# OASIS
# Objets Actifs, Sémantique, Internet et Sécurité

Proposition de projet commun
INRIA – UNSA – CNRS
en collaboration avec Eurécom*

10 juin 1999

**Résumé**

Dans le cadre des applications réparties (réseaux Internet et intranets, cartes à puce et terminaux), l'objectif du projet est de proposer des principes fondamentaux, des techniques et des outils pour la construction, l'analyse, la validation, la vérification et la maintenance de systèmes fiables.

*Suivant les termes des protocoles d'accord INRIA Eurécom UNSA CNRS en cours de signature.

# 1 Composition de l'équipe

### Responsable Scientifique

I. Attali, Chargée de Recherche INRIA

### Chercheurs

F. Baude, Maître de Conférences UNSA
D. Caromel, Professeur UNSA
B. Serpette, Chargé de Recherche INRIA

### Chercheur invité

G. Barthe, Professor Auxiliar, Universidade do Minho, Portugal, invité pour 6 mois.

### Ingénieur de recherche

F. Montagnac, à temps partiel

### Post-doctorants

H. Nilsson, mars 99 à février 2000 (PhD Linköping)
S. Yu, octobre 98 à septembre 99 (PhD Durham)

### Doctorants

C. Courbis, boursière INRIA (1ière année)
R. Guider, allocataire MENRT (3ième année)
C. Roudet, allocataire MENRT (4ième année)
M. Russo, allocataire MENRT (2ième année)
D. Sagnol, allocataire MENRT (3ième année)
J. Vayssière, boursier INRIA (à partir de mars 1999)

### Collaborateurs Extérieurs Eurécom

R. Molva, Professeur Associé
Y. Roudier, Assistant

# 2 Présentation générale et objectifs

### Thème de recherche

Dans le cadre des applications réparties (réseaux Internet et intranets, cartes à puce et terminaux), l'objectif du projet est de proposer des principes fondamentaux, des techniques et des outils pour la construction, l'analyse, la validation, la vérification et la maintenance de systèmes fiables.

### Mots-clés

**Principaux:**

programmation à objets répartie, sémantique et environnements, analyses de programmes, sécurité.

**Secondaires:**

code mobile, collecticiels, commerce électronique, communication de groupe, concurrence, distribution, fiabilité du logiciel, génie logiciel, Internet, méthodes formelles et preuves, optimisation de cache, parallélisme, sûreté, synchronisation, typage, validation.

### Domaines d'applications

applications embarquées, carte à puce, commerce électronique, télécommunications, téléphonie mobile.

## 2.1 Acquis

Le projet rassemble deux domaines d'expertise clairement identifiés:

- sémantique, environnements, compilation et analyses statiques [CDD+87, QS91, SS91, Ser92, SG95, ACW96, ACE96, ACEL96, ACR98], (http://www.inria.fr/croap/eiffel-ll, http://www.inria.fr/croap/java, http://www.inria.fr/croap/javacard);
- programmation à objets répartie [BVN91, Car93, CBR96, BBC+96, FB96, CKV98], (http://www.inria.fr/sloop/europa, http://www.inria.fr/sloop/javall);

De plus, les collaborateurs d'Eurécom apportent des compétences reconnues internationalement en sécurité des réseaux et des systèmes distribués [SM97, MT98, Mol99].

## 2.2 Objectifs

A partir de ces compétences, l'objectif du projet est de créer une synergie sur les thématiques suivantes:

- environnement fondé sur la sémantique pour le développement, l'analyse et la vérification d'applications réparties et communicantes liées à l'Internet (par exemple Java, Java Card);
- construction de bibliothèques facilitant la programmation et la maintenance d'applications multi-threadées, distribuées, sécurisées, en particulier pour les applications collaboratives et le commerce électronique.

Ces deux thématiques ne sont pas disjointes, au contraire:

- les analyses statiques des programmes à objets sont utiles pour la construction d'applications réparties et sécurisées; de plus des compétences en sémantique sont indispensables pour prouver la conservation de la sémantique séquentielle lors de la distribution d'une application;
- les modèles à objets actifs (e.g. pour Java//: threads transparents, objets distants, appels asynchrones avec futurs transparents) sont utiles lors de l'étude et la description du modèle d'exécution Java Card.

D'autre part, l'avènement d'Internet a créé la nécessité d'étendre, de manière fondamentale, les conceptions existantes de mobilité et sécurité. Nous disposons de compétences, d'outils et de méthodes qu'il nous parait intéressant de mettre à profit dans un domaine d'application devenu majeur. Nous souhaitons aborder le domaine de la sécurité en établissant des collaborations avec des experts reconnus dans le domaine, en particulier l'équipe de Refik Molva (Eurécom) et celle de Dominique Bolignano (Dyade). D'une façon concrète, nous allons débuter cette collaboration par le co-encadrement de stages de DEA correspondant à deux thèmes précis de recherche qui devraient créer une synergie entre nos compétences (voir Sections 4.1.4 et 4.2.4).

### Objectifs à trois ans

Le projet envisage les résultats scientifiques et les réalisations pratiques suivants:

- construction d'outils de visualisation, analyse, vérification, et transformation de programmes Java et Java Card (voir Section 4.1.1);
- construction d'outils et de méthodologies pour la modélisation et la vérification de politiques de sécurité (voir Section 4.1.4);
- analyses statiques de programmes Java et optimisations (voir Section 4.1.2).
- De plus, un contrat passé (avec Sun Microsystems) nous encourage à poursuivre les travaux autour du modèle et de la bibliothèque Java// en particulier vers des mécanismes de communication de groupe et de migration sécurisés (voir Sections 4.2.3 et 4.2.4).

Ces recherches auront comme cadre trois contrats de recherche (signés ou en cours de signature):

- contrat Bull/CP8;
- contrat Oppidum avec Bull et Schlumberger;
- contrat The Open Group.

.

### Objectifs à long terme

Le projet mènera des recherches fondamentales dans les domaines suivants:

- spécifications, environnements et vérifications (voir Section 4.1.3);
- étude conceptuelle et formelle des modèles, langages et programmes à objets (voir sections 4.2.1 et 4.2.2)

### Structure du présent document

Dans la suite du document, nous abordons tout d'abord la problématique de notre projet ainsi que nos acquis (Section 3), puis développons notre programme de recherche suivant les deux axes principaux déjà cités (Section 4). La Section 5 situe nos objectifs par rapport aux équipes de l'Institut, puis plus largement en France et dans le monde. Enfin, dans une dernière partie (Section 6), nous citons quelques élements témoignant de notre visibilité dans le monde académique et le monde industriel avec en particulier nos collaborations formellement établies, les actions conduites avec des partenaires industriels, ainsi que notre implication dans le domaine de la formation.

# 3 Problématique

L'explosion de l'Internet et les perspectives de son impact dans la vie quotidienne, notamment par le biais du commerce électronique, ont créé un nouveau modèle de société caractérisée par la capacité de chacun à accéder, fournir ou modifier des quantités vertigineuses d'information. Ce nouveau phénomène, qui s'accompagne d'une modification physique des réseaux informatiques et d'une évolution de leur utilisation, invalident les modèles existants de systèmes distribués et demandent une nouvelle analyse des mécanismes en jeu. En particulier, l'avènement d'Internet a créé la nécessité d'étendre, de manière fondamentale, les conceptions existantes de la mobilité et la sécurité.

Java a été conçu avec l'espoir d'apporter, dans le cadre d'un langage de programmation, une réponse pratique aux questions de mobilité et sécurité. Si ce pari-là n'est pas totalement réalisé, en particulier au niveau de la sécurité (de nombreuses brèches ont été découvertes), Java demeure néanmoins un grand succès, au niveau de l'Internet bien sûr, et plus récemment dans le domaine du commerce électronique et de la carte à puce, où Java a été choisi pour remplacer le langage assembleur, avec la définition du noyau Java Card exécutable sur une carte à puce avec de faibles ressources.

Les principaux atouts de Java sont sa grande portabilité, grâce à l'interprétation d'un code intermédiaire (bytecode) sur une machine virtuelle indépendante de la plate-forme utilisée, sa relative sécurité, grâce à un système de types évolué et un vérificateur de code intermédiaire, et son haut-niveau, grâce à l'utilisation de concepts issus, pour la plupart, de la programmation distribuée à objets.

Le succès de Java, ainsi que les difficultés rencontrées, suggèrent une étude approfondie des concepts sous-jacents, ainsi que de leurs connexions. Dans les paragraphes suivants, nous abordons ces problématiques, en nous intéressant plus particulièrement aux aspects "Génie Logiciel", et plus concrètement au rôle d'outils interactifs de développement, d'analyse et de vérification de logiciels.

**Programmation à objets et concurrence** Le paradigme objet, même s'il date des années 70, reste un des aspects des langages de programmation les plus étudiés de nos jours. Ces dernières années, avec l'apparition du langage Java, on a pu observer une recrudescence de l'activité autour de la méthodologie objet. Autant le concept se veut universel, autant les variations des modèles et leurs implémentations possèdent des propriétés spécifiques souvent mal définies: sous la même terminologie objet, se retrouvent des thèmes particuliers comme l'héritage (simple ou multiple), le sous-typage, la surcharge, etc.

