INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Projet OASIS

Objets Actifs, Sémantique, Internet et Sécurité

Sophia Antipolis

THÈME 2A

Table des matières

1 Composition de l'équipe **3**

2 Présentation et objectifs généraux **5**

3 Fondements scientifiques **5**
3.1 Spécifications, environnements, analyses et transformations 5
3.2 Programmation objet, concurrence et répartition 6

4 Domaines d'applications **7**
4.1 Panorama 7
4.2 Maintenance et manipulation de programmes 7
4.3 Logiciels sécurisés pour le Commerce Electronique 8
4.4 Programmation répartie, collaborative et sécurisée pour Internet 8

5 Logiciels **8**
5.1 Outils Interactifs Génériques SmartTools 8
5.2 Sémantiques et Environnements pour Java/Java Card 10
5.3 Bibliothèque ProActive 10
5.4 La bibliothèque C++// 11

6 Résultats nouveaux **11**
6.1 Environnements de développement et vérification pour Java et Java Card . . . 11
6.2 Analyses statiques et transformations de programmes 13
6.3 Spécifications sémantiques et vérifications 13
6.4 Outils et méthodologie pour la modélisation et la vérification de politiques de sécurité 14
6.5 Etude formelle des modèles à objets distribués 15
6.6 Implémentation des langages à objets 15
6.7 Bibliothèques pour la répartition 16
6.8 Sécurisation des applications à objets distribuées 18

7 Contrats industriels (nationaux, européens et internationaux) **18**
7.1 Environnements de développement et vérification pour Java et Java Card . . . 18
7.2 SmartTools: outils génériques pour la construction d'environnements de développement 19
7.3 Formavie - Modélisation Formelle et Certification Sécuritaire pour Machine Virtuelle Embarquée 19
7.4 Contrat SUN Microsystems 19

8 Actions régionales, nationales et internationales **19**
8.1 Actions régionales 19
8.1.1 Programmation répartie et collaborative pour Internet 19
8.2 Actions nationales 20

8.2.1 Action de Recherche Coopérative Java Card 20
8.2.2 Action de Recherche Coopérative S-Java 20
8.2.3 Programmation répartie collaborative et sécurisée pour Internet 20
8.3 Actions internationales 21
8.3.1 Spécification formelle et transformation de programmes parallèles 21
8.3.2 Langages Objets Concurrents 21
8.3.3 Concurrence et applications 22
8.4 Visites, Participations à des conférences, et invitations de chercheurs 22

9 Diffusion de résultats **24**
9.1 Animation de la Communauté scientifique 24
9.2 Enseignement 25

10 Bibliographie **27**

OASIS *est un projet commun à l'INRIA, au CNRS et à l'université de Nice-Sophia Antipolis, en collaboration avec l'institut Eurécom.*

1 Composition de l'équipe

Responsable scientifique

Isabelle Attali [DR depuis le 01/10/2000]

Responsable permanent

Bernard Serpette [CR]

Assistance administrative

Maryse Renaud [Assistante Dyade et contact Oasis]

Sandrine Boute [TR à temps partiel, jusqu'en Avril 2000]

Personnel Inria

Gilles Barthe [CR]

Francis Montagnac [IR, à temps partiel]

Didier Parigot [CR]

Personnel UNSA

Françoise Baude [Maitre de Conférence, UNSA]

Denis Caromel [Professeur, UNSA]

Personnel BULL

Franck Chalaux [Ingénieur BULL, jusqu'au 30 novembre 2000, action SmartTools]

Ingénieurs experts

Pascal Degenne [action SmartTools]

Alexandre Fau [action SmartTools]

Joël Fillon [depuis le 01/09/00, action SmartTools]

Line Jakubiec [jusqu'au 30/09/00, projet Formavie]

Philippe Ladagnous [jusqu'au 28/02/00, projet Formavie]

Chercheurs doctorants

Carine Courbis [Allocataire MESR, deuxième année]

Simao De Sousa [Boursier Portugal, depuis le 01/10/99]

Romain Guider [Allocataire MESR, soutenance de thèse le 29/09/2000]

Fabrice Huet [Allocataire MESR + Moniteur UNSA, deuxième année]

Guillaume Dufay [Allocataire MESR depuis le 01/10/2000]

Emmanuel Reuter [Salarié (ingénieur IUFM Nice), depuis le 15/11/2000]

Marjorie Russo [Allocataire MESR puis demi-ATER depuis le 01/09/2000]

David Sagnol [Allocataire MESR, soutenance de thèse le 05/06/2000]

Julien Vayssière [Boursier Région Entreprise, UNSA, deuxième année]

Ludovic Henrio [Boursier DGA, depuis le 01/10/2000]

Chercheurs post-doctorants

Henrik Nilsson [financement suédois, puis boursier Inria du 01/02/2000 au 31/05/2000]

Marieke Huisman [depuis le 01/10/2000, financement ARC S-JAVA]

Stagiaires

Alexandre Bergel [Maîtrise, UNSA, du 23/03/2000 au 25/06/2000, puis du 25/06/2000 au 30/09/2000]

Jean Lucien Baudoin [Maîtrise, UNSA, du 03/07/2000 au 31/08/2000]

Thierry Abbondanza [Maîtrise, UNSA, du 03/07/2000 au 31/08/2000]

Damien Nade [Maîtrise, UNSA, du 03/07/2000 au 31/08/2000]

Ludovic Henrio [DEA SPP, Paris VII, du 04/04/2000 au 15/09/2000]

Florian Doyon [Alternance IUT, du 01/10/99 au 30/09/2000]

Guillaume Dufay [DEA SPP, Paris VII, du 01/04/2000 au 30/09/2000]

Olivier Nano [Maîtrise, UNSA, du 26/06/2000 au 30/09/2000]

Emmanuel Reuter [DEA RSD, UNSA du 01/04/2000 au 30/06/2000]

Collaborateurs extérieurs Eurécom

Refik Molva [Professeur associé]

Yves Roudier [Assistant]

2 Présentation et objectifs généraux

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.

Le projet rassemble deux domaines d'expertise clairement identifiés :

- sémantique, environnements, compilation et analyse statique,
- programmation à objets répartie.

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

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.

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. Plus précisément, nous visons les domaines d'applications suivants : applications embarquées, carte à puce, commerce électronique, télécommunications, téléphonie mobile.

3 Fondements scientifiques

3.1 Spécifications, environnements, analyses et transformations

Mots clés : sémantique, environnements, méthodes formelles et preuves, analyses de programmes, fiabilité du logiciel, sécurité.

Participants : Isabelle Attali, Gilles Barthe, Denis Caromel, Carine Courbis, Pascal Degenne, Guillaume Dufay, Alexandre Fau, Ludovic Henrio, Line Jakubiec, Philippe Ladagnous, Francis Montagnac, Henrik Nilsson, Didier Parigot, Marjorie Russo.

Depuis les débuts de l'informatisation des tâches, l'utilisateur (du plus novice au plus expert) a toujours eu besoin d'aide pour mettre au point ses programmes, manipuler une banque de données, construire un circuit imprimé ou encore concevoir un plan d'architecte. L'existence des termes comme CAO (Conception Assistée par Ordinateur), EAO (Enseignement Assisté

par Ordinateur), PAO (Publication Assistée par Ordinateur), ou dans un domaine plus proche, CASE (Computer-Aided Software Engineering), montre bien l'état des besoins.

Les domaines d'utilisation d'outils d'aide sont nombreux et variés : les interfaces homme-machine, la programmation, les aspects distribution et configuration de systèmes, les environnements de développement de preuves, la gestion de librairies, la simulation et l'évaluation de performance, etc.

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.

Nous nous appuyons sur la Sémantique Naturelle (formalisme issu de la sémantique opérationnelle structurelle et de la déduction naturelle). Une spécification est un ensemble de règles d'inférence qui décrivent comment déduire une conclusion à partir de prémisses. Les règles décrivent le comportement des constructeurs du langage manipulé. L'approche structurelle de la méthode apparaît dans la forme des séquents, qui ont généralement un *sujet*, terme d'une syntaxe abstraite. Nous avons étudié plusieurs classes de langages de programmation, en nous spécialisant sur les langages à objets (Eiffel, Java) [2, 1].

Ces environnements gagnent à être associés à des outils d'analyses statiques et de transformations qui assurent une certaine aide à la programmation et à l'optimisation de programmes [10].

Enfin le contexte applicatif (Internet, commerce électronique, cartes à puce et sécurité) nous pousse à nous intéresser à des outils de vérification qui seront particulièrement utiles aux programmeurs pour garantir qu'une politique de sécurité est respectée (identification, intégrité, confidentialité), par exemple, que tel programme gardera des données intègres en cas de retrait inopiné de la carte.

Notre travail sur les transformations de programmes comme outils de généricité nous a amené à comparer différentes techniques de transformations issues de divers styles de programmation : fonctionnelle pour la technique de déforestation et la programmation polytypique, à objets pour les *visitor patterns* ou les *tree traversals* et enfin les grammaires attribuées [9, 7].

