Pôle

Objets et composants logiciels

GDR ALP

Coordinateurs : C. Dony, C.Queinnec

Responsables des groupes :
G. Castagna, C.Dony, J-L. Giavitto, J.Malenfant, C. Queinnec

1 Problématique

Nouveaux défis. Les langages à objets sont aujourd'hui très largement utilisés pour la production industrielle de logiciels. Celle-ci pose à présent de nouveaux défis : (1) la décentralisation liée à l'omniprésence du réseau, (2) l'existance d'une base très importante de logiciels et systèmes existants que les sociétés souhaitent faire évoluer sans les reconstruire entièrement, (3) les demandes importantes en matière d'aptitude des logiciels à l'évolution enfin (4) le développement d'applications par « instantiation » d'applications génériques, ou par assemblage de composants existants. Un composant peut se présenter sous différentes formes : binaire, classe, spécification, documentation, code source, hiérarchie de classes, interface (Corba, Active/X, Java, C++, Eiffel...), framework, objet composite de CAO ou d'application multimédia. Ainsi une application générique peut elle-même être vue comme un composant logiciel.

Objets et composants L'idée de développer des applications par utilisation, assemblage ou réutilisation de composant est ancienne mais les débuts de sa mise en pratique à grande échelle sont récents. Le formalisme de programmation par objets est déjà au coeur des possibilités actuelles dans ce domaine (architectures logicielles génériques ou extensibles : *frameworks*, architectures pour l'interopérabilité : CORBA, COM/OLE, composants répartis : *JAVA Beans*, contrôles Active/X) et l'un des objectifs majeurs de la recherche dans le domaine des objets au cours des prochaines années sera de rendre effective cette démarche de réalisation des applications.

Parmi les propriétés nécessaires pour de tels composants, on trouve traditionnellement la réutilisabilité et la composabilité, émanations directes des travaux réalisés depuis des années au sein du pôle objets du GDR-Programmation. Les recherches initiales dans le domaine ont montrés que ces propriétés ne peuvent être obtenus sans que par ailleurs les composants soient définis à l'aide d'un processus de modélisation clair et précis, et organisés sous formes de hiérarchies bien construites et éventuellement correctement typées. Il faut que ces composants soient conformes à des spécifications claires et précises, le plus souvent formelles. Il faut qu'ils soient développés dans une démarche qualité où le cycle de développement a de plus en plus tendance à n'être qu'un cycle de maintenance, avec tout ce que cela suppose comme support à la conception, à la documentation, aux tests, etc. Parce que toutes les possibilités de composition ont rarement été prévus à l'avance, les composants doivent être adaptables. Enfin, parce que les composants ne sont pas tous conçus dans le même langage, l'interopérabilité devient également une propriété cruciale.

Objectifs. L'objectif général du pôle que nous proposons est de regrouper et de mieux coordonner une communauté de chercheurs travaillant de facto sur les évolutions des technologies objet dans le but de répondre aux défis posés. Le thème central de ces recherches est relatif à l'ensemble des approches et des techniques de spécification, de conception, de réalisation, d'assemblage, de gestion de l'évolution, d'exécution des composants logiciels variés et hétérogènes, répartis sur le réseau.

2 Liste des groupes de travail et contacts

2.1 Groupes de travail

Ce pôle vise à établir une large synergie, il regroupe un nombre important d'équipes et sera structuré en groupes de travail. Un groupe de travail est un ensemble d'équipes collaborant à l'avancement des connaissances ainsi qu'au développement de nouvelles techniques et de nouveaux outils supportant directement ou indirectement la programmation par composants.

Certains groupes ont une problématique de développement de techniques, de moyens pour l'industrie du composant; ainsi l'étude de l'évolution des techniques d'implantation des langages à objets, l'étude de l'évolution des modèles de programmation par objet, l'étude de la spécification et des contrôle sémantiques, enfin l'étude des architectures réflexives sont des points incontournables pour la maîtrise des composants logiciels. D'autres groupes sont plus orientés vers les problèmes : problèmes de l'architecture et de l'utilisation des composants génériques et des bibliothèques de composants, problème de la gestion des composants en situation d'utilisation (cohérence, versions, répatition, interopérabilité). Les problématique sont en fait très imbriquées et nous prévoyons que de nombreux projet interagissent à travers les frontières des groupes, pour appliquer une technique particulière à un problème donné.

Les groupes de travail proposés et décrits dans les sections suivantes sont :

- **Evolution des techniques d'implantation des langages à objets**.
  Animateur : Christian Queinnec.
- **Spécification, vérification, sémantique**.
  Animateur : Giuseppe Castagna.
- **AGI : Architectures génériques (frameworks, hiérarchies) et idiômes (patterns)**,
  Animateur : Christophe Dony.
- **Réflexivité et architectures à méta-niveaux**
  Animateur : Jacques Malenfant.
- **Conception et Support de l'Évolution et de la repartition**
  Animateur : Jean-Louis Giavitto.

2.2 Contacts intra-GDR

Notre proposition constitue un effort important de regroupement de chercheurs et d'équipes autour de la problématique centrale des composants. Certains appartenaient à différents pôles de l'ancien GDR programmation (pôle objet, pôle programmation parallèle, pôle interface et environnements). Par ailleurs, de nombreux nouveaux chercheurs du domaine qui ne participaient pas aux activités de l'ancien GDR sont intégrés à cette proposition.

2.3 Contacts inter-GDR

Nous souhaitons que des contacts aussi étroits que possible s'établissent avec des pôles d'autres GDR ayant des problématique proches. La forme exacte que pourront avoir les relations inter-gdr reste à définir.

– GDR ARP (Architectures, Réseaux et Parallélisme) - Objets répartis. L'évolution vers le réseau tend à faire de tous composant logiciel un maillon d'une application répartie, ce qui fait apparaître des besoins de sûreté, de sécurité et de résistance aux pannes pour les composants, propriétés qui sont abordés dans plusieurs de nos groupes de travail mais qui sont aussi largement étudiées dans le domaine des systèmes et donc dans le GDR ARP.

– GDR I3 (Information, Interaction, Intelligence). L'ancien pôle objet du GDR programmation avait des contact étroits avec le groupe "représentation par objets" de l'ancien PRC IA. Plusieurs des projets décrits ci-dessous proposent de nouvelles collaborations avec les membres de ce groupe, par exemple pour ce qui concerne la construction des architectures logicielles génériques. Nos nouveaux projets proposent également des collaborations plus nouvelles avec des chercheurs en « bases de données à objets » mais ces contacts gagneront à être développés car un certain nombre de problématiques relatives à l'évolution et à la gestion des composants sont communes.

3 Fonctionnement

L'objectif premier du pôle est de mieux coordonner les recherches françaises autour du domaine des composants. L'organisation en groupes de travail que nous proposons est un premier pas vers une meilleures coordination mais il est clair que des rapprochements plus étroits entre projets similaires pourront avoir lieu. Pour cela il est essentiel que le pôle puisse tenir des journées, au moins une fois par an, incluant des sessions propres à chaque groupe et des sessions communes à l'ensemble des groupes. Chaque groupe fera le point annuellement de l'avancée de ses projets et de ses collaborations.

4 Participants

4.1 Liste des participants

– CNET : Nathalie Chevalier, Michel Dao,

– CRIN : Amedeo Napoli, Dominique Colnet, Olivier Zendra, Philippe Coucaud,

– I3S - ESSI - Université de Nice : Anne-Marie Pinna-Dery, C. Michel, Denis Caromel, J.-C. Pazzaglia., Jean-Pierre Regourd, Manuel Serrano, Marjorie Russo, Mireille Blay-Fornarino, Philippe Collet, Philippe Lahire, Roger Rousseau,

– INRIA Rhones-Alpes : Jerome Euzenat, Petko Valtchev,

– INRIA Sophia Antipolis : Isabelle Attali,

– IRIN - Université de Nantes : Cristine Choppy, Jean-Claude Royer, Michel Allemand, Pascal André, Pascal Poizat,

– IRISA : Alain Le Guennen, Jean-Lin Pacherie, Jean-Marc Jézéquel, Wai Ming Ho,

– IRIT, ENSEEIHT, Université de Toulouse : Annie Marcoux, Aris Papandreou, Christine Maurel, Fabien Dagnat, Frédéric Migeon, Jean-Louis Colaco, Louis Féraud, Marc Pantel, Patrick Sallé,

– Ecole des Mines de Nantes : Frédéric Rivard, Jacques Noyé, M. Braux, Mario Südholt, Nicolas Revault, Nouri Bouraqadi, Olivier Thomann, Pierre Cointe, Thomas Ledoux,

– L3I — INSA Lyon : Jean Louis Sourrouille,

– LaMI – Université d'Évry Val d'Essonne : Olivier Michel,

– LGI2P (École des Mines d'Alès) : Christelle Urtado, Dalila Tamzalit, Martine Magnan, Mourad Oussalah, Sylvain Vauttier

– LIENS - ENS Paris : Giuseppe Castagna,

– LIFL Lille : Bernard Carré, Gilles Vanwormhoudt,

– LIP6 : Jean-Pierre Briot, Alain Cardon, Zahia Guessoum, Marc Lhuillier, François Pachet, Christian Queinnec, Thierry Saura, Min-Jung Yoo,

– LIRMM - Université de Montpellier : Roland Ducournau, Christophe Dony, Marianne Huchard, Hervé Leblanc, Thérèse Libourel, Stéphane Pelenc,

– LRI - Université Paris XII - Orsay : Marie-Claude Gaudel, Jean-Louis Giavitto,

– Université Paris I : Sebastien Praud, Veronique Benzaken,

- VALORIA - Université de BRETAGNE -SUD : Jacques Malenfant,
- Université de Savoie : Michel Riveill,

5 Groupe de travail - Implantation de langages à objets

Responsable : Christian Queinnec - Université Paris-VI - Christian.Queinnec@lip6.fr

5.1 Présentation

Le but de ce groupe est de rassembler une communauté d'implanteurs de langages à objets autour des thèmes modernes que sont, entre autres :

- les implantations cohabitant avec C et/ou d'autres langages ou bibliothèques (et notamment le couplage par Corba),
- les bibliothèques d'exécution de programmes répartis (GC compris),
- la compilation/interprétation de programmes, non figés, mobiles, réflexifs et coopératifs,
- la définition de compilateurs et d'interprètes ouverts,
- la définition et l'implantation des langages de scripts.