D'autre part, les aspects programmation concurrente (en particulier, multithreading, accès concurrents) viennent apporter un degré supplémentaire de complexité; les programmeurs peuvent être confrontés aux questions suivantes: est-ce que plusieurs threads accèdent à un même objet en même temps ? Quels sont les objets partagés par un thread donné ? Y a-t-il des appels récursifs, directs ou à travers des cycles?

Le mélange de l'ensemble de ces traits peut faire apparaître des cas où la spécification du langage reste floue et pour lesquels la construction d'applications et leur mise au point restent délicates.

Nous sommes donc convaincus de l'utilité d'environnements de développement dans lesquels le programmeur sera guidé dans la mise au point de ses applications, par des outils interactifs, graphiques, utilisant la sémantique du langage. A ce titre, Java Card est un bon exemple par la taille réduite des applications, et permet d'envisager le traitement de problèmes insolubles dans le cas plus général d'applications contenant des millions de lignes de code.

**Programmation répartie** Le langage Java peut être également abordé comme un langage très prometteur pour la programmation distribuée sur un réseau; l'arrivée de Java a laissé espérer que l'on pourrait distribuer des applications haute-performance sur le réseau Internet, premier pas vers le méta-computing. Malheureusement, les composants Java standards tels que RMI (Remote Method Invocation) n'aident en fait pas à construire de manière transparente des applications séquentielles, multi-threadées, ou distribuées, en permettant l'exécution d'une même application sur une architecture multi-processeurs à mémoire partagée aussi bien que sur un réseau de stations de travail (intranet, Internet), ou encore n'importe quelle combinaison hiérarchique des deux.

La question est donc comment construire à partir des outils standards (par exemple threads, RMI, etc), des modèles et bibliothèques facilitant la programmation distribuée. Un des domaines d'application particulièrement représentatif est la construction et l'évolution de collecticiels (plusieurs utilisateurs distants travaillent de manière coordonnée à une même tâche) dans lesquels se posent des problèmes d'élection, de synchronisation, de répartition des calculs, etc.

**Sécurité** Plusieurs domaines d'applications du réseau Internet et des cartes à puces, dont le commerce électronique, demandent la protection de données précieuses (numéros de compte en banque, codes confidentiels, etc) qui, en un moment déterminé, seront disponibles sur des plate-formes reliées physiquement à un vaste réseau. Ici le besoin de sécurité maximale est réel et partagé:

– les sociétés de services (banques, administration etc.) doivent avoir la garantie qu'une application vérifie des propriétés de sécurité, définies dans une politique de sécurité globale (par exemple identification, intégrité, confidentialité, ou non-répudiation) et pourra ainsi résister à des attaques systématiques;

- leurs clients doivent être assurés que les informations qu'ils fournissent lors d'une demande de services ne seront pas utilisées à mauvais escient ou détournées vers un tiers.

Le même problème apparait dans le cadre de la sécurité des applications distribuées par l'utilisation de code mobile. Les menaces sont doubles: menace sur l'environnement d'exécution par un code mobile hostile et menace d'un environnement d'exécution malveillant envers un code mobile et ses données. Plusieurs solutions basées sur une stratégie défensive, dont l'approche "sandbox" de Java ou l'utilisation de techniques cryptographiques, sont actuellement proposées. Si elles assurent un degré élevé de sécurité, ces solutions sont contraignantes et portent détriment à l'efficacité et la commodité (par exemple, il est souvent conseillé de désactiver Java et JavaScript avant de visiter un lien inconnu, politique peu appliquée en pratique).

Il semble donc nécessaire d'améliorer la sécurisation d'applications collaboratives, par le biais de politiques simples et transparentes qui n'affectent pas les performances des applications. Cela demande sans doute une spécification plus systématique des politiques de sécurité, qui sont souvent formulées en termes génériques et ne tiennent pas compte des spécificités des applications concernées.

Là encore, des outils de vérification seront particulièrement utiles aux programmeurs pour garantir que tel programme pourra être exécuté sur une carte à puce sans atteinte à la sécurité ou que tel autre gardera des données intègres en cas de retrait inopiné de la carte.

## Démarche

Notre projet s'attachera à répondre à ces problématiques par trois approches complémentaires :

1. Théorique : étude et définition rigoureuse des modèles et langages à objets; cette direction inclut des définitions sémantiques, l'étude de leur équivalence dans un contexte séquentiel ou distribué, des analyses de programmes et des vérifications de propriétés des modèles et des programmes.
2. Appliquée : nous trouverons notre source d'inspiration dans des problèmes concrets et actuels, que nous aborderons de manière aussi directe et complète que possible, et pour lesquels nous proposerons des solutions techniques; cette approche, pour laquelle les recherches sont guidées par les buts, permettra entre autres d'aborder des problèmes aussi différents que l'exécution de programmes Java Card sous res-

sources limitées, la distribution d'un ensemble d'objets sur un réseau, ou encore la définition de politiques de sécurité.
Cette approche permettra également de mettre à jour des transformations de programmes applicables dans un contexte objet (inlining, unrolling, etc) et trouvera un terrain d'application naturel avec le langage Java Card.

3. Pragmatique : nous souhaitons accompagner et valider les modèles et solutions proposés par des expérimentations en vraie grandeur; par exemple, les définitions sémantiques seront toujours démontrables et trouveront leur utilisation dans des environnements de développement; de plus, nous souhaitons implémenter et diffuser une bibliothèque pour la programmation répartie sécurisée.

Ces trois approches sont bien entendu complémentaires:

- les applications guident nos recherches et permettent de resserrer naturellement nos différents axes;
- les implémentations valident et rendent visibles des modèles théoriques;
- d'autre part, une implémentation ne peut pas être réussie sans une étude rigoureuse préalable.

## Acquis

Isabelle Attali étudie la sémantique naturelle [Kah87] à la fois du point de vue des concepteurs (expressivité) et du point de vue des utilisateurs (évaluation efficace [AFZ88, Att89, AC90, ACG92]). Ces recherches et expérimentations ont été menées en utilisant le langage Typol [Des88] et le système Centaur [BCD$^+$88]. Parmi les possibles champs d'application de la sémantique naturelle, trois ont été particulièrement riches sur le pouvoir et les limites de la sémantique naturelle: la manipulation des documents structurés [AAFZ97], les langages à objets parallèles [ACE96, ACEL96, ACR98], les langages fonctionnels parallèles [ACW96, ACGW96]. Ces deux dernières directions ont été étudiées en collaboration avec Denis Caromel.

De plus, d'autres modélisations ont également été explorées (avec le langage Eiffel// [Car93]) afin de montrer l'équivalence de programmes séquentiels et parallèles; en particulier, nous avons étudié comment modéliser des programmes Eiffel// en $\pi$-calcul [MPW92] et en TLA [Lam94]. Certaines de ces spécifications ont servi de base à la preuve de propriétés du modèle Eiffel// [ACE98b, ACL99].

D'autre part, nous avons, avec la thèse de doctorat de Romain Guider, acquis une certaine expérience des analyses statiques, qui ont comme champ d'application les optimisations mais aussi la parallélisation et la répartition de programmes Java [ACG98]. Enfin, dans le cadre de la thèse de doctorat de Christophe Roudet, nous avons conçu et implanté TrfL, un langage dédié aux transformations de programmes [APR97], inspiré, dans une certaine mesure, de la sémantique naturelle.

Ces compétences en spécifications, compilation et environnements sont renforcées par l'arrivée de Bernard Serpette, spécialisé dans la définition et l'implémentation des langages de programmation (compilation, environnements de développement), notamment fonctionnels et à objets [SS91, QS91]. Bernard Serpette a également étudié avec succès le domaine des analyses statiques pour l'aide à la programmation et l'optimisation de programmes [Ser92, SG95].

Denis Caromel est spécialiste en programmation concurrente, répartie et parallèle dans le cadre des langages à objets; ces recherches sont menées aussi bien sur des aspects plus théoriques comme les sémantiques formelles, ou les analyses statiques et transformations pour la répartition automatique ou semi-automatique, que sur des aspects plus pragmatiques comme la conception de langages et de méthodes de programmation [Car93], ou la construction de bibliothèques pour le parallélisme [CKV98]. Il est co-responsable du groupe de travail "Concurrence et applications" du Java Grande Forum (voir http://www.javagrande.org).

Françoise Baude travaille depuis la création du projet SLOOP (1995) avec Denis Caromel. Elle a participé à l'élaboration de C++// et - indirectement à celle - de Java// [BBC+96]. Elle s'est tout particulièrement intéressée aux problèmes de leur support d'exécution [FB96] (mise en œuvre portable, optimisations, etc). Ses travaux antérieurs autour des modèles, outils pour le calcul distribué (acteurs [BVN91], formalisme Bird-Merteens [BS94], bibliothèques de communication, telles PVM) sont des atouts pour orienter nos recherches vers de nouvelles technologies (par exemple, les agents mobiles).