Notre travail sur la formalisation des sémantiques des langages nous a amené à proposer des extensions aux systèmes de types existants. Nous avons tout particulièrement étudié le sous-typage par constructeurs [3], qui permet de formaliser d'une manière élégante de nombreux exemples de sémantique naturelle.

3.2 Programmation objet, concurrence et répartition

Mots clés : programmation à objets répartie, analyses de programmes, code mobile, collecticiels, communication de groupe, concurrence, distribution, synchronisation, sécurité.

Participants : Isabelle Attali, Françoise Baude, Alexandre Bergel, Denis Caromel, Florian Doyon, Romain Guider, Fabrice Huet, Francis Montagnac, Olivier Nano, Emmanuel Reuter, David Sagnol, Bernard Serpette, Julien Vayssière.

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, multi-threading, accès concurrents) viennent apporter un degré supplémentaire de complexité. 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.

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 métacomputing. 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 sur 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?

Nous avons développé des compétences en programmation concurrente, répartie et parallèle dans le cadre des langages à objets ; ces recherches sont menées aussi bien sur des aspects théoriques comme les sémantiques formelles, ou les analyses statiques et transformations pour la répartition automatique ou semi-automatique, que sur des aspects pragmatiques comme la conception de langages et de méthodes de programmation [6], ou la construction de bibliothèques pour le parallélisme [4, 5].

Nous avons également étudié les problèmes des supports d'exécution de ces bibliothèques [33] (mise en œuvre portable, optimisations, etc).

4 Domaines d'applications

4.1 Panorama

Résumé : *Les domaines d'application du projet Oasis couvrent tous les aspects du logiciel embarqué (cartes à puces, commerce électronique, téléphonie mobile) ainsi que les aspects liés aux applications réparties et communicantes sur intranets et Internet (par exemple applications collaboratives).*

4.2 Maintenance et manipulation de programmes

La maintenance des systèmes logiciels est l'un des points critiques du cycle de vie d'un logiciel. La durée de vie de ces logiciels a augmenté ainsi que leur complexité due à des mises à jour successives, soit pour corriger des bugs, soit pour ajouter de nouvelles fonctionnalités. Les

problèmes de l'an 2000 et du passage à l'Euro sont deux exemples concrets de maintenance de logiciel. La taille et la complexité de ces systèmes logiciels rendent leur maintenance coûteuse et hasardeuse si elle n'est pas automatisée. Les systèmes de manipulation et de transformation de programmes, cantonnés jusqu'ici dans des applications de taille réduite, semblent désormais adaptés pour répondre à ces nouveaux besoins.

4.3 Logiciels sécurisés pour le Commerce Electronique

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 plates-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.

4.4 Programmation répartie, collaborative et sécurisée pour Internet

Nos champs d'application sont les systèmes collaboratifs (par exemple des systèmes d'entreprise intranet et Internet), mettant ainsi l'accent sur les aspects liés à la sécurité, les systèmes transactionnels commerciaux, bancaires, etc.

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.

5 Logiciels

5.1 Outils Interactifs Génériques SmartTools

Participants : Isabelle Attali, Denis Caromel, Carine Courbis, Franck Chalaux, Pascal Degenne, Alexandre Fau, Joël Fillon, Francis Montagnac, Didier Parigot [correspondant].

SmartTools, générateur d'environnements de développement interactif, permet de générer, à partir de spécifications formelles associées à un langage, un environnement interactif de développement pour ce langage. Cette année, nous avons porté nos efforts de recherche et de développement essentiellement sur les points suivants:

Nous avons défini et réalisé une vraie architecture modulaire. Nous avons déjà tiré avantage de cette approche. En particulier, nous avons pu apprécier la facilité d'évolution et d'extension

du système dans ce contexte. La réalisation très rapide et extrêmement simple d'une version distribuée (répartie) de SmartTools, utilisant la bibliothèque Proactive développée dans l'équipe, nous a permis de tester la validité et la correction de notre architecture.

Les points de rapprochement entre la technologie SmartTools et la technologie XML ont été largement confirmés, tant au niveau spécification que composant logiciel. Nous avons défini et réalisé une correspondance entre les spécifications syntaxiques liées au formalisme XML (DTD) et leur équivalent dans SmartTools. Nous proposons donc, avec SmartTools, un outil de génération d'environnement pour des langages définis avec les formalismes issus de la technologie XML. Grâce à notre architecture ouverte et à cette correspondance, les nombreux développements autour de XML peuvent être facilement utilisés et connectés au sein de notre plate-forme. Ce rapprochement fournit ainsi à l'outil SmartTools un champ d'application beaucoup plus large pour la génération d'environnements, et non plus limité aux seuls langages de programmation.

Pour décrire les traitements sémantiques spécifiques à chaque langage, nous avons automatisé la technique de programmation, dénommée "visitor pattern". En utilisant les travaux sur les multi-méthodes, nous avons étendu cette technique. L'intérêt de cette approche est surtout sa simplicité de mise en oeuvre et correspond parfaitement à nos soucis de réutilisation ou de généricité des composants sémantiques. Tout ceci correspond aux éléments de base de nos efforts de recherche sur la définition et la réutilisation de "composants génériques" pour les descriptions sémantiques.

Notre objectif de limiter fortement tous les développements propriétaires en nous appuyant le plus possible sur les composants logiciels issus des technologies Java ou XML semble être quasiment atteint au cours de la première année de SmartTools. Cet effort d'ouverture concerne en particulier les aspects d'environnement interactif et aussi les composants syntaxiques comme les générateurs d'analyseurs syntaxiques. En particulier, l'outil de transformation XSLT semble être un bon candidat pour le remplacement des développements propriétaires utilisés jusqu'ici dans SmartTools. L'intérêt de cette approche est qu'elle permet de tirer avantage des futurs efforts de développement autour des composants standards de Java ou XML. L'utilisation de standards facilite grandement l'utilisation de SmartTools, ainsi que les futurs développements.

Cette année, nous avons établi plusieurs liens de collaborations autour de notre plate-forme SmartTools. Dans le cadre de Dyade, nous avons régulièrement collaboré avec l'équipe Koala et Claude Pasquier (ingénieur Dyade dans le projet Lemme), en particulier sur la technologie XML et la boite à outil Koala. De plus, le projet Lande à l'IRISA a utilisé et interfacé son solveur générique dans le cadre d'un stage d'été de Cristelle Lecomte.

Avec l'équipe de recherche de Marne-la-Vallée, nous avons eu des échanges de point de vue sur le thème de recherche "composants sémantiques" et utilisons leur implémentation des multi-méthodes dans SmartTools. Avec le projet Protheo du LORIA, en particulier avec Pierre-Etienne Moreau, nous avons élaboré des plans d'action pour la réalisation d'outil de transformation commun. Enfin, dans le cadre d'une action intégrée (PAI) du Ministère des Affaires Etrangère , Proteus, avec une équipe de recherche de Slovénie dirigée par Marjan Mernick, nous avons établi une connection entre nos deux outils, Lisa `http://marcel.uni-mb.si/lisa/` et SmartTools.

Finalement, après seulement une année de développement, nous avons pu commencé nos efforts de diffusion. La version 2 a été installée chez notre partenaire industriel Bull CP8. Au

cours du workshop WAGA00 [12], cette version a été présentée lors d'une session de démonstration. Enfin, nous avons à plusieurs reprises effectué des présentations et démonstrations de SmartTools à nos partenaires industriels et utilisateurs potentiels. Les travaux ont également donné lieu à la rédaction d'un article [27].

Pour plus d'information, voir `http://www-sop.inria.fr/oasis/SmartTools`.

5.2 Sémantiques et Environnements pour Java/Java Card

Participants : Isabelle Attali [correspondante], Denis Caromel, Carine Courbis, Ludovic Henrio, Henrik Nilsson, Marjorie Russo.

Nous avons spécifié la quasi-totalité de la sémantique de Java (exception faite des interfaces et du chargement dynamique des classes). Nous utilisons le système Centaur, qui permet de dériver un interprète interactif fondé sur la sémantique naturelle. Les spécifications sont particulièrement lisibles et ont l'originalité de faire cohabiter sémantique opérationnelle structurelle (*small-step*) et sémantique naturelle (*big-step*). Ce travail de spécification s'est assorti d'un travail innovant sur la mise en œuvre de l'interprète dérivé, puisque la simulation du programme Java est animée et qu'une visualisation graphique permet de comprendre les synchronisations et les partages d'objets.

L'ensemble de ces travaux a donné lieu à un dépot à l'Agence de Protection des Logiciels sous le nom "JavaSem" (voir `http://www-sop.inria.fr/oasis/java`).

Nous avons utilisé cette sémantique opérationnelle de Java pour décrire un modèle simulant tous les acteurs des applications Java Card (noyau pour la programmation des cartes à puce), puis programmé ce modèle en Java, ce qui permet d'obtenir un simulateur d'applets Java Card. Nous avons proposé et spécifié des extensions de la sémantique qui prennent en compte la spécificité du modèle Java Card (pas d'activités concurrentes), et construit une interface spécifique à JavaCard (liée aux transactions entre lecteur de carte et carte).