Les buts sont de fournir des modèles et des techniques d'implantation à l'usage des concepteurs de langages ou d'applications à objets. Les buts à court terme sont :

- la synthèse de GC adapté à un programme,
- la compilation séparée de l'envoi de messages,
- les protocoles d'échanges de données structurées,
- les fonctions génériques réparties,
- la définition d'une architecture de compilation multi-langages (incluant Java et Smalltalk) pour une même machine virtuelle.

Composition

Les équipes membres de ce groupe de travail sont, à ce jour :

Laboratoire	Responsable scientifique
CRIN	Dominique Colnet
	Philippe Coucaud
	Olivier Zendra
École des Mines de Nantes	Pierre Cointe
	Olivier Thomann
IRIT	Louis Féraud
	Aris Papandreou
LIP6	Christian Queinnec
	Thierry Saura
LIRMM	Roland Ducournau
Univ. Nice	Manuel Serrano
VALORIA	Jacques Malenfant

5.2 Compilation séparée et efficace de l'envoi de message

L'envoi de message est considéré comme la clé de l'efficacité (ou de l'inefficacité) des langages de programmation par objets. De nombreuses techniques ont été élaborées, que ce soit dans le cadre des langages à typage dynamique (pour un survol, [Duc97]) ou dans celui des langages à typage statique (pour une approche exemplaire, voire la cas du compilateur SmallEiffel). Dans les deux cas, l'optimisation est confrontée à un compromis espace-temps mais la difficulté principale, non encore résolue, réside dans l'inaptitude des techniques employées à permettre une compilation séparée. Or, le marché du logiciel exige de maintenir la confidentialité des sources des modules dont seul le code objet est délivré. Pour satisfaire cette exigence il y deux voies possibles — la classique compilation séparée pour laquelle il faudrait alors essayer d'adapter les techniques efficaces de compilation de message ou mettre au point des techniques similaires plus appropriées, — une compilation répartie dans laquelle les modules distants seraient compilés à distance en suivant les instructions envoyées par l'utilisateur (compilateur) local. Le projet se propose d'examiner ces deux voies.

5.3 Synthèse de gestionnaire de mémoire

La gestion de la mémoire dans les langages à objets est un problème essentiel tant par les aspects liés à la sûreté qu'à ceux liés à la performance. Plus exactement, nous cherchons à voir si les informations d'inférence de types peuvent être utilisées pour optimiser la gestion de la mémoire. Il est en effet possible de produire lors de la compilation un GC tenant compte des specificités de chaque programme compilé.

5.4 Objets ouverts

Corba procure un mécanisme d'invocation de méthodes sur des objets-serveurs distants. Cette vision correspond à une logique industrielle favorisant le traitement transactionnel court sur des objets opaques non mobiles. Tout traitement non prévu par l'interface est interdit. Ce modèle, émanation des clients/serveurs classiques, n'est pas extensible du côté du client. Pour prendre un exemple, qui n'a pas rêvé, dans une application permettant de consulter une encyclopédie, à un traitement non prévu mais correspondant à un simple petit script de filtrage ? Permettre d'échanger des données brutes, structurées et auto-décrites est le premier but de ce projet.

Le second but défend encore une fois le client qui souhaite conférer dynamiquement de nouveaux comportements aux objets existants qu'il manipule et dont il n'est probablement pas le propriétaire. Dans un monde persistant, cette capacité est fondamentale. Par exemple, les clients http (mosaic, netscape, etc.) ne sont, d'une certaine manière, que des mises en œuvre nouvelles du protocole `ftp` et des sites accessibles qui n'ont pas eu à être modifiés pour procurer ces nouveautés. De la même manière, il faut pouvoir doter des objets existants de nouvelles méthodes propres à leur utilisateur et non à leur propriétaire.

Le projet, nommé objets répartis ouverts au sens des systèmes ouverts, étudiera d'abord, en conjonction avec le projet plate-forme, les mécanismes de bas niveau pour échanger des valeurs structurées, les rendre persistantes et assurer leur cohérence. Dans un second temps, le mécanisme de méthodes externes sera analysé sur la base des fonctions génériques à la CLOS. Ces développements seront eux-mêmes multi-langages afin de pouvoir s'adapter à de multiples situations. Les langages visés sont, entre autres, C, Objective caml, Java et Scheme.

5.5 Projet HECTOR

Le travail précédent était dédié à l'interoperabilité via CORBA dont on a pu mesurer certaines limites. En conséquence, nous souhaitons reposer le problème de l'interoperabilité en revoyant HECTOR pour mieux intégrer des composants hétérogènes. Cette nouvelle conception impliquera de nouvelles implantations et des questions telles que: communication entre traducteurs,typage, essais de standardisation de code objet … qui renvoient à des problèmes de compilation de LOO.

6 Groupe de travail : Spécification, vérification, sémantique

Responsable : G.Castagna, LIENS, Ecole Normale Supérieure, Paris, castagna@dmi.ens.fr

6.1 Présentation

Les langages orientés objets ont acquis une importance prépondérante dans le développement des applications informatiques à grande échelle. Cette utilisation a rendu nécessaire l'étude formelle de ces langages à la fois pour mieux en cerner les caractéristiques fondamentales et pour pouvoir définir de nouveaux langages à objets capables de combiner une plus grande expressivité avec une grande facilité d'utilisation et fiabilité.

Le but de ce groupe est d'étudier les aspects sémantiques et formels des langages objets pour procéder à leur développement et à leur vérification. En particulier la recherche sera axée sur les problématiques suivantes:

1. Spécification formelle de programmes avec pour objectifs la génération de code et la conception assistée; une attention particulière est portée aux méthodes d'écriture et de structuration de la spécification.

2. Sémantique formelle de langages et applications objets avec pour objectifs la vérification de propriétés du langage ou de l'application, et la construction d'outils de développement, visualisation, analyse, transformation et parallélisation; une attention particulière est portée aux aspects liés à la concurrence.

3. Mécanismes d'assertion et typage pour les langages objets avec pour objectifs la vérification statique et/ou dynamique de correction et fiabilité des programmes et la structuration des composants logiciels; une attention particulière est portée aux aspects lies à l'implantation/performance et aux problémetiques liées à la répartition (localité, sites, tolérance aux pannes...) ou à la persistance (contraintes d'intégrité, rapprochement des langages de requêtes et des langages d'assertions avec quantifications...).

6.2 Participants

Membres	Affiliations	Axe
Pascal André	(MdC, IRIN, Nantes)	1,2,3
Michel Allemand	(MdC, IRIN, Nantes)	1,2
Isabelle Attali	(CR, INRIA Sophia Antipolis)	2,3
Véronique Benzaken	(MdC, Université Paris 1)	3
Denis Caromel	(MdC, Université de Nice)	2,3
Giuseppe Castagna	(CR, LIENS, Paris)	3
Christine Choppy	(Prof, IRIN, Nantes)	1,2
Philippe Collet	(PDoc, Université de Nice)	3
Jean-Louis Colaço	(PDoc, ENSEEIHT, Toulouse)	2,3
Carine Courbis	(DEA, Université de Lyon)	2,3
Fabien Dagnat	(Doc, ENSEEIHT, Toulouse)	2,3
Michel Dao	(CNET, Paris)	1
Marc Pantel	(ENSEEIHT, Toulouse)	2,3
Pascal Poizat	(Doc, IRIN, Nantes)	1,2
Sebastien Praud	(Doc, université Paris 1)	3
Roger Rousseau	(MdC, Université de Nice)	3
Jean-Claude Royer	(MdC, IRIN, Nantes)	1,2,3
Marjorie Russo	(doc, Université de Nice)	2,3
Patrick Sallé	(Prof, ENSEEIHT, Toulouse)	2,3

6.3 Projets

Les trois axes cités au début seront traités au sein de plusieurs projets dévéloppés par une ou plusieurs équipes du groupe.

6.3.1 Vérification statique de programmes concurrents

- Définition et étude du calcul d'acteurs primitifs CAP: sémantique des acteurs et objets concurrents; extension du calcul pour la prise en compte de la répartition (sites, pannes, ...).
- Vérification statique par inférence:élimination des erreurs de type dynamiques; élimination des messages orphelins.
- Mise en oeuvre dans une extension acteurs de Caml (ML-Act).

6.3.2 Propriétés des langages et applications objets

- sémantiques du langage Java et du noyau Java Card (destiné à la programmation des cartes à puces);
- À partir de la sémantique formelle, construction d'outils de développement, visualisation, analyse, transformation, ou parallélisation;
- À partir de la sémantique formelle, expression et vérification d'un ensemble de propriétés, soit du langage lui-même, soit des programmes (par exemple aspects sécurité, parallélisation et distribution).

6.3.3 Spécification des programmes à objets pour la génération de code

À l'heure actuelle, les modèles et outils d'analyse et de conception par objets permettent de générer de manière assez satisfaisante la définition du squelette des classes (attributs typés et signatures de méthodes). En revanche, le corps des méthodes elles-mêmes ne bénéficie d'aucun traitement automatique. L'objectif est de proposer des solutions pour améliorer la génération de code dans le cadre d'un outil de conception par objets supportant le méta-modèle UML. Plusieurs directions de recherche sont envisageables :

- enrichir les parties dynamiques du méta-modèle UML pour pouvoir représenter les comportements de manière mieux adaptée ;
- chercher à intégrer des langages de spécification "opérationnelle" tels que B ou Claire ;
- utiliser un modèle objet pour représenter les aspects fonctionnel et dynamiques [Sht97].

