même de robots. Ainsi, un même robot pourrait être exploité dans des systèmes multi-robots distincts. Par ailleurs, une même mission devrait être réalisable par des ensembles de robots différents, qu'ils soient homogènes ou hétérogènes.

La propriété d'automatisation vise également à contribuer à la flexibilité tout en déchargeant les développeurs et les exploitants des systèmes robotiques. Ainsi, les robots doivent prendre en charge l'organisation de la flotte de la découverte des robots disponibles, jusqu'au déploiement d'une mission et la répartition des rôles, en passant par la constitution et le maintien d'un réseau. L'organisation doit être automatiquement révisée pour prendre en compte les apparitions et les disparitions de robots, même intempestives. L'automatisation doit se traduire également à l'échelle de chaque robot pris individuellement. Un robot doit s'auto-adapter et ajuster son comportement en fonction du contexte pour prendre en compte par exemple les ressources disponibles, ainsi que les capacités et les comportement des autres robots.

Les travaux que nous avons menés à ce jour constituent une étape vers l'objectif ambitieux de définir une chaîne complète de modèles et d'infrastructures pour des systèmes multi-robots flexibles. Les travaux futurs que j'envisage peuvent se décliner suivant les directions complémentaires suivantes :

- **Architectures** logicielles hybrides, sociales et adaptables ;
- **Langages** de programmation pour la robotique ;
- **Infrastructures et stratégies** pour la gestion des ressources ;
- **Démarches et outils** pour le test d'applications robotiques.

7.2.1 Architectures logicielles de contrôle des robots

Parmi les architectures logicielles utilisées en robotique [Par08, Ing03, Boi01], nous nous intéressons à celles dites *hybrides* résumées par l'expression "*Think and Act Concurrently*" [MM08]. Elles sont très prisées par la communauté [ACF+98, VNE+01, Alb95, AB04] car elles permettent de réaliser des agents robotiques dotés de capacités cognitives pour traiter des tâches complexes, tout en prenant en compte les aléas d'un environnement dynamique nécessitant une grande réactivité, comme par exemple pour l'évitement d'obstacles mobiles.

Dans le cadre de systèmes multi-robots, les logiciels de contrôle des robots doivent également intégrer une dimension sociale afin de coordonner leur activité. Cette capacité doit également être

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

complétée par la capacité d'adaptation en fonction des évolutions de l'environnement, de la flotte robotique et des ressources embarquées ou accessibles par le robot.

Le modèle MADCAR issu de la thèse de Guillaume Grondin [Gro08] constitue un premier pas dans cette direction. Il permet d'adapter dynamiquement l'architecture d'un robot. Cependant, nous nous sommes restreints à des architectures réactives. De plus la coordination des adaptations est indirecte, puisqu'elle se limite à un échange de composants.

Une perspective immédiate de ce travail serait de généraliser l'approche à des architectures qui incluent des capacités de planification et de coordination. Une autre perspective concerne la cohérence globale des adaptations d'un système multi-robots. Elle consiste à étendre MADCAR pour supporter des stratégies de coordination des adaptations plus directes. Ces deux pistes, doivent converger *in fine* pour déboucher sur une architecture hybride, sociale et adaptable [BS11]. La dimension sociale de cette architecture serait basée sur notre solution d'allocation dynamique de rôles [LBSD12] issue de la thèse de Van Tuan Le [Le10].

7.2.2 Langages de programmation pour la robotique

Dans nos travaux sur les composants logiciels [Gro08] et leur intégration avec la programmation par aspects [Fak06], nous avons initialement adopté l'approche largement répandue de représenter chaque composant à l'aide de plusieurs objets. Par exemple, chaque port d'un composant est matérialisé à l'exécution par un objet. Cette approche présente l'inconvénient de créer un fossé entre la conception du programme et le reste du cycle de vie. La correspondance entre les objets et les composants ou les aspects conduit à la surcharge intellectuelle des développeurs, ce qui engendre des erreurs pendant le codage et rend le débogage difficile. Pour palier ce défaut il est nécessaire de recourir à un langage de programmation qui offre des constructions équivalentes à celles manipulées à la conception, comme c'est le cas avec le langage à composants SCL [FBDH12].

Une autre piste à poursuivre est d'intégrer dans un langage à composants des concepts clés propres aux applications robotiques. Expliciter des abstractions des logiciels de contrôle des robots est un des thèmes de prédilection des *workshops* SDIR [10] et DSLROB [11]. L'originalité de ma proposition est qu'elle cible un langage de programmation à base de composants qui soit également dynamique. Mon objectif est de pouvoir exprimer dans ce langage des architectures hybrides, sociales et adaptables comme décrit dans la section 7.2.1.

Une question sous-jacente à la définition d'un tel langage est le choix d'un modèle de composants approprié aux architectures ciblées. Ce modèle doit être à la fois simple à appréhender et avoir un pouvoir d'expression suffisamment élevé pour exprimer facilement toutes les facettes des architectures de contrôle des robots. Notre expérience passée [BS07, Moü06] avec le modèle de composants orienté flux MALEVA [BMP06] a montré qu'il permet d'exprimer aussi bien une architecture réactive à subsomption [Bro85] qu'une architecture à anticipation [Dav96]. Les travaux futurs dans ce sens devront comparer MALEVA à d'autres modèles de composants logiciels pour le développement d'architectures représentatives de différentes familles.

10. Software Development and Integration in Robotics, http://robotics.unibg.it/tcsoft/sdir2013/
11. Domaine Specific Languages for ROBotics, http://www.doesnotunderstand.org/public/DSLRob2013

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

En complément à l'introduction d'un langage à composants, il est indispensable de prendre en compte la base de code existante. Une approche consiste à fournir des outils de transformation des bibliothèques de classes existantes. C'est la solution promue par FRACTAL qui consiste à construire un type de composants capsules (*wrapper*) pour chaque classe d'objets qui doivent être utilisés dans un assemblage de composants. Une approche alternative que je propose de poursuivre consiste à utiliser la réflexion pour introduire les constructions d'un langage à composants dans un langage à objets. Cette solution dont nous avons montré la faisabilité [BF09] repose sur une extension du langage à objets hôte pour assurer la symbiose entre les objets et les composants. Suivant le contexte, une entité est vue et manipulée soit comme un objet, soit comme un composant, indépendamment de sa vraie nature.

7.2.3 Infrastructures et stratégies pour la gestion des ressources

Le développement d'applications robotiques requiert des outils pour assister les développeurs dans la gestion des ressources des robots en termes de mémoire, de puissance de calcul (CPU), de média de communication et d'énergie. Nous avons déjà fait une proposition dans ce sens dans le cadre de la thèse de Mariano Martinez-Peck [MP12]. En effet, nous avons proposé une mémoire virtuelle orientée-objet et dirigée par les applications. La thèse en cours de Guillermo Polito [Pol15] poursuit ce travail en se plaçant en amont dans le cycle de vie du logiciel. Notre but est d'intervenir avant la mise en production, afin de minimiser l'espace mémoire occupé par les logiciels de contrôle des robots. A cet effet, nous étudions une infrastructure qui permet aux développeurs d'analyser et de modifier de manière atomique le graphe d'objets pendant l'exécution des applications. La solution que nous proposons repose sur l'idée de réifier l'espace mémoire occupé par chaque application [PDFB13]. Dès lors, il va nous être possible de filtrer les objets terminaux et les réifications (classes, méthodes...) pour ne garder que ceux réellement nécessaires à l'application.

La gestion des ressources devrait également se faire de manière collaborative. C'est le cas par exemple dans la thèse de Guillaume Grondin où les composants inutilisés par un agent robotique, sont transmis aux autres membres de la flotte, avant de libérer la mémoire qu'ils occupent. C'est également le cas dans la thèse de Van Tuan Le, où les robots se répartissent la charge de routage des communications et de maintien du réseau [LBS+09]. Une perspective complémentaire à ce travail concerne la gestion combinée de la mémoire et de la puissance de calcul, en exploitant les ressources disponibles dans le voisinage réseau d'un robot. Par exemple, un robot pourra déporter pendant l'exécution une partie des traitements qu'il doit effectuer sur des serveurs distants (*cloud*), ou sur des robots voisins. Il s'agit d'un support à la mobilité forte [FPV98], où les processus sont migrés avec le code et les données. Cela nous est accessible car nous travaillons avec des langages dynamiques et réflexifs. Chaque espace mémoire tel que nous l'avons défini [PDFB13] inclut l'ensemble des réifications y compris les processus et la pile d'exécution. Par ailleurs, l'infrastructure que nous développons permet de faire cohabiter plusieurs espaces mémoires. Nous pouvons donc envisager de fragmenter un logiciel de contrôle d'un robot en différents espaces mémoires qui, suivant le contexte, cohabiteront sur un seul robot ou seraient répartis.

Une autre manière d'aborder la question des ressources est de raisonner à l'échelle d'un

© 2014 Tous droits réservés.
doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

groupe de robots et décider de l'ajout ou de la suppression de robots en fonction des besoins de la mission à accomplir. Nous avons amorcé cette perspective par une première étude de la taille optimale d'une flotte de robots homogènes pour explorer un terrain [BFD12]. Les résultats de ce travail effectué avec un simulateur discret devront être validés sur un simulateur réaliste, puis sur une flotte de robots. C'est l'objet du post-doc de Zhi Yan [Yan14] qui démarre en septembre 2013.

L'analyse à des fins d'optimisation de la taille des groupes et des flottes robotiques est un sujet qui admet nombre de perspectives. En effet, nous nous intéressons actuellement à des flottes de robots identiques qui participent tous à une unique tâche. Mais, différentes pistes d'études complémentaires sont possibles en faisant varier les paramètres suivants :

- homogénéité des robots (robots hétérogènes ou homogènes)
- nombre de tâches auxquels participe chaque robot (robots mono-tâche ou multi-tâches)
- nombre de robots requis pour chaque tâche (tâches qui requièrent un robot unique vs. plusieurs robots)

Certaines classes de problèmes ne peuvent être traitées que par certaines combinaisons. Mais, pour d'autres, différentes combinaisons peuvent répondre aux mêmes besoin. Par exemple des opérations de sauvetage peuvent être réalisées par des robots mono-tâches. Elle peuvent également être effectuées par des robots multi-tâches. L'analyse des performances peut alors être utilisée comme outil pour aider les développeurs à choisir entre différentes configurations étant donnée une mission à réaliser.

7.2.4 Démarches et outils pour le test d'applications robotiques

L'utilisation de langages dynamiques constitue un des piliers qui font l'originalité de mes travaux. Ce choix est motivé par la flexibilité de ces langages qui les rend particulièrement appropriés pour les approches de développement logiciel agiles, tels que eXtreme Programming [12] ou Scrum [13]. Elles se traduisent par des pratiques comme la programmation en binôme (*pair programming*) [14] ou développement dirigé par les tests (TDD : *Test Driven Development*) [15].

Une piste de travaux que je souhaite explorer est d'étudier l'introduction de l'agilité dans le développement d'applications robotiques. Nous avons commencé à travailler dans ce sens en proposant les bases d'une démarche de test de telles applications [LFB13]. Il s'agit de fournir un cadre qui permet le test d'un robot dans sa globalité, c'est à dire de tester l'ensemble constitué d'un logiciel de contrôle de robot et d'un corps robotique matériel. Cette solution vient compléter les pratiques actuelles de la communauté robotique que sont : les démarches formelles avec preuves [BdSIY11], les tests du logiciel seul [Big10, LSS+10], le recours aux simulateurs purs [SS12, SKPK11] ou avec quelques éléments matériels (*hardware in the loop*) [CMW11].

Dans une logique d'agilité notre démarche de test doit être complétée et validée pour faire du développement dirigé par les tests (TDD). Suivant les besoins, ces tests peuvent être unitaires ou

12. http://www.extremeprogramming.org/
13. http://www.scrumalliance.org/
14. http://www.extremeprogramming.org/rules/pair.html
15. http://www.agiledata.org/essays/tdd.html

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

fonctionnels à la *Behavior Driven Developmemnt* [16]. Dans un premier temps, je m'intéresserai à TDD pour réaliser le logiciel de contrôle d'un robot, étant donné un matériel existant. Une perspective à plus long terme dans cette direction consiste à étudier TDD dans un contexte de *co-design*, où le logiciel et le matériel sont développés en parallèle. Les développeurs itérerons entre la production de tests et la production du fragment du logiciel et éventuellement du matériel qui valide les tests. Ce travail viserait aussi bien les robots individuels que les flottes de robots.

Le développement dirigé par les tests requiert un ensemble d'outils et infrastructures pour simplifier le travail des développeurs. Le travail de thèse de Nick Papoulias [Pap13] va dans ce sens et traite du débogage. La solution proposée permet de déboguer les applications robotiques *in vivo* afin de disposer de tout le contexte du dysfonctionnement à traiter. Cet outil est une brique d'un environnement de développement complet qui doit notamment inclure une infrastructure pour les tests. Le travail que nous avons effectué sur le nouveau *framework* BoTest [FLB13] constitue une première étape dans ce sens.

Un travail conséquent reste à faire, notamment pour ce qui est de l'automatisation des tests. Cette automatisation est importante voire indispensable pour l'adoption de TDD et le recours systématique aux tests. Dans son état actuel, BoTest est semi-automatique. Il requiert l'intervention d'un opérateur humain pour les opérations physiques. Celui-ci doit préparer le robot et les objets de l'environnement de test, par exemple en les positionnant à des points particuliers. L'opérateur peut également être sollicité pour effectuer certaines mesures, comme par exemple la position finale du robot. L'automatisation de ces différentes tâches requiert le développement d'un banc de test physique, instrumenté et robotisé. Par exemple, un tel banc serait doté de bras robotisés pour réaliser différentes actions (déplacer des objets) et de lasers pour mesurer les positions.

7.3 Pour des usages éthiques des systèmes multi-robots

Comme c'est le cas pour mes travaux passés, la validation des perspectives doit se faire non-seulement sur le plan théorique, mais également au travers d'expérimentations concrètes. Il s'agit d'une part de confronter les modèles à la réalité et d'autre part de se mettre dans la peau d'un utilisateur et de toucher ainsi les limites de nos propositions. Ces expérimentations doivent se traduire chaque fois que possible par applications qui montrent des exemples d'usages des technologies manipulées. Le projet CAIRE qui démarre (voir l'annexe D.1 page 139) est une illustration récente de cet état d'esprit. En effet, sur le plan scientifique, nous traitons du test, que nous validons au travers d'une application de cartographie de rayonnement électromagnétique. Cet usage correspond à un réel besoin, identifié avec des partenaires experts du domaine et confirmé auprès d'un industriel.

Au delà de la validation expérimentale de nos résultats, les démonstrateurs que nous développons ont une fonction supplémentaire. Il s'agit de nous faire réfléchir sur les impacts sociétaux de nos travaux. Ce questionnement éthique qui doit accompagner tout travail scientifique, est à mes yeux d'autant plus critique que la robotique est hautement subversive. Elle est en train de bouleverser nos modes de vie et va probablement même refaçonner l'humanité.

16. http://dannorth.net/introducing-bdd/

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

7.4 Références bibliographiques du chapitre

[AB04] J.S. Albus and A. J. Barbera. Rcs : A cognitive architecture for intelligent multiagent systems. In *Proceedings of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles (IAV 2004)*, Lisbon, Portugal, July 2004.

[ACF+98] Rachid Alami, Raja Chatila, Sara Fleury, Malik Ghallab, and Félix Ingrand. An architecture for autonomy. *International Journal of Robotics Research. Special Issue on Integrated Architectures for Robot Control and Programming*, 5(1), March 1998.

[Alb95] J.S. Albus. The nist real-time control system (rcs) an applications survey. In *Proceedings of the AAAI 1995 Spring Symposium Series*, Stanford University, Menlo Park, CA, USA, March 1995.

[BdSIY11] Saddek Bensalem, Lavindra de Silva, Félix Ingrand, and Rongjie Yan. A verifiable and correct-by-construction controller for robot functional levels. *Journal of Software Engineering for Robotics*, 2(1) :1–19, September 2011.

[BF09] Noury Bouraqadi and Luc Fabresse. Clic : A component model symbiotic with smalltalk. In *Proceedings of the International Workshop on Smalltalk Technologies*, Brest, France, August 2009. ACM.

[BFD12] Noury Bouraqadi, Luc Fabresse, and Arnaud Doniec. On fleet size optimization for multi-robot frontier-based exploration. In *7th National Conference on"Control Architecture of Robots"*, May 2012.

[Big10] G. Biggs. Applying regression testing to software for robot hardware interaction. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4621–4626, 2010.

[BMP06] J. P. Briot, T. Meurisse, and F. Peschanski. Une expérience de conception et de composition de comportements d'agents à l'aide de composants. *L'Objet*, 11 :1–30, 2006.

[Boi01] Olivier Boissier. *Systèmes Multi-Agents*, chapter Modèles et architectures d'agents. Hermes, 2001.

[Bro85] Rodeney Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1) :14–23, March 1985.

[BS07] Noury Bouraqadi and Serge Stinckwich. Towards a generic anticipatory agent architecture for mobile robots. In Peter Sapaty and Joaquim Filipe, editors, *Proceedigs of The 3rd International Workshop on Multi-Agent Robotic Systems (MARS)*, Angers, France, May 2007.

[BS11] Noury Bouraqadi and Serge Stinckwich. The next 700 control architectures for rescue robotics. In *Proceedings of the 9th IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR) - Outrageous Visions for Computing in Rescue Robotics Track*, Kyoto, Japan, novembre 2011. IEEE.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[BU04] Gilad Bracha and David Ungar. Mirrors : design principles for meta-level facilities of object-oriented programming languages. In *Proceedings of the International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'04), ACM SIGPLAN Notices*, pages 331–344, New York, NY, USA, 2004. ACM Press.

[CMW11] Ian Yen-Hung Chen, Bruce A. MacDonald, and Burkhard C. Wünsche. A flexible mixed reality simulation framework for software development in robotics. *Journal of Software Engineering for Robotics*, 2(1) :40–54, September 2011.

[Dav96] P. Davidsson. A linearly quasi-anticipatory autonomous agent architecture : Some preliminary experiments. In *Distributed Artificial Intelligence Architecture and Modelling*, number 1087 in Lecture Notes in Artificial Intelligence, pages 189–203. Springer Verlag, 1996.

[DBD+09] Arnaud Doniec, Noury Bouraqadi, Michael Defoort, Van Tuan Le, and Serge Stinckwich. Distributed constraint reasoning applied to multi-robot exploration. In *Proceedings of ICTAI 2009, 21st IEEE International Conference on Tools with Artificial Intelligence*, pages 159–166, 2009.

[DDB09] Michaël Defoort, Arnaud Doniec, and Noury Bouraqadi. A decentralized collision avoidance algorithm for multi-robots navigation. In *Proceedings of ICINCO (International Conference on Informatics in control, Automation and Robotics)*, Milano, Italy, July 2009.

