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

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

1 Composition de l'équipe

Responsable scientifique
Isabelle Attali [DR]
Responsable permanent
Bernard Serpette [CR]
Assistance administrative
Maryse Renaud [Assistante Dyade et contact Oasis jusqu'au 2 septmebre 2001]
Philippe Dereymez [MOO depuis le 17 Septembre 2001]
Personnel Inria
Eric Madelaine [CR]
Francis Montagnac [IR, à temps partiel]
Didier Parigot [CR]
Personnel UNSA
Françoise Baude [Maitre de Conférence, UNSA]
Denis Caromel [Professeur, UNSA, membre IUF depuis le 1er Septembre 2001]
Personnel CP8
Claude Pasquier [Ingénieur CP8, depuis le 23 mars 2001, action SmartTools]
Ingénieurs experts
Pascal Degenne [SmartTools]
Alexandre Fau [SmartTools]
Joël Fillon [SmartTools]
Christophe Held [Formavie depuis le 01/05/2001]
Lionel Mestre [Proactive depuis le 01/04/2001]
Chercheurs invités
Erick Fredj [Jerusalem College of Technology, du 15/08/2001 au 31/08/2001]
Andrew Wendelborn [Univ. Adelaide, du 27/06/2001 au 27/07/2001]
Chercheurs doctorants
Laurent Baduel [Allocataire MESR + Moniteur UNSA, depuis le 01/10/2001]
Rabéa Boulifa [Bourse INRIA, depuis le 02/11/2001]
Arnaud Contes [Boursier DGA, depuis le 05/11/2001]
Carine Courbis [Allocataire MESR, troisième année]
Ludovic Henrio [Boursier DGA, deuxième année]
Fabrice Huet [Allocataire MESR + Moniteur UNSA, troisième année]
Emmanuel Reuter [Salarié (ingénieur IUFM Nice)]
Marjorie Russo [Allocataire MESR puis demi-ATER jusqu'au 01/09/2001]
Julien Vayssière [Boursier Région, UNSA, troisième année]
Chercheur post-doctorant
Baomin Xu [depuis le 15/05/2001]
Stagiaires
Thierry Abbondanza [DESS, Projet et stage télécom du 01/04/2001 au 30/09/2001]

Laurent Baduel [Stage DEA RSD, UNSA du 01/03/2001 au 31/07/2001]
Alexandre Bergel [Maîtrise, UNSA, du 01/10/2000 au 28/02/2001 puis Stage DEA Informatique, UNSA, du 01/03/2001 au 31/10/2001]
Lionel Blavy [Maîtrise, UNSA, du 01/07/2001 au 31/08/2001]
Rabéa Boulifa [DEA ENSEEIHT Toulouse, du 16/07/2001 au 31/10/2001]
Damien Ciabrini [Maîtrise, UNSA, du 01/07/2001 au 31/08/2001]
Abishek Chaturvedi [ITT Kharagpur, du 02/05/2001 au 22/07/2001]
Arnaud Contes [Stage DEA RSD, UNSA du 01/03/2001 au 31/07/2001]
Michael D'Agostino [ESIL, 1ère année, du 01/06/2001 au 31/08/2001]
Pascal Fauconnier [Maîtrise, UNSA, du 01/07/2001 au 31/08/2001]
Thomas Henrot [Maîtrise, UNSA, du 01/07/2001 au 30/09/2001]
Hamoudi Kalla [Stage DEA Orléans, du 01/04/2001 au 15/11/2001]
Anand Meka [ITT Kharagpur, du 02/05/2001 au 28/07/2001]
Olivier Nano [Maîtrise, UNSA, du 01/10/2000 au 28/02/2001]
Vincent Ribaillier [Ecole des Mines, Alès, 3ème année du 26/03/2001 au 31/07/2001]
Fabien Tricoire [Maîtrise, UNSA, du 01/07/2001 au 31/08/2001]
Joseph George Variamparambil [IIT Kanpur du 09/05/2001 au 31/07/2001]
Steve Van der Hoeven [ESSI 2ème année, UNSA, du 10/06/2001au 28/09/2001]
Guillaume Van de Kerckhove [Maîtrise, UNSA, du 01/07/2001 au 30/09/2001]
Gilles Zucchini [Maîtrise, UNSA, du 01/07/2001 au 31/08/2001]

Conseiller scientifique

Claudio Sacerdoti-Coen [Univ. Bologne, du 08/01/2001 au 24/02/2001 et du 09/09/2001 au 21/09/2001]

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 et d'obtenir des résultats 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, Rabea Boulifa, Denis Caromel, Carine Courbis, Pascal Degenne, Alexandre Fau, Joël Fillon, Christophe Held, Ludovic Henrio, Eric Madelaine, Francis Montagnac, Didier Parigot, Claude Pasquier, Marjorie Russo, Claudio Sacerdoti-Coen, Bernard Serpette.