Shen-Wei Yu, en séjour post-doctoral jusqu'en septembre 1999, vient de terminer un PhD à l'Université de Durham sur la vérification mécanique de programmes concurrents et apporte des compétences très importantes en systèmes de preuves et model-checkers.

Henri Nilsson viendra rejoindre notre groupe en mars 1999 pour un séjour post-doctoral d'un an. Il a soutenu en Mai 1998 un PhD à l'Université de Linköping sur la mise au point déclarative de langages fonctionnels paresseux. Nous comptons sur lui pour apporter un nouvel éclairage et des compétences pour les aspects mise au point et simulation d'applets Java Card.

Enfin, Gilles Barthe, chercheur invité pour 6 mois dans le projet, non seulement apporte ses compétences en compilation et optimisation de programmes fonctionnels et à objets [BKR97, BHS97, BHT98, Bar98b], mais aussi son expertise en langages typés et systèmes de preuves [BRB96, Bar98a, Bar98c]. Nous espérons que le statut temporaire de sa présence au sein de notre projet pourra se transformer en intégration permanente en 1999 avec l'obtention d'un statut de chercheur ou enseignant-chercheur.

D'autre part, l'équipe NSTEAM de Refik Molva étudie, au sein de l'institut Eurécom, les problèmes de sécurité émergents dans les systèmes distribués et les réseaux, et conçoit des solutions par une application directe ou dérivée de la cryptographie. Actuellement la recherche dans NSTEAM se concentre sur deux axes: la sécurité du code mobile et les protocoles de sécurité multipoint. Les compétences de R. Molva sont reconnues internationalement en sécurité des réseaux [Mol99, MT98, SM97, BGH+95, MST94]; de plus, R. Molva est également spécialiste des techniques formelles de vérification puisqu'il a, lors de sa thèse de doctorat au LAAS, sous la direction de Michel Diaz, conçu, en utilisant des techniques formelles, un analyseur de protocoles pour architectures multi-couches.

Les conditions sont donc réunies pour favoriser une synergie entre les différentes thématiques en présence.

# 4 Programme de Recherche

Nous détaillons ici chaque objectif, sur les moyen et long termes, et identifions les membres du projet impliqués.

## 4.1 Spécifications, environnements, analyses et vérifications

### 4.1.1 Environnements de développement et vérification pour Java et Java Card

**Participants**
I. Attali, G. Barthe, F. Baude, D. Caromel,
C. Courbis, H. Nilsson, M. Russo, S. Yu.

Nous souhaitons, à partir d'une description formelle des sémantiques des langages Java et Java Card, dériver un ensemble d'outils pour la simulation, la mise au point, et la vérification d'applications.

Les recherches s'orienteront vers des directions communes (Java et Java Card) pour ce qui concerne la simulation, et les aides à la mise au point:

- spécification complète de la sémantique dynamique du langage puis simulation de l'exécution du programme par interprétation de la sémantique;
- fonctions de mise au point: poser des points d'arrêt, exécuter pas à pas, modifier les valeurs des variables, sauvegarder puis restaurer le contexte global d'exécution, afin de faciliter la mise au point des applications;

Certains points sont spécifiques à Java:
- abstraction de la visualisation de la topologie du graphe d'objets (pour passer à l'échelle dans le cas d'applications réelles);
- contrôle des threads (gestion de l'entrelacement, suspension-reprise d'un thread ou d'un ensemble de threads, etc);

D'autre part, les particularités de Java Card nous entraînent sur un terrain spécifique, notamment par l'existence d'un environnement d'exécution dédié sur carte à puce, d'un lecteur de carte qui sert d'interface avec l'utilisateur, et de communications via des APDUs (Application Protocol Data Unit). La sémantique opérationnelle du langage Java Card doit donc être associée aux spécifications de ces différents éléments, et devrait aboutir à la construction d'un environnement dérivé, qui permettra, en plus des fonctionnalités précédentes, de:

- simuler le fonctionnement d'une (ou plusieurs) applications, sur une station de travail, avant le chargement sur la carte à puce. Les scénarii de tests, décrits dans un format à définir, pourront être construits de manière interactive, sauvegardés et rejoués à volonté; ils seront également utilisés par un testeur (tests après installation sur la carte).
- visualiser, de manière textuelle et graphique, l'ensemble des objets impliqués (attributs, références), notamment l'utilisateur, le lecteur de carte, les applets s'exécutant sur la carte.
- faciliter la programmation des communications via des APDUs dont le format est particulièrement ésotérique (codes octets); cette aide peut se placer en amont (programmation via des appels de méthode) et en aval (visualisation lisible des APDUs).
- synthétiser une approximation de l'encombrement mémoire nécessaire à une exécution donnée d'un programme afin de savoir à l'avance si le chargement et l'exécution de cette application sur une carte seront possibles.
- prendre en compte des interruptions (retrait de la carte, coupure de courant) et sauvegarder un contexte cohérent ;

De plus, des outils de vérification sont nécessaires: par exemple, on devra s'assurer que deux applets censées partager certains objets mais pas d'autres respecteront ce cloisonnement et que les objets confidentiels d'une applet ne pourront ni être lus, ni modifiés par l'autre applet.

Ces outils pourront permettre au programmeur de vérifier que son applet est bien conforme à une spécification. Cette spécification permettra de définir des propriétés de l'applet en termes d'états et de changements d'états dus à des commandes. Les états sont exprimés par des variables d'états (logiques, entières). Un des formalismes possibles pour décrire ces propriétés pourrait être un sous-ensemble de TLA. Ces spécifications devront ensuite être validées (manuellement ou sur machine) afin d'assurer des propriétés de sécurité des applets.

### 4.1.2 Analyses statiques et transformations de programmes à objets

**Participants**
I. Attali, G. Barthe, D. Caromel, B. Serpette,
R. Guider, C. Roudet.

Le paradigme objet introduit une forte composante dynamique : si un objet **o** de type statique **T** appelle une méthode (virtuelle) de nom **m**, les deux seules propriétés utilisables sont d'une part que le type dynamique **Td** de **o** est un sous-type de **T** et d'autre part que le type de la méthode **m** de **Td** est un sous-type de la méthode **m** de **T**. Ceci implique que le flot de contrôle d'un programme partiel est rarement défini, or la connaissance de ce flot de contrôle est généralement un prérequis des analyses statiques traditionnelles.

Dès lors, deux voies de recherche peuvent être envisagées :

- L'analyse de programmes complets. Dans ce contexte, on peut trouver une approximation des types dynamiques et, ainsi, reprendre les travaux sur les analyses statiques traditionnelles. Cette voie prend tout son sens pour les outils de vérification de propriétés de programmes Java Card.
- La paramétrisation par des types dynamiques. Ceci revient à expliciter le polymorphisme inhérant à l'appel de méthode : pour chaque nom de méthode il faut construire un ensemble d'associations entre des types dynamiques d'entrées et une valeur abstraite (résultat de l'analyse). Cette voie, à notre avis peu explorée, nous semble un bon sujet de recherche.

Plus précisement nous envisageons les analyses statiques suivantes :

- Dans le cadre de la thèse de Romain Guider, nous étudions comment paralléliser des programmes à objets de manière automatique ou semi-automatique; il s'agit de découvrir, grâce à des techniques d'interprétation abstraite, des propriétés du graphe d'objets qui permettront de comprendre comment introduire des activités concurrentes dans des programmes Java. Cette étude comporte une analyse du flot de contrôle et une représentation abstraite du graphe d'objets; les deux problèmes n'étant pas indépendants du fait de la liaison dynamique.

  Ces mêmes techniques peuvent être exploitées pour optimiser la compilation Java; par exemple, en exploitant des informations sur la durée de vie des variables, ou en supprimant des appels virtuels (inférence de type), voire l'inlining d'objets ou la spécialisation.
- Analyse temporelle. Cette analyse consiste à construire un graphe acyclique (correspondant au temps) et à associer à chaque noeud de l'Arbre de Syntaxe Abstraite un couple dans le graphe construit (début et fin d'exécution). Ainsi on pourra répondre aux questions du type : telle

action est-elle nécessairement faite avant telle autre ? Par suite, cette analyse aidera à la vérification des programmes Java Card.

Toutes ces analyses permettent de transformer le programme source. Ces transformations pourront être exprimées dans le formalisme TrfL conçu dans le cadre de la thèse de C. Roudet (des expérimentations autour du langage Java sont d'ailleurs en cours).

### 4.1.3 Spécifications sémantiques et vérifications

**Participants**
I. Attali, G. Barthe
S. Yu.

La spécification formelle des langages de programmations, telle qu'elle est conçue en sémantique naturelle, permet non seulement une description précise et sans équivoque des langages concernés, mais aussi de bénéficier d'un certain nombre d'environnements pour l'implémentation ou l'étude de ces langages:

- les environnements de programmation, tel *Centaur*, ont la capacité d'engendrer certains outils à partir de la spécification formelle d'un langage;
- les environnements de preuves, tel *Coq*, offrent un cadre formel dans lequel les propriétés du langage peuvent être vérifiées;
- les model-checkers, tels *Autograph, CADP, Concurrency Workbench*, permettent d´établir automatiquement des propriétés de systèmes concurrents, et sont couramment utilisés pour analyser et vérifier formellement la correction de protocoles de communication ou de circuits électroniques.