[DDB11] Michaël Defoort, Arnaud Doniec, and Noury Bouraqadi. *Informatics in Control Automation and Robotics*, chapter Decentralized Robust Collision Avoidance Based on Receding Horizon Planning and Potential Field for Multi-Robots Systems, pages 201–215. Number 85 in LNEE. Springer, 2011.

[Def09] Michaël Defoort. Post-doctorant. Navigation et évitement d'obstacles dans une flotte multi-robots (1 an), 2008-2009.

[Fak06] Houssam Fakih. *L'integration des fonctionnalités transversales dans les composants logiciels en utilisant la programmation par aspects*. PhD thesis, Université de Lille 1, December 2006. Thèse co-encadrée et dirigée par Laurence Duchien (LIFL, Université de Lille 1).

[FB05] H. Fakih and N. Bouraqadi. Les aspects et les composants logiciels : Etude de cas avec le modèle de composant fractal. *L'Objet*, 2005.

[FBDH12] Luc Fabresse, Noury Bouraqadi, Christophe Dony, and Marianne Huchard. A language to bridge the gap between component-based design and implementation. *Computer Languages, Systems and Structures*, 2012.

[FLB13] Luc Fabresse, Jannik Laval, and Noury Bouraqadi. Towards test-driven development for mobile robots. In D . Brugali, N . Hochgeschwender, and R . Philippsen, editors, *Proceedings of the ICRA 2013 Workshop on Software Development and Integration in Robotics (SDIR VIII)*, 2013.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[FPV98] Alfonso Fuggetta, Gian Pietro Picco, and Giovanni Vigna. Understanding Code Mobility. *IEEE Transactions on Software Engineering*, 24(5), May 1998.

[GBV06] Guillaume Grondin, Noury Bouraqadi, and Laurent Vercouter. Madcar : an abstract model for dynamic and automatic (re-)assembling of component-based applications. In *Proceedings of the 9th International Symposium on CBSE (Component-Based Software Engineering)*, LNCS, pages 360–367, Sweden, June 2006. Springer.

[GBV08] Guillaume Grondin, Noury Bouraqadi, and Laurent Vercouter. Component reassembling and state transfer in madcar-based self-adaptive software. In *Proceedings of the 46th International Conference TOOLS-EUROPE (Objects, Models, Components, Patterns)*, Switzerland, June 2008.

[GBV09] Guillaume Grondin, Noury Bouraqadi, and Laurent Vercouter. Un modèle pour le développement d'agents auto-adaptables. In *Actes des JFSMA (Journées Francophones des Systèmes Multi-Agents)*, Lyon, France, October 2009.

[Gro08] Guillaume Grondin. *Un modèle d'agents auto-adaptables à base de composants*. PhD thesis, Ecole des Mines de Saint Etienne, November 2008. Thèse co-encadrée avec Laurent Vercouter (Ecole des Mines de St Etienne) sous la direction d'Olivier Boissier (Ecole des Mines de St Etienne).

[Ing03] Félix Ingrand. Architectures logicielles pour la robotique autonome. In *JNRR'03 (Journées Nationales de Rechercher en Robotique)*, Murol/Clermont-Ferrand, France, October 2003.

[LBS+09] Van Tuan Le, Noury Bouraqadi, Serge Stinckwich, Victor Moraru, and Arnaud Doniec. Making networked robot connectivity-aware. In *Proceedings of ICRA (International Conference on Robotics and Automation)*, Kobe, Japan, May 2009.

[LBSD12] V. T. Le, N. Bouraqadi, S. Stinckwich, and A. Doniec. Role-based dynamic coalitions of multi-tasked rescue robots. In *Proceedings of the 9th International ISCRAM Conference*, Vancouver, Canada, apr 2012.

[Le10] Van Tuan Le. *Coopération dans les systèmes multi-robots : Contribution au maintien de la connectivité et à l'allocation dynamique de rôles*. PhD thesis, Université de Caen, October 2010. Thèse co-encadrée avec Serge Stinckwich (GREYC Univ. de Caen/MSI Hanoï) et Victor Moraru (MSI Hanoï) sous la direction de François Bourdon ((GREYC Univ. de Caen).

[LFB13] Jannik Laval, Luc Fabresse, and Noury Bouraqadi. A methodology for testing mobile autonomous robots. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[LSS+10] Jae-Hee Lim, Suk-Hoon Song, Jung-Rye Son, Tae-Yong Kuc, Hong-Seong Park, and Hong-Seak Kim. An automated test method for robot platform and its components. *International Journal of Software Engineering and Its Applications*, 4(3) :9–18, July 2010.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[MM08] Maja J. Matarić and François Michaud. *Handbook of Robotics*, chapter 38. Behavior-Based Systems, pages 891–910. Springer, 2008.

[Moü06] Rémy Moüeza. Gestion des ressources dans le contexte de l'informatique ubiquitaire. Master's thesis, Université de Caen, September 2006.

[MP12] Mariano Martinez-Peck. *Application-Level Virtual Memory for Object-Oriented Systems*. PhD thesis, Université de Lille, October 2012. Thèse co-encadrée avec Luc Fabresse (Ecole des Mines de Douai) et Marcus Denker (INRIA Lille Nord Europe) sous la direction de Stéphane Ducasse (INRIA Lille Nord Europe).

[MPBD+13] Mariano Martinez-Peck, Noury Bouraqadi, Marcus Denker, Stéphane Ducasse, and Luc Fabresse. Object-based virtual memory brought to the application level. *Journal Of Object Technology*, 12(1), jan 2013.

[Pap13] Nick Papoulias. Refléxion et débogage à distance d'applications contraintes en ressources. Thèse co-encadrée avec Luc Fabresse (Ecole des Mines de Douai) et Marcus Denker (INRIA Lille Nord Europe) sous la direction de Stéphane Ducasse (INRIA Lille Nord Europe), décembre 2013.

[Par08] Lynne E. Parker. *Handbook of Robotics*, chapter 40. Multiple Mobile Robot Systems, pages 921–941. Springer, 2008.

[PBD+11] Nick Papoulias, Noury Bouraqadi, Marcus Denker, Stéphane Ducasse, and Luc Fabresse. Towards structural decomposition of reflection with mirrors. In Loïc Lagadec, editor, *International Workshop on Smalltalk Technologies (IWST)*, Edinburgh, Scotland, UK, August 2011. ACM.

[PDFB13] Guillermo Polito, Stéphane Ducasse, Luc Fabresse, and Noury Bouraqadi. Virtual smalltalk images : Model and applications. In *Proceedings of the International Workshop on Smalltalk Technologies (IWST)*, 2013.

[Pol15] Guillermo Polito. Isolation et modularisation de systèmes réflexifs. Thèse co-encadrée avec Luc Fabresse (Ecole des Mines de Douai) et Marcus Denker (INRIA Lille Nord Europe) sous la direction de Stéphane Ducasse (INRIA Lille Nord Europe), 2015.

[SKPK11] Jung-Rye Son, Tae-Yong Kuc, Jong-Koo Park, and Hong-Seok Kim. Simulation based functional and performance evaluation of robot components and modules. In *Information Science and Applications (ICISA), 2011 International Conference on*, pages 1–7. IEEE, April 2011.

[SS12] Hong Seong and Jeong Seok. SITAF : simulation-based interface testing automation framework for robot software component. In Florian Kongoli, editor, *Automation*. InTech, July 2012.

[VNE+01] Richard Volpe, Issa Nesnas, Tara Estlin, Darren Mutz, Richard Petras, and Hari Das. The claraty architecture for robotic autonomy. In *Proceedings of the IEEE Aerospace Conference*, Big Sky, Montana, USA, March 2001.

[Yan14] Zhi Yan. Post-doctorant. Dimensionnement de flottes robotiques, 2013-2014.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

Quatrième partie

Annexe : Curriculum Vitæ

© 2014 Tous droits réservés.

doc.univ-lille1.fr

A. CV Résumé

A.1 État civil

Nom : BOURAQADI SAÂDANI

Prénoms : Mohammed Nouraddine (Noury)

Date et lieu de naissance : 03 mai 1971 à Fès (Maroc)

Nationalité : Française

Situation familiale : Marié, 2 enfants

A.2 Parcours professionnel et académique

2001-actuellement : *Enseignant-Chercheur* au Département Informatique et Automatique de l'École des Mines de Douai.

1999-2001 : *Post-doc* au sein de l'équipe "Objets, Composants & Modèles" du département Informatique de l'École des Mines de Nantes. Participation à un projet pour le compte de France Télécom R&D sur l'utilisation de la réflexion et de la programmation par aspects pour la mobilité forte (code et processus).

1995-1999 : *Doctorat* en informatique de l'Université de Nantes effectué au sein de l'équipe Objets, Composants & Modèles (OCM) du Département Informatique de l'École des Mines de Nantes. Thèse intitulée : *Un MOP Smalltalk pour l'étude de la composition et de la compatibilité des métaclasses - Application à la programmation par aspects*, présentée le 13 juillet 1999, devant un jury composé de :
- Jean-François Perrot, Professeur, LIP6, Président.
- Christophe Dony, Professeur, LIRMM, Université de Montpellier, Rapporteur.
- François Pachet, Maître de Conférences HDR, LIP6, Sony-CSL Paris, Rapporteur.
- Charles Consel, Professeur, IRISA, Université de Rennes, Examinateur.
- Noureddine Mouaddib, Professeur, LINA, Université de Nantes, Examinateur.
- Pierre Cointe, Professeur, Dépt. Informatique, Ecole des Mines de Nantes, Directeur.

1995-1998 : *Consultant indépendant* en informatique. Collaboration avec des architectes et des infographistes pour la réalisation de solutions multimédia (CD-ROM et bornes interactives) pour le grand public. Donneurs d'ordre :

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

- Mairies des villes de Lorient, Nantes et St Ouen
- Direction Régionale des Arts et de la Culture (D.R.A.C.) Ile-de-France

1995 : DEA en informatique. Université de Nantes.

1994 : Diplôme d'ingénieur, spécialité informatique industrielle. École Nationale d'Ingénieurs de Brest.

A.3 Enseignement et responsabilités pédagogiques - depuis 2000

- (2007–2011) **Responsable pédagogique** de l'option ISIC (informatique) dans le cadre des 2 dernières années du cursus ingénieur de l'ÉCOLE DES MINES DE DOUAI.
- Responsable à l'ÉCOLE DES MINES DE DOUAI des enseignements de :
  - Langage C : niveau Bac+4, environ 20h/an de cours/TD/TP de 2001 à 2008 ;
  - Programmation par objets : niveau Bac+4, environ 40h/an de cours/TD/TP depuis 2001 à ce jour ;
  - Systèmes à objets distribués : niveau Bac+5, environ 18h/an de cours/TD/TP depuis 2001 à ce jour ;
  - Composants logiciels : niveau Bac+5, environ 18h/an de cours/TD/TP depuis 2004 à ce jour ;
  - Systèmes embarqués et Robotique Mobile : niveau Bac+5, environ 18h/an de cours/TD/TP depuis 2008 à ce jour ;
  - Initiation à la programmation : niveau Bac+3 pour la formation continue à distance (intervention nécessitant un déplacement à l'étranger), environ 8h/an de cours/TD de 2009 à 2011 ;
  - Introduction à la programmation par objets, 38h/an de cours/TD depuis 2011.
- Encadrement de projets :
  - Projets de découverte de la recherche : 3 à 5 projets réalisés par des équipes de 2 élèves, niveau Bac+4 - environ 80h d'encadrement/an, depuis 2009 à ce jour.
  - Projets scientifiques et techniques : 1 à 2 projets réalisés par des équipes de 4 à 8 élèves, niveau Bac+5 - environ 30h d'encadrement/an, depuis 2004 à ce jour.
- Jurys divers (Admission aux Ecoles des Mines, Stages, Projets encadrés par des collègues...) : niveaux Bac+3 à Bac+5, environ 20h/an, depuis 2001 à ce jour.
- Participation à l'Ecole des Mines de Nantes aux enseignements de :
  - Aperçu de Smalltalk : niveau Bac+5, environ 8h/an de cours/TD/TP de 2002 à 2003 (responsable du contenu).
  - Compilation : niveau Bac+4, 16h de TP en 2000.
  - Réflexion : niveau Bac+5, 12h de TP en anglais (étudiants étrangers du master EMOOSE) en 2000.
  - Proposition de sujets, encadrement et jury de mini-projets : niveau Bac+5, 5 projets effectués par des étudiants étrangers (interaction en anglais) en 2000.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

A.4 Recherche

A.4.1 Indices et mesure d'impact

Calculs effectués en mai 2013 avec "Google Scholar" [1].

h-index = 12

i10-index = 15

A.4.2 Publications et communications - période 2000-2013

- Ouvrages et actes : 5
- Revues et chapitres d'ouvrages : 10
- Communications internationales avec comité de lecture et actes : 32
- Communications nationales avec comité de lecture et actes : 5
- Autres communications : 12
- Exposés invités : 2

A.4.3 Encadrements

- Thèses soutenues : 4
- Thèses en cours : 2
- Post-doctorants : 4 dont 1 en cours
- Ingénieurs de recherche : 4 dont 1 en cours
- Stages Master : 7

A.4.4 Animation scientifique

- Comités de pilotage de conférences : 12
- Organisation de manifestations : 8
- Comités de programme et relecture d'articles : 42
- Examinateur dans des jurys de thèse : 2

1. http://scholar.google.fr/citations?user=JQxHcqQAAAAJ&hl=en

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

A.4.5 Projets et Collaborations

Période	*Financement*	*Intitulé*	*Partenaires*
2002-2006	Fonds Propres	Aspects et Composants	LIFL (Univ. Lille 1), **Ecole des Mines de Douai**
2003-2006	Groupement des Ecoles des Mines	MAAC (Modèles, Agents, Aspects et Composants)	Ecoles des Mines d'Alès, de **Douai**, de Nantes et de St Etienne
2003-2010	Fonds Propres	Systèmes Multi-Agents Robotiques	**Ecole des Mines de Douai**, équipe MAD du GREYC de Caen et MSI de l'UMMISCO de Hanoï, Vietnam
2004-2007	CPER, FEDER	Mosaïques (MOdèles et InfraStructures pour Applications ubIQUitairES)	LIFL (Univ. Lille 1) - LAMIH (Univ. Valenciennes) - **Ecole des Mines de Douai** - TRIGONE (Univ. Lille 1) - LEOST (INRETS Lille)
2007-2009	Institut Carnot MINES	ODiCe (Open DIgital ConcretE)	**Dépt. Informatique et Automatique** et Dépt. Génie Civil et Environmental, Ecole des Mines de Douai
2008	Privé (entreprise)	CCure (Réflexion et Extreme Programming)	**Ecole des Mines de Douai** et le créateur d'entreprise M. Cellé
2009-Actuellement	Région Nord-Pas de Calais + Fonds propres	Infrastructures logicielles flexibles pour les langages dynamiques	**Ecole des Mines de Douai** et l'INRIA Lille Nord Europe
2012-2013	PICOM	RoboShop : Robotique mobile en galerie commerciale	**Ecole des Mines de Douai**
2013-2015	Région Nord-Pas de Calais	CAIRE : Infrastructure logicielle pour la cartographie robotique	**Ecole des Mines de Douai**, INRIA Lille, IEMN Lille

© 2014 Tous droits réservés.

doc.univ-lille1.fr

B. Production Scientifique et Encadrements

B.1 Positionnement des travaux

Mes travaux de recherche portent sur la flexibilité des systèmes robotiques. Ils se positionnent à l'intersection du génie logiciel et de l'intelligence artificielle. Les contributions qui se placent dans le cadre général des langages dynamiques et réflexifs peuvent être réparties suivant 3 axes complémentaires :

- **Modularité pour et par la réflexion.** Cet axe concerne la *modularisation* des applications, avec pour cible privilégiée les logiciels de contrôle des robots mobiles et autonomes.
- **Infrastructures logicielles multi-robots.** Il s'agit de *middleware* et de *framework* destinés à la fois pour les applications mono-robots et pour le niveau macroscopique que représentent les flottes de robots en réseau.
- **Modèles de coordination décentralisée.** L'objet de cet axe est la *coordination* de flottes robotiques vues comme des systèmes multi-agents physiques.

Ces travaux ont donné lieu à différentes publications et communications listées par type en section B.4. La figure B.1 donne la répartition par axes de cette production. Chaque cercle comporte au centre le nombre de papiers qui traitent exclusivement du thème en question. Les articles qui relèvent de plusieurs thématiques apparaissent quant à eux dans les intersections. Les nombres entre parenthèse représentent le nombre d'ouvrages, chapitres et articles en revue pour chaque partie.

Les contributions des différents travaux sont le fruit d'un travail en équipe avec en particulier la participation de différents étudiants et ingénieurs que j'ai pu co-encadrer. La section B.3 en donne la liste. Ces encadrements sont également présentés sur la figure B.2 qui en donne la carte thématique. Les couleurs permettent de distinguer les niveaux des personnes encadrées (master, doctorat, post-doc et ingénieur).

B.2 Réalisations et prototypes

Durant les différents travaux décrits plus haut, nous avons voulu concrétiser nos idées sous la forme de prototypes aussi avancés et finalisés que possible. Il s'agit de valider expérimentalement les modèles théoriques proposés ou de manipuler et donc mieux appréhender les concepts étudiés. C'est à mes yeux, un point important du fait que l'informatique est une science appliquée. Une activité de développement est donc indispensable pour être confronté aux problèmes

© 2014 Tous droits réservés.

doc.univ-lille1.fr

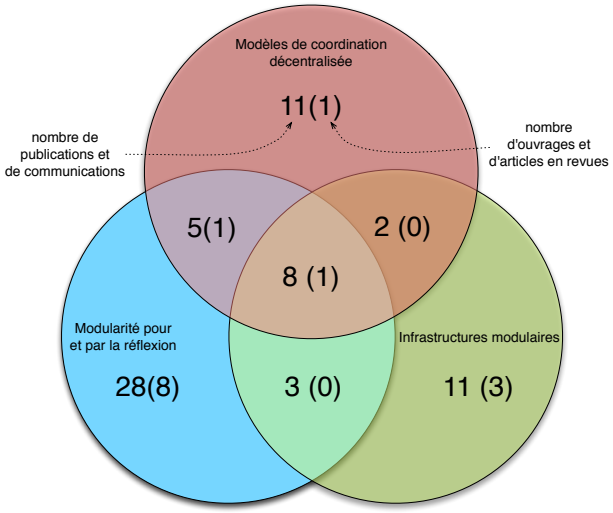

FIGURE B.1 – Thèmes des principales publications et communications

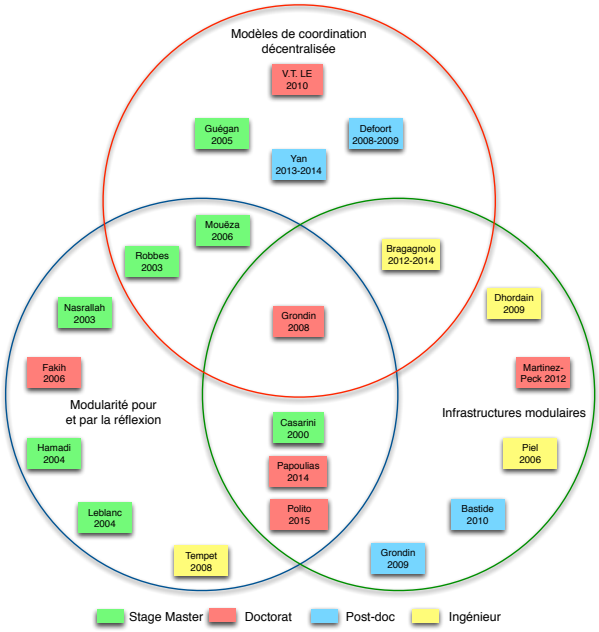

FIGURE B.2 – Carte thématique des encadrements

© 2014 Tous droits réservés.