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.

Nous nous intéressons au cas où nous pouvons, par des techniques d'analyse statique et d'interprétation abstraite, extraire un modèle fini d'une application. Nous utiliserons alors des outils de vérification génériques [8], issus des travaux du projet MEIJE et que nous adapterons aux problèmes de preuve de propriétés de sécurité, ou aux problèmes de comportement d'applications distribuées sur des réseaux.

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, 6].

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, Laurent Baduel, Arnaud Contes, Françoise Baude, Alexandre Bergel, Rabea Boulifa, Denis Caromel, Michaël D'Agostino, Ludovic Henrio, Fabrice Huet, Eric Madelaine, Anand Meka, Lionel Mestre, Francis Montagnac, Olivier Nano, Emmanuel Reuter, Vincent Ribailler, Bernard Serpette, Abhishek Chaturvedi, Julien Vayssière, Baomin Xu.

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 [5], ou la construction de bibliothèques pour le parallélisme [3, 4].

4 Domaines d'applications

4.1 Panorama

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 Répartition, mobilité et sécurité : ProActive

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

Participants : Baomin Xu, Françoise Baude, Laurent Baduel, Denis Caromel [correspondant], Arnaud Contes, Ludovic Henrio, Fabrice Huet, Lionel Mestre, 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 [4] et la page `http://www.inria.fr/oasis/proactive`.

5.2 Outils Interactifs Génériques : SmartTools

Participants : Isabelle Attali, Denis Caromel, Carine Courbis, Pascal Degenne, Alexandre Fau, Joël Fillon, Christophe Held, Hamoudi Kalla, Francis Montagnac, Didier Parigot [correspondant], Claude Pasquier, Steve Van Der Hoeven, Joseph George Variamparambil.

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.

L'intérêt et l'originalité de notre approche est de rendre accessible et facilement utilisable des techniques avancées de programmation initialement développées pour les langages de programmation. En particulier, les nouvelles approches de programmation adaptative, par aspects et par composants que nous utilisons largement, nous semblent être un atout très important pour ce type d'applications en termes de modularité et de réutilisation de composant.

Un challenge supplémentaire est de proposer ces nouveaux mécanismes de programmation à des non-spécialistes en les intégrant dans un environnement uniforme et interactif et sur une architecture modulaire largement ouverte aux technologies XML. Les retombées industrielles de ce type de plateforme sont à la hauteur des enjeux des applications du commerce électronique et des cartes à puces. En d'autres termes, l'ouverture vers les langages métiers offre à notre outil un vaste champ d'application et justifie pleinement notre approche générique.

L'originalité et l'innovation de notre approche peuvent se synthétiser en quatre points importants :

1. Accepter en entrée des formalismes W3C (DTD et schéma), ce qui nous permet du même coup de profiter des nombreux développements autour des standards W3C. Notre innovation consiste à proposer pour des documents XML, une méthodologie de programmation (pour décrire les traitements sémantiques) fondée sur les travaux autour des "design patterns", issus de la programmation à objets.
2. Fournir une interface utilisateur conviviale est aussi un impératif ; l'innovation de notre approche est de traiter tous ces aspects d'affichage (y compris l'interface utilisateur) sur un même modèle. Cela permet de proposer une approche homogène et uniforme avec un fort potentiel de réutilisation tant pour la plate-forme SmartTools que pour les environnements produits. Un autre avantage important est que les techniques mises en œuvre permettent d'exporter ces vues graphiques vers d'autres supports comme des browsers.
3. Pour assurer une bonne évolution de l'outil, il était vital de concevoir dès le début une architecture logicielle modulaire (par composants indépendants) et extensible. Nos choix ont été confirmés avec d'une part une mise en œuvre quasiment naturelle d'une version répartie (via la bibliothèque ProActive développée dans l'équipe), mais surtout par la facilité d'ajout de nouveaux composants et d'interconnection avec d'autres plate-formes comme par exemple .NET [28] (avec le protocole SOAP).
4. Enfin, nous proposons une approche originale de programmation par aspect au-dessus de la technique des visiteurs qui ne requiert pas de transformation des sources. Cette approche dynamique a l'intérêt d'être beaucoup plus simple dans sa mise en œuvre. Mais

surtout, elle permet d'utiliser cette approche dans le cadre d'applications du e-commerce pour traiter les problèmes de reconfiguration, d'adaptation et de sécurité.