Quoique chaque famille dispose de logiciels performants, les connexions entre logiciels de ces différentes familles restent à ce jour insuffisamment explorées. De plus, les environnements de preuves tels Coq reposent sur des logiques très générales, et possèdent un nombre limité de tactiques et librairies pour des logiques plus spécifiques, telles les logiques qui s'utilisent dans les preuves de correction de programme. Nous nous proposons, en marge de nos activités plus appliquées, d'aborder sur le long terme les problèmes suivants:

- la connexion entre descriptions sémantiques et outils de preuve.
  Un outil de traduction a été développé pour traduire le couple (syntaxe abstraite, sémantique naturelle) vers des définitions de types inductifs dans le système Coq [Ter95]. Cet outil a été utilisé sur une partie du compilateur Esterel [KT96], puis plus récemment sur un compilateur d'un langage jouet impératif [Ber98].
  Nous souhaitons poursuivre les efforts entamés pour associer les descriptions sémantiques à des preuves de propriétés liées à ces descrip-

tions, en collaboration avec le projet Coq dans le cadre de la future action de développement SmartTools.
- la connexion entre model-checkers et environnements de preuves. Dans le cadre de son PhD, Shen-Wei Yu a intégré des techniques de model-checking au sein du système Lego [YL97, Yu99], ce qui permet à la fois de générer automatiquement des preuves dans le cas de systèmes finis et d'étendre la puissance du model-checking à des systèmes infinis. Nous souhaitons poursuivre cette intégration, par exemple au sein du système Coq.

Les outils développés serviront à l'étude et la vérification de programmes Java/Java Card. En contrepartie, nous espérons que la vérification de programmes Java nous permettra de contribuer à la conception d'un langage spécifique mieux adapté à nos objectifs.

#### 4.1.4 Outils et méthodologie pour la modélisation et la vérification de politiques de sécurité

Les travaux autour de Java seront effectués en collaboration avec Eurécom, et seront amorcés sous la forme d'un co-encadrement de stagiaire de DEA.

**Participants**
I. Attali, G. Barthe, D. Caromel, R. Molva, Y. Roudier,
C. Courbis, M. Russo, S. Yu.

Nous souhaitons étudier d'une part des critères qui garantissent certaines propriétés de sécurité, et d'autre part les outils qui pourraient permettre de formaliser ces propriétés. Le cadre de ces recherches s'appliquera naturellement à Java et Java Card.

L'ensemble de ces propriétés constitue ce qu'on appelle une "politique de sécurité", qui prend en compte la nature des acteurs agissant sur le système (droits d'un acteur en fonction du moment, de son historique, de l'état du système, etc.). Ces propriétés sont spécifiques (à un acteur) et génériques (par exemple relatives au flux d'information). Une politique de sécurité peut par exemple garantir l'intégrité, la confidentialité des données, ou encore la non-répudiation.

Après avoir étudié les outils de formalisation (logique temporelle, analyses statiques, calcul des constructions inductives, ...), nous formaliserons une politique de sécurité; cette formalisation s'effectuera en relation avec la sémantique opérationnelle des langages Java et Java Card. En effet, nous souhaitons pouvoir associer cette formalisation à des outils de vérification et valider notre démarche par des jeux de tests, pour vérifier qu'un programme donné est conforme à une politique de sécurité particulière.

Le domaine de recherche est relativement nouveau et ambitieux, il semble donc difficile d'avoir une vision à long terme des développements attendus. On pourrait par exemple, à partir d'une spécification de politique de sécurité de haut-niveau générer automatiquement l'implémentation de cette politique (par exemple sous la forme d'appels à l'API sécurité de Java 1.2). D'autre part, nous pensons qu'il serait intéressant de découvrir d'une part, l'impact de la description sémantique sur les vérifications possibles, et d'autre part l'impact des vérifications à réaliser, dans le cadre d'une politique de sécurité donnée, sur la description sémantique.

## 4.2 Objets répartis

### 4.2.1 Etude formelle des modèles à objets distribués

**Participants**
I. Attali, D. Caromel, B. Serpette,
R. Guider, J. Vayssière.

Dans le cadre de travaux antérieurs, nous avons proposé, étudié et enrichi progressivement un modèle de programmation à objets distribués [Car93, CR95, CBR96, CKV98] qui permet, entr'autres, de rendre possible la réutilisation dans le cadre de la programmation parallèle, distribuée, et concurrente. Cela se traduit en particulier par un comportement parallèle qui prolonge aussi souvent que possible celui existant en séquentiel.

On retrouve naturellement ce trait dans les aspects sémantiques formelles. D'une certaine manière, des systèmes distribués obtenus à partir d'un système séquentiel existant conservent en grande partie leur sémantique séquentielle. Dans le cas de systèmes construits directement, leur sémantique reste proche d'une sémantique séquentielle. Nous avons pu d'ores et déjà formaliser ces propriétés dans un certain nombre de travaux [Lip96, ACE98a, ACL99, ACE98b], soit sur des sous-ensembles des langages considérés, soit sur des études de cas.

Nous allons poursuivre ces études sur les modèles à objets pour la concurrence, le parallélisme et la distribution, et en particulier rechercher un résultat général qui définirait que, sous certaines conditions, un système objet séquentiel est équivalent à un système obtenu en rendant certains objets actifs, c'est à dire parallélisables ou distribuables. Pour l'instant cette équivalence est réduite au cas où le programme est de type structure de donnée récursive, avec la totalité des objets devenant actifs. Nous souhaitons en particulier l'étendre à tous les systèmes dont la topologie ne présente ni cycle, ni DAG, ni instruction *Awaited*.

D'autre part, nous souhaitons comparer et relier ces équivalences à celles existant dans le cadre des langages fonctionnels. En effet, les aspects *attente-par-nécessité* de nos modèles objets semblent en rapport direct avec les calculs de type paresseux, et l'on peut imaginer que, dans ce cadre, des résultats des langages fonctionnels puissent être utilisés pour la programmation à objets.

Notons enfin les liens existant entre ces recherches et celles liées aux analyses statiques et transformations de programmes (cf. Section 4.1.2): les propriétés recherchées ici se situent en amont et conditionnent une partie des analyses statiques réalisées.

### 4.2.2 Implémentation des langages à objets et optimisations

**Participants**
D. Caromel, B. Serpette,
D. Sagnol.

La connaissance que la communauté a acquise sur l'implémentation efficace de l'héritage simple (deux à trois indirections pour l'appel d'une méthode) est le tronc commun d'où fleurissent de nouveaux traits sémantiques comme l'héritage multiple, la surcharge, le sous-typage, les fonctions génériques, etc. Ces nouveaux traits ont généralement, selon les langages considérés, des sémantiques divergentes, conséquence de choix d'implémentation.

Nous nous attacherons à expliciter et comprendre ces divergences, et à étudier l'implémentation des langages à objets.

Une première étape consistera en l'étude de l'existant en comparant les implémentations de C++, OCaml et Java. Il n'est pas question de définir un nouveau modèle théorique des langages à objets mais plutôt de trouver les frontières entre l'efficacité et l'expressivité dans les langages existants.

Cette étude devrait aboutir à l'extraction d'un nouveau langage intermédiaire dédié aux langages à objets et permettre ainsi la spécification d'optimisations indépendantes du langage source. De ce point de vue, il parait intéressant de vérifier qu'une nouvelle interprétation des instructions de la machine virtuelle Java (i.e. réécriture partielle de l'interpréteur du byte-code) ne permettrait pas d'exprimer des traits sémantiques manquant à Java (héritage multiple).

Ainsi, via ce langage intermédiaire, nous espérons introduire une nouvelle implémentation de l'héritage multiple proche de la délégation des langages d'acteurs, un peu moins efficace que C++, mais moins restrictive.

### 4.2.3 Bibliothèque pour la répartition

**Participants**
F. Baude, D. Caromel, B. Serpette,
D. Sagnol, J. Vayssière.

Nous développons un modèle de programmation concurrente orienté objet, et une implémentation sous la forme d'une bibliothèque 100 % Java

[CV98, CKV98] (Java//, http://www.inria.fr/sloop/javall), faisant suite aux travaux autour d'Eiffel// [Car93] et C++// [CBR96]. Les qualités essentielles du modèle proposé sont:

- de rendre transparent à l'utilisateur le fait que:
  - les objets de son application sont ou non distants,
  - et sont supportés ou non par des threads,

  permettant ainsi la réutilisation de code, par exemple pour un déploiement sur Internet (un de nos souhaits étant de parvenir à une conservation de la sémantique, voir Section 4.2.1);
- de fournir un modèle haut niveau de gestion de la concurrence entre objets actifs.