Le simulateur permet à un développeur de construire, mettre au point et tester son application avant de charger cette application sur une carte à puce. Le test peut être soit figé dans un programme (ce qui impose une découverte manuelle des commandes acceptées par une application), soit construit de manière interactive, grâce à une analyse du source d'une application Java Card afin d'en déduire les commandes (et leur format) comprises par cette application.

Pour plus d'information, consulter la page `http://www-sop.inria.fr/oasis/javacard`.

5.3 Bibliothèque ProActive

Mots clés : programmation objet, parallélisme et répartition, représentation Meta-Objet, Metacomputing.

Participants : Françoise Baude, Alexandre Bergel, Denis Caromel [correspondant], Florian Doyon, Fabrice Huet, Olivier Nano, Emmanuel Reuter, Julien Vayssière.

Notre objectif est de permettre l'exécution d'une même application sur une architecture multi-processeurs à mémoire partagée, sur un réseau de stations de travail, sur Internet ou encore sur n'importe quelle combinaison hiérarchique.

Pour attaquer ce problème, nous avons développé une bibliothèque 100% Java, qui fournit des threads transparents, des objets distants, des appels asynchrones avec futurs transparents, et des mécanismes de synchronisation de haut niveau. Cette bibliothèque permet très facilement de répartir et rendre collaborative toute application écrite en Java.

Afin de démontrer la puissance de cette bibliothèque, nous avons développé des applications collaboratives parallèles et distribuées qui permettent à plusieurs utilisateurs de travailler ensemble sur une scène 3D avec des mécanismes d'élection et de synchronisation : l'image de la scène est calculée par un ensemble dynamique de moteurs de rendu utilisant un algorithme de lancer de rayon. Nous avons également développé DIVA (Distributed and Interactive Virtual world in Java), une application répartie collaborative pour un monde virtuel interactif entièrement écrit en Java en utilisant Java3D, RMI, and ProActive.

Enfin, une étude récente a démontré que l'utilisation de ces bibliothèques permet d'économiser 30 % de code. ProActive est ainsi particulièrement adaptée aux applications réparties sur l'Internet grâce à la réutilisation de code initialement non réparti, à une synchronisation automatique et à la possibilité de faire migrer des activités d'une machine à l'autre.

Un environnement de mise au point (IC2D : Interactive Control & Debug for Distribution) de contrôle et inspection, pour les applications développées en ProActive est désormais disponible.

Pour plus d'information, consulter [11] et la page `http://www-sop.inria.fr/oasis/proactive`.

5.4 La bibliothèque C++//

Mots clés : programmation objet, parallélisme et répartition, représentation Meta-Objet, Metacomputing.

Participants : Françoise Baude, Denis Caromel [correspondant], David Sagnol.

C++// est une bibliothèque pour C++ permettant de répartir une application en réutilisant le maximum du code séquentiel. Son implémentation au dessus du système de metacomputing Nexus/Globus permet à une application C++// d'utiliser des ordinateurs répartis sur l'Internet, Globus se chargeant du contrôle d'accès et de l'authentification de l'utilisateur sur ces machines.

Pour plus d'information, consulter [14] et la page `http://www-sop.inria.fr/oasis/c++ll`.

6 Résultats nouveaux

6.1 Environnements de développement et vérification pour Java et Java Card

Participants : Isabelle Attali, Denis Caromel, Carine Courbis, Ludovic Henrio, Henrik Nilsson, Marjorie Russo, Bernard Serpette.

Les travaux autour de la concurrence en Java, menés principalement par Marjorie Russo dans le cadre de sa thèse de doctorat ont plus particulièrement porté cette année sur les aspects

vérification de propriétés liées à la concurrence en Java. De nombreux travaux portent sur le typage et sa sûreté, sur un sous-ensemble de Java qui élimine les aspects "multi-threads" en général. C'est cet aspect qui nous intéresse plus particulièrement, notamment la détermination (et la preuve) de propriétés importantes à la fois sur le langage et sur une application donnée, telle l'absence de deadlock. Ces travaux, en cours, ont donné lieu à publication [18, 15].

Les travaux menés autour de l'environnement Java Card permettant la simulation et le test interactif des applets (grâce à une analyse de dépendances de données et une construction interactive des commandes) ont donné lieu à publication [17, 26]. Ces travaux ont été arrêtés dans le contexte de Centaur, mais repris sur la plateforme SmartTools, notamment à l'aide des *visitors*. Par exemple, certaines analyses liées aux spécificités de Java Card ont été décrites et permettent de déterminer très tôt dans le processus de développement que le code écrit par le développeur est bien du Java Card (au lieu de la phase très tardive liée au convertisseur).

Dans le cadre de l'étude et de la certification de la JCVM (Java Card Virtual Machine), nous avons été amenés à définir et implémenter, en Java, un ensemble d'outils regroupés autour d'une interface utilisateur nommée JCVM Tools. Cette interface comprend :

1. un éditeur de liens. Il prend en entrée un ensemble de fichiers au format objet de la JCVM (capfiles). L'éditeur de liens construit, en mémoire, un programme clos où toutes les références externes sont résolues (voir 6.3).

2. un vérificateur de byte-code. Cet outil complète l'éditeur de liens en vérifiant un certain nombre de propriétés de types sur le programme clos (voir 6.2).

3. un évaluateur. Cet outil permet une exécution pas à pas de la JCVM, de poser des points d'arrêt, de visualiser, à tout moment, l'état interne de la JCVM (pile d'appel, pile d'opérandes courante d'un bloc d'activation quelconque, valeurs courantes des variables locales d'un bloc d'activation).

4. un extracteur Coq. Cet outil permet, à partir d'un programme clos, de générer un fichier contenant la description Coq de ce programme. Ce fichier sert de programme source dans la certification de programme JCVM (voir 6.3).

A terme, la certification de la JCVM devra prendre en compte toutes les étapes transformant un ensemble de programmes partiels, écrit dans un langage évolué (Java ou autre), en un programme clos. Il semble raisonnable de s'appuyer sur le fait que la JVM (Java Virtual Machine) et son format objet associé (fichier *.class* correspondant à la définition d'une classe) seront une étape intermédiaire de ces transformations. Par exemple, les outils de la JCVM Tools s'appuient sur les résultats d'un convertisseur de fichiers *.class* vers un fichier capfile. C'est donc naturellement que nous avons été amenés à définir et implémenter des outils autour de la JVM. La différence la plus significative entre la JVM et la JCVM est que l'édition de liens est faite de manière paresseuse, au fur et à mesure du chargement d'une classe. Le chargement d'une classe est fait par nécessité (création, accès aux champs ou aux méthodes, test dynamique). A titre d'exemple, la classe *Object*, mère de toutes les classes, référence indirectement plus de 180 autres classes ; il apparaîtrait coûteux que tout programme, aussi petit soit-il, nécessite le chargement effectif de toutes ces classes.

Nous avons écrit, en Java, un interpréteur ainsi qu'un vérificateur de byte-code de la JVM. Le même type d'interface utilisateur que la JCVM Tools est proposé. A court terme, nous proposerons un extracteur Coq nous affranchissant ainsi du convertisseur JVM vers JCVM et du format des capfiles.

6.2 Analyses statiques et transformations de programmes

Participants : Isabelle Attali, Gilles Barthe, Denis Caromel, Simao De Sousa, Guillaume Dufay, Romain Guider, Ludovic Henrio, Line Jakubiec, Didier Parigot, Bernard Serpette.

Nous avons écrit un vérificateur de byte-code pour la JCVM (Java Card Virtual Machine) et la JVM (Java Virtual Machine). Beaucoup d'efforts ont été fournis et restent à fournir afin de permettre un maximum d'expressivité. Ces deux analyses ont été intégrées dans les environnements JCVM Tools et JVM Tools (voir 6.1 pour le descriptif de ces environnements). L'interface utilisateur permet de visualiser, au niveau de chaque instruction du byte-code, les types des valeurs se trouvant dans la pile d'opérandes et dans les variables locales. En cas d'échec du vérificateur, l'instruction fautive ainsi que la pile d'opérandes et les variables locales associées sont mises en valeur.

Dans le but d'effectuer une analyse statique des exceptions de sécurité pouvant être levées à l'exécution d'un programme Java Card, nous avons conçu un système d'inférence de type. Le but d'un tel système est de déterminer statiquement à quels contextes peut appartenir chacun des objets utilisés par une application et de conclure en comparant ces contextes avec les contextes possibles d'exécution de chaque instruction. Cette analyse a été intégrée dans l'environnement JCVM Tools. L'interface utilisateur permet de visualiser les instructions pouvant lever des exceptions de sécurité. Une description complète de l'analyse se trouve dans le rapport du stage de DEA de Ludovic Henrio [37].

Nos précédents résultats de la réduction statique sur un langage objet à la Java ont été publiés dans [21].