6.3.4 Spécifications formelles mixtes pour les composants réutilisables

La plupart des méthodes actuelles de développement à objets se basent sur plusieurs formalismes de description (statique, fonctionnel, dynamique). Malheureusement la cohérence et les possibilités de preuve sont quasi inexistantes du fait d'un manque de sémantique commune. Notre proposition est d'étudier des spécifications formelles alliant aspect dynamique et aspect fonctionnel. Ce formalisme sera inspiré des concepts objets et devra permettre une conception assistée dans un langage à objets. Les aspects méthodes pour la structuration et l'écriture des spécifications, preuves et conception à objets seront prioritairement mis en valeur. Un aspect délicat des preuves est que l'on s'intéresse a des vues différentes du problème (dynamique et fonctionnel), il est important d'avoir un cadre commun permettant d'intégrer différente techniques de preuves (model-checking et prouveur de théorèmes).

Un tel formalisme doit proposer des notations graphiques pour représenter certains aspects de la spécification notamment les aspects dynamiques.

Il y a plusieurs points de départ connus pour ces travaux. Les travaux sur les Types Abstraits Graphiques qui mélangent un automate gardé et une spécification algébrique. Une autre approche intéressante est celle du langage LOTOS qui associe une description de contrôle avec une spécification algébrique. D'autres travaux complémentaires existent en conception formelle de matériel ou en méthodes à objets.

Des liens avec le pole Specifications et preuves du GDR ALP et le groupe Modelisation à objets prévu dans le GDR I3 sont envisagés.

6.3.5 Intégration des multi-méthodes dans les langages à délégation

1. Définitions de sous-systèmes et/ou extensions des langages de type FHM (Fisher, Mitchell Honsell) et AC (Abadi Cardelli) aptes à supporter une extension avec "sélection multiple". Une attention particulière sera portée à la decidabilité et à l'existence d'un type minimum pour ces langages, existence qui constitue un problème ouvert.

2. Étude des relations formelles entre les langages FHM et AC.

3. Intégration de la "sélection multiple" dans le ou les systèmes développés au point (1), et étude formelle de leur propriétés en particulier pour ce qui concerne le système de types.

4. Réalisation d'un prototype pour implanter l'extension qui s'avérera la plus intéressante.

6.3.6 Assertions dans les langages objets

A mi-chemin entre les preuves et les tests, les techniques à base d'assertions ont une utilité reconnue pour l'obtention d'une documentation en accord avec la réalité du code, pour la conception par contrats et pour embarquer dans le code des diagnostics utilisables par les tests. Leurs principaux mérites sont l'emploi de formalismes très accessibles aux programmeurs et de supporter des définitions partielles qui peuvent être complétées de manière incrémentale, ce qui est appréciable dans un cadre réparti. Leurs principaux défauts concernent leur expressivité trop souvent réduite et la difficulté à les prouver.

L'objectif de cet axe est double : augmenter l'expressivité des assertions et contribuer au rapprochement des techniques statiques (typage, preuves partielles) et dynamiques (évaluations différées, échantillonnées...).

Plus précisément, le premier objectif conduira, dans un premier temps, à l'étude de l'introduction de quantificateurs dans les assertions. L'introduction de quantificateurs dans les assertions permet d'augmenter leur pouvoir expressif (leur confèrant ainsi le statut de requêtes au sens base de données du terme). Cependant, l'introduction de quantificateurs s'avère insuffisant si l'on souhaite énoncer puis garantir certaines propriétés relatives à l'évolution des objets (dans le contexte de langages persistants). De telles propriétés s'expriment généralement en logique temporelle du premier ordre. La pertinence d'un tel formalisme sera étudiée

Le deuxième objectif vise à (partiellement) garantir statiquement ou dynamiquement la correction des programmes en étudiant la possibilité de combiner des techniques d'analyse statique de programme (Transformeurs de prédicats à la Dijkstra, etc. . .) à des techniques de démonstration automatique de théorèmes en logique (temporelle linéaire) du premier ordre ou d'autres techniques (évaluation différée, partielle, test de logiciel etc. . .).

Une attention particulière sera portée pour proposer des solutions indépendantes d'un langage particulier et pour utiliser des standards (Corba, ODMG, UML...).

6.3.7 Implémentation des mécanismes nécessaires aux controÎes dynamiques

Pour que les techniques assertionnelles ou de contrôles de contraintes d'intégrité puissent reposer sur une forte expressivité, notamment par l'utilisation de quantifications, il faut que les supports d'exécution permettent l'évaluation des formules logiques de manière efficace. Cela suppose des moyens de représentation explicite par réification des extensions de types (volatiles ou persistantes, centralisées ou réparties), d'utilisation de techniques par nécessité et d'informations sur les intentions des formules, pour choisir les techniques d'évaluation les plus adaptées : contrôle extensif ou échantillonné... De plus, il faut développer la possibilité de vérifier statiquement des propriétés stables pour rapprocher les techniques de tests des techniques de typage statique voire des preuves.

6.3.8 Securité des Objets Mobiles

La récente explosion de l'utilisation d'Internet et l'apparition conjointe de la mobilité de systèmes à la fois software (applets) et hardware (ordinateurs portables), posent de nouveaux enjeux tant au niveau de la flexibilité des outils conçus pour programmer Internet que, plus encore, au niveau de la sécurité du réseau.

En fait, la configuration vers laquelle Internet tend de plus en plus à évoluer est celle d'une énorme machine concurrente partitionnée en systèmes emboîtés (firewalls, LAN's, workstations, applets) et dont la topologie se modifie en permanence.

Il se pose donc le problème de l'étude d'un formalisme pour la programmation de cette machine qu'est Internet, et pour en assurer sa sécurité.

L'utilisation d'objets emboités bien s'adapte à la representation du partitionnement, et les calculs formels développes pour étudier la programmation concurrente capturent de manière très satisfaisante les aspects qui concernent la communication. Restent les aspects de localité et de mobilité.

L'absence de ces aspects rend ces calculs complètement inadaptés à l'étude des problèmes de sécurité et de robustesse tel que la "fault tolerance" ou l'accès a des domaines protégés. Seul les aspects de typage peuvent être pris en compte. Mais ceux-ci, quoique nécessaires, ne gèrent qu'une petite partie de la problématique liée à la sécurité et à la robustesse. D'ailleurs cette défaillance n'est pas caractéristique des calculs formels mais elle est présente dans l'utilisation actuelle d'Internet. Par exemple, des problèmes de sécurité tels que l'allocation de ressources et le "denial of service" sont trop souvent délégués par le langage de "programmation" d'Internet au support de la machine soujacente (dans le cas de la Java Virtual Machine, elle s'appuis pour cela sur le système d'exploitation résident). L'existence de ce double niveau de contrôle constitue une source potentielle de trous de sécurité, source qui est souvent exploitée pour des attaques malicieuses. Il est donc dans notre intention d'étudier d'une manière formelle tous ces aspects et proposer de solutions à ces problèmes de securité.

6.4 Collaborations

Des liens avec le pôle spécifications et preuves du GDR ALP et le groupe Modelisation à objets prévu dans le GDR I3 sont envisagés. En particulier une coordination (reunion commune et echange de documents) avec le groupe "Spécification formelle et test de logiciels orientés-objet" du pôle spécifications et preuves est prevue.

Références : [ACR97, CPS96, CPS97, CV98, Cas97, Sht97, BS97, BS98, Col97, Col, CR96, AC98, AR98, PCR98]

7 Groupe de travail : Hiérarchies, architectures génériques (frameworks) et idiômes (patterns)

Responsable : C.Dony, LIRMM, Université de Montpellier-II, dony@lirmm.fr.

7.1 Présentation

Ce groupe de travail s'intéresse globalement aux méthodes de développement, à l'architecture et à la gestion des applications réutilisables et des bibliothèques de composants. Un composant dans ce contexte peut être un patron de conception *design-patterns*, description d'une solution logicielle à un problème spécifique, un *framework*, architecture logicielle permettant de générer simplement des applications ou d'autres composants comme des classes, une hiérarchie de classes ou des ensembles de classes. Le groupe travaille sur les problèmes suivants :

- conception et production d'applications à partir de composants, de frameworks ou de patrons,
- étude des architectures logicielles des applications et des bibliothèques, découpage modulaire des applications en vue de leur réutilisation,
- ré-ingénierie, restructuration d'applications,
- mesure de la qualité des applications et des composants

Liste des équipes

- Ecole des mines de Nantes : Pierre Cointe, Jacques Noyé, Mario Südholt, Nouri Bouraqadi, Thomas Ledoux, Frédéric Rivard, M. Braux, Nicolas Revault.
- IRISA Rennes : Jean-Marc Jézéquel, Alain Le Guennen, Jean-Lin Pacherie, Wai Ming Ho,
- LIFL Lille : Bernard Carré, Gilles Vanwormhoudt,
- LIRMM Montpellier : Christophe Dony, Marianne Huchard, Hervé Leblanc, Thérèse Libourel.
- LIP6 Paris : Jean-Pierre Briot, Zahia Guessoum, Marc Lhuillier, Min-Jung Yoo.
- CNET Issy les Moulineaux : Michel Dao et Nathalie Chevalier,
- CRIN Nancy : Amedeo Napoli,
- INRIA Rhones-Alpes : Jerome Euzenat, Petko Valtchev
- AFC Consulting - Sophia-Antipolis Annick Fron

7.2 Projets relatifs à la programmation par aspects (AOP)

La programmation par aspects est apparue récemment (1996) comme un autre paradigme de programmation avec lequel il est possible de spécifier et de représenter les différents aspects ou services d'un module logiciel. Cette séparation introduit une nouvelle forme de découpage modulaire, par exemple entre les différents ensembles de méthodes constituant une classe, et offre de nouveaux schémas de réutilisation.

7.2.1 Conception modulaire d'objets

Participants : Bernard Carré, Gilles Vanwormhoudt

L'objectif de l'étude est de faciliter la conception modulaire d'objets, liée au découpage transversal d'un système en fonctions (tâches, activités, contextes). Il s'agit de rendre compte de la double structuration (orthogonale) objets/fonctions, et ainsi de tirer profit des moyens offerts par ces deux dimensions, en particulier : hiérarchies de classes, composition et génericite. Ce découpage pose le problème inverse de l'articulation entre elles des tâches centrées sur les mêmes objets. Cette étude [VG97, G.98] est en relation avec les récents travaux sur la conception par plans d'objets et sur la programmation par aspects ou par sujets (aspect-programming, subject-oriented programming).