Le jeu de primitives que fournit le MOP (Méta Object Protocol) est utilisable pour la définition de mécanismes plus élaborés que le "simple" appel distant de méthode, utile pour le développement d'applications distribuées par exemple les collecticiels. En particulier, nous introduirons du calcul collectif portant sur des groupes d'objets actifs, ces groupes pouvant être définis explicitement, ou mieux implicitement à l'aide d'informations de typage par exemple. Ceci implique des communications de groupe, dont la conception sera réalisée de telle sorte que l'implémentation puisse se baser sur du multicast de la couche réseau ou transport, par exemple le protocole RM (Reliable Multicast) en cours de spécification. De même, nous introduirons un mécanisme de migration d'objets actifs, dont la gestion pourra être – grâce au MOP – totalement transparente. C'est également à ce niveau que la sécurisation d'échanges de données sera introduite (voir Section 4.2.4).
L'implémentation du MOP est aussi le lieu idéal pour:

- injecter des optimisations d'exécution, comme par exemple le recouvrement du transfert de certains paramètres d'un appel distant de méthode, avec l'exécution de cette méthode distante [BCFS98]. Nous comptons persévérer dans cette voie.
- faire profiter la bibliothèque, et ce sans nuire à sa portabilité, des plus efficaces couches de transport, sécurisé ou non. Nous ferons donc évoluer Java// en ce sens.

### 4.2.4 Sécurisation des applications à objets distribuées

Ces recherches seront menées en collaboration avec Eurécom et seront amorcées par le co-encadrement d'un stagiaire de DEA.

**Participants**
F. Baude, D. Caromel, R. Molva, Y. Roudier, B. Serpette.

Il s'agit d'intégrer, en suivant les approches appliquée et pragmatique, des caractéristiques de sécurité dans les bibliothèques pour la répartition, et par là-même, aux applications développées en partant par exemple d'une application collaborative existante. Notamment, les communications sur Internet circulant en clair par défaut, les applications peuvent requérir que les informations échangées (que ce soient des données ou du code) soit protégées.

Nous débuterons notre étude avec celle de mécanismes primordiaux pour le type d'applications que nous visons (applications collaboratives et commerce électronique) à savoir : sécuriser des communications de groupe au sein d'une application, sécuriser du code mobile (cf. les aglets). La bibliothèque Java// nous servira de cadre pour valider nos idées.

Notons qu'avant toute chose, la communication de groupe ainsi qu'un mécanisme de migration d'objets actifs doivent être ajoutés à Java// (cf. Section 4.2.3). Autant que possible, leur conception intègrera le besoin de sécurité.

Notons que ces recherches ne relèvent nullement des travaux de sécurité qui visent à intégrer aux protocoles réseaux Internet des aspects sécurisés [Mol99].

Il s'agit ici de fournir des outils (bibliothèques) et modèles de haut niveau d'abstraction (objets actifs) qui facilitent la programmation à objets distribuée sécurisée. En revanche, ces travaux bénéficieront d'une connaissance approfondie des mécanismes en cours d'intégration dans la couche IP, connaissances qui sont au centre des compétences des collaborateurs Eurécom.

En perspective de ces travaux, et à plus long terme, des applications développées en Java// sécurisé pourraient être vérifiées vis-à-vis de politiques de sécurité propres au domaine d'applications (cf. Section 4.1.4). La conception des mécanismes de sécurité que nous introduirons dans Java// gagnera donc à se faire de concert avec l'étude et la formalisation de politiques de sécurité.

# 5 Positionnement

Notre proposition de projet rentre dans le cadre du thème 2A Sémantique et Programmation. Nous nous positionnons donc tout d'abord par rapport aux autres projets de ce thème, puis aux projets connexes du Thème 1 Réseaux et Systèmes, puis citons les équipes du paysage universitaire local et national qui étudient des problèmes en relation avec notre thématique et avec lesquelles nous pourrions poursuivre ou établir des collaborations.

Enfin, nous mentionnons des projets connexes européens et nord-américains avec lesquels nous n'avons pas de liens formels.

## 5.1 Positionnement dans l'institut

### 5.1.1 Thème "Sémantique et Programmation"

– La proposition de projet TROPICS (Transformations et Outils pour le Calcul Scientifique, INRIA Sophia Antipolis) aborde des problématiques également étudiées dans OASIS (transformations de programmes) mais sous l'angle du calcul scientifique (différentiation automatique, parallélisation (grain fin) de programmes Fortran ou C).
– La proposition de projet CERTILAB (à Sophia Antipolis) aura pour thème la certification de logiciel. Les domaines abordés (théories typées, ordre supérieur, bibliothèques pour Coq) se situent très en amont de nos préoccupations. Un point d'intérêt commun concerne la connexion entre spécifications sémantique en Typol et preuves en Coq et pourra faire l'objet de collaborations. D'autre part, CERTILAB mentionne en perspective des travaux futurs sur les preuves de correction de protocoles qui ne font pas partie de nos objectifs.
– La proposition de projet LEMME (à Sophia Antipolis) aura pour thème la certification d'algorithmes de calcul formel. Il est question d'y aborder également les preuves de compilateurs, ce qui ne fait pas partie de nos objectifs au sein d'OASIS puisque nous nous limitons à l'étude et la vérification de propriétés de programmes et de langages (héritage, non-interférence entre objets, distribution, sécurité).

De par notre histoire commune et leur implication (à des degrés divers) à la future action de développement SmartTools, ces trois projets seront des partenaires privilégiés.

– Le projet LANDE (IRISA) poursuit des recherches sur une thématique très proche: "conception d'outils d'aide au développement et à la validation de logiciels". Les objectifs communs sont l'étude et la preuve des propriétés de sécurité de programmes Java et Java Card (action coopérative Java Card). Les techniques communes sont les spécifications formelles (sémantique opérationnelle) et les analyses statiques (optimisations, transformations de programmes). Ces points de convergence sont connus et exploités (utilisation prévue d'outils communs, échanges, visites).
– le projet CRISTAL (INRIA Rocquencourt) s'intéresse à la conception de langages de programmation expressifs et sûrs afin de faciliter le développement d'applications, d'augmenter leur robustesse, et d'améliorer leurs performances. Les principales techniques étudiées et utilisées dans ce but sont les systèmes de types et les systèmes de modules. Il est question d'utiliser des plate-formes communes pour la spécification et la construction d'outils complémentaires (dans le cadre de l'action coopérative Java Card).

- L'objectif du projet COQ (INRIA Rocquencourt) est l'élaboration d'un langage (issu de la théorie des types) et d'un environnement (Coq) pour développer des preuves formelles. Nous allons étudier ensemble comment construire des descriptions sémantiques permettant la simulation et la preuve.
- Le projet VIP de l'action Dyade (dirigé par Dominique Bolignano) a développé une expertise reconnue dans le développement de machines virtuelles embarquées sécurisées, la vérification formelle de protocoles cryptographiques, et la modélisation de politiques de sécurité. La preuve formelle qui garantit les propriétés de sécurité des protocoles de paiement (architecture C-SET) a été formalisée et vérifiée mécaniquement en Coq. Nous avons cette année entamé une collaboration avec D. Bolignano autour de la sémantique de Java Card et cette collaboration sera renforcée dans le futur, notamment sur les aspects politiques de sécurité.
- Les objectifs du projet PROTHEO (LORIA) sont la conception et la réalisation d'outils pour la spécification et la vérification de logiciels. Non seulement ces objectifs se situent en amont de nos préoccupations, mais les moyens et techniques employés sont complètement différents (résolution de contraintes, systèmes de réécriture).
- Le projet COMPOSE (IRISA) s'intéresse à la conception de systèmes adaptatifs, en utilisant l'évaluation partielle. Le processus d'adaptation repose sur des techniques d'analyses et transformations de programmes, techniques que nous utilisons également, mais à d'autres fins.
- Les expérimentations liées aux environnements de développement et vérification Java et Java Card seront poursuivies avec la plate-forme Centaur actuelle mais utiliseront dans le futur les outils produits dans le cadre de l'action de développement SmartTools en cours de montage, au fur et à mesure de leur disponibilité. Il n'y a donc aucun recouvrement thématique mais une relation de fournisseur à client.

Il n'est pas justifié de se positionner par rapport au projet CODES sur les aspects cryptographie. Nous serons seulement utilisateurs de ces techniques dans le domaine de la sécurité.