Romain Guider a soutenu sa thèse [13] sur une méthode d'analyse statique de programmes à objets qui permet de découvrir automatiquement des propriétés des graphes d'objets. On peut notamment, avec cette technique, prouver qu'à un point de programme donné, deux chemins distincts dans le graphe d'objets ne conduisent jamais à une même cellule mémoire, ou encore qu'un objet ne sera plus accessible après la prochaine opération du programme. Ces propriétés permettent différentes manipulations de programmes.

6.3 Spécifications sémantiques et vérifications

Participants : Isabelle Attali, Gilles Barthe, Guillaume Dufay, Marieke Huisman, Line Jakubiec, Simao De Sousa, Didier Parigot.

Face aux impératifs de sécurité réclamés pour les utilisations des *smart cards* (comme la confidentialité, l'authentification, l'intégrité des données stockées sur la carte), le langage JavaCard apporte des solutions spécifiques, comme le mécanisme de *firewall*. De plus, en raison de l'absence de certaines fonctionnalités évoluées de Java dans JavaCard, ce langage se prête

mieux que Java à la vérification formelle de propriétés sur la plate-forme JavaCard elle-même ou bien sur des programmes.

Ainsi, pour établir de manière formelle des propriétés de sécurité, la machine virtuelle JavaCard et son environnement d'exécution (Java Card Virtual Machine, JCVM, et Java Card Runtime Environment, JCRE, respectivement) doivent été formalisés dans un assistant de preuves. Seule une telle formalisation, imposée pour atteindre les plus hauts niveaux définis par les procédés de certification tels que *Common Criteria* ou *ITSEC*, peut apporter la confiance requise pour l'utilisation massive des *smart cards*.

Nous avons développé en Coq des modèles exécutables de la Machine Virtuelle JavaCard. Cette formalisation, déposée à l'APP sous le nom "CertiCartes", offre la possibilité de prouver des propriétés sur la plate-forme elle-même ou sur des programmes JavaCard.

En effet, tout programme JavaCard peut être traduit dans l'environnement Coq grâce aux JCVM Tools (voir 6.1). Il peut ensuite être exécuté dans l'environnement Coq, à condition que l'utilisateur fournisse une implémentation des APIs utilisés par le programme en Coq. Il est alors possible de raisonner formellement sur l'exécution du programme.

Au niveau des propriétés de la plate-forme, nous avons également formalisé le byte-code Verifier comme une interprétation abstraite de la JCVM. Nous avons également établi la correction du byte-code Verifier par la technique des "diagrammes commutatifs". Nos travaux s'orientent maintenant vers la génération automatique de machines virtuelles abstraites (le BCV en est un exemple).

Les résultats sont présentés dans [19, 30, 36].

Nous avons également poursuivi notre réflexion sur les formalismes pour les spécifications sémantiques et les systèmes de preuves [28, 29, 31, 16, 20, 32].

Nos travaux dans ce domaine complémentent nos recherches sur l'utilisation des systèmes de types et de l'analyse statique pour sécuriser le code mobile (voir 6.2).

6.4 Outils et méthodologie pour la modélisation et la vérification de politiques de sécurité

Participants : Isabelle Attali, Gilles Barthe, Denis Caromel, Simao De Sousa, Refik Molva, Didier Parigot, Yves Roudier, Marjorie Russo.

Nous étudions, 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'applique naturellement à Java et JavaCard.

Nous étudions l'utilisation des systèmes de preuves pour établir l'adhérence d'un programme donné à une certaine politique de sécurité. Dans un premier temps, nous avons dressé un panorama des outils pour la spécification et vérification formelle de logiciel embarqué [24].

L'étape suivante consiste à définir une méthodologie de simplification de modèles en fonction d'une politique de sécurité (donc de propriétés à vérifier). Nos travaux dans ce domaine complémentent nos recherches sur l'utilisation des systèmes de types et de l'analyse statique pour sécuriser le code mobile (voir 6.2).

Par ailleurs, dans [34], nous avons étudié comment la réflexion et les MOP (Meta-Object Protocol) pouvaient se combiner dans le cadre des architectures logicielles à base de compo-

sants. Ces travaux nous ont amenés à proposer un guide de combinaison des ensembles de permissions dans le cadre du modèle de sécurité Java, ceci afin d'obtenir une politique de sécurité assurant l'absence d'erreurs à l'exécution tout en appliquant le principe de permissions minimales (least priviledge permission set).

6.5 Etude formelle des modèles à objets distribués

Participants : Denis Caromel, Ludovic Henrio, Fabrice Huet, Bernard Serpette.

En collaboration avec Sara Alouf et Philippe Nain (Mistral), nous avons étudié la modélisation de différentes méthodes de localisation d'agents mobiles. Pour assurer la communication avec un agent mobile, deux approches principales sont possibles, l'une centralisée et l'autre distribuée. Nous utilisons la modélisation par chaînes de Markov pour comparer les performances de ces deux approches. L'objectif est de trouver un modèle qui soit assez réel dans chaque cas et de définir les valeurs des paramètres du système qui assurent la stabilité. De plus, nous souhaitons parvenir à des recommandations sur l'approche à choisir pour une application donnée.

Dans l'approche centralisée, une chaîne de Markov à 22 états a été étudiée. Des expérimentations sur le système réel sont en cours et une première comparaison entre les résultats analytiques et expérimentaux a été faite. L'allure générale des courbes obtenues est la même, ce qui valide la méthode choisie. Actuellement, les expériences concernent des machines sur un réseau local, mais il est prévu de les étendre un MAN, voire un WAN (Internet).

Dans l'approche distribuée, nous établissons une formule close pour le temps de réponse du système. Dès que celle-ci est disponible, une comparaison avec les résultats expérimentaux sera faite. Au vu de cette comparaison, le modèle étudié pourra être adopté ou bien remanié.

Par ailleurs, dans le cadre de la thèse de Ludovic Henrio (débutée en Octobre 2000), nous avons abordé l'étude des caractéristiques que devrait avoir un calcul d'objets pour nous permettre de démontrer un certain nombre de nouvelles propriétés sur le modèle de répartition utilisé pour la bibliothèque ProActive. Ces travaux devraient aboutir à la démonstration d'un certain nombre de conjectures formulées sur ce calcul.

6.6 Implémentation des langages à objets

Participants : Bernard Serpette.

En collaboration avec Manuel Serrano de l'Université de Nice Sophia Antipolis, nous avons développé un compilateur de Scheme vers la JVM. Nous avons utilisé comme entrée le dernier arbre de syntaxe abstraite (AST) disponible dans le compilateur Bigloo (compilateur de Scheme vers C écrit par Manuel Serrano). Cet AST est celui qu'utilise Bigloo juste avant la génération de code C. A partir de cet AST, nous générons directement un fichier objet (*.class*) de la JVM. Ce travail a été écrit en Scheme et directement intégré au compilateur Bigloo. Outre le plaisir d'écrire dans un style plus libertaire que Java, ce développement a permis de vérifier les résultats suivants :

1. La JVM est un candidat **possible** comme machine cible pour des langages autre que Java.

L'adresse `http://grunge.cs.tu-berlin.de/~tolk/vmlanguages.html` recense déjà plus de 100 langages utilisant la JVM comme cible.

2. La JVM est un candidat **acceptable** comme machine cible. Les deux concessions majeures que nous avons dû faire sont les nombres boxés (i.e. la valeur d'un nombre peut se retrouver encapsulée dans un objet) et l'appel d'ordre supérieur (appel d'une fonction calculée) nécessitant un accès indirect au travers d'une table globale à chaque module. Néanmoins, les analyses frontales de Bigloo (analyse du flot de contrôle et inférence de type) permettent de supprimer ces inconvénients dans une majorité des cas. Ainsi, sur l'ensemble des bancs d'essai de Bigloo, nous avons obtenu un ratio compris entre 2 et 4 entre la JVM et C (ces résultats ont été obtenus sur une machine à base de Sparc). Ce ratio est très acceptable en regard de la portabilité que nous avons gagnée.

3. Enfin, l'accès à des fichiers objets de la JVM créés par des générateurs autres que des compilateurs Java nous a permis de mettre en avant le manque de finesse des vérificateurs de byte-code standard. Ces derniers rejettent assez rapidement les programmes écrits dans des dialectes autre que Java. Ceci nous amènera à effectuer des recherches approfondies sur ce type d'analyse statique.

D'autre part, nous avons spécifié et implémenté en Scheme un assembleur de la JVM prenant en entrée une version lisible d'un fichier *.class*. Ce travail a été effectué dans un double but :

1. Cet assembleur est le langage cible du compilateur Scheme vers JVM que nous avons implémenté.

2. Dans le cadre de l'étude sur les vérificateurs de byte-code, il s'avérait indispensable de pouvoir générer des classes partiellement fausses (ou à la limite de l'incorrection) afin de tester les résultats des vérificateurs de byte-code.

Cet assembleur permet d'effectuer certaines optimisations standard comme l'*inlining*, le *branch tensionning*, l'élimination de code mort et la suppression de variables locales lorsque la valeur de ces dernières peut rester dans la pile d'opérandes. Un résultat intéressant que nous avons obtenu est qu'une transformation de programme relativement naïve comme l'*inlining* peut transformer un programme accepté par le vérificateur de byte-code de Java1.2 en un programme équivalent non accepté par ce même vérificateur.