7.2.2 Programmation par aspects

Participants : Pierre Cointe, Jacques Noyé, Mario Südholt, Nouri Bouraqadi, Thomas Ledoux et Frédéric Rivard (Ecole des Mines de Nantes)

Ce groupe étudie la compilation à partir de la spécification des aspects par le moyen d'une technologie de compilation particulière, la «machine à tisser» les aspects (aspect weaver). Deux sujets de recherche plus particuliers ont été initiés dans ce domaine :

- L'utilisation de techniques réflexives [Coi96] pour la représentation uniforme de différents aspects d'une application.
- La notion d'aspects d'un programme parait relativement proche du concept de vues d'une architecture logicielle. Le projet consiste à étudier les relations entre ces deux concepts dans le cadre d'un travail sur la définition d'architectures logicielles utilisant schémas de conception et schémas d'architecture.

7.3 Projet liés à l'organisation et à la gestion des architectures logicielles

7.3.1 Méthodes de ré-ingéniérie

Participantes : Isabelle Borne et Annya Romanczuk, Ecole des Mines de Nantes

L'objectif de ce projet est d'étudier comment améliorer la qualité de modèles de conception donnés dans un formalisme de type OMT (par exemple des schémas obtenus par une traduction automatique [BRS98]). Pour cela nous proposons de restructurer les modèles objets, en passant par une représentation de ces modèles à l'aide de schémas (patterns) d'analyse et de conception. Ensuite, des mesures seront étudiées et définies afin évaluer qualitativement et quantitavement certains critères de qualité des nouveaux modèles de conception obtenus.

7.3.2 Détection de défauts de conception

Participants : Isabelle Borne et Nicolas Revault, Ecole des Mines de Nantes

Nous étudions et cherchons à élaborer des méthodes permettant la reconnaissance ou la détection de schémas de conception, ou de situations pour appliquer des schémas, dans le code d'une application afin d'en extraire une spécification de plus haut niveau d'abstraction. La détection de situations dans lesquelles appliquer des schémas de conception passe fatalement par une comparaison du code observé à une version standard de l'implémentation de schémas candidats. Pour réaliser cette comparaison, nous explorons la possibilité d'utiliser différents systèmes de mesure de code, en particulier des systèmes de calcul de métriques sur du code, comme ceux fournis dans ENVY/QA. Ces systèmes pouvant aussi fournir des éléments de critique de code, nous pensons également utiliser ceux-ci pour évaluer la distance entre deux fragments de programmes.

7.3.3 Projet environnement de construction et de manipulation des hiérarchies d'héritage

Participants : Christophe Dony, Marianne Huchard, Hervé Leblanc, Thérèse Libourel - LIRMM Montpellier.

Participants - Contact industriel Ce projet est mené dans le cadre d'une CTI avec le CNET avec Michel Dao et Nathalie Chevalier.

Participants - Collaboration inter-GDR Nous comptons collaborer sur la conception incrémentale de hiérarchie avec Amedeo Napoli et Jérome Euzenat du groupe "Représentation par Objets" du GDR I3 qui travaillent sur la classification et l'apprentissage.

L'objectif général du projet est de contribuer à la maîtrise de l'assemblage et de l'évolution de composants logiciels hétérogènes par la réalisation d'un environnement complet de manipulation de hiérarchies. L'environnement est basé sur un modèle générique de notation et de représentation des hiérarchies qui permettra de manipuler des hiérarchies issus de différents formalismes ou langages : OMT, JAVA, C++ ou SMALLTALK par exemple. Un premier point du projet s'intéresse à la dénition des critères formels et informels permettant de juger de la qualité des hiérarchies et à la réalisation d'outils de mesure correspondants. Sur ces bases nous développerons des outils de manipulation basiques et évolués. Les outils de base permettront d'enlever ou d'ajouter une classe en respectant un ensemble de critères. Les outils évolués permettront la construction de hiérarchies à partir d'un ensemble de classes, la réorganisation et la fusion de hiérarchies. Un outil de réorganisation a déjà été spécifié[DDHL96] dans une première phase du projet.

Références : [DDHL95, DDHL96, DHL97]

7.3.4 Construction automatique de taxonomies dans les bases de connaissances

Participants : Jerome Euzenat, Petko Valtchev - INRIA Rhone-Alpes.

L'organisation et la réorganisation de graphes de spécialisation a de multiples applications (comme l'aide à la programmation par objets). Nous sommes plus particulièrement intéressé par la construction automatique de taxonomies dans les bases de connaissance à des fins d'analyse exploratoire (se faire une idée des régularités qui peuvent exister). Pour cela nous avons développé: - des algorithmes de catégorisation (construction de taxonomies à partird'instances) prenant en compte les rapports spécifiques entre objets (composition, relation...) fondés sur une dissimilarité adaptée elle aussi aux notions d'objets (exploitation de la structure hiérarchique) [1]; - un algorithme d'insertion d'un élément dans un treillis (calcul incrémental de la complétion de McNeille) [2].

Références :

1 Petko Valtchev, Jérôme Euzenat Dissimilarity measure for collections of objects and values Lecture notes in computer science 1280 (Xiaohui Liu, Paul Cohen, MichaelBerthold (éds.), Actes 2nd international symposium on intelligent data analysis, London (UK), pp259-272, (21-25 août) 1997), 1997 ftp://ftp.inrialpes.fr/pub/sherpa/publications/valtchev97c.ps.gz

2 Petko Valtchev An Algorithm for Minimal Insertion in a Type Lattice Actes 2nd international symposium on Knowledge, Retrieval, Use and Storage for Efficiency (KRUSE), Vancouver (CA), pp82-94, (11-13 août) 1997 ftp://ftp.inrialpes.fr/pub/sherpa/publications/valtchev97b.ps.gz

7.4 Projets liés à la conception et à l'utilisation des architectures logicielles réutilisables

7.4.1 Conception et implémentation d'architectures génériques

Participants : J. Noyé, M. Südholt, M. Braux.

Les architectures génériques objet (*frameworks*) ont un rôle clé à jouer comme moyen d'exprimer autant des conceptions que des implémentations ré-utilisables. Cette réutilisation n'est toutefois pas gratuite et pose un certain nombre de problèmes, tant au niveau de la conception que de l'implémentation.

Du point de vue de l'implémentation ,nous cherchons actuellement à établir jusqu'à quel point les techniques de spécialisation de programmes comme l'évaluation partielle [CHM+98] et le découpage de programmes (*slicing*) peuvent aider à éliminer les problèmes de passage du générique au spécifique posés par les outils de développement actuels (compilateurs, débogueurs, profileurs…) . Nous étudions aussi les relations entre ces techniques et la réflexion dans le cadre de la spécialisation à l'exécution. Afin de focaliser nos travaux, nous nous intéressons tout particulièrement aux applications réparties s'appuyant sur des technologies Java et Corba (travaux en collaboration avec l'équipe de Charles Consel de l'Irisa).

Du point de vue de la conception, nous travaillons actuellement sur la traduction d'architectures logicielles définies en termes de schémas de conception vers des schémas d'architectures bien définis formellement.

7.4.2 programmation génériques

Participants : Didier Parigot,Gilles Roussel, Etienne Duris, Loïc Correnson.

Le projet s'intéresse à la réutilisation de composants logiciels. En particulier, nous nous intéressons à la programmation générique ou adaptive. Notre objectif est de développer une méthodologie et un cadre général pour la conception et l'utilisation de composants génériques (indépendant de tout paradigme de programmation). Nos premiers travaux [CDPR98] portent sur la définition de méta-langage pour la spécification de l'utilisation d'un module générique. Une autre partie de nos travaux concerne la mise en oeuvre de cette utilisation de façon efficace grâce à des outils d'évaluation symbolique : déforestation, composition symbolique, évaluation partielle. Cette approche est développée avec un soucis constant de préserver l'interopérabilité des méthodes et des outils développés. En particulier, nous envisageons d'étudier la validité de notre approche dans le cadre de la construction automatique de pilotes de gestion d'équipements réseaux, en collaboration avec le CNET.

7.4.3 Expérimentation des Frameworks et des Patrons pour les applications réparties

Participants : Jean-Marc Jézéquel - Irisa/CNRS - jezequel@irisa.fr Alain Le Guennen - Irisa - aleguenn@irisa.fr Wai Ming Ho - Irisa - WaiMing.Ho@irisa.fr

Contacts industriels : - Softeam (réalisation d'un framework de validation de logiciels répartis)

- CNET (CTI MetaFor: Meta-Frameworks d'objets répartis)

Ce groupe a comme objectif général le développement de méthodes et d'environnements de programmation pour architectures parallèles distribuées. Une des approches que nous explorons consiste à utiliser des technologies objets pour construire des *modèles* spécifiques à des domaines d'applications (calcul scientifique intensif, systèmes de télécoms, travail coopératif), et en s'appuyant sur la notion de "framework", de fournir des cadres de conception, de réalisation et de validation adaptés à ces modèles.

L'objectif de cette action est d'identifier et de formaliser les schémas de conception revenant de manière répétitive dans un contexte distribué (en particulier dans le domaine des télécommunications) et d'explorer comment ils pourraient être intégrés dans l'outil UMLAUT de manipulation de modèles de conception d'architectures d'objets répartis fondé sur UML (*Unified Modeling Language*).

References : [GJP96, JJL97, JT98]

7.4.4 Expérimentation des Frameworks et des composants pour les systèmes multi-agents

Participants : Jean-Pierre Briot, Zahia Guessoum, Marc Lhuillier, Min-Jung Yoo.

Le groupe "Frameworks" du thème OASIS possède une expérience accumulée en matière de conception de frameworks (ex : programmation concurrente : Actalk, par règles : NeOpus, par contraintes : BackTalk...). En plus des activités en matière de frameworks de programmation multi-paradigmes (voir le groupe de travail Langages multiparadigmes), nous étudions aussi activement les frameworks de programmation par agents (architecture modulaire d'agent DIMA, architecture componentielle d'agent MALEVA, framework d'agents mobiles SCD). Un point clé est la décomposition du comportement d'un agent sous forme de composants élémentaires et spécialisés. Une approche possible (DIMA) est de contrôler la coordination et le séquencement de ces composants par l'intermédiaire d'un module de supervision explicite à un niveau méta (lien avec le groupe de travail Réflexivité). Une autre approche (MALEVA) est décentralisée et repose sur le découplage des flux de données et de contrôle.