### 5.1.2 Thème "Réseaux et systèmes"

- Le projet SLOOP (sans les membres qui participent à la création d'OASIS) a très récemment recentré sa thématique sur la simulation, l'algorithmique et les structures discrètes pour les communications, et donc n'aborde pas les domaines étudiés dans OASIS.
- Le projet MEIJE (INRIA Sophia Antipolis) a comme objectif la modélisation des systèmes réactifs, parallèles et communicants. Il étudie

aussi la définition et l'implémentation de langages et de méthodes formelles de vérification basés sur ces modèles. Les deux modèles adoptés sont les calculs de processus et les formalismes réactifs synchrones. Depuis 4 ans, nous avons travaillé ensemble, en particulier sur les aspects liés aux modélisations en $\pi$-calcul (encadrement commun d'étudiants [Sag95, Bru96], participation séminaires). Dans le futur, nous pensons maintenir ces liens sur ces aspects. D'autre part, les travaux autour de la programmation synchrone et réactive en Java (SugarCubes) pourraient également faire partie de sujets possibles de collaboration.
– Le projet SOR (INRIA Rocquencourt) s'intéresse plus à la réplication des données et aux problèmes de persistance qui ne correspondent pas directement à nos axes de recherche. Néanmoins, le projet SOR collabore avec CRISTAL sur le sujet des machines virtuelles virtuelles (MVV), et nous serons amenés à étudier leurs modèles (en particulier les SSPChains) dans le cadre de la définition de notre bibliothèque pour la répartition et pour l'implémentation des langages à objets.
– Le projet PARA (INRIA Rocquencourt) bien que s'intéressant peu à l'algorithmique distribuée, étudie la programmation des processus mobiles en tenant compte de la reconfiguration dynamique des interconnexions. Leurs travaux pourront servir de base formelle pour l'implémentation de notre bibliothèque pour la répartition. Notons de plus que l'arrivée d'Alain Deutsch au projet PARA apporte une composante "analyse de programmes" dont les fondements (l'interprétation abstraite) s'avèrent communs.
– L'avant-projet ECOO (Environnements pour la coopération) a pour objectif de concevoir et de réaliser des services de base pour le développement d'applications coopératives. OASIS ne travaille pas spécifiquement sur de tels modèles ou du midleware spécialisé pour le travail coopératif. Les bibliothèques pour la répartitions conçues et développées par OASIS, en particulier ProActive, sont génériques et non-spécialisées. A ce titre, les projets sont complémentaires et des collaborations sont envisageables entre ECOO et OASIS: on pourrait étudier dans quelle mesure le modèle et la bibliothèque ProActive fournissent des briques de bases utiles à la construction de services spécialisés dédiés aux applications coopératives.

## 5.2 Positionnement sur le plan local et national

– au sein du laboratoire I3S CNRS UNSA, plusieurs enseignants-chercheurs poursuivent des travaux de recherches dans des domaines connexes (certains de ces travaux ont donné lieu à des collaborations par le passé):
  – Anne-Marie Dery et Mireille Fornarino modélisent des dépendances inter-objets, en décrivant leurs interactions, et ce unique-

ment en se basant sur l'interface des objets liés. Ce modèle permet d'exprimer la communication et la coordination entre des objets distants. Comme nous, elles sont intéressées par les aspects spécifications (publication commune sur les modélisations de collecticiels de type workflow [ABB+98]), les aspects sémantiques (connexion de la sémantique d'Eiffel avec la sémantique des dépendances), et les aspects mise en oeuvre basés sur l'exploration d'un Méta-Object Protocole. Cependant, leur approche de niveau très "méta" (aspect-oriented programming) ne se situe donc pas au même niveau que la notre; les intérêts communs et leur approche complémentaire en font des partenaires privilégiés sur le site de Sophia Antipolis.

  - Manuel Serrano a travaillé avec Bernard Serpette à l'INRIA Rocquencourt. Il a des compétences en compilation, analyse de flot de contrôle dans les langages fonctionnels. Il s'intéresse maintenant aux communications sécurisées dans les systèmes d'objets mobiles. Sur l'ensemble de ces aspects, nous envisageons de renouer les liens.
  - Bruno Martin est spécialiste en théorie des codes et il s'intéresse par ailleurs à la cryptographie et à son application à la confidentialité et l'intégrité des messages. Des collaborations sont envisagées; en effet, son domaine de recherche et ses compétences, qui se situent en amont de celles d'OASIS, sont extrêmement complémentaires.
- Equipe PLUME (ENS-Lyon).
  Plusieurs réunions de travail ont eu lieu avec Pierre Lescanne, Eva Rose et Kristoffer Rose sur le thème de la vérification du code mobile. En particulier, le système Centaur est utilisé pour décrire un vérificateur de ByteCode embarqué sur une carte à puce.
- G. Castagna (ENS-Ulm) est spécialiste du typage dans les langages à objets et s'intéresse aux aspects mobilité et sécurité des objets (définition du seal-calculus, extension du $\pi$-calcul); nous participons ensemble à un groupe de travail du GDR ALP.

## 5.3 Positionnement sur le plan international

Nous ne décrivons pas ici les collaborations internationales (décrites en Section 6.1.3). Nous nous attachons ici à décrire les domaines et objectifs de quelques groupes de recherche dont nous connaissons les travaux, et nous précisons lorsque cela est envisageable sur quels aspects nous pourrions établir des relations suivies.

- Joachim Posegga et Harald Vogt (Deutsche Telekom AG) ont proposé récemment [PV98] une technique de vérification de bytecode Java Card fondée sur l'utilisation de model-checking. Nous avons rencontré Joachim Posegga en juillet dernier et évoqué ensemble la possibilité de collaborations dans un cadre européen.

- Martin Abadi (Compaq) est spécialiste des langages à objets, des méthodes de spécification et vérification et de la sécurité dans les systèmes distribués. Ses travaux récents sur des abstractions pour la protection rentrent complètement dans notre domaine d'intérêt. Nous rencontrons fréquemment M. Abadi, et nous informons mutuellement des progrès de nos recherches.
- Le groupe Secure Internet Programming de Princeton University étudie depuis Novembre 1995 la sécurité des logiciels utilisés sur Internet, notamment les systèmes de code mobiles comme Java, ActiveX, et JavaScript. Ils s'intéressent aux aspects langage, typage, machine virtuelle, cartes à puce, et encryptage. Nous n'avons aucun contact avec cette équipe pour le moment.
- Microsoft: le groupe de Microsoft Research, Cambridge (en particulier Luca Cardelli, http://www.luca.demon.co.uk/), est engagé dans le développement de modèles formels pour la migration de code et la sécurité. Leurs modèles (spi-calcul, calcul des ambients) sont basés sur le $\pi$-calcul et utilisent des paradigmes issus des langages de programmation, telle la sémantique opérationnelle ou les systèmes de types, pour formaliser le comportement du code mobile et la sécurité des communications. Leurs résultats et formalismes pourraient être utilisés dans la partie théorique de notre projet.
- Promodis: le consortium "Programming of Modular and Distributed Systems", qui inclut des universités et entreprises suédoises, dont Ericsson, poursuit des objectifs et méthodes très proches: conception et implémentation d'environnements de preuves pour applications réparties et sécurité, analyses statiques de programmes, enforcements des politiques de sécurité par des mécanismes de typage. Quoique le domaine d'application privilegié de Promodis soit celui de la télephonie mobile, plus particulièrement dans le cadre de Safe-Erlang, nous espérons pouvoir établir des relations à moyen terme.
- Carnegie Mellon University: le projet Fox a exploré un nouveau concept, le code auto-certifié (*proof-carrying code*), selon lequel un fournisseur doit accompagner son code d'une preuve de sécurité qui peut-être vérifiée par le client. Cette approche soulève deux questions fondamentales, qui sont au centre de notre projet, à savoir (1) la spécification formelle des politiques de sécurité et (2) la preuve formelle qu'un objet mobile adhère à la politique de sécurité qui lui est attachée. Notre recherche s'inscrit donc en amont du travail sur le code auto-certifié.

# 6 Eléments de visibilité

Nous donnons ici des informations factuelles qui permettent d'apprécier la visibilité des membres du projet nationalement et internationalement concernant l'animation de la recherche et la diffusion des résultats. Nous présentons également la valorisation de nos recherches dans le contexte industriel.

## 6.1 Collaborations

Nous décrivons ici les collaborations formalisées entre notre équipe et d'autres groupes de recherche.

### 6.1.1 Interactions avec d'autres projets INRIA

- Action coopérative sur le thème "Java Card: Sémantique, optimisations et sécurité". Partenaires: Lande, Cristal, Dyade et Onera-Cert.
  (voir http://www.irisa.fr/lande/jensen/javacard.html)

### 6.1.2 Nationales

- Groupe de Travail "Spécification, Vérification, Sémantique" du GDR ALP: Algorithmique, Langages et Programmation.
  Responsable Giuseppe Castagna, ENS-Ulm.

### 6.1.3 Internationales

- Contrat NSF - INRIA - CNRS (1997-2000, montant 165 KF) sur la spécification formelle et la transformation de programmes parallèles concernant J.-L. Gaudiot (USC), Isabelle Attali et Denis Caromel.
- collaborations INRIA-Brésil (1995-1998, montant 60 KF) avec Fabio Da Silva, Paolo Borba (Recife) et Sidi Ould Ehmety (São Luis) sur la programmation à objets.
- Denis Caromel est co-responsable du groupe de travail "Concurrence et applications" du Java Grande Forum (voir http://www.javagrande.org).
- University of Adelaide, Australie,
  Equipe de Michael Oudshoorn et Andrew Wendelborn, coopération sur la formalisation des langages Eiffel// et Sisal et plus récemment sur une extension fonctionnelle de Java.
- London Parallel Applications Centre (LPAC), University College London (UCL), DEC LTD (Galway), dans le cadre de EUROPA Working Group on Parallel C++ (Esprit III Project).
- Indiana University (High Performance Computing Lab, Dennis Gannon),
  California Institute of Technology (Caltech, Carl Kesselman),

Real World Computer Partnership (RWCP, Japan, Yutaka Ishikawa), dans le cadre du groupe de travail international HPC++.