6.7 Bibliothèques pour la répartition

Participants : Françoise Baude, Alexandre Bergel, Denis Caromel, Florian Doyon, Fabrice Huet, Olivier Nano, David Sagnol, Julien Vayssière.

Nous développons un modèle de programmation concurrente et répartie à objets ainsi que deux implémentations sous la forme de bibliothèques : l'une pour C++, appelée C++// (`http://www-sop.inria.fr/oasis/c++ll`), l'autre appelée ProActive, 100 % Java (`http://www-sop.inria.fr/oasis/proactive`), faisant suite aux travaux autour d'Eiffel// .

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 objectifs étant de parvenir à une conservation de la sémantique) ;
- de fournir un modèle haut niveau de gestion de la concurrence entre objets actifs.

Le jeu de primitives que fournit le MOP (Meta Object Protocol) est utilisable pour la définition de mécanismes plus élaborés que le "simple" appel distant de méthode. En particulier, nous avons introduit un mécanisme de migration d'objets utilisable par le programmeur grâce à un ensemble minimal de primitives, mais dont la gestion est totalement transparente [22]. Il devient dès lors possible de construire des API d'agents mobiles grâce à la construction d'itinéraires. Nous avons démontré qu'une classe d'applications pouvant parfaitement être supportée par le modèle offert par ProActive consiste en de l'administration de systèmes en réseau à l'aide d'agents mobiles [38].
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. La seule implication du programmeur est de déclarer certains des paramètres d'appel de la méthode comme étant des *later*, qui sont en fait gérés à la façon dont le sont les *futurs*. Nous avons poursuivi nos expériences basées sur la version de C++// au dessus du système de metacomputing Globus/Nexus. Nous avons ainsi démontré l'intérêt pratique de cette technique qui permet d'améliorer les performances d'invocation de services distants, que ce soit en environnement Intranet ou en environnement metacomputing [14, 33].
Nous avons conçu et développé un environnement de mise au point, contrôle, inspection, pour les applications développées en ProActive : IC2D (Interactive Control & Debug for Distribution).

Ses caractéristiques principales sont :

- interactivité forte (visualisation graphique de la topologie d'une application répartie, nombreux contrôles à la souris) ;
- possibilité pour l'utilisateur d'influer sur le comportement de l'application répartie (par exemple, choix dynamique de nœuds où créer de nouveaux objets actifs, déplacement d'un objet actif d'un nœud à un autre pendant l'exécution de l'application, etc),

sans introduire aucune modification dans le code de l'utilisateur, ni dans le cœur de la bibliothèque ProActive elle-même. Le principe consiste à rajouter un moniteur qui va être informé de façon transparente d'événements jugés importants pour suivre le déroulement d'une application à objets actifs répartis (tels par exemple les envois et réceptions de requêtes de service et de leurs réponses). Un prototype [35] est en cours de développement.

Toute application répartie doit s'exécuter dans un environnement lui donnant accès à un certain nombre d'entités ou ressources tant matérielles que logicielles. De telles entités s'utilisent par le biais d'interfaces prenant la forme d'un service. Afin de pouvoir utiliser une entité, il est nécessaire de pouvoir la nommer. Et, si on n'en connait pas le nom, il est donc nécessaire de parvenir à le découvrir.

Jini est une bibliothèque proposée par SUN pour la programmation distribuée, où la découverte des entités distantes se fait 'par type' au contraire des systèmes classiques de découverte 'par nom'. Jini souffre cependant d'une certaine rigidité. Ainsi, nous proposons une solution à base d'*adapteurs* [39] pour effectuer la conversion entre deux types non compatibles, mais qui décrivent cependant des entités offrant les mêmes sortes de services. Nous projetons d'intéger ce mécanisme à la bibliothèque ProActive, pour dans un premier temps, découvrir les noms de machines qui hébergeront des nœuds ProActive.

6.8 Sécurisation des applications à objets distribuées

Participants : Françoise Baude, Denis Caromel, Fabrice Huet, Refik Molva, Yves Roudier, Bernard Serpette, Julien Vayssière.

Dans le cadre d'un financement CNRS (montant 250 KF pour deux ans, voir 8.2.3), démarré en décembre 1999, en partenariat avec R. Molva (Eurecom) et G. Castagna (ENS-Ulm), ce projet de recherche vise à développer une méthodologie et les outils associés afin de faciliter le développement d'applications réparties, collaboratives sur Internet.

Il s'agit, par exemple, d'expérimenter sur ProActive la topologie prônée par le Seal Calculus afin d'obtenir des propriétés spécifiques de sûreté et de sécurité pour les applications ProActive. La définition d'une boite à outils (API, etc.) permettra à des non-spécialistes de construire plus facilement des applications complexes réparties et sécurisées (e-commerce, Intranet, services, etc.).

D'autre part, et dans le cadre de cette thématique, des membres de cette collaboration se sont fortement impliqués dans l'organisation d'une table ronde sur la sécurité lors de la conférence ECOOP'2000. L'objectif était de faire le point sur les outils et techniques permettant le développement de logiciels sécurisés, en particulier dans le cadre du code mobile et des applications Internet. Une question importante évoquée et largement débattue lors du panel semble être : a-t-on besoin de primitives spécifiques de sécurité au niveau des langages de programmation? Sans être unanime, la réponse semble être positive; il reste maintenant à savoir lesquelles. Après édition, la transcription de la table ronde a été publiée dans [23].

7 Contrats industriels (nationaux, européens et internationaux)

7.1 Environnements de développement et vérification pour Java et Java Card

Participants : Isabelle Attali, Denis Caromel, Carine Courbis, Henrik Nilsson, Didier Parigot, Marjorie Russo.

Il s'agit d'un contrat de recherche avec Bull/CP8 (1998 - 2001) d'un montant de 1 362 KF

(pour une description des activités scientifiques, voir 6.1).

7.2 SmartTools: outils génériques pour la construction d'environnements de développement

Participants : Isabelle Attali, Denis Caromel, Pascal Degenne, Alexandre Fau, Joël Fillon, Francis Montagnac, Didier Parigot.

Action Dyade SmartTools démarrée en Juin 1999 sur le thème "outils génériques pour la construction d'environnements de développement" (voir 5.1), avec la participation de Microsoft (pour un montant de 440 KF).

7.3 Formavie - Modélisation Formelle et Certification Sécuritaire pour Machine Virtuelle Embarquée

Participants : Isabelle Attali, Gilles Barthe, Guillaume Dufay, Simao De Sousa, Line Jakubiec, Didier Parigot.

Projet OPPIDUM, démarré le 1er juin 1999 d'un montant de 1 360 KF.
Les partenaires industriels sont Bull CP8 et Schlumberger SC & T.
Le travail effectué par l'équipe dans le cadre de ce projet est développé dans 6.4.

7.4 Contrat SUN Microsystems

Participants : Alexandre Bergel, Denis Caromel, Fabrice Huet, Olivier Nano, Julien Vayssière.

Sun Microsystems supporte (15 000 $) les travaux sur la bibliothèque ProActive, notamment les outils liés au Meta-computing (activités de JavaGrande). Les résultats scientifiques sont développés en 6.7.

8 Actions régionales, nationales et internationales

8.1 Actions régionales

8.1.1 Programmation répartie et collaborative pour Internet

Participants : Denis Caromel, Julien Vayssière.

Cette action, soutenue par la Région PACA, en partenariat avec la société Questel (fournisseur de recherche d'antériorité pour le dépôt de brevets), permet de financer la thèse de doctorat de Julien Vayssière.

8.2 Actions nationales

8.2.1 Action de Recherche Coopérative Java Card

Participants : Isabelle Attali, Gilles Barthe, Denis Caromel, Carine Courbis, Alexandre Fau, Romain Guider, Bernard Serpette.

L'action de recherche coopérative JavaCard: sémantique, optimisations et sécurité (voir `http://www.irisa.fr/lande/jensen/javacard.html`) a démarré en Janvier 1998 pour deux ans.

L'action JavaCard vise à coordonner les travaux effectués à l'INRIA autour des aspects formels de Java. L'action se focalise sur la sémantique et la vérification des programmes JavaCard, la version de Java dédiée aux cartes à puces.

L'action JavaCard s'est terminée par l'organisation d'un workshop en association avec le JavaCard Forum (voir `http://www.irisa.fr/lande/jensen/jc-workshop.html`). Le workshop a réuni plus de 80 participants à Cannes, le 14 Septembre.

Les activités de recherche sont décrites en 6.1 et 6.3.

8.2.2 Action de Recherche Coopérative S-Java

Participants : Isabelle Attali, Gilles Barthe, Guillaume Dufay, Marieke Huisman, Line Jakubiec, Bernard Serpette.

L'action de recherche coopérative S-Java : Combinaison d'Outils Formels pour la Sécurisation de Programmes Java `http://www-sop.inria.fr/oasis/SJava` a démarré en Janvier 2000 pour deux ans.