Références : [Lhu98, YMB98, GBD97, Gue97]

8 Groupe de travail : Analyse, conception et support de l'évolution et de la distribution des objets et des composants

Responsable : Jean-Louis Giavitto, LRI ura 410 du CNRS, - Université de Paris-Sud, giavitto@lri.fr.

8.1 Présentation

La thématique de ce groupe recouvre l'analyse, la conception et le support de l'évolution, ainsi que la distribution et la répartition des objets et des composants. L'objectif est de faciliter la conception et la réalisation d'applications adaptables parallèles et ouvertes.

Nous nous intéressons particulièrement aux méthodes d'analyse et de conception ainsi qu'aux mécanismes de programmation et aux supports d'exécution permettant de maîtriser l'interaction et la composition d'objets répartis, dans un environnement évolutif et ouvert : comment gérer les versions des objets ou des composants, rétablir la cohérence lors du remplacement d'un objet ou d'un composant par un autre compatible et de qualité supérieure ; comment localiser un objet, une ressource de calcul, s'adapter aux aléas d'un réseau et aux pannes des systèmes matériels ; comment assurer l'interopérabilité au niveau des objets et des services, permettre la migration, s'adapter à des environnements d'exécution hétérogènes.
Cette problématique est à la croisée de plusieurs thèmes (systèmes répartis, bases de données, programmation parallèle, ...) et trouve son application dans de très nombreux domaines d'activités : la conception assistée par ordinateur, le génie logiciel, l'intelligence artificielle, les systèmes d'information, les interfaces homme-machine, les bases de données, le calcul parallèle et la simulation, ainsi que les domaines applicatifs de l'ingénierie.

Le but de ce groupe de travail n'est pas de se substituer aux pôles et thèmes existants menant des recherches spécifiques à chaque domaine, mais de faciliter la rencontre des différentes communautés travaillant sur ces domaines, répertorier les équipes et les compétences, analyser les problématiques, identifier les domaines d'applications potentiels et d'éventuels partenaires d'autres disciplines, ainsi que de favoriser l'échange et la diffusion de techniques, outils et prototypes.

Liste des équipes

- ESSI – Université de Nice ; contact : Mireille Blay-Fornarino (blay@jessica.essi.fr)
- LRI – Université de Paris-Sud ; contact Jean-Louis Giavitto (giavitto@lri.fr)
- I3S – Université de Nice ; contact : Philippe Lahire (lahire@unice.fr)
- VALORIA – Université de Bretagne sud ; contact Jacques Malenfant (Jacques.Malenfant@univ-ubs.fr)
- LaMI – Université d'Évry Val d'Essonne ; contact : Olivier Michel (michel@lami.univ-evry.fr)
- LGI2P (École des Mines d'Alès) ; contact : Mourad Oussalah (Oussalah@eerie.fr)
- Projet INRIA SiracIMAG – Université de Savoie ; contact : Michel Riveill (Michel.Riveill@univ-savoie.fr)

8.2 Projets des équipes

Six projets sont développés autour de trois axes par les participants de ce groupe de travail :

- conception et construction d'applications évolutives,

- interaction entre composants et répartition,
- mécanismes réflexifs et architecture distribuée.

Ces projets s'échelonnent tout le long du cycle de vie du logiciel des phases d'analyse et de conception aux supports d'exécution.

L'analyse et la conception d'applications objet évolutives est étudiée dans le projet 8.2.1. Le projet 8.2.2 aborde plus spécifiquement le développement d'applications réparties par *intégration et extension d'applications existantes*.

Les deux projets suivants se concentrent sur l'*interaction entre composants et répartition*, l'un (8.2.3) à partir des concepts de méta-programmation, l'autre (8.2.4) en étendant la notion de « mixin ».

Enfin, les deux derniers projets visent à étendre les concepts objets classiques en s'appuyant sur des mécanismes réflexifs et un support d'exécution. Ainsi, le projet 8.2.5 s'intéresse à la *prise en compte explicite des ressources* et au développement d'un *support d'exécution* pour la synchronisation et la gestion répartie d'une mémoire objet partagée. Le projet 8.2.6 s'attaque aux problèmes de d'unification des différentes sortes de liens entre objets et d'*interopérabilité* et de *persistance* des objets.

8.2.1 Analyse et conception de l'évolution d'objets

Participants : Mourad Oussalah, Martine Magnan, Christelle Urtado, Dalila Tamzalit, Sylvain Vauttier – LGI2P (École des Mines d'Alès).

Description. Les modèles de l'évolution d'objets issus de l'industrie ou de la recherche qui ont été développés ces deux dernières décennies ont souvent adopté une politique curative et non pas préventive. En effet, ces modèles ont intégré le concept d'évolution et non pas le processus d'analyse et de conception de l'évolution. L'expérience a montré aussi bien au niveau de la conception que du développement les limites des approches évolutives actuelles.

En effet, la modélisation de l'évolution d'objets telle qu'elle est traitée classiquement est rigide et parfois inadaptée par rapport à la réalité du monde. L'absence d'une modélisation correcte de l'évolution des données physiques, structurales et comportementales peut entraîner des problèmes allant de la redondance jusqu'à l'incohérence de ces données.

Les stratégies d'évolution existantes sont diverses et variées dans la littérature et ne répondent que partiellement aux besoins. Aussi le groupe analyse et conception de l'évolution d'objets s'intéresse aux différentes stratégies d'évolution d'objets existantes en analysant chacune de ces techniques (avantages et limites) et surtout en précisant les classes de problème et d'application auxquelles elles s'adressent.

8.2.2 Assemblage de composants répartis (OLAN)

Participants : Michel Riveill – Université de Savoie - Projet INRIA SiracIMAG.

Description. L'objectif de ce projet est de fournir des outils et services pour le développement et l'exécution d'applications réparties par intégration et extension d'applications existantes. On vise à fournir des outils permettant :

- de construire des applications réparties par assemblage de composants, dont certains sont des applications existantes ;
- de configurer ces applications à la demande et de les déployer sur des plates-formes d'usage courant.

L'approche suivie combine des techniques de programmation par composants et des techniques de programmation par agents.

Partenariats actuels : Projet ESPRIT LTR C3DS ; action Dyade (GIE Bull-INRIA) AAA ; CNET (administration de services) ; Xerox XRCE (outils de compilation) ; SGS-Thomson (projet Storia, mise en collaboration d'outils interactifs mono-utilisateur). Contact en cours : Soft-Mountain et CSTI.

Références : [BBKR96, BBM95, BBBR96, BR96, VDBM97]

8.2.3 Méta-programmation et objets distants

Participants : Mireille Blay-Fornarino, Anne-Marie Pinna-Dery, un thésard – ESSI, Université de Nice.

Description. Cette recherche concerne à la fois la définition des interactions entre objets distants et l'apport de la méta-programmation à la programmation par composants.

Dans un contexte aussi difficile que les applications distribuées, l'objectif est à la fois de permettre une programmation plus simple et plus évolutive des applications distribuées fortement interactives et en même temps de vérifier l'absence de cycles, de déterminer la nécessité de point de synchronisation, de permettre la mise en place de priorités, etc. C'est pourquoi nous aimerions décrire l'évolution des composants à la fois en terme de dépendances et de Méta-Objets. La robustesse des application répartie ainsi visée est également un point crucial de notre démarche.

Partenariats : Nous avons un étudiant en thèse sur ce sujet et une collaboration en cours d'élaboration avec l'ENSI de Tunis sur les aspects migration d'objets. Nous travaillons aussi avec Stéphane Ducasse à l'université de Berne. Nous collaborons aussi étroitement avec Jean-Paul Rigault, (directeur de l'ESSI).

Références : [DDF96, DFP95, BDFJ97, ABBF$^+$98]

8.2.4 Mobilité et composition incrémentale

Participants : Olivier Michel – LaMI, Université d'Évry ; Jean-Louis Giavitto – LRI, Orsay.

Description. Les notions de *nom* et d'expression *ouverte* sont centrales dans la problématique de construction incrémentale de programmes. Par exemple, la notion de *mixin*, dans laquelle des noms sont utilisés comme références à d'autres mixins, généralise la notion d'héritage des langages à objets [BC90], la composition de module [AZ96] et la compilation séparée [Anc97]).

On peut d'ailleurs observer que, pour ce qui est de la macro-structure des programmes (classes, modules), l'éditions des liens avec les bibliothèques [LF96, Car97]) ou bien pour l'interaction entre applications distribuées [Que96], c'est ce mode de combinaison par nom qui est privilégié.

Le formalisme des amalgames [MDVS96, Mic96], proche des mixins, repose sur une notion de nom et un mécanisme de capture de nom dans un cadre déclaratif. Il autorise la définition de systèmes incrémentaux dont le comportement évolue au cours du temps. Partant des travaux déjà réalisés, nous voudrions développer deux axes. Le premier consiste à introduire dans le formalisme des amalgames une notion de nom *distante*. Cette extension de la notion de nom permettra la référence, la composition et l'évolution d'une expression en fonction de données distribuées.

Le deuxième axe consiste à étudier et intégrer aux amalgames une notion de *mobilité* et de migration. Cette notion de mobilité, qui est absente de `Java` par exemple, est nécessaire au développement d'applications incrémentales sur un réseau ouvert.

Réalisation : Un prototype du langage 81/2 a déjà été réalisé dans le cadre du pôle Parallélisme de l'ex-GDR de programmation. Nous désirons étendre ce prototype en intégrant une notion de mobilité et de capture distante.

8.2.5 Programmation énergétique (ORCADE)

Participants : Jacques Malenfant – Université de Bretagne sud.

Description. Le travail s'organise autour de deux axes. Le premier consiste en une évolution du paradigme objet vers la *programmation énergétique* (où le modèle explicite la consommation des ressources), à la *mobilité* en milieu hétérogène (où l'adaptabilité devient une propriété cruciale) et à la *composition dynamique* (où les services se construisent à l'exécution autour des composants).