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

Réalisations avec des retombées pour la communauté Smalltalk	
BoTest	Framework de test pour des applications robotiques
Ghost	Bibliothèque de proxies génériques
Ocean	Bibliothèque réseau développée en TDD
PhaROS	Bibliothèque pour développer des noeuds ROS
rST	Middleware pour applications distribuées
UbiquiTalk	Middleware P2P avec support de découverte automatique et de déploiement de code à la volée

Réalisations avec des retombées pour l'équipe	
BOSS	Simulateur 2D discret de systèmes multi-robots
FracTalk	Implémentation Smalltalk du modèle de composants Fractal
MalevaST	Implémentation Smalltalk du modèle de composants Maleva
WifiBotST	Framework de contrôle de robots Wifibots avec un moteur de simulation 2D

TABLE B.1 – Principaux logiciels développés

pratiques que rencontrent les ingénieurs de notre domaine. Une autre raison derrière cette démarche est le souhait de capitaliser les efforts fournis par les différents membres de l'équipe et *in fine* les fédérer notamment via des outils communs.

La liste alphabétique des principaux logiciels que nous avons développés est donnée par la table B.1. Une liste plus exhaustive, ainsi que le code et le descriptif détaillé de chaque logiciel sont disponibles en ligne [1]. A travers cette liste, nous pouvons constater, d'une part, que nous sommes fortement ancrés dans la communauté du langage dynamique Smalltalk. D'autre part, nous avons investi du temps dans des outils transversaux à l'échelle de l'équipe ou même à l'échelle de notre communauté.

B.3 Récapitulatif des encadrements passés et en cours

Thèses soutenues

1. Houssam Fakih. Intégration des fonctionnalités transversales dans les composants logiciels en utilisant la programmation par aspects. Lieu de déroulement des travaux : ÉCOLE DES MINES DE DOUAI. Doctorat de l'Université de Lille 1, présenté le 12 décembre 2006, devant un jury composé de :
   - Jean-Luc Dekeyser, Professeur, LIFL, Univ. de Lille 1, Président.
   - Jean-Claude Royer, Professeur, OBASCO, Ecole des Mines de Nantes, Rapporteur.
   - Stéphane Ducasse, Professeur, LISTIC, Univ. de Savoie, Rapporteur.
   - Laurence Duchien, Professeur, LIFL, Univ. de Lille 1, Directrice.
   - **Noury Bouraqadi**, Maître Assistant, DIA, Ecole des Mines de Douai, Co-encadrant (taux d'encadrement : 50%).

   Résumé : Cette thèse définit le cadre général pour appliquer les concepts de la programmation par aspects (AOP) dans les modèles à composants logiciels en vue de définir des

1. http://car.mines-douai.fr/software/

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

applications à base de composants et d'aspects. L'objectif est double : (1) intégrer les aspects pour la définition des fonctionnalités transversales des applications, (2) utiliser les composants pour définir les différentes constructions des aspects en vue d'améliorer leur réutilisation. Notre approche propose d'ouvrir les composants logiciels via deux interfaces AOP réflexives. Ces deux interfaces AOP sont définies à un niveau méta et permettent à l'aspect d'introspecter et d'intercesser tous les appels aux opérations du composant. Elles peuvent être liées l'une à l'autre et forment dans ce cas une liaison dite AOP, spécifique à notre modèle. La prise en charge des fonctionnalités transversales se fait alors en enrichissant cette liaison AOP par un aspect. Un aspect joue le rôle d'un connecteur dans notre approche. Il regroupe les interactions non-anticipées entre un ensemble de composants et la fonctionnalité transversale, plus précisément, entre les interfaces AOP de l'ensemble des composants et les interfaces de base de la partie métier de la fonctionnalité transversale désignée par le service. La définition d'une "fonctionnalité" transversale dans notre approche comprend : le service, l'aspect et la (les) coupe(s). Nous proposons de définir ces différentes constructions en tant qu'entités de première classe via des composants logiciels. Ceci améliore leur réutilisation et a d'autres avantages sûrs : les coupes peuvent, par exemple, être configurées tout au long des phases de cycle de vie de l'application. Nous avons appliqué notre approche au modèle Fractal. Ceci nous a amené à définir Fractal-AOP qui représente la projection de notre solution générale au modèle Fractal. Fractal-AOP est implantée sur la base de FracTalk, notre implantation Smalltalk de modèle Fractal.

2. Guillaume Grondin. MaDcAr-Agent : un modèle d'agents auto-adaptables à base de composants. Lieu de déroulement des travaux : ÉCOLE DES MINES DE DOUAI. Doctorat de l'Ecole des Mines de St Etienne, présenté le 24 novembre 2008, devant un jury composé de :
   - Stéphane Ducasse, Directeur de Recherche, INRIA, équipe RMOD, Lille, Président.
   - Jean-Pierre Briot, Directeur de Recherche, CNRS, LIP6, Paris, Rapporteur.
   - Michel Occello, Professeur, Univ. Pierre Mendès France, LCIS, Valence, Rapporteur.
   - Sylvain Lecomte, Professeur, LAMIH, Univ. de Valenciennes, Examinateur.
   - Jean-Paul Arcangeli, Maître de conférences, IRIT, Univ. Paul Sabatier, Toulouse, Examinateur.
   - Olivier Boissier, Professeur, LSTI, Ecole des Mines de St Etienne, Directeur.
   - **Noury Bouraqadi**, Maître Assistant, DIA, Ecole des Mines de Douai, Co-encadrant (taux d'encadrement : 50%).
   - Laurent Vercouter, Maître Assistant, LSTI, Ecole des Mines de St Etienne, Co-encadrant.

   Résumé : Dans le cadre de l'informatique ubiquiste, l'environnement d'exécution d'une application est constitué de machines hétérogènes en ressources matérielles et appartenant à des utilisateurs différents (PC, PDA, téléphone mobile, etc.). Ces caractéristiques imposent de structurer l'application en une organisation d'unités logicielles relativement indépendantes qui coopèrent et interagissent. Dans cette thèse, nous proposons MaDcAr-Agent, un modèle d'agents auto-adaptables à base de composants et muni d'une infrastructure dédiée à l'adaptation. Ce modèle se caractérise par la présence d'un niveau méta qui comporte notamment un moteur d'assemblage en charge des adaptations dynamiques

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

et automatiques en fonction du contexte de l'agent. Le fonctionnement du niveau méta est guidé par la spécification de deux politiques : la politique d'assemblage qui permet à l'agent de s'adapter aux changements de contexte en fonction des composants disponibles et la politique de gestion de contenu qui permet à l'agent d'avoir les composants dont il a le plus besoin grâce aux interactions avec les autres agents. A travers ces spécifications explicites et découplées du comportement applicatif de l'agent, le concepteur d'agents peut prendre en charge la perturbation d'un système dû à des changements imprévus et répétés, sans pour autant nuire à l'autonomie des agents qui composent ce système. Pour valider notre approche, diverses expérimentations ont été menées avec ce modèle, notamment dans le cadre d'un scénario impliquant des robots mobiles qui doivent explorer une zone inconnue.

3. Van Tuan Le. Coopération dans les systèmes multi-robots : Contribution au maintien de la connectivité et à l'allocation dynamique de rôles. Lieu de déroulement des travaux : 50% du temps à l'ÉCOLE DES MINES DE DOUAI et 50% au laboratoire UMMISCO [2] (IFI [3]/IRD [4]/UPMC [5]), de Hanoï, Vietnam. Doctorat de l'université de Caen, présenté le 6 octobre, 2010 devant un jury composé de :
   - Michel Occello, Professeur, Univ. Pierre Mendès France, LCIS, Valence, Président.
   - Simon Lacroix, Directeur de Recherche, CNRS, LAAS, Toulouse, Rapporteur.
   - Catherine Tessier, Maître de Recherche (HDR), DCSD, Onera, Toulouse, Rapporteur.
   - Olivier Simonin, Maître de Conférences, LORIA, Univ. Henri Poincaré, Nancy, Examinateur.
   - François Bourdon, Professeur, GREYC, Univ. de Caen-Basse Normandie, Directeur.
   - **Noury Bouraqadi**, Maître Assistant, DIA, Ecole des Mines de Douai, Co-encadrant (taux d'encadrement : 50%).
   - Victor Moraru, Professeur, UMMISCO, IFI, Hanoï, Vietnam, Co-encadrant.
   - Serge Stinckwich, Maître de conférences, GREYC, Univ. de Caen-Basse Normandie, chercheur associé au sein du laboratoire UMMISCO, IRD/UMPC, Hanoï, Vietnam, Co-encadrant.

   Résumé : Dans cette thèse, nous proposons une solution qui permet à n'importe quelle collection de robots hétérogènes de s'organiser en équipes et ce, en fonction à la fois des exigences de la tâche à réaliser, des robots disponibles et de leurs ressources. Notre approche basée sur la décomposition d'une tâche complexe en rôles, sépare les préoccupations du niveau de conception et du niveau d'implémentation. Nous proposons des heuristiques basées sur le protocole Contract-Net pour affecter les rôles aux robots afin de former des coalitions. Chaque coalition se compose de robots coopérant de manière étroite pour effectuer une tâche unique. L'affectation de rôles aux robots, ainsi que la coopération de ces derniers requière que les robots puissent communiquer de manière fréquente. Or, la connectivité du réseau de robots est un pré-requis de la communication. Nous proposons

2. Unité de Modélisation Mathématique et Informatique des Systèmes COmplexes
3. Institut de la Francophonie pour l'Informatique.
4. Institut de Recherche pour le Développement.
5. Université Pierre et Marie Curie, Paris.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

une solution originale à ce problème basée sur notre concept de "sensibilité à la connectivité". Il s'agit de doter chaque robot d'une connaissance de la structure du réseau. Nous montrons qu'une connaissance partielle et locale à chaque robot, peut être exploitée pour maintenir la connectivité du réseau de manière distribuée et robuste. Chaque robot peut ainsi planifier localement ses déplacements sans mettre en péril la connectivité du réseau global. En effet, cette connaissance locale que représente la sensibilité à la connectivité peut être exploitée pour déterminer les robots et les connexions critiques du réseau de robots.

4. Mariano Martinez Peck. Application-Level Virtual Memory for Object-Oriented Systems. Lieu de déroulement des travaux : 90% du temps à l'ÉCOLE DES MINES DE DOUAI et 10% à l'INRIA Lille. Doctorat de l'université de Lille 1 et de l'Ecole des Mines de Douai, présenté le 29 octobre 2012 devant un jury composé de :
   - Jean-Bernard Stefani, Directeur de recherche, INRIA Grenoble-Rhône-Alpes, Président.
   - Stéphane Ducasse, Directeur de Recherche, INRIA, équipe RMOD, Lille, Directeur.
   - Robert Hirschfeld, Professeur, Hasso-Plattner-Institut, Universität Potsdam, Allemagne, Rapporteur.
   - Christophe Dony, Professeur, Université Montpellier 2, Rapporteur
   - Roel Wuyts, Professeur, IMEC & Katholieke Universiteit Leuven, Belgique, Examinateur.
   - **Noury Bouraqadi**, Maître Assistant, DIA, Ecole des Mines de Douai, Co-encadrant (taux d'encadrement : 30%).
   - Marcus Denker, Chargé de Recherche, INRIA, équipe RMOD, Lille, Co-encadrant.
   - Luc Fabresse, Maître Assistant, DIA, Ecole des Mines de Douai, Co-encadrant.

   Résumé : During the execution of object-oriented applications, several millions of objects are created, used and then collected if they are not referenced. Problems appear when objects are unused but cannot be garbage-collected because they are still referenced from other objects. This is an issue because those objects waste primary memory and applications use more primary memory than what they actually need. We claim that relying on operating systems (OS) virtual memory is not always enough since it is completely transparent to applications. The OS cannot take into account the domain and structure of applications. At the same time, applications have no easy way to control nor influence memory management.

   In this dissertation, we present Marea, an efficient application-level virtual memory for object-oriented programming languages. Its main goal is to offer the programmer a novel solution to handle application-level memory. Developers can instruct our system to release primary memory by swapping out *unused yet referenced objects* to secondary memory.

   Marea is designed to : 1) save as much memory as possible *i.e.* the memory used by its infrastructure is minimal compared to the amount of memory released by swapping out unused objects, 2) minimize the runtime overhead *i.e.* the swapping process is fast enough to avoid slowing down primary computations of applications, and 3) allow the programmer to control or influence the objects to swap.

   Besides describing the model and the algorithms behind Marea, we also present our im-

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

plementation in the Pharo programming language. Our approach has been qualitatively and quantitatively validated. Our experiments and benchmarks on real-world applications show that Marea can reduce the memory footprint between 25% and 40%.

Thèses en cours

1. Nick Papoulias. Refléxion et débogage à distance d'applications contraintes en ressources. Lieu de déroulement des travaux : 90% du temps à l'ÉCOLE DES MINES DE DOUAI et 10% à l'INRIA Lille. Thèse co-encadrée par :
   – Stéphane Ducasse, Directeur de Recherche, INRIA, équipe RMOD, Lille, Directeur.
   – **Noury Bouraqadi**, Maître Assistant, DIA, Ecole des Mines de Douai, Co-encadrant (taux d'encadrement : 30%).
   – Marcus Denker, Chargé de Recherche, INRIA, équipe RMOD, Lille, Co-encadrant.
   – Luc Fabresse, Maître Assistant, DIA, Ecole des Mines de Douai, Co-encadrant.
2. Guillermo Polito. Isolation et modularisation de systèmes réflexifs. Lieu de déroulement des travaux : 90% du temps à l'ÉCOLE DES MINES DE DOUAI et 10% à l'INRIA Lille. Thèse co-encadrée par :
   – Stéphane Ducasse, Directeur de Recherche, INRIA, équipe RMOD, Lille, Directeur.
   – **Noury Bouraqadi**, Maître Assistant, DIA, Ecole des Mines de Douai, Co-encadrant (taux d'encadrement : 30%).
   – Marcus Denker, Chargé de Recherche, INRIA, équipe RMOD, Lille, Co-encadrant.
   – Luc Fabresse, Maître Assistant, DIA, Ecole des Mines de Douai, Co-encadrant.

Post-doctorants

[1] Gautier Bastide. Post-doctorant dans le cadre du projet ODICE. Mise en oeuvre de la plate-forme UbiquiTalk pour une application de calcul scientifique réparti (6 mois), 2008.

[2] Michaël Defoort. Post-doctorant. Navigation et évitement d'obstacles dans une flotte multi-robots (1 an), 2008-2009.

[3] Guillaume Grondin. Post-doctorant dans le cadre du projet ODICE. Mise en oeuvre de la plate-forme UbiquiTalk pour une application de calcul scientifique réparti (6 mois), 2009-2010.

[4] Zhi Yan. Post-doctorant. Dimensionnement de flottes robotiques, 2013-2014.

Ingénieurs de recherche

[1] Michaël Piel. Ingénieur de recherche (12 mois). Développment de la plate-forme UbiquiTalk, 2005-2006.

[2] Franck Tempet. Ingénieur de recherche. Mise en oeuvre de la réflexion et du TDD dans le cadre du projet CCure, 2008.

[3] Gautier Dhordain. Ingénieur de recherche (6 mois). Mise en oeuvre de la plate-forme UbiquiTalk pour une application de commerce ubiquitaire, 2008-2009.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[4] Santiago Bragagnolo. Ingénieur de recherche (18 mois). Projet RoboShop, 2012-2014.

Masters

[1] Gabriel Casarini. Towards transparent strong mobility using a reflective smalltalk. Master's thesis, Vrije Universiteit Brussel (Belgium) In Collaboration with Ecole des Mines de Nantes (France)., 2000.

[2] Rabi Nasrallah. Programmation par aspects et services web. Master's thesis, Université de Technologie de Troie, July 2003.

[3] Romain Robbes. Mise en oeuvre de la programmation par aspects dans le cadre des systèmes multi-agents. Master's thesis, Université de Caen, 2003.

[4] Ali Hamadi. Une implémentation du modèle de composants fractal en smalltalk. Master's thesis, Université de Nantes, September 2004.

[5] Gabriel Leblanc. Vers une généralisation du concept d'aspect. Master's thesis, Université de Lille 1, July 2004.

[6] Ludovic Guégan. Hybridation paramètrable d'agents pour systèmes embarqués. Master's thesis, Université de Caen, 2005.

[7] Rémy Mouëza. Architecture réactive à subsomption à l'aide des composants maleva. Master's thesis, Université de Caen, June 2006.

B.4 Publications et communications de la période 2000-2013

Ouvrages et revues

[1] Noury Bouraqadi, editor. *Proceedings of the 5th National Conference on"Control Architecture of Robots" (CAR)*, Douai, France, May 2010. Ecole des Mines de Douai.

[2] Noury Bouraqadi, editor. *Actes des journées Multi-Agent et Composant*, Nîmes, France, March 2006.

[3] Noury Bouraqadi and Roel Wuyts, editors. *Computer Languages, Systems and Structures Journal - Special Issue on Smalltalk*, volume 31. Elsevier, October 2005.

[4] U. Assmann, E. Pulvermueller, N. Bouraqadi I. Borne, and P. Cointe, editors. *SC 2003 : Workshop on Software Composition Affiliated with ETAPS 2003*, volume 82. Elsevier Science Publishers, April 2003.