L'action est coordonnée par le projet Oasis, et a pour partenaires les projets INRIA Coq, Lande, Meije, l'équipe Sémantique et Interprétation Abstraite (DMI) de l'ENS-Ulm et Trusted Logic. Deux séances de travail réunissant l'ensemble des partenaires ont eu lieu en mars et en septembre. Un groupe plus restreint s'est également réuni à Sophia Antipolis en octobre. Marieke Huisman, qui a effectué sa thèse sous la direction de Bart Jacobs à Nimejgen, a rejoint l'équipe depuis le 1er octobre 2000 dans le cadre d'un post-doc S-Java.

L'ARC S-Java succède à l'ARC JavaCard et vise à coordonner les travaux effectués à l'INRIA autour des aspects formels de Java et JavaCard. L'action se focalise sur la sémantique et la vérification des programmes Java et JavaCard.

La participation du projet Oasis repose sur la description formelle de la plate-forme JavaCard. Les activités de recherche sont décrites en 6.1 et 6.3.

Il est à noter qu'une partie des partenaires de l'action sont impliqués dans le projet européen IST VerifiCard, qui débutera en Janvier 2001.

8.2.3 Programmation répartie collaborative et sécurisée pour Internet

Participants : Isabelle Attali, Françoise Baude, Denis Caromel, Fabrice Huet, Refik Molva, Yves Roudier, Julien Vayssière.

Financement CNRS (montant 250 KF pour deux ans), démarré en décembre 1999, en partenariat avec R. Molva (Eurecom) et G. Castagna (ENS-Ulm). Ce projet de recherche

vise à développer une méthodologie et les outils associés afin de faciliter le développement d'applications réparties collaboratives sur Internet.

Les activités de recherche sont décrites en 6.8.

8.3 Actions internationales

8.3.1 Spécification formelle et transformation de programmes parallèles

Participants : Isabelle Attali, Denis Caromel, Romain Guider.

Contrat NSF - INRIA - CNRS (1997-2000, montant 165 KF) sur la spécification formelle et la transformation de programmes parallèles concernant J.-L. Gaudiot (USC), en partenariat avec Andrew Wendelborn, université d' Adelaide, Australie. La clôture de cette action nous amène à en tirer un bilan. Les objectifs initiaux étaient, pour le début de cette action, d'utiliser les formats et techniques issus de la programmation fonctionnelle (Sisal) dans le cadre des langages à objets (Eiffel), puis d'étudier l'intégration des paradigmes fonctionnel et impératif. Les objectifs en 1999-2000 ont été d'appliquer ces recherches à Java avec la définition de JavaF et l'analyse statique de programmes Java. Sur le plan des résultats scientifiques, on peut citer notamment :

- définition d'une sémantique et de transformations formelles pour Sisal,
- définition et utilisation d'une extension impérative de Sisal,
- analyse de flot pour Java,
- analyse pour le partionnement de code et l'allocation mémoire.

De manière plus factuelle, les éléments suivants permettent de quantifier les résultats de cette collaboration :

- deux2 réunions par an (France, Los Angeles, Adelaïde);
- soutenance de 4 PhD/Theses (Ehmety en 97, Chen en 98, Woo, Guider en 2000 [13]) ;
- en plus des publications de chaque équipe, des publications communes :
  - 1 chapitre de livre,
  - 2 articles dans des revues (dont 1 à paraitre en 2001 [25]),
  - 3 conférences internationales (dont 1 en 2001 [40]).

8.3.2 Langages Objets Concurrents

Participants : Isabelle Attali, Denis Caromel, Marjorie Russo.

Collaboration INRIA-Brésil (1995-2000, montant 60 KF) avec Fabio Da Silva, Paolo Borba (Recife) et Sidi Ould Ehmety (São Luis) sur la programmation à objets. Le département informatique de l'université de Recife et notre équipe partagent de nombreux sujets d'intérêt,

que ce soit au niveau de l'étude des langages concurrents ou de la spécification de sémantiques formelles de langages à objets. Les contacts sont réguliers et les rencontres à l'occasion de conférences également.

8.3.3 Concurrence et applications

Participants : Denis Caromel, Julien Vayssière.

Denis Caromel est co-responsable du groupe de travail "Concurrence et applications" (The Concurrency, Applications, and Benchmarks Group") du Java Grande Forum (voir `http://www.javagrande.org`). Dennis Gannon (Indiana University and NASA Ames, USA) est co-responsable de ce groupe.

L'objectif du Java Grande Forum est de constituer un groupe de conseil et de pression pour faciliter l'utilisation de la plate-forme Java dans les applications hautes performances.

8.4 Visites, Participations à des conférences, et invitations de chercheurs

- Visites :
  - Denis Caromel a fait une présentation invitée au laboratoire CRS4 - Cagliari - Italy, le 17 février.
  - Denis Caromel a fait une présentation invitée à Nantes, Ecole des Mines, le 9 mars.
  - Denis Caromel a fait une présentation invitée au séminaire Java du CEA de Saclay, sur la programmation répartie et parallèle en Java, le 23 mars.
  - Didier Parigot a été invité chez Microsoft Research à Cambridge les 16 et 17 avril.
  - Exposé invité de Henrik Nilsson : "Declarative Debugging for Lazy Functional Languages", à l'Université de Yale (UOGI), mission du 30 mars au 5 avril.
  - Visites de travail de Gilles Barthe à l'Universidade do Minho, au Portugal, du 9 au 14 mai, et du 5 au 8 juillet.
  - Visite de Henrik Nilsson, pour présentation de ses travaux au projet LANDE à Rennes le 12 mai.
  - Visite de travail de Gilles Barthe à l'université de Chalmers, Göteborg, Suède, le 16 juin.
  - Isabelle Attali et Denis Caromel ont participé au groupe de travail NSF-INRIA-CNRS, du 9 au 21 juillet 2000, à Los Angeles.
- Participations écoles :
  - Participation de Fabrice Huet et Julien Vayssière à l'Ecole d'hiver ISYPAR 2000 à Toulouse, du 1er au 3 février.
  - Participation à la formation Microsoft (22-24 février) de l'équipe SmartTools.
  - Participation de Guillaume Dufay et Julien Vayssière (étudiants) et Gilles Barthe et Simão Sousa (organisateurs) à l'Ecole APPSEM (Portugal, 9-15 septembre).

- Participations Conférences :
  - Participation de Gilles Barthe (conférencier invité) à EWSCS'5 du 27 février au 3 mars.
  - Participation de Gilles Barthe au workshop "Model checking & Program Analysis" à Schloss Ringberg, en Allemagne du 19 au 23 février.
  - Participation de Line Jakubiec et Bernard Serpette à la journée "Système pour carte à puce" chez GEMPLUS, à Gemenos, le 6 mars.
  - Participation de Gilles Barthe à FOSSACS'2000 (Berlin, 27-31 mars).
  - Participation de Julien Vayssière à la conférence PA JAVA 2000 du 11 au 16 avril.
  - Participation de Julien Vayssière à la conférence HPCN'2000, à Amsterdam le 9 mai.
  - Dans le cadre de la conférence ECOOP 2000 du 12 au 16 juin 2000, participation de plusieurs membres de l'équipe au workshop "Formal Techniques for Java Programs", et de Julien Vayssière au workshop "Reflection".
  - Gilles Barthe et Guillaume Dufay ont participé au "Workshop on Security, Middleware, and Languages", les 15 et 16 juin, à Stockholm.
  - Participation de Françoise Baude à la conférence EUROPAR 2000 à Münich, du 29 au 31 août.
  - Romain Guider a participé à la conférence FMOODS, Stanford, 6-8 septembre.
  - Carine Courbis a participé à la conférence CARDIS, Bristol, 20-22 septembre.
  - Participation de Julien Vayssière au workshop GRIDFORUM-5, à Boston, du 15 au 17 octobre 2000.
  - Participation de Gilles Barthe et Bernard Serpette à la conférence LPAR'2000, La Réunion, 6-12 novembre 2000.
  - Participation de Denis Caromel à SuperComputing 2000 à Dallas, du 6 au 10 novembre.
  - Denis Caromel a participé à la Conférence NOTERE, Paris du 22 au 24 novembre, il a présidé la session "Réflexion".
  - Denis Caromel a participé le 15 décembre au workshop "Advanced Object Programming fo Scientific Computing", Paris.
- Invitations de chercheurs :
  - Maria Joao Frade, doctorante encadrée par Gilles Barthe, du 10 janvier 2000 au 10 février 2000.
  - Visite de Pierre-Yves Saintoyant dans le cadre de l'action SmartTools (présentation et démonstrations) le 23 février.
  - Dans le cadre de l'action intégrée PROTEUS, Marjan Mernick et deux de ses étudiants ont visité le projet (3 semaines en juillet).

- Visite de Gustavo Betarte, Universidad de la Republica, Uruguay (19 juin-6 juillet).
- Séminaire de Matthias Felleisen, de l'université de Rice le 4 juillet.
- Exposé de Marieke Huisman (Katholieke Universiteit Nijmegen) le 26 juillet.
- Visite de Bart Jacobs et Erik Poll, de Katholieke Universiteit Nijmegen, le 15 septembre.
- Visite Peter Ryan et Matthew Hennessy le 17 novembre.