La qualité des résultats des recherches est attestée par des articles publiés dans des conférences internationales ciblées sur le sujet [19, 18] et au cours desquelles des démonstrations d'une première version du prototype ont été effectuées. Plusieurs articles sont également en soumission ou en cours d'acceptation dans des revues et conférences [31, 30, 39]. La plate-forme SmartTools est en cours d'évaluation chez un partenaire industriel (Dynam-IT) dans le but de développer et commercialiser des logiciels d'accélération de livraison de contenus web. Ces travaux intéressent également la société Ilog, directement impliquée dans les composants logiciels et leur évolution dans le contexte des nouvelles technologies de l'information. De plus, nous allons participer, à partir de Janvier 2002, à un projet européen avec le W3C, nommé "QUESTION-HOW". L'outil SmartTools sera utilisé comme démonstrateur des spécifications/technologies XML.

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

5.3 Sémantiques et Environnements pour Java/Java Card

Participants : Isabelle Attali [correspondante], Lionel Blavy, Rabéa Boulifa, Denis Caromel, Damien Ciabrini, Carine Courbis, Pascal Fauconnier, Ludovic Henrio, Thomas Henrot, Eric Madelaine, Claude Pasquier, Marjorie Russo, Bernard Serpette, Fabien Tricoire, Guillaume Van De Kerckhove, Gilles Zucchini.

Des activités de deux types autour des sémantiques de Java et JavaCard sont actuellement poursuivies : les travaux plus anciens liés au système Centaur sont encore actifs, notamment dans le cadre de la thèse de Marjorie Russo, et d'autre part des adaptations et des nouveaux développements sont en cours dans le cadre de la plate-forme SmartTools autour de Java et Java Card.

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.

Ces travaux ont également débouché sur la formalisation et la vérification de propriétés relatives à la concurrence et aux synchronisations. Ces propriétés sont d'une part inhérentes au langage Java (comme par exemple le fait que deux threads ne peuvent, à un instant donné, avoir un lock sur le même objet) et d'autre part, spécifiques à une application (mais indépendantes d'une machine virtuelle) comme par exemple l'absence de deadlock.

L'ensemble de ces travaux a donné lieu à un dépot à l'Agence de Protection des Logiciels sous le nom "JavaSem". Pour plus d'information, consulter la page `http://www-sop.inria.fr/oasis/java`) ainsi que les documents [17, 14].

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.

Ces travaux ont été repris et étendus au sein de la plate-forme SmartTools, dans le cadre de nos relations avec CP8, ce qui nous permet de produire un environnement intégré permettant la manipulation d'applications Java Card tant au niveau du source que du byte-code. Un serveur de graphe nous permet de proposer une visualisation graphique d'informations statiques. Les outils JVMTools (développés dans l'équipe) nous permettent de produire un évaluateur (simulateur) ainsi qu'un vérificateur au niveau du byte-code. Une analyse de partage d'objets [22] a également été intégrée à l'environnement.
Pour plus d'information, consulter la page `http://www-sop.inria.fr/oasis/javacard`.

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.

Depuis la fin du travail de thèse de David Sagnol, C++// n'est plus vraiment maintenue, mais reste publiquement accessible : en effet, elle nous sert de vitrine auprès d'équipes de calcul scientifique, numérique, qui voudraient utiliser un modèle parallèle, réparti, de programmation orientée objet, mais qui, de par leur souci important de recherche de performance, sont attirés en premier lieu par C++// et non pas par ProActive. Cependant, ces deux bibliothèques offrant le même modèle de programmation, il est assez facile de les diriger vers ProActive, car nous arrivons à les convaincre que, quitte à utiliser un modèle objet, autant utiliser Java. Ainsi, les résultats et l'expérience acquis durant les recherches menées autour de C++// sont reportés sur celles autour de ProActive. Mais notre souci de recherche de performance et d'application de notre modèle de programmation répartie au calcul scientifique parallèle reste d'actualité. Concrètement, nous sommes en train de démarrer une collaboration avec l'équipe CAIMAN de l'UR de Sophia-Antipolis, en premier lieu, en proposant un sujet de stage de DEA intitulé

Metacomputing et objets Java pour la simulation numérique en électromagnétisme.