[5] E. Pulvermueller, I. Borne, N. Bouraqadi, P. Cointe, and U. Assmann, editors. *Proceedings of the ETAPS 2002 Workshop on Software Composition*, volume 65 of *Electronic Notes in Theoretical Computer Science*, Grenoble, France, April 2002. Elsevier.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

Articles de revues et chapitres d'ouvrages

[1] Mariano Martinez-Peck, Noury Bouraqadi, Marcus Denker, Stéphane Ducasse, and Luc Fabresse. Object-based virtual memory brought to the application level. *Journal Of Object Technology*, 12(1), jan 2013.

[2] Mariano Martinez-Peck, Noury Bouraqadi, Stéphane Ducasse, and Luc Fabresse. Object swapping challenges : an evaluation of imagesegment. *Computer Languages, Systems and Structures*, 2012.

[3] Luc Fabresse, Noury Bouraqadi, Christophe Dony, and Marianne Huchard. A language to bridge the gap between component-based design and implementation. *Computer Languages, Systems and Structures*, 2012.

[4] Noury Bouraqadi. *More Pharo By Example*, chapter Sockets. Square Bracket Associates, 2013 (à paraître).

[5] Michaël Defoort, Arnaud Doniec, and Noury Bouraqadi. *Informatics in Control Automation and Robotics*, chapter Decentralized Robust Collision Avoidance Based on Receding Horizon Planning and Potential Field for Multi-Robots Systems, pages 201–215. Number 85 in LNEE. Springer, 2011.

[6] R. Razavi, N. Bouraqadi, J. W. Yoder, J. F. Perrot, and R. Johnson. Language support for adaptive object-models using metaclasses. *Journal of Computer Languages, Systems and Structures*, 31(3-4) :199–218, October 2005. Also published in the proceedings of the reserach track of the ESUG 2004 conference.

[7] H. Fakih and N. Bouraqadi. Les aspects et les composants logiciels : Etude de cas avec le modèle de composant fractal. *L'Objet*, 2005.

[8] Noury Bouraqadi and Thomas Ledoux. *Aspect-Oriented Software Development*, chapter 12 – Supporting AOP using Reflection, pages 261–282. Addison-Welsey, 2005.

[9] N. Bouraqadi. Safe metaclass composition using mixin-based inheritance. *Journal of Computer Languages and Structures*, 30(1-2) :49–61, April 2004. Special issue : Smalltalk Language.

[10] N. Bouraqadi and T. Ledoux. Le point sur la programmation par aspects. *Technique et Science Informatique*, 20(4) :505–528, 2001.

Communications internationales avec comité de lecture et actes

[1] Jannik Laval, Luc Fabresse, and Noury Bouraqadi. A methodology for testing mobile autonomous robots. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[2] V. T. Le, N. Bouraqadi, S. Stinckwich, and A. Doniec. Role-based dynamic coalitions of multi-tasked rescue robots. In *Proceedings of the 9th International Conference on Information Systems for Crisis Response and Management (ISCRAM)*, Vancouver, Canada, apr 2012.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[3] Noury Bouraqadi and Serge Stinckwich. The next 700 control architectures for rescue robotics. In *Proceedings of the 9th IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR) - Outrageous Visions for Computing in Rescue Robotics Track*, Kyoto, Japan, novembre 2011. IEEE.

[4] Mariano Martinez Peck, Noury Bouraqadi, Marcus Denker, Stéphane Ducasse, and Luc Fabresse. Problems and challenges when building a manager for unused objects. In *Proceedings of the Smalltalks 2011 Conference*, Bernal, Buenos Aires, Argentina, 2011.

[5] Nick Papoulias, Noury Bouraqadi, Marcus Denker, Stéphane Ducasse, and Luc Fabresse. Towards structural decomposition of reflection with mirrors. In Loïc Lagadec, editor, *International Workshop on Smalltalk Technologies (IWST)*, Edinburgh, Scotland, UK, August 2011. ACM.

[6] Mariano Martinez-Peck, Noury Bouraqadi, Marcus Denker, Stéphane Ducasse, and Luc Fabresse. Efficient proxies in smalltalk. In Loïc Lagadec, editor, *International Workshop on Smalltalk Technologies (IWST)*, Edinburgh, Scotland, UK, August 2011. ACM.

[7] Noury Bouraqadi and Luc Fabresse. Towards small portable virtual machines. In Marcus Denker and Gabriela Arévalo, editors, *Proceedings of the Smalltalks 2010 Conference*, Concepcion del Uruguay, Argentina, 2010.

[8] Luc Fabresse, Noury Bouraqadi, Christophe Dony, and Marianne Huchard. Component-oriented programming : From requirements to language support. In Marcus Denker and Gabriela Arévalo, editors, *Proceedings of the Smalltalks 2010 Conference*, Concepcion del Uruguay, Argentina, 2010.

[9] Mariano Martinez Peck, Noury Bouraqadi, Marcus Denker, Stéphane Ducasse, and Luc Fabresse. Experiments with a fast object swapper. In Marcus Denker and Gabriela Arévalo, editors, *Proceedings of the Smalltalks 2010 Conference*, Concepcion del Uruguay, Argentina, 2010.

[10] Mariano Martinez Peck, Noury Bouraqadi, Marcus Denker, Stéphane Ducasse, and Luc Fabresse. Visualizing objects and memory usage. In Marcus Denker and Gabriela Arévalo, editors, *Proceedings of the Smalltalks 2010 Conference*, Concepcion del Uruguay, Argentina, 2010.

[11] Guillaume Grondin, George Aouad, Noury Bouraqadi, and Denis Damidot. Odice : an open distributed platform for computational simulation applied to concrete, concrete modeling. In *Proceedings of ConMod 2010 Symposium on Concrete Modelling*, Lausanne, Switzerland, June 2010. RILEM Publications.

[12] Serge Stinckwich, Noury Bouraqadi, Van Tuan Le, and Arnaud Doniec. Dynamic role assignment for large-scale multi-agent robotic systems. In Serge Stinckwich, editor, *Post-Proceedings of the 2nd Workshop on Agent Technology for Disaster Management (ATDM) of the 12th International Conference on Principles of Practice in Multi-Agent Systems (PRIMA)*, Studies in Computational Intelligence, Nagoya, Japan, December 2009. Springer.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[13] Michaël Defoort, Arnaud Doniec, and Noury Bouraqadi. A decentralized collision avoidance algorithm for multi-robots navigation. In *Proceedings of ICINCO (International Conference on Informatics in control, Automation and Robotics)*, Milano, Italy, July 2009.

[14] Van Tuan Le, Noury Bouraqadi, Serge Stinckwich, Victor Moraru, and Arnaud Doniec. Making networked robot connectivity-aware. In *Proceedings of ICRA (International Conference on Robotics and Automation)*, Kobe, Japan, May 2009.

[15] Van Tuan Le, Noury Bouraqadi, Serge Stinckwich, and Victor Moraru. Connectivity awareness in networked robotic systems. In *Proceedings of IEEE-RIVF International Conference on Computing and Communication Technologies*, Da Nang City, Vietnam, July 2009.

[16] Noury Bouraqadi and Luc Fabresse. Clic : A component model symbiotic with smalltalk. In *Proceedings of the International Workshop on Smalltalk Technologies*, Brest, France, August 2009. ACM.

[17] Arnaud Doniec, Noury Bouraqadi, Michaël Defoort, Van Tuan Le, and Serge Stinckwich. Distributed constraint reasoning applied to multi-robot exploration. In *Proceedings of the 21st International Conference on Tools with Artificial Intelligence (ICTAI)*, Newark (NYC Metropolitan Area), New Jersey, USA, November 2009. IEEE.

[18] Guillaume Grondin, Noury Bouraqadi, and Laurent Vercouter. Component reassembling and state transfer in madcar-based self-adaptive software. In *Proceedings of the 46th International Conference TOOLS-EUROPE (Objects, Models, Components, Patterns)*, Switzerland, June 2008.

[19] Noury Bouraqadi and Serge Stinckwich. Towards a generic anticipatory agent architecture for mobile robots. In *Proceedings of the MARS (Multi-Agent and Robotic Systems) of the ICINCO conference*, pages 102–105, France, May 2007.

[20] Noury Bouraqadi and Serge Stinckwich. Bridging the gap between morphic visual programming and smalltalk code. In *Proceedings of International Conference on Dynamic Languages*, pages 107–126, Switzerland, August 2007.

[21] Guillaume Grondin, Noury Bouraqadi, and Laurent Vercouter. Madcar : an abstract model for dynamic and automatic (re-)assembling of component-based applications. In *Proceedings of the 9th International Symposium on CBSE (Component-Based Software Engineering)*, LNCS, pages 360–367, Sweden, June 2006. Springer.

[22] N. Bouraqadi, D. Seriai, and G. Leblanc. Towards unified aspect-oriented programming. In *ESUG 2005 Research Conference*, Brussels, Belgium, August 2005.

[23] R. Robbes, N. Bouraqadi, and S. Stinckwich. An aspect-based multi-agent system. In *Research Track of the ESUG 2004 Smalltalk Conference*, Köthen (Anhalt), Germany, September 2004.

[24] R. Razavi, N. Bouraqadi, J. W. Yoder, J. F. Perrot, and R. Johnson. Language support for adaptive object-models using metaclasses. In *Research Track of the ESUG 2004 Smalltalk Conference*, Köthen (Anhalt), Germany, September 2004. Selected for publication in the Elsevier international journal "Computer Languages, Systems and Structures".

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[25] H. Fakih, N. Bouraqadi, and L. Duchien. Aspects and software components : A case study of the fractal component model. In *International Workshop on Aspect-Oriented Software Development (WAOSD 2004)*, Beijing, China, September 2004.

[26] Houssam Fakih, Noury Bouraqadi, and Laurence Duchien. Towards integrating aspects and components. In *Third AOSD Workshop on Aspects, Components, and Patterns for Infrastructure Software (ACP4IS)*, March 2004.

[27] N. Bouraqadi. Metaclass composition using mixin-based inheritance. In *Research Track of the ESUG 2003 Smalltalk Conference*, Bled, Slovenia, August 2003. European Smalltalk Users Group (ESUG).

[28] P. Gahide, N. Bouraqadi, and L. Duchien. Promoting component reuse by integrating aspects and contracts in an architecture model. In Yvonne Coady, editor, *Proceedings of the First AOSD Workshop on Aspects, Components, and Patterns for Infrastructure Software*, pages 51–55, Enschede, The Netherlands, April 2002. University of British Columbia. TR-2002-02. Workshop held in conjunction with the First International Conference on Aspect-Oriented Software Developments (AOSD 2002).

[29] E. Tanter, N. Bouraqadi, and J. Noyé. Reflex - towards an open reflective extension of java. In S. Matsuoka A. Yonezawa, editor, *Proceedings of Reflection 2001*, number 2192 in LNCS, pages 25–43, Kyoto, Japan, September 2001. Springer Verlag.

[30] N. Bouraqadi and T. Ledoux. How to weave ? ECOOP 2001 Workshop on Advanced Separation of Concerns, June 2001.

[31] P.C. David, T. Ledoux, and N. Bouraqadi. Two-step weaving with reflection using aspectj. OOPSLA'01 Workshop on "Advanced Separation of Concerns in Object-Oriented Systems", October 2001.

[32] N. Bouraqadi. Concern oriented programming using reflection. OOPSLA 2000 Workshop on Advanced Separation of Concerns in Object-Oriented Systems, October 2000.

[33] T. Ledoux and N. Bouraqadi. Adaptability in Mobile Agent Systems using Reflection. In *Workshop on Reflective Middleware*, April 2000.

Communications nationales avec comité de lecture et actes

[1] Guillaume Grondin, Noury Bouraqadi, and Laurent Vercouter. Un modèle pour le développement d'agents auto-adaptables. In *Actes des JFSMA (Journées Francophones des Systèmes Multi-Agents)*, Lyon, France, October 2009.

[2] Guillaume Grondin, Noury Bouraqadi, and Laurent Vercouter. Assemblage automatique et adaptation d'applications à base de composants. In *Actes de LMO (Langage et Modèles à Objets)*, Monréal, Canada, March 2008.

[3] H. Fakih and N. Bouraqadi. Les aspects et les composants logiciels - étude de cas avec le modèle de composants fractal. In *First French Workshop on Aspect-Oriented Software Development(JFDLPA 2004)*, Paris, France, September 2004.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[4] R. Robbes, N. Bouraqadi, and S. Stinckwich. Un modèle multi-agent unifiant les notions de groupe et d'aspect. In *Systèmes multi-agents défis scientifiques et nouveaux usages - Actes des JFSMA 2004*, Paris, France, November 2004.

[5] E. Tanter, N. Bouraqadi, and J. Noyé. Reflex : une extension réflexive de Java portable, souple et performante. In R. Godin and I. Borne, editors, *LMO 2001, numéro spécial de la revue l'Objet*, volume 7. Hermès Science, 2001.

Autres communications

[1] Luc Fabresse, Jannik Laval, and Noury Bouraqadi. Towards test-driven development for mobile robots. In D . Brugali, N . Hochgeschwender, and R . Philippsen, editors, *Proceedings of the ICRA 2013 Workshop on Software Development and Integration in Robotics (SDIR VIII)*, 2013.

[2] Noury Bouraqadi, Luc Fabresse, and Arnaud Doniec. On fleet size optimization for multi-robot frontier-based exploration. In *7th National Conference on"Control Architecture of Robots"*, May 2012.

[3] Mariano Martinez Peck, Noury Bouraqadi, Marcus Denker, Stéphane Ducasse, and Luc Fabresse. Object graphs swapping. Presentation at the IWST 2010 workshop of ESUG 2010, September 2010.

[4] Noury Bouraqadi and Luc Fabresse. Ocean : Towards a portable networking library. Presentation at the IWST 2010 workshop of ESUG 2010, September 2010.

[5] Arnaud Doniec, Noury Bouraqadi, Michaël Defoort, Van Tuan Le, and Serge Stinckwich. Multi-robot exploration under communication constraint : a discsp approach. In *5th National Conference on "Control Architecture of Robots" (CAR)*, Douai, France, May 2010.

[6] Van Tuan Le, Noury Bouraqadi, Serge Stinckwich, Victor Moraru, and Arnaud Doniec. Maintaining connectivity in multi-robot systems through connectivity awareness. In *Proceedings of the 5th National Conference on"Control Architecture of Robots" (CAR)*, Douai, France, May 2010.

[7] Noury Bouraqadi and Serge Stinckwich. Flocking-based multi-robot exploration. In *Proceedings of the 4th National Conference on Control Achitectures for Robots (CAR)*, Toulouse, France, April 2009.

[8] Noury Bouraqadi and Serge Stinckwich. Towards an adaptive robot control architecture. In *Proceedings of CAR (National Conference on Control Architectures of Robots : from Models to Execution on Distributed Control Architectures)*, Paris, France, May 2007.

[9] Guillaume Grondin, Noury Bouraqadi, and Laurent Vercouter. Assemblage automatique de composants pour la construction d'agents avec madcar. In *Actes des journées Multi-Agent et Composant (JMAC)*, Nîmes, France, March 2006.

[10] N. Bouraqadi. Efficient support for mixin-based inheritance using metaclasses. In *Workshop on Reflectively Extensible Programming Languages and Systems at The International Conference on Generative Programming and Component Engineering (GPCE'03)*, Erfurt, Germany, September 2003.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[11] N. Bouraqadi. Metaclasstalk - a testbed for exploring programming pradigms. Talk given at the Europeen Smalltalk Users Group (ESUG) 10th Smalltalk Conference, Douai - France, August 2002.

[12] N. Bouraqadi, T. Ledoux, and M. Südholt. A reflective infrastructure for coarse-grained strong mobility and its tool-based implementation. In *Proceedings of the International Workshop on "Experiences with reflective systems"*, September 2001. also published as Technical Report 01/7/INFO.

Exposés invités

[1] Noury Bouraqadi. Robots mobiles et autonomes avec pharo. Invited Speaker at the "Journées Méditérranéennes des Logiciels Libres", November 2010.

[2] Noury Bouraqadi. Les défis de l'informatique ubiquitaire. Invited Speaker to the M2M (Machine to Machine) workshop of the Net 2006 conference, December 2006.

Rapports techniques

[1] Guillaume Grondin and Noury Bouraqadi. Rapport final du projet "odice (open digital concrete) : Plate-forme ouverte pour réaliser des expériences virtuelles sur les matériaux". Technical Report 07DD46GCE+IA1 CALCUL REPARTI, Institut Carnot MINES, December 2009.

[2] Noury Bouraqadi. Rapport d'activité 2008 du projet "odice (open digital concrete) : Plate-forme ouverte pour réaliser des expériences virtuelles sur les matériaux". Technical Report 07DD46GCE+IA1 CALCUL REPARTI, Institut Carnot MINES, February 2009.

[3] Guillaume Grondin, Noury Bouraqadi, and Laurent Vercouter. Modèle de transfert d'état pour l'adaptation d'applications à base de composants. Technical Report 2007-2-2, Ecole des Mines de Douai, February 2007.

[4] Guillaume Grondin, Noury Bouraqadi, and Laurent Vercouter. Conception d'agents auto-adaptables à base de composants. Technical Report 2007-2-1, Ecole des Mines de Douai, February 2007.

[5] Guillaume Grondin, Noury Bouraqadi, and Laurent Vercouter. MaDcAr : an abstract model for dynamic and automatic (re-)assembling of component based applications. Technical Report 2006-700-009, Ecole des Mines de St Etienne, July 2006.

[6] R. Nassrallah and N. Bouraqadi. Les services web : un condensé. Technical Report 2003-5-5, Ecole des Mines de Douai, May 2003.

[7] N. Bouraqadi. Simplification de la production de logiciel par la programmation par aspects. Technical Report 2003-3-1, Ecole des Mines de Douai, March 2003.

[8] N. Bouraqadi and T. Ledoux. Aspect-oriented programming using reflection. Technical Report 2002-10-3, Ecole des Mines de Douai, October 2002.

[9] N. Bouraqadi. Metaclasstalk benchmark. Technical Report 2002-7-2, Ecole des Mines de Douai, July 2002.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[10] N. Bouraqadi, T. Ledoux, and M. Südholt. Un survole de l'architecture du prototype. Technical report, Ecole des Mines de Nantes, May 2001. 4th technical report for the RAM project (Reflection for Adaptable Mobility), partially funded by France Telecom R & D.

[11] N. Bouraqadi, R. Douence, T. Ledoux, O. Motelet, and M. Südholt. Status of work on AOP at the OCM group, april 2001. Technical Report 01/4/INFO, Ecole des Mines de Nantes, April 2001.

[12] N. Bouraqadi, R. Douence, T. Ledoux, and M. Südholt. Une infrastructure réflexive pour la mobilité forte en java. Technical report, Ecole des Mines de Nantes, December 2000. 3rd technical report for the RAM project (Reflection for Adaptable Mobility), partially funded by France Telecom R & D.