9 Diffusion de résultats

9.1 Animation de la Communauté scientifique

- Participation importante du projet OASIS à la conférence ECOOP 2000, organisée à Cannes (Palais des Festivals) et Sophia Antipolis (INRIA et ESSI), du 12 au 16 juin 2000, 700 participants :
  - Denis Caromel : co-président du comité d'Organisation,
  - Isabelle Attali : co-organisateur des tutoriaux,
  - Bernard Serpette : co-organisateur du programme social,
  - Françoise Baude : organisateur des tables rondes,
  - Fabrice Huet, Marjorie Russo, Julien Vayssiere, Ludovic Henrio : étudiants volontaires,
  - Maryse Renaud pour la logistique liée aux tutoriaux.
- Françoise Baude est membre de la commission des spécialistes, 27^{e} section de l'UNSA.
- Françoise Baude a participé au comité de programme de RenPar 2000.
- Isabelle Attali est membre du comité de direction du GDR ALP : Algorithmique, Langages et Programmation.
- Denis Caromel est co-organisateur du Workshop "Java for Parallel and Distributed Computing", (avec S. Chaumette, G. Fox), 1-5 mai, dans le cadre de IPDPS'2000, Cancun, actes Springer, Lectures Notes in Computer Science (LNCS).
- Françoise Baude participe au comité d'édition de la revue Calculateurs Parallèles, éditions Hermès (depuis Mars 96) et coordonne un numéro spécial de cette revue sur le MetaComputing, à paraître en 2001.
- Françoise Baude a participé au comité de programme de EUROPAR 2000, du 29 août au 1er septembre (vice-chair du workshop Object-Oriented Architectures, Tools and Applications).
- Denis Caromel participe au comité d'édition de la revue L'OBJET, éditions Hermès (depuis janvier 97). Numéro spécial "Parallélisme, distribution et approches objets", Technique et Science Informatiques (TSI), à paraiître, Hermès, 2000.

- Didier Parigot a organisé le 3e Workshop sur "Les grammaires attribuées et leur applications" (WAGA'00), le 7 juillet au Portugal, dans le cadre de MPC'2000 (voir `http://www-sop.inria.fr/oasis/waga00.html`).

- Gilles Barthe a organisé un workshop (DTP'2000) sur les sous-typages et les types dépendants en programmation, le 7 juillet à Ponte de Lima, Portugal, dans le cadre de MPC'2000 (voir (voir `http://www-sop.inria.fr/oasis/DTP00`).

- Collaboration de Gilles Barthe avec le groupe de méthodes formelles et logique de l'université du Minho. Visites, encadrement de Maria João Frade (doctorant), Olivier Pons et Tarmo Uustalu (post-doctorants). Obtention d'un financement INRIA-ICCTI pour 2001-2002 (les montants ne sont pas encore connus).

- JAVA CARD Workshop, co-organisé par Isabelle Attali et Thomas Jensen (Irisa), le 14 septembre 2000, à Cannes : 80 participants (voir `http://www.irisa.fr/lande/jensen/jc-workshop.html`).

- Plusieurs membres du projet ont participé à :

  - des jurys de thèse de doctorat en tant que :
    - directeur : Romain Guider (UNSA), David Sagnol (UNSA),
    - examinateur : Eric Noulard (Prisme, Versailles);
  - des jurys d'habilitation à diriger des recherches en tant que rapporteur : Dominique Colnet (Loria).

  Plusieurs membres du projet ont évalué des articles soumis aux :

  - conférences suivantes :
    WAGA'00, ECOOP'00, HPCN'00, IPDPS'00,ISPAN'00, RenPar'00, EUROPAR'00, NOTERE,
  - et revues suivantes : TSI, TAPOS, Concurrency Practice and Experience, Software Practice and Experience, The Computer Journal, TCS, Mathematical Structures in Computer Science, Journal of Functional Programming, Journal of Logic and Computation.

9.2 Enseignement

- Isabelle Attali est responsable du module optionnel "Sémantique et Sécurité" du DEA d'Informatique de l'UNSA. Isabelle Attali et Gilles Barthe interviennent dans le cours de tronc commun "Méthodes formelles et fiabilité du logiciel" organisé par Robert de Simone (module de 36h).

- Isabelle Attali

  - est membre du conseil scientifique de ce DEA,

  - est responsable scientifique depuis 1995 de l'Ecole Jeunes Chercheurs en Programmation, qui réunit environ 40 doctorants pour 15 jours de cours chaque année. En 2000, l'Ecole a eu lieu à l'ENS à Lyon, du 20 au 30 mars (voir `http://www-sop.inria.fr/EJC2000/`).
  - est responsable de la commission Formation Emploi de l'association Télécom-Valley.

- Gilles Barthe

  - intervient dans le module "Sécurité: réseaux, systèmes, applications" du DESS Télécommunications de l'université de Nice Sophia-Antipolis, année 1999-2000 (6 heures de cours, 6 heures de T.D.).
  - participe en tant qu'organisateur, membre du comité scientifique et orateur à "APPSEM 2000: International Summer School on Applied Semantics", Portugal, 9-15 septembre 2000, plus de 100 participants (voir `http://www-sop.inria.fr/oasis/Caminha00`). Il a donné un cours avec Thierry Coquand de Chalmers University, sur les systèmes de types dépendants ("Dependent Types in Programming").

- Françoise Baude

  - est coordinateur au département d'informatique de la Licence d'informatique.
  - est responsable du module "Algorithmique et environnements de programmation parallèles et distribués", de la filière SAR de l'ESSI 3^{e} année,
  - est responsable de l'enseignement "Internet et Bureautique" en DEUG 1re année, tronc commun de l'UNSA (1200 étudiants),
  - est responsable de l'enseignement de Licence d'informatique de l'UNSA "Concepts des systèmes d'exploitation et gestion de la concurrence", ainsi que de celui "Utilisation avancée de systèmes d'exploitation",
  - participe à l'enseignement de Java en Licence MASS et en Licence Informatique,
  - participe au module de tronc commun du DEA RSD "Algorithmique parallèle et distribuée".

- Denis Caromel

  - est responsable des modules de Maitrise d'informatique "C++" et "Programmation distribuée et Administration Système",
  - 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,
  - est responsable depuis 1995 du Module "Langages de Programmation Concurrente, Parallèle, Distribuée" commun aux DEAs Informatique et RSD de l'UNSA,
  - est coordinateur au Département Informatique du DESS Télécommunications, université de Nice - Sophia Antipolis.

- Carine Courbis intervient dans les cours suivants :
  - TPs de Java, IUT GTR Sophia Antipolis pour les 2^{es} années, 24 heures,
  - TPs de Base de Données, IUT GTR Sophia Antipolis pour les 2^{es} années, 54 heures.
- Fabrice Huet intervient dans les cours suivants :
  - 6 Heures en DU à l'IUT Fabron : Corba, RMI et Java (Redaction des TDs avec Michel Syska),
  - 9 Heures en 1^{re} année à l'ESINSA : Java
  - 50 Heures à l'IUT de Sophia, Infocom 1^{re} et 2^{e} année : HTML (Rédaction d'une partie des TDs),
- Julien Vayssière intervient dans les modules suivants :
  - 12 heures de Td dans le module "Programmation distribuée et architectures d'applications réseaux" du DESS Telecom de l'UNSA,
  - 4 heures de cours et 4 heures de TD dans le module "Sécurité : réseaux, systèmes, applications" du DESS Telecom de l'UNSA,
  - 19,5 heures de TD dans le module "Internet et bureautique" du tronc commun de DEUG de l'UNSA,
  - 36 heures de TD dans le module "Architecture et Assembleur" de la licence d'informatique de l'UNSA.

10 Bibliographie

Ouvrages et articles de référence de l'équipe

[1] I. Attali, D. Caromel, S. O. Ehmety, S. Lippi, «Semantic-based visualization for parallel object-oriented programming», *in : Proceedings of OOPSLA'96*, *ACM Sigplan Notices*, *31, 10*, ACM Press, San Jose, CA, octobre 1996.

[2] I. Attali, D. Caromel, S. O. Ehmety, «A Natural Semantics for Eiffel Dynamic Binding», *ACM Transactions on Programming Languages and Systems (TOPLAS) 18*, 5, novembre 1996.

[3] G. Barthe, «Order-sorted inductive types», *Information and Computation 149*, 1, février 1999, p. 42–76.

[4] D. Caromel, F. Belloncle, Y. Roudier, «The C++// System», *in : Parallel Programming Using C++*, MIT Press, 1996, ISBN 0-262-73118-5.

[5] D. Caromel, W. Klauser, J. Vayssiere, «Towards Seamless Computing and Metacomputing in Java», *Concurrency Practice and Experience 10*, 11–13, novembre 1998, p. 1043–1061.

[6] D. Caromel, «Towards a Method of Object-Oriented Concurrent Programming», *Communications of the ACM 36*, 9, septembre 1993, p. 90–102.