Le deuxième axe s'intéresse à la prise en compte de ces évolutions dans l'implantation de supports d'exécution à objets répartis. L'intégration des services de GC répartis, de sérialisation, de contraintes de qualités services ou de réflexion sera étudiée dans ce contexte.

Plus concrètement, en termes d'outils, nous allons travailler sur la mémoire d'objets répartie DMeroon, et en étudier particulièrement le moteur de sérialisation/désérialisation ainsi que la mise en œuvre sur des machines liées par un réseau à haut débit. Nous comptons proposer une extension de Java vers la programmation énergétique sous la forme d'une architecture logicielle (frameworks) et nous comptons également construire le lien entre Java et la mémoire DMeroon.

8.2.6 Plateforme objet flexible et expression de liens entre composants

Participants : Philippe Lahire, Robert Chignoli, Pierre Crescenzo – Université de Nice - Sophia Antipolis (équipe OCL).

Description. Le premier objectif de ce projet est de réaliser une étude des liens entre classes qui devrait aboutir à la définition d'un modèle de paramétrage des liens. Suite à cette étude nous souhaitons spécifier et développer une plateforme objet qui permette la définition ou la redéfinition par paramétrage, des types de liens nécessaires à la description d'une application ou d'une classe d'applications.

Cette plateforme doit permettre en particulier:

- d'associer une sémantique aux concepts très généraux de classe et de lien de manière à pouvoir s'adapter aux différents mécanismes mis en oeuvre dans les principaux langages et à pouvoir créer différents types de liens (vues, lien inverse, lien est-une-sorte-de, lien est-une-version-de, ...);
- de réifier la plupart des concepts que l'on peut trouver dans les langages de programmation d'aujourd'hui, utilisés dans le monde industriel ;
- d'intégrer complètement la notion d'objet persistant de manière à permettre l'exécution de celui-ci soit dans son contexte d'origine (sémantique du langage utilisé dans le programme qui a créé l'objet), soit dans le contexte de l'application qui l'utilise (sémantique du langage utilisé dans le programme qui consulte ou met à jour un objet persistant).

Quelques caractéristiques de notre approche:

- son aspect dynamique basé sur une architecture méta-niveaux,
- la gestion d'informations réflexives,
- l'indépendance des aspects syntaxiques des langages par rapport au moteur de la plateforme,

- la possibilité d'intégrer les problèmes d'interopérabilité et de distribution des objets sur un réseau.

Références : [CCL97]

9 Groupe de Travail «Réflexivité et architectures à méta-niveaux»

Responsable : Jacques Malenfant, Université de Bretagne sud, Jacques.Malenfant@univ-ubs.fr.

9.1 Manifeste

Pendant longtemps tenu pour curiosité scientifique, la réflexivité attire aujourd'hui l'attention de nombreux chercheurs aux prises avec des problèmes d'adaptabilité, éventuellement dynamique, des logiciels. Dans les systèmes répartis construits sur la toile mondiale, les applications ont vocation à s'exécuter sur de longues périodes de temps dans un contexte où les conditions d'exécutions peuvent changer brusquement. La congestion des réseaux, la mobilité, les pannes, l'apparition, la mise à jour et la disparition de services sont autant de réalités auxquelles ces applications vont devoir s'adapter. Dans ce groupe de travail, nous allons étudier l'approche réflexive et méta-architecturale pour la résolution de ces problèmes d'adaptation dynamique. Nous étudierons plus particulièrement les langages à objets réflexifs, leur implantation et leur sémantique ainsi que leurs liens avec les systèmes et les réseaux.

9.2 Equipes et projets

Équipe : L3I — INSA Lyon

Contacts : Jean Louis Sourrouille

Mél. : jean-louis.sourrouille@if.insa-lyon.fr

Description : Nous conduisons actuellement un projet de langage objet temps-réel fondé sur l'utilisation de méta-objets. L'objectif visé est de placer dans l'objet «standard» tous les traitements indépendants du contexte d'exécution afin de conserver une grande capacité de réutilisation. Le méta-objet prend en charge le contexte et contrôle entièrement l'objet unique auquel il est associé. En particulier il gère les contraintes de comportement, de concurrence intra et inter-objets, mais aussi de temps [JJ98]. À partir de ce principe architectural, une direction envisagée est la définition d'un objet autonome et adaptatif qui change son comportement en fonction du contexte, en particulier vis à vis des contraintes de temps.

Équipe : Janus — Laboratoire ESSI, Nice

Contacts : Mireille Blay-Fornarino et Anne-Marie Pinna-Dery

Mél. : blay@essi.fr, pinna@essi.fr

Description : Notre projet concerne à la fois la définition des interactions entre objets distants et l'apport et/ou l'impact de la méta-programmation à la conception. De façon plus détaillée, sur le premier point qui est la définition des interactions, dans un contexte aussi difficile que les applications distribuées, l'objectif est à la fois de permettre une programmation plus simple et plus évolutive des applications fortement interactives. En particulier, nous nous intéressons à vérifier l'absence de cycle, à déterminer la nécessité de point de synchronisation, etc. Nous pensons atteindre ces objectifs par une approche «méta». Dans ce cadre, nous nous intéressons actuellement plus précisémment à la programmation par aspect et développons en Smalltalk et en Open-C++.

Sur le second point, les apports et impacts de la méta-programmation, nous avons plus précisément avancé sur l'approche conception, en nous intéressant dans le cadre des collecticiels, aux besoins rencontrés, aux passages entre modèles statiques et dynamiques, à la détection de problèmes dès la conception. Ce travail nous conduit à envisager une description plus avancée de l'application afin d'exprimer dès la conception le comportement attendu du système.

Équipe : OASIS — LIP6, Université de Paris VI, Pierre et Marie Curie

Contact : Jean-Pierre Briot

Mél. : Jean-Pierre.Briot@lip6.fr

Participants : Khedija Aoun, Alain Cardon, Zahia Guessoum.

Description : Le groupe «Frameworks» du thème OASIS s'intéresse de manière générale à la réflexivité comme pont entre les concepteurs de systèmes et les utilisateurs. Une étude approfondie sur le sujet a déjà débuté, en pleine collaboration avec Rachid Guerraoui de l'EPFL, Lausanne. Nous explorons ces idées en particulier dans le domaine des systèmes d'information coopératifs et distribués, par exemple appliqués à la gestion de crises. Nous comptons mettre en oeuvre un couplage entre une plateforme multi-agents (DIMA) et un framework de programmation répartie (BAST de Benoit Garbinato et Rachid Guerraoui a l'EPFL). L'idée est ensuite d'étudier les possibilités de communication entre le niveau de connaissances des agents et les protocoles de répartition (et de tolérance aux fautes). Cette communication/influence peut être vers le haut par «upcalls» en cas de détection de pannes, et vers le bas par adaptation dynamique de protocoles en fonction des décisions des agents. Notons également que le modèle d'agent de DIMA possède lui-même une architecture à méta-niveau puisqu'un module de supervision est chargé de contrôler la coordination et le séquencement des différents composants d'un agent (voir le groupe de travail Frameworks).

Équipe : ? — LORIA, Inria Nancy

Contact : Amedeo Napoli

Mél. : Amedeo.Napoli@loria.fr

Description : Étude de la réflexivité dans les systèmes de représentation des connaissances, surtout à la lumière des travaux récents dans le domaine des langages de prototypes réflexifs ; depuis les travaux de Pattie Maes ou de Jacques Ferber à la fin des années 80, peu de choses ont été faites en réflexivité et IA, hors ce sujet apparaît important en particulier dans le domaine de la représentation des connaissances.

Équipe : ORCADE — Laboratoire VALORIA, Université de Bretagne sud

Contact : Jacques Malenfant

Mél. : Jacques.Malenfant@univ-ubs.fr

Description : L'équipe ORCADE s'intéresse généralement aux langages et systèmes à objets répartis. Dans le domaine de la réflexivité, l'essentiel de ses travaux portent sur la formalisation du concept de réflexivité en tant que domaine en soi, ainsi que dans son expression dans les langages et les systèmes. Nous commençons actuellement des travaux sur les supports d'exécution réflexifs et leur contrôle depuis les langages et les applications. Dans ce cadre, nous cherchons à réaliser une version réflexive de la mémoire d'objets répartie DMeroon dont les choix d'implantation et les politiques de gestion pourraient être adaptées dynamiquement.

Équipe : «Objets Concurrents Reflexifs» — Thème Janus — Laboratoire I3S, Université de Nice-Sophia Antipolis & CNRS

Contact : Jean-Pierre Regourd

Mél. : rjp@unice.fr

Participants : C. Michel et J.-C. Pazzaglia.

Description : Ce projet se propose de prolonger les travaux de l'équipe *Objets Concurrents Réflexifs* de l'I3S par une généralisation de l'ap proche par composants. Une première partie de l'étude visera à individualiser complètement les composants qui sous-tendaient la démarche de composition des programmes réflexifs développée par J-C Pazzaglia [Paz97] et à appliquer ce tte démarche au langage Java. Pour ce qui est de l'implémentation de modèles réflexifs, le découpage du modèle de LActE/R [**?**] sera repris afin de définir un modèle d' implément ation directement basé sur la notion de composant.

Équipe : Systèmes et langages à objets – École des mines de Nantes

Contact : Pierre Cointe

Mél. : Pierre.Cointe@emn.fr

Participants : Jacques Noyé, Mario Südholt, Nouri Bouraqadi, Thomas Ledoux et Frédéric Rivard

Description : L'essence de la programmation réflexive est l'*ouverture*. Face aux problèmes d'adaptabilité, la programmation réflexive propose qu'une méthodologie générale non seulement transcende toutes les solutions *ad hoc* mais préserve l'ouverture des langages et des applications à des modifications ultérieures. Le but ultime de la réflexion appliquée au domaine informatique est donc de définir l'art et la science de la conception et de l'implantation à la fois des langages de programmation et des systèmes informatiques adaptatifs.