[13] N. Bouraqadi, R. Douence, T. Ledoux, and M. Südholt. Un modèle de mobilité forte en java. Technical report, Ecole des Mines de Nantes, July 2000. 2nd technical report for the RAM project (Reflection for Adaptable Mobility), partially funded by France Telecom R & D.

[14] N. Bouraqadi. Java et la réflexion. Technical Report 2000-4-INFO, Ecole des Mines de Nantes, January 2000. 1st technical report for the RAM project (Reflection for Adaptable Mobility), partially funded by France Telecom R & D.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

© 2014 Tous droits réservés.

doc.univ-lille1.fr

C. Animation scientifique

C.1 Comités de pilotage de manifestations

2003-2010 Conférence internationale ESUG (European Smalltalk Users Group)

2010-2013 Conférence nationale CAR (Control Architectures for Robots)

C.2 Organisation de manifestations

2012 Co-organisation de la manifestation internationale "Pharo Conference", 24-25 Mai 2012 à Villeneuve d'Ascq.

2010 Organisation de la $5^{ème}$ édition de la conférence nationale CAR (Control Architectures for Robots), 18-19 mai 2010, Douai.

2008 Co-organisation de la $9^{ème}$ édition du Annual International Workshop "Engineering Societies in the Agents World" (ESAW 08), 24-26 septembre 2008, St Etienne. Collaboration avec l'Ecole de Mines d'Alès et l'Ecole des Mines de St Etienne.

2006 Co-organisation de la $2^{ème}$ édition de la Journée Multi-Agents, Aspects et Composants (JMAC), 21 Mars 2006, Nîmes, en marge de la conférence Langages et Modèles à Objet (LMO). Collaboration avec l'Ecole de Mines d'Alès, l'Ecole des Mines de Nantes et l'Ecole des Mines de St Etienne.

2004 Co-organisation de la $1^{ère}$ édition de la Journée Multi-Agents, Aspects et Composants (JMAC) le 23 novembre 2004, à Paris, en marge de la conférence JFSMA (Journées Francophones des Systèmes Multi-Agents). Collaboration avec l'Ecole de Mines d'Alès, l'Ecole des Mines de Nantes et l'Ecole des Mines de St Etienne.

2003 Co-organisation de la $2^{ème}$ édition du workshop international *Software Composition* à la conférence ETAPS 2003 à Varsovie, Pologne. Collaboration avec l'université de Karlsruhe (Allemagne) et l'École des Mines de Nantes.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

2002

- Co-organisation de la $10^{ème}$ édition de la conférence internationale ESUG, à Douai. Collaboration avec le GREYC de Caen et Télécom Lille.
- Co-organisation de la $1^{ère}$ édition du workshop international *Software Composition* à la conférence ETAPS 2002 à Grenoble. Collaboration avec l'université de Karlsruhe (Allemagne) et l'École des Mines de Nantes.

C.3 Participation à des jurys de thèse

Thèse de Sébastien Leriche

- Titre : Architectures à composants et agents pour la conception d'applications réparties adaptables
- Université : Toulouse III - Paul Sabatier (UPS)
- Laboratoire : IRIT
- Date : 05 décembre 2006
  - Dr. Jean-Paul Arcangeli, IRIT, UPS (Directeur)
  - Pr. Jean-Paul Bahsoun, IRIT, UPS (Examinateur)
  - Pr. Guy Bernard, INT Every (Rapporteur)
  - Dr. Noury Bouraqadi, ÉCOLE DES MINES DE DOUAI (Examinateur)
  - Pr. Jacques Ferber, LIRMM, (Rapporteur)
  - Pr. Daniel Hagimont, IRIT, ENSEEIHT (Examinateur)
  - Pr. Patrick Sallé, IRIT, ENSEEIHT (Examinateur)

Thèse de Luk Stoops

- Titre : Progressive Mobility
- Université : Vrije Universiteit Brussel (VUB)
- Laboratoire : Programming Language Lab (Prog)
- Date : 27 août 2004
- Jury :
  - Prof. Viviane Jonckers, Software Languages Lab, VUB (Présidente)
  - Dr. Noury Bouraqadi, ÉCOLE DES MINES DE DOUAI (Examinateur)
  - Prof. Theo D'Hondt, Programming Technology Lab, VUB (Co-directeur)
  - Prof. Tom Mens, Université de Mons (Co-directeur)
  - Prof. Dirk Vermeir, Dept. of Computer Science, VUB (Examinateur)
  - Prof. Bernard Manderick, Artificial Lab, VUB (Examinateur)

C.4 Comités de programme et relecture d'articles

2013

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

- Membre du comité de programme du IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR) 21-26 octobre 2013, Linköping, Suède.
- Relecteur pour la revue Technique et Science Informatique (TSI), Numéro special doctorants du GdR GPL, 2013.
- Membre du comité de programme de la IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Allemagne, 6–10 Mai 2013.

2012

- Membre du comité de programme du IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), College Station, Texas, USA, 5-8 Novembre 2012.
- Relecteur pour la revue Science of Computer Programming (SCP), Elsevier Journal.

2011

- Relecteur pour IEEE Transactions on Systems, Man, and Cybernetics- Part B, parution en 2012
- Relecteur pour Journal of Software Engineering for Robotics, parution en 2012
- Relecteur pour la Revue de l'Intelligence Artificielle, Numéro spécial JFSMA, parution en 2012
- Membre du comité de programme de la conférence Smalltalks 2011. Universidad Nacional de Quilmes, Buenos Aires - Argentine. 3-5 novembre 2011.

2010

- Relecture pour la revue internationale ACM Transactions on Autonomous and Adaptive Systems (TAAS), parution en 2011.
- Membre du comité de programme des $18^{èmes}$ Journées Francophones des Systèmes Multi-Agents (JFSMA), 18-20 octobre 2010, Mahdia, Tunisie.
- Membre du comité de programme de la $16^{ème}$ conférence francophone sur les Langages et Modèles à Objets (LMO), 9-10 mars 2010, Pau.

2009

- Membre du comité de programme de la $2^{ème}$ édition du International Workshop on Agent Technology for Disaster Management (ATDM'09) en marge de la conférence internationale Principles of Practice in Multi-Agent Systems (PRIMA), 13 décembre, 2009, Nagoya, Japon.
- Membre du comité de programme des $17^{èmes}$ Journées Francophones sur les Systèmes Multi-Agents (JFSMA'09), 21-23 Octobre 2009, Lyon.
- Membre du comité de programme de la $17^{ème}$ édition de la conférence internationale ESUG, 31 août - 4 septembre 2009, Brest.

2008

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

- Membre du comité de sélection pour la revue l'Objet, numéro spécial "Composants, Services et Aspects : techniques et outils pour la vérification", volume 14, numéro 4 (paru en 2008).
- Membre du comité de programme de la International Conference on Creating, Connecting and Collaborating through Computing (C5), 14-16 janvier 2008, Poitiers.

2007

- Membre du comité de programme de la $15^{ème}$ édition de la conférence internationale ESUG, 25-31 août 2007, Lugano, Suisse.
- Relecteur pour la revue Technique et Science Informatiques (TSI), volume 27, numéro 9-10 (paru en 2008).
- Membre du comité de programme de la $3^{ème}$ Journée Francophone De La Programmation par Aspects (JFDLPA), 26 mars 2007, Toulouse.

2006

- Membre du comité de programme de la $5^{ème}$ édition de Conférence Francophone autour des Composants Logiciels (Journées Composants), 4-6 octobre 2006, Perpignan, France.
- Membre du comité de programme de la $14^{ème}$ édition de la conférence internationale ESUG, 4-8 sept 2006, Prague, République Tchèque.
- Membre du comité de programme de la $5^{ème}$ édition de l'atelier Objets, Composants et Modèles dans l'ingénierie des Systèmes d'Information (OCM-SI), 31 mai 2006, Hammamet, Tunisie, en marge de la conférence INFORSID (INFormatique des ORganisations et Systèmes d'Information et de Décision).
- Président du comité de programme de la $2^{ème}$ édition de la Journée Multi-Agents, Aspects et Composants (JMAC), 21 Mars 2006, Nîmes, en marge de la conférence LMO (Langages et Modèles à Objet).

2005

- Membre du comité de sélection pour la revue l'Objet, numéro spécial "Composants et systèmes multi-agents", volume 12, numéro 4 (paru en 2006).
- Membre du comité de programme de la $2^{ème}$ Journée Francophone De La Programmation par Aspects (JFDLPA), 15 septembre 2005, Lille.
- Membre du comité de programme de la $13^{ème}$ édition de la conférence internationale ESUG, 16-20 août 2005, Bruxelles, Belgique.
- Membre du comité de programme de la $4^{ème}$ édition de Conférence Francophone autour des Composants Logiciels (Journées Composants), du 6 au 8 Avril 2005 à Le Croisic, Presqu'île de Guérande.
- Membre du comité de programme de la $12^{ème}$ édition des journées de Rochebrune, Rencontres interdisciplinaires sur les systèmes complexes naturels et artificiels sur le thème : Réflexivité et auto-référence dans les systèmes complexes, 24-28 Janvier 2005, Rochebrune, Megève.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

2004

- Co-président du comité de sélection du Elsevier International Journal on Computer Languages, Systems and Structures, Special Issue on Smalltalk, volume 31, numéros 3-4 (paru en octobre 2005).
- Membre du comité de programme de la $1^{ère}$ édition de la Journée Multi-Agents, Aspects et Composants (JMAC) le 23 novembre 2004, à Paris, en marge de la conférence JFSMA (Journées Francophones des Systèmes Multi-Agents).
- Membre du comité de programme de la Journée Francophone De La Programmation par Aspects (JFDLPA), 14 septembre 2004, Paris.
- Président du comité de programme de la $12^{ème}$ édition de la conférence internationale ESUG, 6-10 septembre 2004, Köthen, Allemagne.
- Membre du comité de sélection du journal "Science of Computer Programming" (Elsevier), Volume 56, Numbers 1-2 (paru en avril 2005). Special issue "New software composition concepts".
- relecteur pour le journal "Science and Computer Programming" (Elsevier), Volume 54, Issue 1 (paru en janvier 2005). Special issue "Principles and Practice of Prgramming in Java (PPPJ 2003)".

2003

- relecteur pour le journal "Science and Computer Programming" (Elsevier), Volume 54, Issue 1 (paru en janvier 2005). Special issue "Principles and Practice of Prgramming in Java (PPPJ 2003)".
- Membre du comité de programme de la $11^{ème}$ édition de la conférence internationale ESUG, 23-29 août 2003, Bled, Slovénie.
- Membre du comité de programme du workshop international *Software Composition* à la conférence ETAPS 2003.

2002 Membre du comité de programme du workshop international *Software Composition* à la conférence ETAPS 2002.

2001 Relecteur pour la conférence internationale European Conference on Object-Oriented Programming (ECOOP), 18-22 juin 2001, Budapest, Hongrie.

2000

- Relecteur pour la conférence internationale European Conference on Object-Oriented Programming (ECOOP) 12-16 juin 2000, Sophia Antipolis et Cannes.
- Relecteur pour la conférence française Langages et Modèles à Objet (LMO), 2000, Montréal, Québec.
- Relecteur pour la Revue des Calculateurs Parallèles. Numéro spécial "Evolutions des plates-formes orientées objets répartis" 2000.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

© 2014 Tous droits réservés.

doc.univ-lille1.fr

D. Projets financés et collaborations

D.1 Projet CAIRE (2013-2015) : Cartographie Robotique

Le projet CAIRE financé pendant 24 mois par la Région Nord Pas-de-Calais regroupe différents partenaires : l'Ecole des Mines de Douai, l'IEMN Lille et l'INRIA Lille Nord Europe. L'ambition de notre projet est la mise au point de nouvelles techniques de développement de logiciels dédiées à la robotique. En effet, la multiplication de robots impliquera inévitablement un besoin croissant pour développer rapidement et efficacement des logiciels permettant de les utiliser pleinement. Les méthodes de développement actuellement utilisées en robotique souffrent encore d'un manque d'abstraction et de souplesse et imposent l'utilisation de langages et outils de bas niveau. Par exemple, le développement d'une application robotique s'effectue sur une machine de développement (un PC). Puis, l'application est déployée et testée sur le robot cible. Les corrections nécessitent de revenir à la machine de développement d'où une lourdeur qui complique et ralentit le développement. Le recours à la simulation permet de remédier partiellement à ce problème. Toutefois les simulateurs sont encore incomplets et souvent non réalistes notamment pour ce qui est des ressources du robot (capteurs, actionneurs). Des travaux ont d'ailleurs essayé de pallier ces défauts en introduisant la possibilité de connecter le simulateur avec certaines parties du robot (*Hardware in the loop*). Néanmoins, la méthode de test la plus pertinente reste encore de déployer l'application à tester sur le robot. Toutefois, si une erreur survient durant l'exécution, les méthodes et outils permettant de la détecter et de l'analyser son encore très limités. Une méthode basique et inefficace consiste à faire de l'analyse de fichiers logs *post mortem*, c'est-à-dire après l'exécution du programme. Une méthode plus avancée consiste à utiliser un débogueur à distance qui permet d'analyser le programme en cours d'exécution sur le robot. Cependant, les débogueurs à distance (comme GDB ou ceux basés sur l'infrastructure JPDA pour les programmes Java) sont encore limités puisqu'ils imposent une infrastructure particulière sur le robot, nécessitent dans certains cas de redéployer l'application et enfin ne permettent pas la modification dynamique du code pour corriger directement certaines erreurs. Ainsi, les pratiques actuelles rendent le processus de développement d'applications robotiques plus rigide et plus long que pour une application informatique « standard ».

Le caractère novateur de notre projet réside dans l'utilisation de langages dynamiques et réflexifs (LDR) qui sont de plus en plus utilisés comme en témoigne le nombre croissant de langages dynamiques s'exécutant au dessus de la machine virtuelle Java (Groovy, JPython, JRuby, ...). Les LDR apporteront plus de flexibilité et d'agilité dans le développement logiciel pour

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

la robotique. Cette approche se démarque des recherches sur la fiabilité à travers des méthodes formelles ou de la preuve de programme. Notre proposition repose sur les succès des méthodes agiles dans l'industrie du logiciel. Ces méthodes pragmatiques nous semblent plus appropriées en termes de coût et de temps pour le développement d'applications robotiques non-critiques.

Notre objectif est de fournir une infrastructure flexible pour le développement agile dirigé par les tests (Test-Driven Development, TDD) d'applications robotiques. L'objectif de ce projet est de raccourcir le cycle développement traditionnel pour le rendre plus adapté au TDD. Les différentes tâches liées au développement (compilation, débogage, test, analyse de performances, ...) pourront s'effectuer soit sur la machine de développement soit sur le robot en fonction de ses capacités (CPU, mémoire, ...) et des besoins du développeur. Plus précisément, nous souhaitons permettre :

- le développement et le déploiement incrémental du code sur le robot de manière transparente afin que le développeur puisse travailler directement sur le robot tout en profitant de la puissance de sa machine pour l'environnement et les outils de développement ;
- de pouvoir analyser in vivo un programme en cours d'exécution sur un robot ;
- la modification du code et des données du programme à chaud c'est-à-dire pendant son exécution afin de faciliter la mise au point des programmes ;
- l'optimisation du programme en fonction des ressources réelles du robot ;
- l'intégration continue. Il s'agit de regrouper fréquemment et automatiquement toutes les briques logicielles et les déployer sur le robot afin d'en vérifier leur cohérence et leur compatibilité avec les ressources et les capacités matérielles du robot.

Les résultats seront validés dans le cadre d'expérimentation grandeur nature avec un robot de taille humaine doté de roues. Un tel robot sera chargé d'une mission de cartographie d'un espace inconnu. Le robot sur roues devra se déplacer dans cet espace et de le couvrir totalement. Cette mission devra être réalisée en toute autonomie, donc sans assistance humaine. En terme d'infrastructure nous utiliserons le *middleware* ROS, standard *de facto* porté par la Open Source Robotic Foundation (OSRF). L'OSRF est issue de la société Willow Garage et est financée notamment par le DARPA dans le cadre du Humanoid Challenge prévue pour 2014.

D.2 Projet ROBOSHOP (2012-2013) : Robots en galerie marchande

Le projet ROBOSHOP s'intéresse à l'usage de robots *mobiles* et *autonomes* [Par08] dans le cadre d'applications liées au commerce. Par opposition aux robots industriels, un robot *mobile* évolue dans un environnement ouvert, partiellement connu et continuellement changeant. Il est donc quasiment impossible de prédire toutes les situations auxquelles le robot sera confronté et encore moins leurs successions du fait de l'explosion combinatoire.

Le qualificatif *autonome* exclut quant à lui les robots télé-opérés. Dans ROBOSHOP, le robot *autonome* sera doté d'un logiciel qui lui permettra de prendre des décisions sans intervention humaine. Il doit donc mener à bien, seul, les missions qui lui incombent et ce, en toute sécurité pour les biens et les hommes, tout en préservant sa propre intégrité.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

Le projet ROBOSHOP a deux dimensions, l'une scientifique et l'autre technologique. En termes de travaux scientifique, le projet vise à définir une architecture de contrôle d'un robot mobile spécifiquement conçue pour la construction d'applications répondant à des scénarios d'usage dans le cadre du commerce. Cette architecture de contrôle devra :

– être indépendante de toute application ou scénario d'usage ;
– être extensible afin de permettre la construction d'applications répondant aux usages futurs.

Nous proposons d'utiliser le paradigme de *composant logiciel* [SGM02, LW05, CSVC10] afin de prendre en compte la complexité inhérente aux architecture des robots [KS08, Bru07, Mal11]. Le logiciel de contrôle de robot que nous proposons de développer sera constitué de deux sortes de composants logiciels. Pour la partie réactive de l'architecture, nous utiliserons des composant *enrobants*. Il s'agit de composants qui viendraient encapsuler des éléments de code qui respectent les contraintes de temps réel. Ces composants fournissent des fonctionnalités qui assurent la sécurité des humains et des biens situés dans le voisinage du robot et qui préservent l'intégrité du robot. La deuxième sorte de composants correspond à des composants développés totalement dans notre langage Scl [FDH08, FBDH12]. Ils seront utilisés dans la partie délibérative de l'architecture moins contrainte temporellement.

Sur le plan technologique, nous travaillons sur la définition et la mise en place de l'infrastructure matérielle et logicielle du robot. Cette tâche vise donc à fournir un robot complet et prêt à l'emploi disposant des fonctions basiques telles que : localisation, navigation dans un environnement contrôlé, interaction avec l'utilisateur (synthèse vocale, tablette tactile). Il s'agit d'une part d'adapter matériellement l'un des robots disponible à l'ÉCOLE DES MINES DE DOUAI (ajout de capteurs, ...) et d'autre part de rendre accessible à l'architecture de contrôle du robot toutes ces fonctionnalités de base via le middleware ROS (Robot Operating System).