[7] L. Correnson, E. Duris, D. Parigot, G. Roussel, «Generic Programming by Program Composition (position paper)», *in : Workshop on Generic Programming*, Marstrand, Sweden, juin 1998. conjunction with MPC'98, `ftp://ftp-sop.inria.fr/oasis/Didier.Parigot/publications/Correnson98a.p%s.gz`.

[8] N. Furmento, F. Baude, «Schooner: An Object-Oriented Run-time Support for Distributed Applications», *in : In K. Yetongnon and S. Hariri, editors , Proceedings of Parallel and Distributed Computing Systems (PDCS'96), Dijon, 1*, International Society for Computers and their Applications (ISCA), p. 31–36, septembre 1996.

[9] D. Parigot, G. Roussel, M. Jourdan, E. Duris, «Dynamic Attribute Grammars», *in : Int. Symp. on Progr. Languages, Implementations, Logics and Programs (PLILP'96)*, H. Kuchen, S. D. Swierstra (éditeurs), *Lecture Notes in Computer Science, 1140*, Springer-Verlag, p. 122–136, Aachen, septembre 1996, `ftp://ftp-sop.inria.fr/oasis/Didier.Parigot/publications/plilp96.ps.gz`.

[10] B. Serpette, «Approximations d'évaluateurs fonctionnels», *in : Proceedings of WSA (Workshop on Static Analysis)*, Bigre, p. 79–90, 1992.

Livres et monographies

[11] D. Caromel, S. Chaumette, G. Fox, P. Graham (éditeurs), *Java for Parallel and Distributed Computing*, IPDPSP'00, Cancun, Mexico, Springer Verlag No 1800, mai 2000. ISBN 3-540-67442-X.

[12] D. Parigot, M. Mernik (éditeurs), *Third Workshop on Attribute Grammars and their Applications WAGA'00*, MPC'2000, Ponte de Lima, Portugal, INRIA Rocquencourt, july 7 2000. Satellite event of MPC'2000, `http://www-sop.inria.fr/oasis/WAGA00/waga00.html`.

Thèses et habilitations à diriger des recherches

[13] R. Guider, *Analyse statique de programmes Java: Application à la parallélisation*, thèse de doctorat, université de Nice - Sophia Antipolis, septembre 2000.

[14] D. Sagnol, *Correction et optimisation de programmes à objets parallèles*, thèse de doctorat, Université de Nice - Sophia Antipolis, Juin 2000.

Articles et chapitres de livre

[15] I. Attali, D. Caromel, M. Russo, «Graphical Visualization of Java Objects, Threads, and Locks», *IEEE Computer Society*, 2000, disponible à partir de décembre 2000,, `http://computer.org/dsonline`.

[16] G. Barthe, J. Hatcliff, M. Sørensen, «Weak Normalization implies Strong Normalization in Generalized Non-Dependent Pure Type Systems», *Theoretical Computer Science*, 2001, Accepted for publication.

Communications à des congrès, colloques, etc.

[17] I. ATTALI, D. CAROMEL, C. COURBIS, L. HENRIO, H. NILSSON, «SmartTools for Java Card», *in : Smart Card Research and advanced Applications - Proceedings of CARDIS'2000*, A. W. Josep Domingo-Ferrer, David Chan (éditeur), Kluwer Academic Publishers, Bristol, UK, 2000.

[18] I. ATTALI, D. CAROMEL, M. RUSSO, « From Executable Formal Specification to Java Property Verification», *in : Proceedings of FTfJP'00—ECOOP Workshop on Formal Techniques for Java Programs*, S. Drossopoulou, S. Eisenbach, B. Jacobs, G. T. Leavens, P. Müller, A. Poetzsch-Heffter (éditeurs), 2000, `ftp://ftp-sop.inria.fr/oasis/Isabelle.Attali/ftjp.ps.gz`.

[19] G. BARTHE, G. DUFAY, L. JAKUBIEC, B. P. SERPETTE, S. M. D. SOUSA, S.-W. YU, « Formalisation of the Java Card Virtual Machine in Coq », *in : Proceedings of FTfJP'00—ECOOP Workshop on Formal Techniques for Java Programs*, S. Drossopoulou, S. Eisenbach, B. Jacobs, G. T. Leavens, P. Müller, A. Poetzsch-Heffter (éditeurs), 2000.

[20] G. BARTHE, F. V. RAAMSDONK, «Constructor Subtyping in the Calculus of Inductive Constructions», *in : Proceedings of FOSSACS'00*, J. Tuyrin (éditeur), *Lecture Notes in Computer Science, 1784*, Springer-Verlag, p. 17–34, 2000.

[21] G. BARTHE, B. SERPETTE, « Static Reduction Analysis for Imperative Object Oriented Languages », *in : Proceedings of LPAR'00*, M. Parigot, A. Voronkov (éditeurs), *Lecture Notes in Computer Science, 1955*, Springer-Verlag, p. 344–361, 2000.

[22] F. BAUDE, D. CAROMEL, F. HUET, J. VAYSSIÈRE, « Communicating Mobile Active Objects in Java», *in : Proceedings of the 8th International Conference - High Performance Computing Networking'2000*, R. W. Marian Bubak, Hamideh Afsarmanesh, B. Hetrzberger (éditeurs), *Lecture Notes in Computer Science, 1823*, Springer-Verlag, p. 633–643, Amsterdam, mai 2000.

[23] R. MOLVA (ORGANIZER), F. BAUDE (EDITOR), «Panel Session: Mobile Code, Internet Security and E-Commerce», *in : Workshop reader of ECOOP'2000*, J. Malenfant, S. Moisan, A. Moreira (éditeurs), *Lecture Notes in Computer Science*, Springer-Verlag, 2000. A paraître.

Divers

[24] I. ATTALI, G. BARTHE, L. JAKUBIEC, P. LADAGNOUS, S. SOUSA, S.-W. YU, « Securing Embedded Software: Methodology and Tools», manuscrit, 2000.

[25] I. ATTALI, D. CAROMEL, Y.-S. CHEN, J.-L. GAUDIOT, A. L. WENDELBORN, « Enhancing Functional and Irregular Parallelism: Stateful Functions and their Semantics», a paraitre dans International Journal on Parallel Programming, 2001.

[26] I. ATTALI, D. CAROMEL, C. COURBIS, L. HENRIO, H. NILSSON, «An Integrated Development Environment for Java Card», a paraitre, 2001.

[27] I. ATTALI, C. COURBIS, P. DEGENNE, A. FAU, D. PARIGOT, « SmartTools: a Generator of Language Developpment Environments Tools», soumis à publication, ETAPS 2001, Gênes, Italie, 2 - 6 avril, 2001.

[28] G. BARTHE, G. BETARTE, M. FRADE, L. PINTO, J. ZWANENBURG, « Extensible Overloaded Functions», soumis, 2000.

[29] G. Barthe, T. Coquand, «Dependent Type Theory», Lecture Notes for the APPSEM Summer School, 2000.

[30] G. Barthe, G. Dufay, L. Jakubiec, S. M. de Sousa, B. Serpette, «A Formal Executable Semantics of the JavaCard Platform», soumis, 2000.

[31] G. Barthe, M. Frade, E. Giménez, L. Pinto, T. Uustalu, «Type-Based Termination of Recursive Definitions», soumis, 2000.

[32] G. Barthe, O. Pons, «Type Isomorphisms and Proof Reuse in Dependent Type Theory», soumis, 2000.

[33] F. Baude, D. Caromel, N. Furmento, D. Sagnol, «Optimizing Metacomputing with Communication–Computation overlap», soumis, 2000.

[34] D. Caromel, J. Vayssière, «Combining Security with Meta Programming in Java», soumis à publication, 2001.

[35] F. Doyon, «Applets communiquantes pour la programmation répartie», juillet 2000.

[36] G. Dufay, *Certification d'abstractions pour la sécurisation de programmes Java*, Mémoire, Université Paris VII, 2000, Rapport de stage D.E.A. Sémantique, Preuves, Programmation (SPP).

[37] L. Henrio, *Analyses de forme et de partage pour applications JavaCard*, Mémoire, Université Paris VII, septembre 2000, `ftp://ftp-sop/oasis/publications/2000/LudovicHenrioStageDEA.ps`.

[38] E. Reuter, *Les Agents Mobiles pour l'administration de Réseaux*, Mémoire, Université de Nice Sophia Antipolis, juillet 2000, `ftp://ftp-sop/oasis/publications/2000/EmmanuelReuterStageDEA.prn.gz`.

[39] J. Vayssière, «Extending Jini's lookup service with adapters», 2001, Exposé présenté lors de la 5^{me} réunion internationale du Grid Forum, Boston, 15-18 octobre 2000.

[40] J. Woo, I. Attali, D. Caromel, J.-L. Gaudiot, A. L. Wendelborn, «Alias Analysis On Type Inference For Class Hierarchy In Java», a paraitre dans Australasian Computer Science Conference, Brisbane, Australie, 29 jan. - 2 fev, 2001.