Les langages à objets sont des candidats naturels à l'étude de la réflexion [Coi96]. Les mécanismes d'héritage et de délégation propres à ces langages à objets permettent de définir des protocoles permettant d'organiser, de ré-utiliser et de composer ces méta-objets. Ces méta-composants génériques peuvent ainsi être utilisés afin de décrire les propriétés standard des langages à objets et leur décomposition en éléments de base. Dans la continuité de nos travaux précédentes nous souhaitons investiguer les domaines suivants :

1. Fredéric Rivard a consacré une grande partie de sa thèse à la définition d'un protocole de changement de classe dynamique permettant de rendre compte de l'évolution du comportement des objets [Riv97]. Il a proposé une implémentation de ce protocole dans une nouvelle version de ClassTalk appelée NeoClassTalk. Nous souhaitons maintenant utiliser NeoClassTalk pour étudier le problème de la composition dynamique des propriétés de classes [LC96], [MMC95], [BSRL98] et comparer «l'approche métaclasse» à la Smalltalk/CLOS à «l'approche interface» à la Java. Ce travail s'inscrit dans le cadre général de la composition de (méta)-objets qui constituera le cœur de la thèse de Noury Bouraqadi.
2. Nous sommes fortement convaincus que l'informatique distribuée se doit de considérer la réflexion afin d'intégrer de manière transparente de nouveaux mécanismes et de nouvelles stratégies de distribution. Le projet OpenCORBA applique la réflexivité intrinséque de Smalltalk à l'implémentation d'un ORB ouvert devant améliorer la flexibilité et la fiabilité des architectures CORBA. L'un des résultats principal de ce projet est d'avoir montré que les métaclasses explicites de ClassTalk permettaient de modéliser et séparer les différents aspects d'une architecture distribuée (répartition, communication, concurrence, tolérance aux fautes, sécurité, ...) tandis que le changement de classe dynamique de NeoClasstalk permettait de réaliser l'adaptabilité dynamique. Dans le prolongement de la thèse de T. Ledoux qui décrit en détail OpenCorba et son implémentation [Led98], nous souhaitons approfondir nos expérimentations et réaliser des "benchmarks" permettant d'évaluer le sur-coût introduit.
3. Nous nous intéressons à l'implémentation efficace des propriétes des langage réflexifs en adoptant deux approches différentes : support de la réflexion au niveau de la machine virtuelle par de nouvelles primitives et optimisation de programmes réflexifs en utilisant l'évaluation partielle (cf. groupe évolution des langages et de leur implémentation). Pour ce qui

concerne la première approche la collaboration avec OTI et l'accès à sa machine virtuelle est une aide indispensable. Nous participons à la la définition d'un compilateur générique permettant d'utiliser une même machine virtuelle comme support d'exécution à plusieurs langages (dont Smalltalk et Java).

4. Depuis 1996, nous avons a choisi d'utiliser en complément de Smalltalk, le langage Java, ses bibliothéques et ses environnements de développement (en particulier VisualAge for Java). Nous avons entrepris une étude systématique des aspects introspectifs de Java, et de leur mise en œuvre dans des applications avancées comme les applications réparties (java.rmi) et l'assemblage de composants visuels (java.beans).

Équipe : IRIT, Université de Toulouse

Contact : Patrick Sallé

Mél. : migeon@irit.fr

Participants : Annie Marcoux, Christine Maurel, Frédéric Migeon

Description : Réflexivité dans les sytèmes d'acteurs et d'entités concurrentes:

- Définition d'une architecture réflexive individuelle opératoire pour un acteur.
  - sémantique des différents composants de l'architecture
  - implémentation d'un prototype dans le langage d'acteurs *PlasmaII*
- Extension de l'architecture
  - généricité de l'architecture adaptée à différents systèmes concurrents
  - extension à la réflexivité de groupe

Références : [DDF96, BDFJ97, BG96, BSRL98, Coi96, DM95, DM98, DFP95, Gue97, JM95, LC96, Led98, Mal97, MDC96]

Équipe : I3S – Université de Nice Sophia Antipolis (équipe OCL)

Contacts : Philippe Lahire et Robert Chignoli

Mél. : lahire@unice.fr, chignoli@unice.fr

Description : Parallèlement à l'étude des liens entre classes (étude qui devrait aboutir à la définition d'un modèle de paramétrage des liens), nous souhaitons spécifier et développer un prototype qui autorise deux niveaux de programmation (méta et application) dans le but de permettre la définition ou la redéfinition par paramétrage, des types de liens nécessaires à la description d'une application ou d'une classe d'applications. Les principales caractéristiques de notre approche sont:

- l'utilisation du langage Eiffel3 comme modèle de départ des relations entre classes (héritage multiple et répété, clientèle, ...). Eiffel3 sera aussi le langage de développement du prototype.
- la réification des principaux concepts que l'on peut trouver dans les langages de programmation d'aujourd'hui, utilisés dans le monde industriel;
- l'offre de fonctions Méta pour définir la sémantique de classe et de liens pour un langage industriel existant ou futur. Ces fonctions permettront entre autre la définition de liens d'usages différents (vues, lien inverse, lien est-une-sorte-de, lien est-une-version-de, ...);

- l'intégration complète de la notion d'objet persistant de manière à permettre l'exécution de celui-ci soit dans son contexte d'origine (sémantique du langage utilisé dans le programme qui a créé l'objet), soit dans le contexte de l'application qui l'utilise (sémantique du langage utilisé dans le programme qui consulte ou met à jour un objet persistant).
- l'unification de ces différents aspects dans la réalisation d'une plateforme flexible qui devrait servir de cible à des compilateurs ou à des interprètes de langages qui partageraient la même représentation interne en offrant des syntaxes adaptées à des classes d'applications.

Références : [CCL97]

Références

[ABBF+98] I. Attali, R. Bastide, M. Blay-Fornarino, AM. Dery, and Ph. Palanque. Spécification formelle et approche objet pour les applications workflow. *TSI*, 27(2):181–209, February 1998.

[AC98] Michel Allemand and Christine Choppy. The Invoicing Case Study: An algebraic Specificati on. In *Comparing Systems Specification Techniques: "What q uestions are prompted by ones particular method of specification?"*, Nantes, France, March 1998.

[ACR97] I. Attali, D. Caromel, and M. Russo. Vers une sémantique formelle de java. In *Actes des Journées du GDR Programmation*, Rennes, Novembre 1997.

[Anc97] Davide Ancona. An algebraic framework for separate compilation. Technical Report DISI-TR-97-10, Dipartimento di Informatica e Scienze dell'Informazione - Genova, 1997.

[AR98] Pascal André and Jean-Claude Royer. The Invoicing System: Using TAG. In *Comparing Systems Specification Techniques: "What q uestions are prompted by ones particular method of specification?"*, Nantes, France, March 1998.

[AZ96] Davide Ancona and Elena Zucca. An algebraic approach to mixins and modularity. In Michael Hanus and Mario Rodríguez-Artalejo, editors, *Algebraic and Logic Programming, 5th International Conference, ALP'96*, volume 1139 of *lncs*, pages 179–193, Aachen, Germany, 25–27 September 1996. Springer.

[BBBR96] L. Bellissard, S. Ben Atallah, F. Boyer, and M. Riveill. Distributed application configuration. In *16th International Conference on Distributed Computing Systems*, pages 579–585, Hong-Kong, May 1996. IEEE.

[BBKR96] L. Bellissard, S. Ben Atallah, A. Kerbrat, and M. Riveill. Component-based programming and application management with olan. In J.P. Briot and J.M. Geib et A. Yonezawa, editors, *Workshop on Object-Based Parallel and Distributed Computation*, Tokyo, 1996. Lecture Notes in Computer Science (LNCS) 1107.

[BBM95] L. Bellissard, F. Boyer, and M.Riveill. Construction and management of cooperative distributed applications. In *International Workshop on Object Orientation in Operating Systems (IWOOOS'95)*, pages 149–152, university of Lund, Sweden, August 1995. IEEE.

[BC90] Gilad Bracha and William Cook. Mixin-based inheritance. *ACM SIGPLAN Notices*, 25(10):303–311, October 1990. *OOPSLA ECOOP '90 Proceedings*, N. Meyrowitz (editor).

[BDFJ97] L. Berger, A.-M. Dery, M. Fornarino, and O. Jautzy. Contribution : Interaction and Communication Models. In *ECOOP'97 Workshop Reader*. Spinger-Verlag, 1997.

[BG96] Jean-Pierre Briot and Rachid Guerraoui. Objets pour la programmation parallèle et répartie : intérêts, évolutions et tendances. *Technique et Science Informatiques (TSI), AFCET-Hermès, Paris, France*, 15(6):765–800, June 1996.

[BR96] L. Bellissard and M. Riveill. From distributed objects to distributed components: the olan approach. In *Workshop: Putting Distributed Objects to Work*, Linz, Austria, July 1996. ECOOP'96.

[BRS98] I. Borne, A. Romanczuk, and F. Stéfani. Towards a systematic object-oriented transformation of a merise analysis. In *Proceedings of the 2nd Euromicro Conference on Software Maintenance and Reengineering, CSMR'98*, Florence, Italy, March 1998. IEEE.

[BS97] V. Benzaken and X. Schaefer. Static integrity constraint management in object-oriented database programming languages via predicate transformers. In M. Aksit and S. Matsuoka, editors, *European Conference on Object-Oriented Programming (ECOOP'97)*, number 1241 in Lecture Notes in Computer Science, Jyvaskyla, Finland, June 1997. Springer-Verlag.

[BS98] V. Benzaken and X. Schaefer. Static management of integrity in object-oriented databases: Design and implementation. In *6th International Conference on Extending Database Technology (EDBT'98)*, Valencia, Spain, March 23-27 1998. Springer Verlag.

[BSRL98] M. N. Bouraqadi-Saâdani, F. Rivard, and T. Ledoux. Composition de Métaclasses. In *Journeés francophones des langages applicatifs, JFLA'98*, Como, Italy, février 1998.

[Car97] Luca Cardelli. Program fragments, linking, and modularization. In *Conference Record of POPL '97: The* 24$^{\mathrm{rd}}$ *ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, Paris, France, 15–17 January 1997. ACM.

[Cas97] G. Castagna. *Object-Oriented Programming: A Unified Foundation*. Progress in Theoretical Computer Science. Birkäuser, Boston, 1997. ISBN 3-7643-3905-5.