Pour plus d'information sur C++//, consulter 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, Lionel Blavy, Rabéa Boulifa, Denis Caromel, Damien Ciabrini, Carine Courbis, Pascal Fauconnier, Ludovic Henrio, Thomas Henrot, Eric Madelaine, Claude Pasquier, Marjorie Russo, Bernard Serpette, Fabien Tricoire, Guillaume Van De Kerckhove, Gilles Zucchini.

Les travaux autour de la concurrence en Java, menés principalement par Marjorie Russo dans le cadre de sa thèse de doctorat ont été finalisés [17, 14]. Les recherches ont principalement porté cette année sur la possibilité donnée à l'utilisateur d'agir directement sur l'ordonnanceur et l'entrelacement afin d'explorer les différentes exécutions possibles. Les alternatives possibles au mécanisme round-robin de sélection sont donc : un parcours en profondeur d'abord ainsi qu'un choix aléatoire du prochain thread à exécuter. Ces extensions ont été mises en oeuvre au sein du système Centaur.

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 [16]. 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, et non dans la phase très tardive liée au convertisseur, que le code écrit par le développeur est bien du Java Card.

Comme annoncé l'année passée, nous avons unifié les environnements de développement de la JVM et la JCVM. La spécification des objets utilisés par la machine virtuelle (packages, classes, champs, methodes) nous a permis de décrire de manière concise les calculs suivants :

1. graphe d'héritage simple.
2. graphe d'héritage multiple.
3. graphe de dépendance inter-classes.
4. flow de controle intra-procédural.
5. graphe d'appel.

Nous comptons encore réduire le jeu d'instructions de la machine virtuelle. Nous étions passé des 201 instructions de la JVM à 143 instructions et nous pensons pouvoir arriver à une trentaine d'instructions généralistes. Au delà d'une facilité accrue quant à l'écriture d'analyse sur le code octet, ce travail permettra d'intégrer plus élégament les spécificités de la JVM et/ou de la JCVM.

Nous avons intégré plusieurs de ces analyses et vérifications au sein de la plate-forme SmartTools qui devrait nous permettre à terme d'obtenir une environnement complet de déve-

loppement multi-niveaux (source, byte-code et capfile) avec correspondance entre ces différents niveaux. Des outils de décompilation, de résolution des notations pointées et de visualisation graphique d'informations statiques et dynamiques sont également en cours d'intégration.

Nous avons abordé par ailleurs l'étude de la preuve de propriétés de programmes Java par des techniques de *Model Checking*, en nous intéressant dans un premier temps à deux types de propriétés particulières :

- Les propriétés de sécurité de l'API de sécurité Java2, qui permet au programmeur de définir les droits d'accès de certains éléments de programmes à certaines ressources (voir 6.4).
- Les propriétés comportementales de programmes distribués ProActive, telles que l'absence d'inter-blocages ou de famine, plus généralement propriétés de sûreté ou de vivacité.

Dans les deux cas, il s'agit d'extraire du source de l'application (Java ou Bytecode) un modèle fini et de définir la propriété à prouver sous forme d'une formule de logique temporelle, ou d'équivalence à une spécification. On utilise ensuite des outils de vérification automatiques issus des travaux de l'équipe MEIJE à l'INRIA Sophia-Antipolis pour vérifier cette propriété et éventuellement produire des contres-exemples. Une adaptation de ces outils à nos besoins est nécessaire et a été entreprise.

Dans les deux cas, nous avons défini la structure du modèle fini visé. Les outils de construction de ces modèles, basés sur l'analyse statique des programmes sources, restent à réaliser. Le cas des propriétés de programmes ProActive a donné lieu au stage de Rabéa Boulifa et sera prolongé en thèse [27, 37]. Ces travaux sont menés en collaboration avec l'équipe Lande à l'IRISA.

6.2 Analyses statiques et transformations de programmes

Participants : Isabelle Attali, Denis Caromel, Ludovic Henrio, Didier Parigot, Bernard Serpette.