L'un des points difficiles auquel nous sommes confronté est la localisation *indoor*. Afin que le robot puisse réaliser les différentes missions qui lui incombent, il est nécessaire qu'il puisse se localiser dans son environnement. Il s'agit ici de doter le robot de capacités lui permettant de déterminer de manière autonome (donc sans assistance externe) sa position sur la carte d'un environnement fermé (galerie marchande dans notre cas). Il existe différentes solutions de localisation reposant pour la plupart sur des algorithmes de SLAM (Simultaneous localization and mapping) [Sta09]. Ces algorithmes permettent d'intégrer les informations provenant de différents capteurs pour construire et maintenir une carte de l'environnement du robot. Suivant les capteurs choisis, la carte sera plus ou moins précise et robuste aux perturbations. Les capteurs peuvent être installés dans l'environnement (instrumentation de l'environnement) comme c'est le cas dans l'expérimentation de Kulyukin et *al.* [KGN05] qui ont utilisé des puces RFID. La solution la moins contraignante, mais la plus difficile à mettre en œuvre est d'utiliser exclusivement des capteurs embarqués sur le robot : laser, sonar, caméra laser, etc. Le travail consiste donc à choisir et combiner efficacement les capteurs permettant une bonne localisation et la possibilité de facilement déployer le robot dans un nouvel environnement (un nouvel espace de vente dans notre cas).

© 2014 Tous droits réservés.
doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

D.3 Collaboration avec l'équipe RMoD de l'INRIA Lille Nord-Europe (2009-actuellement)

La proximité thématique de nos travaux à l'ÉCOLE DES MINES DE DOUAI avec ceux de l'équipe RMoD dirigée par Stéphane Ducasse a naturellement débouché sur une collaboration. Celle-ci a été formalisée sous la forme d'une convention entre l'ÉCOLE DES MINES DE DOUAI et le centre lillois de l'INRIA. En effet, les interactions et échanges ont rapidement pris de l'ampleur en se concrétisant par différentes d'actions. Nous listons ici les plus significatives.

- Co-encadrement de thèses. Actuellement, 2 doctorants travaillent dans ce cadre, à savoir Nick Papoulias [Pap13] (soutenance prévue en 2014) et Guillermo Polito (soutenance prévue en 2015). Par ailleurs, nous avons déjà à notre actif 1 doctorant qui a soutenu en 2012. Il s'agit de Mariano Martinez-Peck [MP12].
- Développement de la plate-forme Pharo [1] dans le cadre d'un consortium international. Pharo est un logiciel libre (licence MIT) piloté par l'équipe RMod de l'INRIA Lille Nord Europe. Il dispose d'un fort rayonnement tant au niveau national qu'au niveau international. Au 17 mai 2013, il avait été téléchargé plus de 252185 fois ce qui le place en 2^{eme} position sur les 4241 projets hébergés par l'INRIA. Outre notre participation avec des correctifs, nous travaillons à l'enrichissement de la plate-forme Pharo et des outils associés. Nos plus importantes contributions relèvent du domaine des applications réparties. Ainsi, nous assurons la maintenance et l'évolution du middleware à objets rST (Remote SmallTalk) [2] qui sert de base à UbiquiTalk [3]. Par ailleurs, nous avons entrepris une ré-écriture complète de la bibliothèque réseau avec une approche agile (eXtreme programming). La nouvelle bibliothèque baptisée Ocean [4] a une conception basée sur le *design pattern Bridge* afin de permettre différentes implémentations. Nous avons montré qu'il est possible d'utiliser des bibliothèques systèmes et d'alléger ainsi la machine virtuelle [BF10].
- Animation de la communauté internationale ESUG (European Smalltalk Users Group). Depuis sa création en 1991, ESUG œuvre pour favoriser les échanges autour du langage Smalltalk. Ainsi, elle organise tous les ans depuis 1993 une conférence internationale. Environ la moitié des participants sont du monde industriel, alors que l'autre moitié est d'origine académique. Nous nous sommes chargés en 2002 de l'organisation de la 10*ième* édition de cette conférence. Puis, nous avons rejoint en 2003 le bureau de l'association ESUG qui anime la communauté Smalltalk avec différentes actions [5] en plus de l'organisation de la conférence annuelle.

1. http://www.pharo-project.org
2. http://vst.mines-douai.fr/research/11#rST
3. http://vst.mines-douai.fr/UbiquiTalk
4. http://vst.mines-douai.fr/research/11#Ocean
5. http://www.esug.org/wiki/pier/Promotion

© 2014 Tous droits réservés.

doc.univ-lille1.fr

D.4 Collaboration avec les équipes MAD du GREYC de Caen et MSI de l'UMMISCO de Hanoï, Vietnam (2003-2010)

Cette collaboration ainsi que celle avec l'équipe SMA de l'Ecole des Mines de St Etienne (voir section D.8) ont été à l'origine de mon ouverture vers la communauté des Systèmes Multi-Agents. Par ailleurs, l'interaction avec le GREYC [6] de Université de Caen, a été à l'origine de mes travaux sur la robotique mobile et plus particulièrement la problématique des systèmes multi-robots.

En 2003, nous avons commencé à collaborer avec l'équipe MAD (Modèles, Agents, Décision) du laboratoire GREYC, via Serge Stinckwich rencontré dans le cadre de la communauté Smalltalk. Il s'agissait de travailler à l'interface de nos domaines respectifs et d'identifier des niches thématiques à explorer. Ainsi, nous avons commencé par étudier l'utilisation de la programmation par aspects pour le développement des systèmes multi-agents. Ce travail a été réalisé dans le cadre du stage Master de Romain Robbes [Rob03].

Nous avons poursuivi notre interaction en élargissant à la problématique générale d'utilisation des techniques de génie logiciel pour le développement d'architectures de contrôle d'agents physiques : les robots mobiles et autonomes. Le stage de Master de Ludovic Guégan [Gué05] a abordé la question de la "paramétrisation" d'architectures hybrides de type d'InteRRap [Mül96]. Le stage de Master de Rémy Mouëza [Mou06] traite quant a lui de l'utilisation des composants logiciels Maleva [MB01] orientés flux de données pour développer des architectures réactives à subsumption [Bro85].

En 2007, le cercle de partenaires s'est élargi pour intégrer l'équipe MSI (Modélisation et Simulation Informatique) située à Hanoï (Vietnam) et qui fait partie du laboratoire laboratoire UMMISCO [7]. UMMISCO est sous la double tutelle de l'IRD (Institut de Recherche pour le Développement) et de l'UPMC (Université de Pierre et Marie Curie à Paris). C'est dans le cadre de cette collaboration tripartite que s'est déroulée la thèse de Van Tuan Le [Le10]. Le travail de Tuan a porté sur la coopération des systèmes multi-robots dans les applications de sauvetage de type *Urban Search And Rescue* (USAR).

D.5 Projet CCure (2008)

Financement privé (entreprise)

Niveau de responsabilité : Responsable du projet au niveau de l'EMD.

Résumé du projet CCure est un projet d'étude et de transfert de technologie réalisé pour le compte de M Philippe Cellé créateur d'une *startup* dans le cadre de l'incubateur MITI [8] Nord Pas de Calais. L'objet du projet étant confidentiel, nous ne présentons ici que les technologies

6. Groupe de REcherche en Informatique, Image, Automatique et Instrumentation de Caen
7. Unité de Modélisation Mathématique et Informatique des Systèmes COmplexes
8. http://www.miti.fr

© 2014 Tous droits réservés.

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

utilisées et transférées. Il s'agit en l'occurrence de mettre en œuvre la réflexion combinée à une démarche de développement agile du type *Extreme Programming* (XP) [Bec01] dans le cadre du langage Smalltalk. Si la réflexion constitue l'un de nos thèmes de recherche (voir le chapitre B), XP est quant à elle une expertise que nous avons développée notamment lors de la réalisation de nos prototypes de recherche.

Dans le cadre du projet CCure, nous avons, dans un premier temps, défini et priorisé avec le client des scénarii qui décrivent différentes facettes du logiciel à produire. Puis, nous avons réalisé chaque scénario suivant les quatre temps du *Test-Driven Development* (TDD) :

- Matérialiser le scénario sous forme de tests exécutables et automatisables en utilisant un framework de la famille xUnit.
- Développer la portion de code qui fait réussir les tests. C'est à ce niveau que nous avons mis en œuvre la réflexion.
- Réviser la structure du code notamment avec des outils de *refactoring* pour améliorer la conception.
- Optimiser le code pour garantir les performance attendues.

Au cours de la phase de réalisation, nous avons effectué des modifications de différentes natures demandées par le client. Elles consistaient à introduire de nouveaux scénarii, de réviser ceux déjà définis ou leur priorités. Le choix d'adopter la démarche agile XP a été fait justement pour permettre de tels modifications qui peuvent arriver tardivement dans le projet. Ces changements sont d'autant plus probables dans les projets avec un contexte mouvant ou un contour partiellement flou comme c'était le cas pour le projet CCure porté - rappelons le - par une *startup*. Ainsi, le processus de développement itératif a permis au client de voir rapidement une solution, certes partielle, mais totalement opérationnelle. Il a pu ainsi cerner les implications de ses choix, notamment pour les parties non-clairement spécifiées. Par ailleurs, les scénarii qui n'étaient pas encore réalisés pouvaient être modifiés ou supprimés, sans impact pour les autres. Enfin, le recours aux tests exécutables et automatisables a permis de rapidement identifier l'impact des modifications de certains scénarii sur le reste du projet. Ainsi, les tests ont constitués un outil de communication avec le client qui permet de montrer de manière concrète et "palpable" le coût des changements demandés. D'autre part, grâce aux tests, nous avons pu effectuer toutes les corrections nécessaires pour garantir la non-régression du projet.

Actuellement le projet CCure est en cours de finalisation. Notre intervention a été très appréciée par l'entreprise. En effet, la *startup* a adopté l'approche que nous avons proposée pour bâtir la solution finale.

D.6 Projet ODICE- Open DIgital ConcretE (2007-2009)

Financement Institut Carnot MINES

Niveau de responsabilité : Co-responsable scientifique du projet.

Partenaires

- Dépt. Informatique et Automatique, Ecole des Mines de Douai

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

– Dépt. Génie Civil et Environmental, Ecole des Mines de Douai

Résumé du projet La simulation numérique est de plus en plus employée pour tester les propriétés des matériaux (*material-by-design*). Les codes aux éléments finis ont permis de faire de grands progrès dans ce domaine, mais la logique d'homogénéisation des propriétés au niveau de chaque maille associée à des tailles de maille assez grandes, conduit à ne pas représenter finement la microstructure des matériaux qui pourtant conduit aux propriétés macroscopiques. Ainsi, des codes spécifiquement dédiés à la création de microstructures numériques sur différentes échelles en fonction de la complexité du matériau ont récemment été développés : par exemple VCCTL (*Virtual Cement & Concrete Testing Laboratory*) pour le béton. Cependant de tels codes ne disposent pas de toutes les fonctionnalités des codes aux éléments finis et seules certaines propriétés sont calculées. Pour pallier ce problème, il est nécessaire d'utiliser conjointement différents logiciels réalisant les divers calculs requis. Cette solution requiert des opérations de lancement des différents traitement et de transfert de données d'autant plus fastidieuses que les calculs qui peuvent très longs (plusieurs jours) peuvent être effectués sur différentes machines.

Le projet ODICE (*Open Digital ConcretE*) propose d'étudier l'automatisation de ces opérations dans le cadre d'une simulation de la dégradation du béton [Bou09, GB09]. Il s'agit donc de proposer une solution pour composer des logiciels potentiellement répartis sur différentes machines et donc gérer les flux de contrôle, ainsi que les flux de données entre eux. Pour cela, nous avons adopté une approche à base de composants. Chaque code de calcul est piloté par un composant qui l'alimente en données en entrée, déclenche l'exécution et récupère les résultats en sortie. Ainsi, les logiciels de calcul développés pour une utilisation individuelle (*standalone*) sont encapsulés dans des composants. Cela nous permet de les composer directement entre eux ou avec de nouveaux composants que nous avons introduits pour réaliser des calculs intermédiaires et adapter le format des données échangées. Le résultat est un assemblage qui reflète les flux de données et de contrôle.

Cette approche a été expérimentée [GABD10] pour réaliser la simulation de la dégradation physico-chimique et mécanique du béton dans un environnement donné. Cette simulation a mis en œuvre différents logiciels qui sont intervenus successivement dans plusieurs itérations de calcul. Certains travaillent avec une représentation du matériaux sous forme de micro-structure représentée par une matrice 3D de voxels. D'autres logiciels utilisent la composition du béton décrite sous la forme de pourcentages pour chaque composé chimique. Afin de supporter la distribution et de la découverte de composants, notre prototype a été développé sur la base middleware UbiquiTalk que nous avons initialement réalisé dans le cadre du projet MOSAÏQUES (voir section D.7). Nous avons étendu UbiquiTalk avec une interface web permettant ainsi à l'opérateur de piloter la simulation depuis n'importe quelle machine munie d'un navigateur web et d'une connexion Internet.

D.7 Projet MOSAÏQUES (2004-2007)

Financement Contrat Plan Etat-Région (CPER), programme Technologies Avancées pour la Communication soutenu par un financement FEDER (Région Nord-Pas-de-Calais et l'Europe).

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

Niveau de responsabilité : Responsable du projet au niveau de l'EMD.

Partenaires

- GOAL, LIFL, Université des Sciences et Technologies de Lille,
- RD2P, LIFL, Université des Sciences et Technologies de Lille,
- STC, LIFL, Université des Sciences et Technologies de Lille,
- SMAC, LIFL, Université des Sciences et Technologies de Lille,
- LAMIH, Université de Valenciennes,
- DIA, Ecole des Mines de Douai,
- NOCE, TRIGONE, Université des Sciences et Technologies de Lille,
- LEOST, INRETS, Institut National de Recherche sur les Transports et leur Sécurité, Lille

Résumé du projet L'objectif du projet MOSAÏQUES (MOdèles et InfraStructures pour Applications ubIQUitairES) est de définir un cadre de développement (méthodologie et outillage) pour la définition d'applications fonctionnant dans un environnement ubiquitaire. Ce projet s'intéresse à la notion d'adaptabilité dynamique dans les applications et les infrastructures logicielles pour les applications ubiquitaires selon les axes suivants :

1. La définition d'un support méthodologique pour la construction d'applications ubiquitaires. Ce premier axe avait pour objectif l'étude des méthodologies pour la conception d'applications fonctionnant dans un environnement de type ubiquitaire.
2. La réalisation d'un support d'exécution avec des propriétés d'adaptabilité dans un environnement ubiquitaire qui fournit les moyens nécessaires pour qu'un assemblage d'éléments logiciels puisse être adapté dynamiquement en fonction de l'environnement d'accueil et par conséquent de l'application qui le contient.
3. La validation de l'approche passe par la définition d'un ensemble de méthodes et d'outils permettant de prendre en compte l'adaptabilité que ce soit au niveau statique ou dynamique. Cet axe se focalisera sur la définition et la vérification d'une relation entre services offerts, ressources disponibles et qualité de service requise.
4. La démonstration de la faisabilité de l'approche par la réalisation d'applications dans des domaines particuliers comme les transports ou encore les e-services ubiquitaires pour la e-Santé, la e-formation et le e-Commerce.

Notre contribution au projet s'est notamment située au niveau des axes 2 et 4. Au niveau de l'axe 2, nous avons proposé dans le cadre de la thèse de Guillaume Grondin [Gro08] le modèle MaDcAr (Model for automatic and Dynamic component Assembly reconfiguration). Il s'agit d'une solution générique (i.e. indépendante du modèle de composants) permettant de réviser un assemblage de composants. En plus du remplacement d'un ou plusieurs composants d'un assemblage quelconque, MaDcAr permet de remplacer, à l'exécution l'architecture de l'assemblage tout en préservant l'état. Le concepteur de l'application spécifie l'ensemble des architectures valides. Un moteur d'assemblage basé sur un solveur de contraintes permet de retrouver dans une bibliothèque les composants adéquats, de les assembler et de les initialiser de manière à maintenir l'application dans un état cohérent.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

Au niveau de l'axe 4, nous avons réalisé le middleware UbiquiTalk [9]. Il s'agit d'une plate-forme pair à pair (P2P) qui permet la découverte automatique. Chaque pair dans le réseau peut offrir des services dont la liste est également obtenue dynamiquement. L'utilisation d'un service peut requérir le déploiement à la volée sur la machine cliente d'un fragment de code. C'est notamment le cas pour les services destinés à un utilisateur humain. L'interface graphique d'utilisation du service est déployée sur la machine de l'utilisateur lors de la première utilisation. Elle est conservée provisoirement dans un cache côté client pour améliorer les performances. Les services sont versionés. Ainsi, le client télécharge à nouveau le code requis, si la version du service a changé.

UbiquiTalk a atteint un niveau de stabilité et de maturité suffisamment élevé qui nous a permis d'effectuer plusieurs démonstrations dans différents cadres. Nous avons en particulier développé des exemples de services liés au e-commerce utilisables sur des PDA. Nous les avons présentés notamment dans le cadre du pôle de compétitivité "Industries du commerce" [10] (PICOM).

D.8 Projet MAAC (2003-2006)

Financement GEM (Groupement des Ecoles des Mines).

Niveau de responsabilité : Responsable du projet au niveau de l'EMD.

Partenaires

- MOO, Ecole des Mines d'Alès ;
- DIA, Ecole des Mines de Douai ;
- OBASCO, Ecole des Mines de Nantes ;
- SIMMO/SMA, Ecole des Mines de St Etienne.

Résumé du projet L'objectif du projet MAAC (Modèles, Agents, Aspects et Composants) était de définir et d'implémenter un modèle de composant intégrant des caractéristiques de la programmation par aspects pour le développement d'infrastructures logicielles supports à des applications multi-agents. Ce projet a eu différentes retombées intéressantes. Je ne cite ici que celles où mon équipe était directement impliquée.

Sur un plan thématique, notre participation au projet MAAC a contribué à notre ouverture vers la thématique des Systèmes Multi-Agents comme nous l'avons déjà indiqué. Par ailleurs, le projet MAAC a également servi de support à l'animation de la communauté scientifique. Ainsi, nous avons co-organisé de l'atelier Multi-Agent et Composant (JMAC). La première édition [11] a été tenue en 2004, en marge des JFSMA (Journées Francophones des Systèmes Multi-Agents). La seconde édition [12] a eu lieu en 2006, en marge de la conférence LMO (Langages et Modèles

9. http ://vst.mines-douai.fr/UbiquiTalk
10. http://www.picom.fr/
11. http://vst.mines-douai.fr/MAAC/3
12. http://vst.mines-douai.fr/MAAC/7

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

à objets). Enfin, nous avons participé à la promotion et à la relecture d'articles pour le numéro spécial "Composants et Systèmes Multi-Agents" de la revue L'Objet [Boi06] initié dans le cadre du projet.