### 6.2 Actions industrielles

- Sun France, 1 Ultra Sparc, 1997, 65 KF
- Contrats de recherche avec SIMULOG (1997, 1998) d'un total de 75 KF HT sur le thème "transformations de programmes Fortran dans Foresys".
- Contrat de recherche avec Bull/CP8 (1998 - 2001) d'un montant de 1 362 KF HT sur le thème "Environnement de développement et vérification pour JavaCard".
- Sun Palo Alto (1998) US$ 25 000, sur le thème "Programmation répartie en Java".

### 6.3 Actions soumises ou en cours de montage

- Projet OPPIDUM soumis en Octobre 1998: FORMAVIE - Modélisation Formelle et Certification Sécuritaire pour Machine Virtuelle Embarquée.
  Partenaires: Bull CP8, Schlumberger SC & T.
- externat industriel Romain Guider à Open Group, Grenoble sur le thème "Analyse de formes pour l'optimisation de la compilation Java".
- action de développement SmartTools (partenariat INRIA Bull Simulog) sur des outils génériques pour la construction d'environnements de développement.

### 6.4 Formation

- Isabelle Attali est responsable du module de tronc commun "Sémantique des Langages de Programmation" du DEA d'Informatique de l'UNSA et membre du conseil scientifique de ce DEA.
- Isabelle Attali est responsable de la Formation pour l'INRIA Sophia Antipolis depuis 1997 (voir http://www.inria.fr/Formation/Sophia).
- Isabelle Attali est également responsable scientifique depuis 1995 de l'Ecole Jeunes Chercheurs en Programmation, qui réunit environ 40 doctorants pour 15 jours de cours chaque année.
- Denis Caromel est responsable de la filière "Systèmes Distribués" du DEA RSD (Réseaux et Systèmes Distribués) de l'UNSA (en collaboration avec CMA, CNET, Eurécom, INRIA Sophia Antipolis), depuis Septembre 1995.
  Le DEA RSD est d'ailleurs un véritable pôle de compétences autour de la programmation répartie et des aspects de sécurité avec des intervenants reconnus comme R. Molva et B. Martin.

- Denis Caromel est responsable depuis 1995 du Module “Langages de Programmation Concurrente, Parallèle, Distribuée” commun aux DEAs Informatique et RSD de l’UNSA.

### 6.5 Animation de la recherche

- Isabelle Attali est membre du comité de direction du GDR ALP: Algorithmique, Langages et Programmation.
- Isabelle Attali participe au comité de programme de WAGA’99 (2nd International Workshop on Attribute Grammars and their Applications, voir http://www-rocq.inria.fr/oscar/waga99.html).
- Françoise Baude a organisé Stratagem’96, (8-10 juillet 1996 à Sophia-Antipolis, en collaboration avec le groupe CAPA de l’ex GDR PRS): Colloque sur le calcul parallèle pour la résolution d’applications irrégulières de grande taille.
- Françoise Baude participe au comité d’édition de la revue Calculateurs Parallèles, éditions Hermès (depuis Mars 96).
- Françoise Baude a participé aux comités de programme de:
  - RenPar’7, Mons, 30 Mai - 2 Juin 1995, RenPar’8, Bordeaux, 20-24 Mai 1996, RenPar’9, Lausanne, 20-23 Mai 1997, RenPar’10, Strasbourg, 9-12 Juin 1998.
- Denis Caromel co-organise avec Jean-Marc Geib (LIFL) et Patrick Sallé (IRIT) l’action Inter-GDR PRO: Parallélisme, Répartition et Objets (GDR ALP: Algorithmique, Langages et Programmation et GDR ARP: Architecture, Réseaux et systèmes, Parallélisme)
  http://www.inria.fr/sloop/personnel/Denis.Caromel/PRO/PRO.html
- Denis Caromel est président du comité de programme de ISCOPE’98, 8-11 Décembre 1998, Santa Fe, USA, International Symposium on Computing in Object-Oriented and Parallel Environments, actes Springer Verlag, Lectures Notes in Computer Science (LNCS)
  http://www.acl.lanl.gov/iscope98/
- Denis Caromel co-organise avec Jean-Paul Rigault (UNSA) la conférence ECOOP (European Conference on Object-Oriented Programming) à Sophia Antipolis, en Juin 2000.
- Denis Caromel participe (ou a participé) aux comités de programme de:
  - International Conference and Exhibition on High-Performance Computing and Networking (HPCN Europe), 1996 (Brussels), 1997 (Vienna), actes Springer Verlag, Lectures Notes in Computer Science (LNCS).
  - ECOOP’97 (Eleventh European Conference on Object-Oriented Programming), Juin 1997, Jyvaskyla, Finland, actes Springer Verlag, Lectures Notes in Computer Science (LNCS).

  - RenPar'10, Strasbourg, 9-12 Juin 1998,
  - LMO'99, Villefranche sur Mer, 27-29 Janvier 1999.
- Denis Caromel a participé aux comités d'organisation de:
  - ISCOPE'97, Décembre 1997, Los Angeles,
  - Euro-Par'98, Septembre 1998, Southampton,
  - International Workshop on Parallel C++ (IWPC++), Kanazawa, Japan, Mars 1996,
  - Workshop on Runtime Systems for Parallel Programming, RTSPP'97 (Avril 1997, Geneva, Switzerland), RTSPP'98 (Avril 1998, Orlando, Florida).
- Denis Caromel participe au comité d'édition de la revue L'OBJET, éditions Hermès (depuis Janvier 97).

## Références

[AAFZ97] D. Arnon, I. Attali, and P. Franchi-Zannettacci. A document manipulation system based on natural semantics. *Mathematical and Computer Modelling Journal*, 25(4):37–56, 1997.

[ABB+98] I. Attali, R. Bastide, M. Blay, A. M. Dery, and P Palanque. Spécification formelle et approche objet pour les applications workflow. *Techniques et Sciences Informatiques*, 98(2), 1998.

[AC90] I. Attali and J. Chazarain. Functional evaluation of natural semantics specifications. In *Proc. of WAGA (Attribute Grammars and their Applications)*, volume 461, pages 157–176, Paris, 1990. LNCS Springer Verlag.

[ACE96] I. Attali, D. Caromel, and S. O. Ehmety. A natural semantics for eiffel dynamic binding. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 18(5), Novembre 1996.

[ACE98a] I. Attali, D. Caromel, and S. O. Ehmety. About the automatic continuations in the Eiffel// Model. In *Proc. of the 1998 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'98)*, Las Vegas, USA, Juillet 1998.

[ACE98b] I. Attali, D. Caromel, and S. O. Ehmety. Formal Properties of the Eiffel// Model. In *Object Based Parallel and Distributed Computing*. Hermes, France, 1998. A paraitre.

[ACEL96] I. Attali, D. Caromel, S. O. Ehmety, and S. Lippi. Semantic-based visualization for parallel object-oriented programming. In *Proceedings of OOPSLA'96*, volume 31, 10 of *ACM Sigplan Notices*, San Jose, CA, October 1996. ACM Press.

[ACG92] I. Attali, J. Chazarain, and S. Gilette. Incremental evaluation of natural semantic specifications. In *Proc. of PLILP'92 (Programming Language Implementation and Logic Programming)*, volume 631, pages 87–99, Leuven, 1992. LNCS Springer Verlag.

[ACG98] I. Attali, D. Caromel, and R. Guider. Static analysis of Java for distributed and parallel programming. 1998. Soumis à publication.

[ACGW96] I. Attali, D. Caromel, R. Guider, and A. Wendelborn. Optimizing sisal programs: a formal approach. In *Actes Euro-Par'96, International Conference on Parallel Processing, Lyon.* Springer-Verlag, Août 1996. LNCS 1123-1124.

[ACL99] I. Attali, D. Caromel, and S. Lippi. From a specification to an equivalence proof in object-oriented parallelism. In *FMPPTA'99: Modeling and Proving (Fourth Workshop in Formal Methods for Parallel Programming, Theory and Applications)*, volume 1586. Springer, LNCS, 1999.

[ACR98] I. Attali, D. Caromel, and M. Russo. A formal and executable semantics for Java. In *Proceedings of Formal Underpinnings of Java, an OOPSLA'98 Workshop*, Vancouver, CA, October 1998. Technical Report, Princeton University. file://babar.inria.fr:/pub/croap/mrusso/OopslaWorkshop98.ps.gz.

[ACW96] I. Attali, D. Caromel, and A. Wendelborn. A formal semantics and an interactive environments for sisal. In *Tools and Environments for Parallel and Distributed Systems*. Kluwer Academic Publishers, Février 1996. ISBN 0-7923-9675-8.

[AFZ88] I. Attali and P. Franchi-Zannettacci. Unification-free execution of typol programs by semantic attribute evaluation. In *Proc. of the Fifth International Conference Symposium on Logic Programming*, pages 160–177, Seattle, 1988. Cambridge MIT Press.