Les travaux que nous avions entamé l'année passée avec Manuel Serrano (Université de Nice Sophia Antipolis jusqu'à l'automne 2001, Inria-sophia depuis) ont abouti à une nouvelle version de Bigloo intégrant la génération de code JVM (Java Virtual Machine). La description de ce générateur, ainsi qu'un certain nombre de comparaison sont en cours de soumission. Concernant les résultats de temps d'exécution, nous avons obtenu, en prenant la plate-forme JVM la plus rapide selon le banc d'essai, un ratio proche de 2 par rapport à l'exécution en C.

Les travaux de Ludovic Henrio sur l'inférence de contexte ont été publiés et présentés à eSmart'01 [22]. Cette analyse permet de vérifier statiquement que les contraintes de partage d'objets dans une carte à puce Java sont respectées. L'algorithme a été intégré dans l'environnement SmartTools par Thomas Henrot.

Nous avons défini et implémenté en Java un vérificateur de bytecode s'appuyant sur une sémantique abstraite à petit pas. Cette analyse est décrite par des règles de transition sur les états abstraits d'une machine virtuelle. Par contruction, ce vérificateur est polyvariant : plus d'un état abstrait peut être construit pour un point de programme donné. Nous séparons chaque étape abstraite en deux sous-étapes : une étape de recoupement qui permet de fusionner plusieurs états abstraits et une étape opérationnelle qui calcule les effets d'une instruction

spécifique sur un état abstrait. Nous avons exhibé les propriétés nécessaires pour ces deux sous-étapes afin d'obtenir aisément la preuve de correction du vérificateur. Avec une simple définition de l'étape de recoupement le vérificateur accepte les sous-programmes pleinement polymorphes. Finalement, en choisissant une représentation adéquate pour les piles abstraites, notre vérificateur accepte des sous-programme comportant des récursions. Ce travail est en cours de soumission.

6.3 Composants, spécifications sémantiques et vérifications

Participants : Isabelle Attali, Christophe Held, Hamoudi Kalla, Didier Parigot, Claudio Sacerdoti-Coen.

Dans le cadre de l'outil SmartTools, l'approche pour la description des traitements sémantiques est fortement basée sur la technique des «visitor patterns». En effet cette technique a de très bonnes propriétés d'extension en terme de programmation par objet, il était donc naturel de la proposer comme outil de base. Elle a été largement automatisée par la génération de code source Java à partir de la définition des structures (Abstract Syntax Tree, AST) et nous fournissons plusieurs variantes de cette technique permettant une programmation plus lisible. De plus, une programmation par aspect a été introduite pour enrichir cette technique des «visitors patterns». Cette nouvelle fonctionnalité de programmation par aspect ne nécessite aucune transformation de programme. Ainsi, l'ajout d'aspects sur un visiteur peut être totalement dynamique (sans recompilation) [31, 30, 39]. Avec Steve Van der Hoeven, nous avons juste entamé un travail de généralisation de cette programmation par aspect, proche des travaux de Karl Leiberher, pionnier dans le domaine. Le but de ces travaux est de fournir une programmation adaptative pour Java totalement dynamique.

D'une manière plus générale, l'ensemble des travaux autour de SmartTools, en particulier l'établissement des liens entre les formalismes XML et les notions de syntaxe abstraite, la programmation par aspect, les interfaces utilisateurs fortement couplées aux technologies du Web et l'architecture par composants forment les fondations des futurs ateliers logiciels.

Dans le cadre du stage de DEA de Hamoudi Kalla [40], nous nous sommes intéressés à la sémantique opérationnelle sous deux angles : les aspects liés au modèle d'exécution afin d'obtenir un interprète d'une part, et les aspects liés aux preuves formelles sur cette sémantique d'autre part. Nous avons défini un formalisme sémantique fonctionnel qui d'une part se traduit naturellement vers Coq et d'autre part permet de générer un visiteur en Java qui exécute cette sémantique sur un arbre de syntaxe abstraite AST sur la plate-forme SmartTools. Les travaux sont illustrés par un petit langage (expressions, affectation, contrôle). Les applications possibles de ces recherches et développements sont très fortes : avec une même spécification de haut niveau (a priori plus facile à écrire que du Coq, à cause du sous-typage et des fonctions partielles) être capable de prouver un théorème en Coq (par exemple montrer qu'un programme bien typé s'évalue toujours correctement) et dériver une implémentation de référence en Java, en bénéficiant de tous les modules de SmartTools (interface graphique, mise au point, etc).

D'autre part, à partir d'un mécanisme d'exportation de théories Coq (fichiers .vo) vers XML développé par Claudio Sacerdoti-Coen, et inclu dans la distribution standard de Coq V7.0, il est possible de visualiser (et naviguer dans) ces théories via le Web, grâce au format XML.