En ce qui concerne la production scientifique, nous avons abordé dans le projet MAAC deux problématiques intéressantes. La première concerne de fusion de la programmation par aspects et par composants, objet de la thèse d'Houssam Fakih [Fak06] sur le sujet. La second concerne l'étude de l'utilisation de la programmation par aspects pour le développement de systèmes multi-agents. C'est l'objet du stage Master de Romain Robbes [Rob03] co-encadré avec Serge Stinckwich de l'équipe MAD du GREYC (voir section D.4).

D.9 Références bibliographiques du chapitre

[Bec01] Kent Beck. *Extreme Programming Explained*. Addison-Wesley, 2001.

[BF10] Noury Bouraqadi and Luc Fabresse. Towards small portable virtual machines. In Marcus Denker and Gabriela Arévalo, editors, *Proceedings of the Smalltalks 2010 Conference*, Concepcion del Uruguay, Argentina, 2010.

[Boi06] Olivier Boissier, editor. *L'Objet, Numéro Spécial : Composants et Systèmes Multi-Agents*, volume 12. Lavoisier, 2006.

[Bou09] Noury Bouraqadi. Rapport d'activité 2008 du projet "odice (open digital concrete) : Plate-forme ouverte pour réaliser des expériences virtuelles sur les matériaux". Technical Report 07DD46GCE+IA1 CALCUL REPARTI, Institut Carnot MINES, February 2009.

[Bro85] Rodeney Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1) :14–23, March 1985.

[Bru07] Davide Brugali, editor. *Software Engineering for Experimental Robotics*, volume 30 of *STAR (Springer Tracts in Advanced Robotics)*. Springer, 2007.

[CSVC10] Ivica Crnković ;, Severine Sentilles, Aneta Vulgarakis, and Michel R.V. Chaudron. A classification framework for software component models. *IEEE Transactions on Software Engineering*, 99(PrePrints), 2010.

[Fak06] Houssam Fakih. *L'integration des fonctionnalités transversales dans les composants logiciels en utilisant la programmation par aspects*. PhD thesis, Université de Lille 1, December 2006. Thèse co-encadrée et dirigée par Laurence Duchien (LIFL, Université de Lille 1).

[FBDH12] Luc Fabresse, Noury Bouraqadi, Christophe Dony, and Marianne Huchard. A language to bridge the gap between component-based design and implementation. *Computer Languages, Systems and Structures*, 2012.

[FDH08] Luc Fabresse, Christophe Dony, and Marianne Huchard. Foundations of a Simple and Unified Component-Oriented Language. *Journal of Computer Languages, Systems & Structures*, 34/2-3(2-3) :130–149, 2008.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[GABD10] Guillaume Grondin, George Aouad, Noury Bouraqadi, and Denis Damidot. Odice : an open distributed platform for computational simulation applied to concrete, concrete modeling. In *Proceedings of ConMod 2010 Symposium on Concrete Modelling*, Lausanne, Switzerland, June 2010. RILEM Publications.

[GB09] Guillaume Grondin and Noury Bouraqadi. Rapport final du projet "odice (open digital concrete) : Plate-forme ouverte pour réaliser des expériences virtuelles sur les matériaux". Technical Report 07DD46GCE+IA1 CALCUL REPARTI, Institut Carnot MINES, December 2009.

[Gro08] Guillaume Grondin. *Un modèle d'agents auto-adaptables à base de composants*. PhD thesis, Ecole des Mines de Saint Etienne, November 2008. Thèse co-encadrée avec Laurent Vercouter (Ecole des Mines de St Etienne) sous la direction d'Olivier Boissier (Ecole des Mines de St Etienne).

[Gué05] Ludovic Guégan. Hybridation paramètrable d'agents pour systèmes embarqués. Master's thesis, Université de Caen, 2005.

[KGN05] V. Kulyukin, C. Gharpure, and J. Nicholson. RoboCart : Toward Robot-Assisted Navigation of Grocery Stores by the Visually Impaired. In *In IROS*, pages 2845–2850, 2005.

[KS08] David Kortenkamp and Reid Simmons. *Handbook of Robotics*, chapter 8- Robotic Systems Architectures and Programming, pages 187–206. Springer, 2008.

[Le10] Van Tuan Le. *Coopération dans les systèmes multi-robots : Contribution au maintien de la connectivité et à l'allocation dynamique de rôles*. PhD thesis, Université de Caen, October 2010. Thèse co-encadrée avec Serge Stinckwich (GREYC Univ. de Caen/MSI Hanoï) et Victor Moraru (MSI Hanoï) sous la direction de François Bourdon ((GREYC Univ. de Caen).

[LW05] Kung-Kiu Lau and Zheng Wang. A taxonomy of software component models. In *EUROMICRO-SEAA*, pages 88–95. IEEE Computer Society, 2005.

[Mal11] J. Malenfant. De la robotique communicante aux « cyber-physical systems ». Exposé invité aux Journées Nationales de Recherche en Robotique (http://jnrr2011.irccyn.ec-nantes.fr/, 2011.

[MB01] Thomas Meurisse and Jean-Pierre Briot. Une approche à base de composants pour la conception d'agents. *Technique et science informatiques (TSI)*, 20(4) :583–602, 2001.

[Mou06] Rémy Mouëza. Architecture réactive à subsomption à l'aide des composants maleva. Master's thesis, Université de Caen, June 2006.

[MP12] Mariano Martinez-Peck. Remodularisation dynamique de logiciels pour systèmes embarqués. Thèse co-encadrée avec Luc Fabresse (Ecole des Mines de Douai) et Marcus Denker (INRIA Lille Nord Europe) sous la direction de Stéphane Ducasse (INRIA Lille Nord Europe), 2012.

[Mül96] J. P. Müller. *The Design of Intelligent Agents : A Layered Approach*. Number 1177 in Lecture Notes in Artificial Intelligence (LNAI). Springer-Verlag, 1996.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

[Pap13] Nick Papoulias. Refléxion et débogage à distance d'applications contraintes en ressources. Thèse co-encadrée avec Luc Fabresse (Ecole des Mines de Douai) et Marcus Denker (INRIA Lille Nord Europe) sous la direction de Stéphane Ducasse (INRIA Lille Nord Europe), décembre 2013.

[Par08] Lynne E. Parker. *Handbook of Robotics*, chapter 40. Multiple Mobile Robot Systems, pages 921–941. Springer, 2008.

[Rob03] Romain Robbes. Mise en oeuvre de la programmation par aspects dans le cadre des systèmes multi-agents. Master's thesis, Université de Caen, 2003.

[SGM02] C. Szyperski, D. Gruntz, and S. Murer. *Component Software : Beyond Object-Oriented Programming*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, second edition, 2002.

[Sta09] C. Stachniss. *Robotic Mapping and Exploration*. Springer Tracts in Advanced Robotics. Springer, 2009.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

E. Responsabilités pédagogiques (depuis 2000)

E.1 Contexte : La formation à l'ÉCOLE DES MINES DE DOUAI

L'ÉCOLE DES MINES DE DOUAI est un établissement public à caractère administratif (EPA) qui diplôme plus de 200 ingénieurs par an. La formation est suivie essentiellement par des élèves de niveau Bac+2 issus d'écoles préparatoires (*Maths Spé.*). Le cursus de 3 ans ponctués de stages [1] est structuré en *unités d'enseignement (UE)*. Chaque UE correspond a un volume d'environ 120 heures à l'emploi du temps et regroupe plusieurs *modules* d'enseignement qui s'inscrivent dans une même thématique (Par exemple : Mathématiques et Informatique). Un module d'enseignement est un ensemble cohérent de cours, de TD et de TP qui abordent un sujet spécifique dans une UE (Par exemple : Langage C).

La formation dispensée à l'ÉCOLE DES MINES DE DOUAI est qualifiée de généraliste. En effet, la part belle est accordée à des UE scientifiques et techniques de tronc commun et à des enseignements abordant le management et l'organisation de l'entreprise. Les diplômés se distinguent néanmoins par des colorations techniques et scientifiques différentes. En effet, une partie de la fin du cursus (750 heures environ) et les deux derniers stages se déroulent dans le cadre d'une *option*. Celle-ci correspond à un ensemble d'UE scientifiques et techniques d'une même discipline. L'EMD compte 8 options, adossées à 5 départements d'enseignement et de recherche :

- Environnement et Industrie (Dépt. Chimie et Environnement)
- Génie Civil (Dépt. Génie Civil et Environnemental)
- Génie Energétique (Dépt. Energétique Industrielle)
- Ingénierie Mécanique (Dépt. Technologie des Polymères et Composite & Ingénierie Mécanique)
- Ingénierie de la Qualité (DÉPT. INFORMATIQUE ET AUTOMATIQUE)
- Ingénierie des Systèmes d'Information et de Communication (DÉPT. INFORMATIQUE ET AUTOMATIQUE)
- Optimisation & Automatisation des Processus Industriels (DÉPT. INFORMATIQUE ET AUTOMATIQUE)
- Technologie des Polymères et Composite (Dépt. Technologie des Polymères et Composite & Ingénierie Mécanique)

1. 3 mois la première année, 4 mois la seconde et 5 en dernière année.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

L'enseignement de l'informatique dans lequel je suis impliqué est découpé en 2 parties. La première, d'environ 120 heures (Cours + TD + TP) est dédiée à l'initiation à l'informatique dans le cadre du tronc commun. Il s'agit d'un tour d'horizon des bases de l'algorithmique et de la programmation, des systèmes de gestion de bases de données, des réseaux et les systèmes d'exploitation. La seconde et plus importante partie (750 heures environ) est l'option ISIC : Ingénierie des Systèmes d'Information et de Communication, ouverte depuis septembre 1999. L'enseignement ISIC est structuré de manière à aborder les différentes missions auxquelles peuvent être confrontés les ingénieurs en entreprise. Ainsi, l'option ISIC est organisée en 4 UE complémentaires :

- Infrastructures Systèmes et Réseaux
- Méthodes de modélisation pour l'analyse et la conception
- Langages et outils de développement
- Organisation et Gestion des Systèmes d'Information

Les modules de chaque UE sont répartis sur les deux années de l'option. Cette répartition est guidée non-seulement par les dépendances entre modules et le niveau d'abstraction du contenu, mais également par les responsabilités prises par les élèves durant les stages successifs. Ainsi, la première année de l'option est essentiellement dédiée aux enseignements à dominante technologique. La seconde année fait quant à elle la part belle aux enseignements liés à des tâches de conception et de management.

E.2 Responsable pédagogique de l'option ISIC de l'ÉCOLE DES MINES DE DOUAI (2007-2011)

En 2007, j'ai pris la responsabilité pédagogique de l'option ISIC. J'ai assumé cette charge jusqu'à la fin de mon mandat en 2011. Le chef du DÉPT. INFORMATIQUE ET AUTOMATIQUE m'a proposé de continuer à exercer cette fonction. J'ai cependant décliné cette offre afin de dégager davantage de temps pour mon activité de recherche et pour préparer mon HDR. En effet, la responsabilité pédagogique d'une option d'enseignement de l'EMD est très prenante et laisse peu de temps aux autres activités.

E.2.1 Amélioration de l'attractivité de l'option

La charge liée à l'option a été d'autant plus importante qu'à mon arrivée, ISIC était devenue très peu attractive pour les élèves. La situation était même critique l'année de ma prise de fonction (avril 2007) comme le montre la figure figure E.1 En effet, pour l'année scolaire 2006-2007, seuls 6 élèves avaient rejoint ISIC (diplômés en 2008) ! Face à cette désaffection, nous avons entrepris un certain nombre d'actions -décrites plus bas- qui ont eu un impact positif. Ainsi, l'attractivité de l'option a augmenté comme en témoigne la progression du nombre de diplômés qui a rejoint la moyenne des premières années de vie de l'option. Le maximum historique de 25 diplômés sera dépassé en 2013 puisqu'en septembre 2011, un nombre record de 27 élèves a rejoint l'option.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

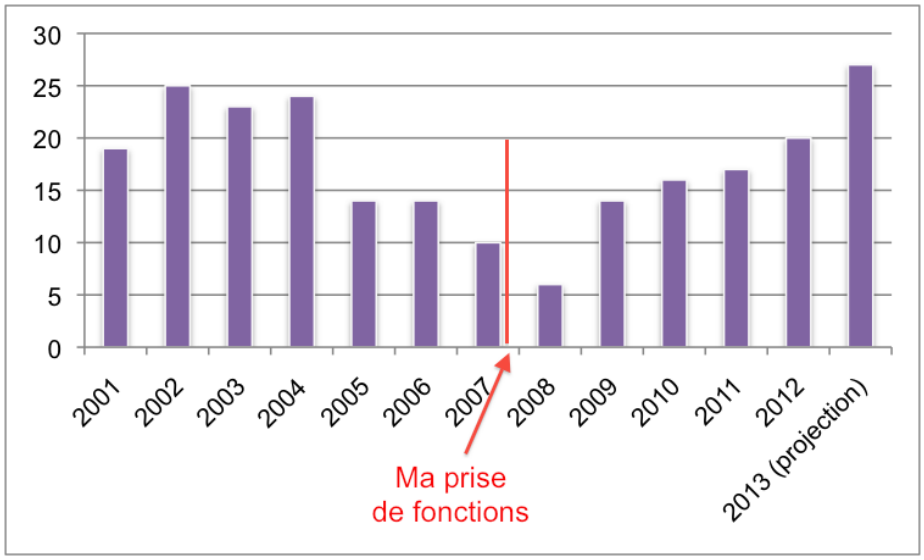

FIGURE E.1 – Evolution du nombre de diplômés d'ISIC

Ce renversement positif du flux d'élèves dans l'option est le fruit d'une série d'actions complémentaires menées depuis l'automne 2007. Je tiens à préciser que j'ai fait ce travail conjointement avec différentes personnes et en particulier M. Arnaud Doniec qui, a occupé depuis 2009 les fonctions de responsable pédagogique des enseignements informatiques du tronc commun.

Evolution de l'enseignement informatique du tronc commun. A ma prise de fonctions, la gestion de ces cours incombait au responsable d'ISIC. En effet, l'enseignement l'informatique au tronc commun était limité. Nous avons identifié la faiblesse du volume horaire (36 h seulement !) ainsi que la pauvreté et l'obsolescence du contenu comme les principales causes de la baisse d'attractivité de l'option. A notre demande, ce volume a plus que triplé de 2008 à 2011, puisqu'il est à passé à 120h. L'initiation à l'algorithmique et la programmation en Visual Basic a cédé la place à des cours d'initiation aussi bien sur la programmation (en Java) que sur les bases de données, les réseaux ou l'administration système. Notre objectif est de doter les élèves d'un bagage suffisamment riche afin qu'ils aient une vision plus large du métier d'informaticien et qu'ils puissent prétendre à des stages techniciens en informatique en entreprise.

Evolution de l'enseignement de l'option ISIC. Au delà des ajustements effectués chaque année pour prendre en compte les remarques et suggestions des élèves et des professeurs, nous avons opéré différentes évolutions du contenu de l'option. Certains de ces changements ont été initiés par la direction des études et concernent toutes les options de l'école. C'est le cas notamment de l'introduction des Projets de Découverte de la Recherche (PDR). Les autres changements sont plus spécifiques à l'option. Parmi les évolutions les plus marquantes que nous avons engagées, citons la révision de toutes les UE. Tout d'abord, nous avons redéfini les thèmes UEs de manière à en améliorer la lisibilité et clarifier leurs articulations. Partant de là, nous avons révisé leurs contenus en supprimant des modules

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

d'enseignement devenus obsolètes. Nous avons exploité le volume horaire ainsi dégagé pour renforcer des modules que nous jugeons importants ou pour en introduire de nouveaux. Par exemple, nous avons procédé au remplacement du vieillissant cours de gestion de projet où il était question des démarches en V et en "waterfall". Le nouveau module que nous avons mis en place, traite toujours de la gestion de projets, mais avec une démarche agile (comme par exemple SCRUM). Celle-ci constitue désormais la nouvelle référence en entreprise comme en témoigne le succès de manifestations telles que la conférence Agile (ex-XP Days). Un autre exemple d'évolution a consisté en la suppression du module sur CORBA en faveur d'un module qui traite de J2EE. Nous avons également doublé le volume horaire de cet enseignement pour consacrer suffisamment de temps à cette technologie devenue incontournable dans le monde de l'entreprise. Un dernier exemple est celui des nouveaux modules : la programmation par contraintes, les systèmes multi-agents, les systèmes embarqués et la méta-modélisation. Nous les avons introduits pour étoffer la palette de connaissances de nos élèves et élargir leurs horizons de pensées.

Communication auprès des élèves du tronc commun. Cette action a pris deux formes. La première a été d'aller à la rencontre des élèves du tronc commun. Mon but était de leur présenter l'option ISIC et ses débouchés. J'ai utilisé dans mes présentations, les résultats des projets effectués par les élèves déjà dans l'option. Cela s'est avéré être un bon choix car il permet aux élèves de mieux visualiser les compétences cultivés. Un autre support que j'ai diffusé est une collection de documents (articles dans la presse, brochure du Syntec numérique,...) sur les métiers d'un ingénieur informaticien. La deuxième forme d'action de communication est également en relation avec les débouchés. Il s'agit d'interventions d'industriels partenaires qui, à ma demande, sont venus parler de leur activité et du marché de l'emploi des informaticiens.

E.2.2 Autres responsabilités pédagogiques liées à l'option ISIC

Parallèlement au grand chantier qu'a représenté l'amélioration de l'attractivité de l'option, j'ai assumé d'autres responsabilités pédagogiques liées à l'option ISIC. Je les réparti en 4 groupes :

- Définition et évolution du contenu de l'option
- Suivi des élèves
- Gestion des intervenants
- Coordination avec les autres acteurs de la formation au sein de l'Ecole.

Définition et évolution du contenu de l'option. Nous ne nous attardons pas sur ce groupe de tâches que nous avons déjà décrit dans la liste d'actions de la section E.2.1. J'ajouterai seulement qu'en plus du contenu scientifique et technique, je devais définir les volumes horaires et les crédits de chaque module d'enseignement. Bien entendu, ces différentes tâches étaient effectuées en concertation avec les enseignants.