[CCL97] Robert Chignoli, Pierre Crescenzo, and Philippe Lahire. Ofl : un environnement ouvert pour programmer et métaprogrammer en eiffel. principes généraux de l'environnement. Technical report, I3S – UPRES-A 6070 CNRS –, 1997.

[CDPR98] Loïc Correnson, Etienne Duris, Didier Parigot, and Gilles Roussel. Generic programming by program composition (position paper). In *Workshop on Generic Programming*, Marstrand, Sweden, June 1998. conjunction with MPC'98.

[CHM^{+}98] C. Consel, L. Hornof, R. Marlet, G. Muller, S. Thibault, E.-N. Volanschi, J. Lawall, and J. Noyé. Partial evaluation for software engineering. *ACM Computing Surveys*, 1998. 1998 Symposium on Partial Evaluation - To Appear.

[Coi96] P. Cointe. Reflective languages and metalevel architectures. In M. Aksit, A. Black, L. Cardelli, J. Coplien, P. Cointe, et al., editors, *Strategic Research Directions in Object Oriented Progamming*. ACM, June 1996. ACM Computing Surveys 28A(4).

[Col] Philippe Collet. Oqual : a common language for assertions and queries in eiffel. In *TOOLS'EUROPE 97*, pages 240–255. edited with Tools 23, USA'97, Prentice Hall, 1997.

[Col97] Philippe Collet. *Un modèle fondé sur les assertions pour le génie logiciel et les bases de données : application au langage OQUAL, une extension d'Eiffel*. Thèse d'informatique, UNSA, Université de Nice Sophia Antipolis, Dec. 1997.

[CPS96] J-L. Colaço, M. Pantel, and P. Sallé. Cap: An actor dedicated process calculus. In *ECOOP'96 Workshop on Proof Theory of Concurrent Object-Oriented Programming*, May 1996.

[CPS97] J-L. Colaço, M. Pantel, and P. Sallé. A set-constraint-based analysis of actors. In *Proc. of the IFIP International conference on Forma l Methods for Open Object-based Distributed Systems*, July 1997.

[CR96] Philippe Collet and Roger Rousseau. Assertions are objects too! In Frieder Monninger, editor, *WOON'96 (First White Object-Oriented Nights Conference)*, pages 1–13. SIG Computer and Univ. St Petersburg, 1996.

[CV98] G. Castagna and J. Vitek. Towards a calculus of mobile computations. In *IEEE Workshop on Internet Programming Languages*, 1998.

[DDF96] A.-M. Dery, S. Ducasse, and M. Fornarino. Object and Dependency Oriented Programming in FLO. In *ISMIS'96 : 9=E8me International Symposium on Methodologies for Intelligent Systems*, LNAI, June 1996.

[DDHL95] H. Dicky, C. Dony, M. Huchard, and T. Libourel. Ares, adding a class and restructuring inheritance hierarchies. In *Actes des 11ièmes Journées Bases de Données Avancées, BDA'95*, pages 25–42, Nancy, 1995.

[DDHL96] H. Dicky, C. Dony, M. Huchard, and T. Libourel. On automatic class insertion with overloading. *ACM Sigplan Notice - Proceedings of ACM OOPSLA'96, Object-Oriented Programming Languages, Systems and Applications*, 31(10):251–267, October 1996. Egalement en Rapport de Recherche LIRMM No 95054, Février 1996.

[DFP95] S. Ducasse, M. Fornarino, and A.-M. Pinna. A Reflective Model for First Class Relationships. In *Proceedings of OOPSLA'95*, pages June265–280, 1995.

[DHL97] C. Dony, M. Huchard, and T. Libourel. Automatic hierarchies reorganization, an algorithm and case studies with overloading. Technical Report 97279, LIRMM, 1997. Version étendue de [DDHL96] - SOUMIS À PUBLICATION.

[DM95] F.-N. Demers and J. Malenfant. Reflection in logic, functional and object-oriented programming: a short comparative study. In *Actes du «IJCAI'95 Workshop on Reflection and Metalevel Architectures and their Applications in AI»*, pages 29–38, août 1995.

[DM98] F.-N. Demers and J. Malenfant. Réflexion monadique extensible. In S. A. Cerri and C. Queinnec, editors, *Actes de la conférence Journées Francophones sur les Langages Applicatifs, JFLA'98*, Collection Didactique, pages 31–45. INRIA, février 1998.

[Duc97] R. Ducournau. La compilation de l'envoi de message dans les langages dynamiques. *L'Objet*, 3(3):?, 1997.

[G.98] Vanwormhoudt G. Programmation par contextes en smalltalk. In *L'OBJET*, volume 3. Hermes, 1998.

[GBD97] Zahia Guessoum, Jean-Pierre Briot, and Michel Dojat. Des objets concurrents aux agents autonomes. In Joël Quinqueton, Marie-Claude Thomas, and Brigitte Trousse, editors, *Cinquièmes Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents (JFIADSMA'97)*, pages 93–107. Hermès, April 1997.

[GJP96] F. Guidec, J.-M. Jézéquel, and J.-L. Pacherie. An object oriented framework for supercomputing. *Journal of Systems and Software*, Special Issue on *Software Engineering for Distributed Computing*, June 1996.

[Gue97] Zahia Guessoum. DIMA : Une plate-forme multi-agents en smalltalk. *L'Objet*, 3(4), December 1997. Numéro spécial sur Smalltalk, édité par François Pachet et Hafedh Mili.

[JJ98] Babau J.P. and Sourrouille J.L. Expressing real-time constraints in a reflective object model. In *IFAC CEP (Control Engineering Practice)*, 1998. A paraître.

[JJL97] J.-M. Jézéquel and Pacherie J.-L. Operator design pattern, application to parallel computation. In *Collected papers from the PLoP '96 and EuroPLoP '96 Conferenc es*, number wucs-97-07. Washington University Department of Computer Science, February 1997.

[JM95] M. Jacques and J. Malenfant. Proto-Reflex: un langage à prototypes avec réflexion de comportement. In A. Napoli, editor, *Actes de la conférence Langages et Modèles à Objets, LMO'95*, pages 75–91. INRIA-Lorraine, octobre 1995.

[JT98] J.-M. Jézéquel and M. Train. *Software Components and Design Patterns.* Addison-Wesley, June 1998. To be published.

[LC96] T. Ledoux and P. Cointe. Explicit Metaclasses As a Tool for Improving the Design of Class Libraries. In *ISOTAS'96*, volume 1049 of *LNCS*, Kanazawa, Japan, Mar. 1996. JSSST-JAIST, Springer-Verlag.

[Led98] T. Ledoux. *Réflexion dans les systèmes répartis : application à CORBA et Smalltalk.* PhD thesis, École des Mines de Nantes, mars 1998. à paraitre en rapport de recherche de l'École des mines de Nantes.

[LF96] Shinn-Der Lee and Daniel P. Friedman. Enriching the lambda calculus with contexts: toward a theory of incremental program construction. In *International Conference on Functional Programming.* ACM, May 1996.

[Lhu98] Marc Lhuillier. *Une approche à base de composants logiciels pour la conception d'agents. Principes et mise en oeuvre à travers la plate-forme Maleva.* Thèse d'université, Université de Paris VI, February 1998.

[Mal97] J. Malenfant. Abstraction, encapsulation et réflexion dans les langages à prototypes. Technical Report 97-4-INFO, École des mines de Nantes, avril 1997. Mémoire d'Habilitation à diriger les recherches, Université de Nantes.

[MDC96] J. Malenfant, C. Dony, and P. Cointe. A Semantics of Introspection in a Reflective Prototype-Based Language. *Lisp and Symbolic Computation, Kluwer*, 9(2/3):153–179, mai/juin 1996.

[MDVS96] O. Michel, D. De Vito, and J.-P. Sansonnet. 81/2 : data-parallelism and data-flow. In E. Ashcroft, editor, *Intensional Programming II:Proc. of the 9th Int. Symp. on Lucid and Intensional Programming.* World Scientific, May 1996.

[Mic96] O. Michel. Introducing dynamicity in the data-parallel language 81/2. In Luc Bougé, Pierre Fraigniaud, Anne Mignotte, and Yves Robert, editors, *EuroPar'96 Parallel Processing*, volume 1123 of *LNCS*, pages 678–686. Springer-Verlag, August 1996.

[MMC95] P. Mulet, J. Malenfant, and P. Cointe. Towards a Methodology for Explicit Composition of Metaobjects. *Actes d'OOPSLA'95, ACM Sigplan Notices*, 30(10):316–330, octobre 1995.

[Paz97] Jean-Christophe Pazzaglia. *Méthodologie et outils pour la programmation réflexive dans les langages à objets.* Thèse d'université, December 1997.

[PCR98] Pascal Poizat, Christine Choppy, and Jean-Claude Royer. A new method for LOTOS specification. In *13th Workshop on Algebraic Development Techniques*, Lisbon, Protugal, april 1998.

[Que96] Christian Queinnec. DMEROON, overview of a distributed class-based causally-coherent data model. *Lecture Notes in Computer Science*, 1068:297–309, 1996.

[Riv97] F. Rivard. *Évolution du comportement dans les langages à classes réflexifs dynamiquement typés.* PhD thesis, École des Mines de Nantes, 1997.

[Sht97] M. Harito Shteto. *Technique à objets pour les algorithmes de configuration du réseau téléphonique commuté.* Thèse d'université, Université de Paris 6, 1997.

[VDBM97] J.-Y. Vion-Dury, L. Bellissard, and V. Marangozov. A component calculus for modeling the olan configuration language. In *Proceedings of Second International Conference on Coordination M odels and Languages (COORDINATION'97)*, Berlin, September 1997.

[VG97] Debrauwer L. Vanwormhoudt G., Carré B. Programmation par objets et contextes fonctionnels. application de crome à smalltalk. In Garlatti Serge Ducourneau Roland, editor, *LMO'97 (Langages et modèles à objets).* Hermes, Octobre 1997.

[YMB98] Min-Jung Yoo, Walter Merlat, and Jean-Pierre Briot. Modeling and validation of mobile agents on the web. *Simulation Series*, 30(1):23–28, January 1998.