Ces fichiers XML sont construits à partir des fichiers .vo (après vérification de type en Coq), ce qui assure de pouvoir retrouver une définition à partir d'une utilisation (liens hypertextes).

Nous proposons une approche complémentaire à celle de la librairie Helm `http://www.cs.unibo.it/helm`, développée à l'Université de Bologne, par l'intermédiaire du nouveau format XML provenant du W3C, SVG (Scalable Vector Graphics), qui est en passe de devenir un standard pour ce qui est affichage graphique et structuré de formules mathématiques.

Notre approche est fondée sur un ensemble de transformations : à partir des fichiers XML exportés depuis Coq, une première étape passe par MathML 2.0 (issu du W3C), puis dans une seconde étape, un document SVG est produit (à partir de la partie présentation de Math-ML).

En conclusion, nous avons produit des outils qui permettent, à partir de modèles et preuves spécifiés en Coq, de visualiser ces modèles et de les manipuler, grâce à des liens hypertextes intra et inter modules.

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

Participants : Isabelle Attali, Françoise Baude, Rabea Boulifa, Denis Caromel, Eric Madelaine, Julien Vayssière.

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 (voir 6.2).

Par ailleurs, dans [24, 23], nous avons étudié comment la réflexion et les MOP (Meta-Object Protocol) pouvaient se combiner dans le cadre des architectures logicielles à base de composants. 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 (requête de localisation d'un agent mobile, auprès d'un serveur de noms) et l'autre distribuée (utilisation d'une chaine de répéteurs afin de parvenir jusqu'à l'agent mobile). 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 réaliste 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 et des expérimentations sur le système réel dans un contexte de LAN et de MAN ont permis de conclure que la modélisation reflète correctement le système réel [29]. Il reste cependant à aller un peu

plus loin, par exemple, à complexifier la modélisation dans le cas d'un nombre variable d'agents mobiles dans le système.

Nous allons maintenant nous attacher à appliquer ce modèle afin de permettre le choix dynamique de la meilleure politique de localisation, pour chaque application.

Par ailleurs, dans le cadre de la thèse de Ludovic Henrio (débutée en Octobre 2000), nous avons commencé 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 : Damien Ciabrini, Bernard Serpette, Gilles Zucchini.

Damien Ciabrini et Gilles Zucchini ont écrit le lecteur de capfiles, entité représentant un package dans la Java Card Virtual Machine (JCVM), en utilisant les structures que nous avions définies pour la JVM. Ce travail nous permet de décrire et implémenter des analyses de bas niveau (interprétation et vérification de bytecode) independamment de la plate-forme utilisée (JVM ou JCVM).

Durant le stage de Damien Ciabrini et Gilles Zucchini, nous avons commencé une étude sur la décompilation de la JVM. Pour ce faire, nous avons défini un petit langage applicatif, sous-ensemble de Scheme, et implémenté la traduction d'une méthode de la JVM vers ce langage. Ce traducteur possède deux propriétés très intéressantes :

1. toute méthode JVM validée par un vérificateur de bytecode standard peut être décompilée vers le langage applicatif.
2. le code décompilé ne contient pas d'affectation sur les variables locales.

Cette deuxième propriété est très importante car elle permet de réinjecter le code vers un langage purement fonctionnel et, d'autre part, elle permet de manière très élégante d'aborder le problème de la SSA (Static Single-Assignment).

6.7 Bibliothèques pour la répartition

Participants : Laurent Baduel, Françoise Baude, Alexandre Bergel, Denis Caromel, Michael D'Agostino, Erick Fredj, Fabrice Huet, Olivier Nano, Lionel Mestre, Emmanuel Reuter, Vincent Ribaillier, 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 communication de groupe qui étend le principe de l'appel de méthode asynchrone et distante entre deux objets actifs, à l'invocation asynchrone et distante de méthode sur un groupe d'objets actifs. Ce travail de DEA [32] se poursuit en thèse.

Nous avons également 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 [36]. Il devient dès lors possible de construire des API d'agents mobiles grâce à la construction d'itinéraires d'un plus haut niveau d'abstraction. Nous avons démontré qu'une classe d'applications pouvant parfaitement être supportée par le modèle offert par ProActive, et en particulier, la construction d'itinéraires de haut niveau, consiste en de l'administration de systèmes en réseau à l'aide d'agents mobiles [41].