Suivi des élèves. Cet ensemble regroupe les tâches d'accompagnement des élèves-ingénieurs dans leur scolarité. Les tâches les plus importantes sont :

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

- Les absences : La présence à tous les cours/TD/TP est obligatoire. Je devais donc m'assurer de l'assiduité des élèves et le cas échéant vérifier leurs justificatifs d'absences. Des absences injustifiées à répétition exposent l'élève à des sanctions qui peuvent aller jusqu'à l'exclusion définitive.
- Les notes : En tant que responsable de l'option, je centralisais toutes les notes. Je devais repérer le plus tôt possible les élèves en difficulté afin d'identifier et tenter de résoudre les éventuels problèmes. Je devais également discuter de telles situations lors des conseils des professeurs. En fin d'année, je devais faire des propositions quant au devenir des élèves en difficulté (passage conditionné à des rattrapages, redoublement, exclusion).
- Les Stages : J'accompagnais les élèves dans leur recherche de stage (Diffusion d'offres, aide pour la rédaction de CV, validation des sujets de stage) pour les deux stages effectués durant l'option. Cet accompagnement commence dès l'arrivée des élèves dans l'option, notamment pour les élèves qui doivent effectuer un stage à l'étranger. En effet, chaque élève doit effectuer durant sa scolarité au moins un stage dans un autre pays. De tels stages sont naturellement plus difficiles à trouver notamment du fait des distances culturelles et linguistiques. Après le démarrage des stages (en France ou ailleurs), je devais m'assurer de leur bon déroulement et gérer les éventuels changements ou problèmes (Changement du sujet ou de la durée du stage, départ du tuteur de stage de l'entreprise, ...). Une fois les stages effectués, je devais planifier avec les élèves et leur tuteurs de stages les dates de soutenance.
- Les projets professionnels : Les élèves de l'EMD sont accompagnés dès le début du cursus dans une réflexion individuelle pour définir un projet professionnel. Ce projet est progressivement affiné notamment en option où j'ai conseillé les élèves d'ISIC. Environ 25 à 30% d'entre eux (comme dans d'autres options) choisissent de passer la dernière année dans un autre établissement. En effet, l'EMD a plusieurs accords d'échange d'étudiants pour des séjours académique ou des double-diplômes, avec plusieurs établissements en France et à l'étranger. Les élèves qui partent, restent néanmoins rattachés à l'Ecole. J'ai ainsi suivi des élèves qui sont partis soit suivre des formations de management (notamment Audencia, Télécom-management) ou scientifiques (par exemple : IIT Delhi, Inde - University of Chalmers, Suède - Georgia Tech Atlanta, USA - University of Durham, Royaume Uni). Ce suivi s'est fait en deux parties. Avant le départ, j'ai conseillé les élèves et j'ai validé leurs choix d'enseignements ou de cursus. Après le départ des étudiants, je m'assurais régulièrement du bon déroulement de leur scolarité, notamment à la lumière de leurs notes. Je continuais à les conseiller quant aux choix de leurs projet et leurs stages.
- Les élèves étrangers : Dans le cadre d'accords avec d'autres établissements, l'EMD reçoit des étudiants notamment des étrangers (Chinois, Brésiliens, Indiens, ...). Ils peuvent soit valider un nombre de crédits, ou bien suivre la totalité de la fin du cursus de l'EMD et notamment les options. En préparation de leur arrivée, j'interagissais avec ceux d'entre eux intéressés par l'informatique. Le cas échéant, je définissais avec eux un parcours qui tient compte des pré-requis des modules qui les intéressent et de leur formation d'origine.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

Gestion des intervenants. Vu le nombre réduit d'enseignant-chercheurs informaticiens (nous sommes 5), l'option ISIC repose d'une part sur les ingénieurs du service informatique de l'Ecole, mais également pour une part non-négligeable sur des intervenants extérieurs. Une des tâches importantes qui m'incombait consistait à gérer cet ensemble hétérogène, concilier les contraintes et les visions forcément divergentes afin d'assurer une formation cohérente, de la meilleure qualité qu'elle soit. Plus concrètement, cela consistait en :

- Emploi du temps. Il s'agit évidemment d'allouer des créneaux horaires et des salles aux différents enseignements et autres projets. Ainsi, je devais veiller au bon enchaînement des modules (commencer par les pré-requis) tout en prenant en compte les disponibilités des enseignants (notamment extérieurs).
- Rechercher les intervenants. Chaque année entre la mi-mai et la mi-juillet, après avoir fait le point sur le programme pédagogique de l'option, je devais re-contacter les différents intervenants et m'assurer de leur disponibilité. Cela concerne plus particulièrement les extérieurs (vacataires ou prestataires). En cas d'indisponibilité de l'un d'eux ou pour les nouveaux modules, je devais chercher de nouvelles personnes ayant l'expertise requise pour assurer les enseignements. Une fois tous les intervenants trouvés, je devais transmettre le montant total des coûts (vacations + prestations) à la direction des études de l'Ecole.
- Suivi des intervenants extérieurs. Cette tâche concerne la logistique des interventions. Je devais souvent planifier les déplacements des enseignants qui viennent d'autres régions, les accueillir à la gare, les emmener à l'hôtel, puis à l'Ecole. Par ailleurs, je devais faire un suivi comptable et m'assurer de leur présence et transmettre tous les mois le nombre d'heures effectués et le type d'activité (cours, TD, ...) au service comptable pour paiement. Enfin, j'avais la charge de transmettre les supports de cours à la reprographie.
- Bilan des enseignements. Ce bilan est la synthèse de deux sources d'informations qui m'a permis d'avoir un processus d'amélioration "continue" du contenu pédagogique de l'option conformément à ce qui est pratiqué dans l'EMD. Chaque responsable de module faisait lors des dernières séances un bilan avec les élèves et me retournait une proposition avec les éventuels améliorations et changements s'il y en a. De mon côté, à la fin de chaque année, je faisais un bilan avec les élèves seuls sur l'année dans sa globalité.

Coordination avec les autres acteurs de la formation au sein de l'Ecole. Celle-ci regroupe différentes tâches et notamment :

- Bilan mensuel au sein du département. Une fois par mois, je devais rendre compte au chef du département de la situation de l'option. Ce bilan se faisait dans le cadre d'une réunion qui réunit les personnes assumants des responsabilités d'équipe ou pédagogiques. Ainsi, je partageais des informations sur les enseignements avec notamment mes collègues responsables des options OAPI (Optimisation & Automatisation des Processus Industriels) et IQ (Ingénierie de la Qualité), également rattachées au département IA.
- Bilan mensuel avec la direction des études. Cette réunion réunit différentes personnes de la direction des études avec les responsables des 8 options de l'EMD. C'est l'occasion de faire le point sur chaque option, les éventuels difficultés et les autres questions transversales aux options. C'est le cas notamment de l'organisation (modalités, sujets, suivi et jurys) des

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

projets scientifiques et techniques, qui du fait de leur pluridisciplinarité, réunissent des élèves de différentes options.

- Préparation et participation aux conseils des professeurs. Pour chaque promotion, le conseil des professeurs se réunit deux fois par an. En présence de la direction des études, des professeurs et des représentants des élèves, les situations des élèves notamment ceux en difficulté sont discutés. En préparation de ces conseils, je devais récupérer toutes les notes et les transmettre à la direction des études (travail fait tout au long de l'année du fait du contrôle continu). Par ailleurs, je devais identifier les élèves en difficulté, analyser leurs situations et faire des propositions quant à leur passage en année supérieur, le redoublement ou l'exclusion.
- Participation à la réflexion sur l'évolution de la formation à l'EMD. La formation dispensée à l'EMD évolue régulièrement. Ces changements touchent aussi bien le tronc commun que les options. C'est donc naturellement que j'ai contribué aux évolutions et en particulier le passage d'un cycle de 4 ans à 3 ans, par des propositions lors des discussions préparatoires ainsi qu'à leur mise en place. Le recrutement des élèves qui se faisait initialement en large partie au niveau BAC+1 est passé à BAC+2. Le contenu de la formation a changé, ainsi, à partir de 2012, une partie des enseignements d'options a été remplacée par des *mineures*. Il s'agit d'UEs attachées chacune à au moins 2 options. A l'issue de la première année totalement dédiée au tronc commun, chaque élève choisi une *mineure*. Les élèves ayant suivi une mineure s'orientent vers l'une des options associées. Afin de multiplier les possibilités et laisser le choix aux élèves, plusieurs parcours sont possibles. Ainsi, une même option est associée à au moins 3 mineures différentes.
- Information et communication. Il s'agit d'actions collectives qui visent à diffuser l'information en interne à l'école ou vers des cibles extérieures, sous forme écrite, de rencontres ou de visites accompagnées de démonstrations. Je n'en cite ici que les plus importantes. Par exemple, j'ai participé à la préparation du guide de l'élève qui décrit la formation. Celui-ci est actualisé avant chaque rentrée avec les contenus des enseignements, les crédits, les modes d'évaluation, ... J'ai également rédiger tous les ans des parties du rapport d'activité de l'EMD. Ce dernier vise quand à lui un public extérieur, ainsi que le conseil d'administration de l'école. Une autre action importante est le *forum d'entreprises*. C'est un salon annuel organisé par l'EMD où les entreprises viennent rencontrer les élèves et les enseignants. Dans ce cadre, en plus d'inviter des entreprise du secteur informatique, j'ai tenu un stand dédié à l'option ISIC. Ainsi, j'informais les élèves du tronc commun sur la formation ISIC et les perspectives d'emploi.

E.3 Responsable de modules d'enseignements à l'ÉCOLE DES MINES DE DOUAI (2001-actuellement)

Depuis mon arrivée à l'EMD, j'ai pris la responsabilité d'un certain nombre de modules d'enseignement, que je décris ci-dessous. Depuis le début, j'ai constamment cherché à améliorer et à faire évoluer tant le contenu scientifique et technique de mes enseignements, que ma manière

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

de les dispenser. Je souhaite en effet capter l'attention des élèves et optimiser leur acquisition des connaissances. Un des outils fondamentaux que j'utilise est l'organisation de séances qui mixent cours/TD/TP. J'aborde un chapitre de cours que j'illustre en détaillant un exemple simple. J'invite les élèves à refaire l'exemple sur leur machine (TD/TP) afin de m'assurer de leur compréhension du cours. Enfin, je leur propose plusieurs problèmes pour lesquels ils doivent développer une solution en mobilisant les connaissances acquises.

En terme d'évaluation, je privilégie des contrôles pratiques avec documents autorisés. L'idée est de mettre les élèves dans une situation qui s'approche le plus de ce qu'ils rencontreront dans leur vie professionnelle.

Dans le reste de cette section, je présente chacun des modules et je fourni une description de son contenu. Ces résumés correspondent à la dernière version du cours. En effet, je fais évoluer mes cours annuellement en fonction des évolutions technologiques et des retours des élèves.

E.3.1 Programmation par objets (2001-actuellement)

Niveau : Bac+4

Effectifs : 8 à 32 élèves (option ISIC)

Volume horaire annuel : 10h de cours, 30h de TD/TP

Descriptif : Dans ce module je commence par un approfondissement des concepts des langages à objets, avant d'introduire les schémas de conception (*design patterns*), le développement dirigé par le test (TDD) et la concurrence. Afin que les élèves soient en mesure de prendre du recul par rapport à la technologie, j'utilise deux langages de programmation différents : Smalltalk et Java.

E.3.2 Systèmes à objets distribués (2001-actuellement)

Niveau : Bac+5

Effectifs : 8 à 32 élèves (option ISIC)

Volume horaire annuel : 4h de cours, 14h de TD/TP

Descriptif : Dans ce module, je commence par initier les élèves au développement d'applications à objets distribués basées sur les sockets. Une fois qu'ils ont assimilé les ingrédients de base des middleware (proxy, serialisation, déserialisation, passage par copie, par référence), nous abordons le concept d'ORB (Object Request Broker). Enfin, nous abordons les questions de déploiement et les différentes architectures de systèmes distribués (client-serveur, 3 tiers, pair à pair).

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

E.3.3 Langage C (2001-2008)

Niveau : Bac+4

Effectifs : 56 élèves (2 options : ISIC et OAPI) de 2001 à 2004, puis 8 à 15 élèves (option ISIC) de 2005 à 2008.

Volume horaire annuel : 6h de cours, 14h de TD/TP

Descriptif : L'objectif de ce module est de former les élèves aux fondements du langage C, incontournable dans la programmation système. En plus des éléments du langage, j'ai jugé utile d'introduire dans ce cours une dimension relative au processus d'ingénierie. Ainsi, j'ai abordé des notions tels que le développement modulaire et les libraires.

E.3.4 Composants logiciels (2004-actuellement)

Niveau : Bac+5

Effectifs : 8 à 32 élèves (option ISIC)

Volume horaire annuel : 4h de cours, 14h de TD/TP (2004-2008) - 2h de cours, 5h de TD/TP (2009-actuellement)

Descriptif : C'est le module qui a le plus évolué depuis que j'en ai la responsabilité. Historiquement, il traitait exclusivement de J2EE avec une perspective technologique. J'ai introduit deux nouveaux chapitres. Le premier, définit le concept de composant logiciel et le second aborde un modèle de composant académique. Depuis 2009, en tant que responsable de l'option ISIC, j'ai augmenté le volume horaire lié à la partie EJB/J2EE et je l'ai délégué à un vacataire. L'intervenant est un responsable R&D d'un éditeur de logiciel utilisant J2EE. Il a une pratique industrielle de cette plate-forme et à ce titre il a un point de vue technologique plus pertinent et plus intéressant pour les élèves.

E.3.5 Systèmes embarqués et Robotique Mobile (2008-actuellement)

Niveau : Bac+5

Effectifs : 8 à 32 élèves (option ISIC)

Volume horaire annuel : 4h de cours, 14h de TD/TP

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

Descriptif : L'objectif de ce module que j'ai introduit dans l'option ISIC est de fournir aux élèves des éléments de programmation de systèmes embarqués en général et à la robotique mobile en particulier. J'ai fait le choix d'axer le cours sur la problématique de performance que j'aborde suivant deux facettes complémentaires : architectures logicielles et optimisation de programmes. Cela présente l'avantage dé présenter des élèves des concepts dont une partie peut être utilisée dans d'autres domaines. C'est le cas notamment de l'optimisation de programmes.

E.3.6 Initiation à la programmation (2009-2011)

Niveau : Bac+3

Effectifs : 12 élèves formation continue, tronc commun

Volume horaire annuel : 2h de cours, 6h de TD/TP

Descriptif : Il s'agit de la partie *présentielle* d'un module d'initiation à l'informatique, dont le reste se déroule à distance. Il est suivi par des étudiants étrangers qui suivent la formation continue diplômante dont une partie se fait à distance. Les étudiants sont dans leur pays d'origine. Je me suis donc déplacé à l'étranger pour assurer cette intervention.

E.3.7 Introduction à la programmation par objets (2011-actuellement)

Niveau : Bac+4

Effectifs : 200 élèves (tronc commun)

Volume horaire annuel : 6h de cours, 20h de TD

Descriptif : Il s'agit d'un cours introductif aux concepts de la programmation par objets pour les élèves du tronc commun. J'ai fait cette initiation avec le langage Java, bien que je pense qu'il ne soit pas idéal pédagogiquement. En effet, les élèves ont eu une initiation à la programmation procédurale dans un environnement Java. Ainsi, en continuant avec Java ils restaient en terrain connu. Mais, pour cela, j'ai dû faire face à la difficulté d'isoler au mieux les concepts afin d'en introduire un à la fois.

E.3.8 PDR : Projets de Découverte de la Recherche (2009-actuellement)

Niveau : Bac+4

Effectifs : 3 à 5 projets effectués par 2 élèves chacun

© 2014 Tous droits réservés.

doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

Volume horaire annuel : 60 à 100 heures

Descriptif : Les PDR sont des projets d'élèves en rapport avec la recherche sur des sujets proposés par les enseignants. Un volume horaire de 80 heures est prévu à l'emploi du temps. Je commence l'encadrement des PDR par une introduction au sujet et une formation rapide aux concepts et outils requis pour le projet. J'effectue ensuite un point d'avancement avec les élèves lors de leurs séances de projet (1 à 2 séances hebdomadaires). Enfin, je suis jury à l'évaluation (document, code et soutenance). La difficulté majeure de ces projets est qu'ils sont planifiés au début de l'option, quand les élèves ont très peu de bagage technique. Des sujets pertinents, réellement en rapport avec la recherche sont donc difficiles à poser.

E.3.9 PST : Projets Scientifiques et Techniques (2004-actuellement)

Niveau : Bac+5

Effectifs : 1 à 2 projets effectués par 4 à 8 élèves chacun

Volume horaire annuel : 20 à 40 heures

Descriptif : Les PST sont des projets pluridisciplinaires que les élèves effectuent en fin de cursus. Ils doivent déboucher sur une réalisation pour le compte d'un client extérieur à l'Ecole. Chaque année, je joue le rôle de tuteur pour 1 ou 2 projets. Il s'agit d'encadrer les travaux des élèves (120h à l'emploi du temps) et de les accompagner dans la relation avec le client extérieur à l'Ecole. Enfin, je participe à l'évaluation du rapport du projet et au jury de la soutenance.

E.3.10 Jurys divers

Je participe tous les ans à différents jurys pour un volume global d'environ 20h/an :
- Admission aux Ecoles des Mines
- Stages technicien de 13 semaines minimum en entreprise (Bac+3)
- Stages ingénieur adjoint de 16 semaines minimum en entreprise (Bac+4)
- Stage ingénieur de 25 semaines minimum en entreprise (Bac+5)
- Projets (PDR, PST) encadrés par mes collègues

E.4 Participation aux enseignements de l'Ecole des Mines de Nantes (2000-2003)

Après ma thèse à l'Ecole des Mines de Nantes, j'ai continué à effectuer des TD, TP et autres encadrements projets. J'ai effectué une partie de ces enseignements en anglais, puisque la cible était les étudiants du master international EMOOSE dont la majorité était d'origine étrangère (Chili, Argentine, Colombie, Belgique). Ma participation aux enseignements de l'Ecole

© 2014 Tous droits réservés.
doc.univ-lille1.fr

HDR de Mohammed Nouraddine Bouraqadi Saâdani, Lille 1, 2013

des Mines de Nantes s'est prolongée après mon départ pour Douai jusqu'en 2003. En effet, j'ai donné un cours qui fait le tour d'horizon des spécificités du langage Smalltalk, destiné aux élèves de dernière année de l'option Génie des Systèmes Informatiques. La liste ci-dessous donne des éléments complémentaires sur ces différents enseignements.

- Aperçu de Smalltalk : 24 élèves niveau Bac+5, 3h de cours et 5h de TD/TP de 2002 à 2003 (responsable du contenu).
- Compilation : 24 élèves, niveau Bac+4, 16h de TP en 2000.
- Réflexion : 9 étudiants, niveau Bac+5, 12h de TP en anglais (étudiants étrangers du master EMOOSE) en 2000.
- Proposition de sujets, encadrement et jury de mini-projets : niveau Bac+5, 5 projets effectués par des étudiants étrangers du master EMOOSE (interaction en anglais) en 2001.

© 2014 Tous droits réservés.

doc.univ-lille1.fr