[APR97] I. Attali, V. Pascual, and C. Roudet. A language and an integrated environment for program transformations. rapport de recherche 3313, INRIA, Novembre 1997.

[Att89] I. Attali. Compiling typol with attribute grammars. In *Proc. of PLILP'89 (Programming Language Implementation and Logic Programming)*, volume 348, pages 252–272, Orléans, 1989. LNCS Springer Verlag.

[Bar98a] G. Barthe. Order-sorted inductive types. Accepted for publication in Information and Computation, 1998.

[Bar98b] G. Barthe. Partial evaluation of object calculi. Technical Report UMDITR9806, Department of Computer Science, University of Minho, 1998.

[Bar98c] G. Barthe. The relevance of proof-irrelevance. In K.G. Larsen, S. Skyum, and G. Winskel, editors, *Proceedings of ICALP'98*, volume 1443 of *Lecture Notes in Computer Science*, pages 755–768. Springer-Verlag, 1998.

[BBC+96] F. Baude, F. Belloncle, D. Caromel, N. Furmento, P. Mussi, Y. Roudier, and G. Siegel. Parallel object-oriented programming for parallel simulations. *Information Sciences : an international journal, Elsevier Sc. Pub.*, 93:35–64, 1996.

[BCD+88] P. Borras, D. Clément, Th. Despeyroux, J. Incerpi, G. Kahn, B. Lang, and V. Pascual. Centaur: the System. In *SIGSOFT'88 Third Annual Symposium on Software Development Environments*, Boston, 1988. ftp://babar.inria.fr/pub/centaur/papers/sde3.ps.

[BCFS98] F. Baude, D. Caromel, N. Furmento, and D. Sagnol. Overlapping communication with computation in the framework of distributed object systems. 1998. Soumis à publication.

[Ber98] Y. Bertot. A certified compiler for an imperative language. Rapport de Recherche RR-3488, INRIA, 1998.

[BGH+95] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva, and M. Yung. The kryptoknight family of light-weight protocols for authentication and key distribution. *ACM Transactions on Networking*, 1995.

[BHS97] G. Barthe, J. Hatcliff, and M.H. Sørensen. Reflections on reflections. In H. Glaser, P. Hartel, and H. Kuchen, editors, *Proceedings of PLILP'97*, volume 1292 of *Lecture Notes in Computer Science*, pages 241–258, 1997.

[BHT98] G. Barthe, J. Hatcliff, and P. Thiemann. Monadic type systems: Pure type systems for impure settings. In A. Gordon, A. Pitts, and C. Talcott, editors, *Proceedings of HOOTS'97*, volume 10 of *Electronic Notes in Theoretical Computer Science*. Elsevier Publishing, 1998.

[BKR97] G. Barthe, F. Kamareddine, and A. Ríos. Explicit substitutions for the $\lambda\delta$-calculus. In M. Hanus and J. Heering, editors, *Proceedings of ALP '97 - HOA '97*, volume 1298 of *Lecture Notes in Computer Science*, pages 209–223. Springer-Verlag, 1997.

[BRB96] G. Barthe, M. Ruys, and H. Barendregt. A two-level approach towards lean proof-checking. In S. Berardi and M. Coppo, editors, *Proceedings of TYPES'95*, volume 1158 of *Lecture Notes in Computer Science*, pages 16–35. Springer-Verlag, 1996.

[Bru96] E. Bruneton. Implantation d'un traducteur eiffel// – $\pi$-calcul. Rapport de stage deuxième année, Ecole Polytechnique, juillet 1996.

[BS94] F. Baude and D. Skillicorn. Vers de la programmation parallèle structurée fondée sur la théorie des catégories. *Techniques et Sciences Informatiques*, 13:494–525, 1994.

[BVN91] F. Baude and G. Vidal-Naquet. Actors as a parallel programming model. In *In Proc. of the 8th Symposium of Theoretical Aspects of Computer Science*, pages 184–195. Springer-Verlag, 1991. LNCS 480.

[Car93] D. Caromel. Towards a Method of Object-Oriented Concurrent Programming. *Communications of the ACM*, 36(9):90–102, September 1993.

[CBR96] D. Caromel, F. Belloncle, and Y. Roudier. The C++// Language. In *Parallel Programming using C++*, pages 257–296. MIT Press, 1996. ISBN 0-262-73118-5.

[CDD+87] J. Chailloux, M. Devin, F. Dupont, J. M. Hullot, B. Serpette, and J. Vuillemin. Le-lisp version 15.2. Technical Report Manuel de Référence, INRIA, 1987.

[CKV98] D. Caromel, W. Klauser, and J. Vayssiere. Towards Seamless Computing and Metacomputing in Java. *Concurrency Practice and Experience*, 10(11–13):1043–1061, November 1998.

[CR95] D. Caromel and Y. Roudier. Reactive Programming in Eiffel//. In Jean-Pierre Briot, Jean-Marc Geib, and Akinori Yonezawa, editors, *Object-Based Parallel and Distributed Computation OBPDC'95*, number 1107 in Lecture Notes in Computer Science, pages 125–147. Springer-Verlag, 1995.

[CV98] D. Caromel and J. Vayssiere. A java framework for seamless sequential, multi-threaded, and distributed programming. In *Java for High-Performance Network Computing*, pages 141–150. ACM Workshop, Stanford University, March 1998. Palo Alto.

[Des88] T. Despeyroux. Typol: A formalism to implement Natural Semantics. Res. Rep. 94, INRIA Sophia Antipolis, France, 1988.

[FB96] N. Furmento and F. Baude. Schooner: An object-oriented runtime support for distributed applications. In *In K. Yetongnon and S. Hariri, editors , Proceedings of Parallel and Distributed Computing Systems (PDCS'96), Dijon*, volume 1, pages 31–36. International Society for Computers and their Applications (ISCA), Septembre 1996.

[Kah87] G. Kahn. Natural Semantics. In *Proc. of Symposium on Theoretical Aspects of Computer Science, Passau, Germany, LNCS 247*, 1987.

[KT96] D. Kaplan-Terrasse. Vers la certification du compilateur v5 d'Esterel. Rapport de Post-Doc, Inria-Sophia, France, September 1996. à paraitre en RR Inria.

[Lam94] L. Lamport. The Temporal Logic of Actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, May 1994.

[Lip96] S. Lippi. Langages à objets parallèles et temporal logic of actions (tla)". Rapport de stage de DEA, DEA Mathématiques Discrètes et Informatique de Marseille, juillet 1996.

[Mol99] R. Molva. Internet security architecture. *ISDN & Computer Networks Journal*, 1999.

[MPW92] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, i and ii. *Journal of Information and Computation*, 100:1–77, 1992.

[MST94] R. Molva, D. Samfat, and G. Tsudik. Authentication of mobile users. *IEEE Network Magazine*, 1994.

[MT98] R. Molva and G. Tsudik. Secret sets and applications. *Information Processing Letters*, 1998.

[PV98] J. Posegga and H. Vogt. Byte code verification for java smart cards based on model checking. In *5th European Symposium on Research in Computer Security (ESORICS)*, Louvain-la-Neuve, Belgium, 1998. Springer LNCS.

[QS91] C. Queinnec and B. Serpette. A dynamic extent control operator for partial continuations. In *Proceedings of POPL'91 - Eighteenth Annual ACM symposium on Principles of Programming Languages*, pages 174–184, Orlando, 1991.

[Sag95] D. Sagnol. $\pi$-calcul pour les langages à objets parallèles. Rapport de stage de DEA, Université de Nice - Sophia Antipolis, juin 1995.

[Ser92] B. Serpette. Approximations d'évaluateurs fonctionnels. In *Proceedings of WSA (Workshop on Static Analysis)*, pages 79–90. Bigre, 1992.

[SG95] B. Serpette and J. M. Geffroy. L'évaluation partielle par la réduction statique. In *Proceedings of JFLA (Journées Francophones des Langages Applicatifs)*, 1995.

[SM97] D. Samfat and R. Molva. Idamn: an intrusion detection architecture for mobile networks. *IEEE Journal on Selected Areas in Communications*, 1997.

[SS91] B. Serpette and N. Séniak. Langages intermédiaires pour la compilation de langages fonctionnels. In *Proceedings of JFLA (Journées Francophones des Langages Applicatifs)*, pages 12–20, 1991.

[Ter95] D. Terrasse. Encoding Natural Semantics in Coq. In *In Proc. of the Fourth International Conference on Algebraic Methodology and Software Technology*, volume LNCS 936. Springer-Verlag, 1995. ftp://babar.inria.fr/pub/croap/terrasse/amast95.ps.

[YL97] S. Yu and Z. Luo. Implementing a Model Checker for LEGO. In John Fitzgerald, Cliff B. Jones, and Peter Lucas, editors, *FME'97: Industrial Applications and Strengthened Foundations of Formal Methods*, volume 1313 of *Lecture Notes in Computer Science*, pages 442–458, Graz, Austria, september 1997. Springer-Verlag.

[Yu99] S. Yu. *Formal Verification of Concurrent Programs based on Type Theory.* PhD thesis, Dpt of Computer Science, University of Durham, February 1999.

## Table des matières