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. 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 [21]. Des techniques du même genre seront utilisables dans l'étude plus approfondie que nous avons entamée sur la communication asynchrone par streaming (flux).

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) [20, 33, 34]. Un tel environnement est particulièrement utile pour le déploiement d'applications de type metacomputing (par exemple, traitement d'images médicales en parallèle, objet d'une collaboration avec Erick Fredj), ou d'applications à base de composants logiciels, telles par exemple celles développées au dessus de la plateforme EJB, plateforme de test commune des participants du projet RNTL ARCAD. Les caractéristiques principales d'IC2D 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).

L'utilisation d'un MOP pour combiner différents services non-fonctionnels, et relativement

orthogonaux, est une des solutions étudiées dans le cadre du projet RNTL ARCAD. L'objectif supplémentaire des travaux de recherche menés au sein d'ARCAD, est de réaliser l'adaptation, voire l'adjonction, de services non-fonctionnels, mais de manière dynamique, c'est-à-dire, sans arrêter l'application. La possibilité offerte par IC2D d'ordonner la migration d'objets actifs en cours d'exécution va donc clairement dans ce sens.

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 avons proposé une solution à base d'*adapteurs* [25] pour effectuer la conversion entre deux types non compatibles, mais qui décrivent cependant des entités offrant les mêmes sortes de services.

6.8 Sécurisation des applications à objets distribuées

Participants : Denis Caromel, Arnaud Contes, Abishek Chaturvedi, Fabrice Huet, Anand Meka, Bernard Serpette, Julien Vayssière.

Dans le cadre d'un financement CNRS (montant 250 KF pour deux ans, voir 8.2.2), 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.

L'étude de la sécurisation d'applications réparties écrite avec ProActive s'est concrétisée par la définition d'un mécanisme de déclaration de politique de sécurité [38], qui se traduit ensuite par sa mise en œuvre au sein de la bibliothèque ProActive : possiblité de définir des domaines virtuels, dans lesquels les objets actifs s'exécutent, avec description de contraintes d'authentification, intégrité, confidentialité de toute communication entre objets actifs inter-domaines. Ceci est à rapprocher des techniques de mise en œuvre de Virtual Private Networks. Ce travail se poursuit en thèse.

7 Contrats industriels (nationaux, européens et internationaux)

7.1 Environnements de développement et vérification pour Java et Java Card

Participants : Isabelle Attali [correspondante], Lionel Blavy, Denis Caromel, Damien Ciabrini, Carine Courbis, Pascal Fauconnier, Ludovic Henrio, Thomas Henrot, Didier

Parigot, Claude Pasquier, Marjorie Russo, Fabien Tricoire, Guillaume Van De Kerckhove, Gilles Zucchini.

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). Ce projet se termine le 31 décembre 2001.

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 5.2). Cette action se termine le 31 décembre 2001.

7.3 Mise à disposition de SmartTools pour Dynam-IT

Participants : Isabelle Attali, Didier Parigot.

Ce contrat, signé en Septembre 2001 pour une durée de 6 mois, permet à la jeune pousse Dynam-IT d'évaluer en interne la plate-forme SmartTools pour son activité propre de développement et commercialisation de logiciels d'accélération des livraisons de contenus web. Les clients visés sont d'une part les sites internet professionnels et d'autre part les fournisseurs d'accès. Les terminaux visés sont les PC.

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

Participants : Isabelle Attali, Christophe Held, Claudio Sacerdoti-Coen, 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 6.3. Ce projet se termine le 31 décembre 2001.

7.5 ARCAD - Architecture Répartie extensible pour Composants ADaptables

Participants : Françoise Baude, Denis Caromel, Fabrice Huet, Julien Vayssière, Baomin Xu.

Il s'agit d'un contrat RNTL démarré en 2001, d'une durée de 3 ans, et d'un montant de 1 020 kF.
Les partenaires sont les équipes Rainbow (I3S CNRS UNSA), DTL/ASR (France Télécom

R&D), SIRAC (INRIA Rhône-Alpes), et OCM (Ecole des Mines de Nantes).

Le travail effectué par l'équipe dans le cadre de ce projet est développé dans 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 partiellement par la Région PACA, permet de financer la thèse de doctorat de Julien Vayssière.

8.2 Actions nationales

8.2.1 Action de Recherche Coopérative S-Java

Participants : Isabelle Attali, Rabea Boulifa, Ludovic Henrio, Eric Madelaine, 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` se termine le 31 décembre 2001.

Les partenaires de cette ARC sont les projets INRIA Lemme (coordinateur), Coq, Lande, Meije, l'équipe Sémantique et Interprétation Abstraite (DMI) de l'ENS-Ulm et Trusted Logic.

Les activités de recherche sont décrites en 6.1 et 6.3.

8.2.2 Programmation répartie collaborative et sécurisée pour Internet

Participants : Isabelle Attali, Françoise Baude, Denis Caromel, Fabrice Huet, Julien Vayssière.

Financement CNRS (montant 250 KF pour deux ans), qui se termine le 31 décembre 2001, en partenariat avec R. Molva et Y. Roudier (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 européennes

8.3.1 Verificard

Participants : Eric Madelaine.

Eric Madelaine participe aux travaux du projet Verificard sur la modélisation et vérification de la plate-forme et des programmes Javacard. Les partenaires sont, en dehors de l'INRIA :

GemPlus, Bull, Universités de Nimegue, Munich, Hagen, Sics. (voir `http://www.verificard.com/`).

8.4 Actions internationales

8.4.1 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.

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

- Visites et présentations invitées :
  - Isabelle Attali et Denis Caromel ont été invités pour des séminaires à l'université de Pise, en Octobre 2001 ;
  - Denis Caromel a fait :
    - une présentation aux Journées PRO : "Parallélisme, Répartition et Objets", Toulouse, 15 et 16 Mars 2001, IRIT – ENSEEIHT ;
    - une journée de cours sur le thème "Objets, concurrence, répartition et mobilité" à l'École Jeunes Chercheurs en Programmation, 4 - 16 juin 2001, Cargèse, Corse, France ;
    - une présentation invitée "Software Components : from Business to Metacomputing Components", à ORAP Forum / SpeedUp, 25-26 Octobre 2001, Lyon ;
  - Eric Madelaine
    - a participé, sur invitation, au Dagsthul seminar, Dagsthul, 11-15 décembre 2000 ;
    - a été invité pour une présentation dans le projet LANDE (Irisa), le 11 Mai 2001.
- Participations écoles :
  - Ludovic Henrio a participé à l'Ecole Jeunes Chercheurs en Programmation, Cargèse, Juin 2001.
  - Fabrice Huet a participé à l'École d'été sur les Réseaux Haut Débit et Multimedia, Calcatoggio, Corse, 6-12 mai 2001.
- Participations conférences :
  - Une grande partie de l'équipe a participé à ETAPS 2001, Gènes, 2-6 avril 2001 et eSmart, Cannes, septembre 2001.
  - Françoise Baude
    - a participé à la réunion du groupe de travail du GDR ARP *Grappes*, en mai 2001.
    - a été invitée à la 3ème conférence internationale "Large-Scale Scientific Computations", Sozopol, Bulgarie, Juin 2001.

  - a participé à la conférence PaCT'2001 (Parallel Computing Technologies), Septembre 2001, Akademgorodok, Novosibirsk, Russie.
- Eric Madelaine et Rabéa Boulifa ont participé à une réunion S-Java, Paris, Août 2001.
- Eric Madelaine a participé à la journée VerifiCard/WP4, 9 avril 2001 à Sophia Antipolis, à la Conférence TACAS'01, Venise, 2-6 avril 2001, ainsi qu'au workshop Systèmes Infinis, Paris, 26-27 avril.
- Olivier Nano a été invité à présenter nos travaux autour de ProActive, au premier workshop Européen des utilisateurs de Globus, Lecce, Italie, Juin 2001.
- Didier Parigot a fait plusieurs présentations de SmartTools :
  - à l'équipe OCM (Ecole des Mines de Nantes), juillet 2001,
  - à l'équipe ObjectWEB (INRIA grenoble), décembre 2001.
- Didier Parigot et Pascal Degenne ont présenté SmartTools aux journées Composants (Besançon), novembre 2001.
- Didier Parigot et Alexandre Fau ont présenté SmartTools au workshop XML Technologies and Software Engineering, à Toronto, Canada, juillet 2001.
- des démonstrations de SmartTools ont été effectuées à CC'01 et LDTA'01, Génes, 2-6 avril 2001.
- Invitations de chercheurs :
  - Andrew Wendelborn (University of Adelaide), Juin 2001.
  - Erick Fredj (Jerusalem College of Technology), Août 2001.
  - Marjan Mernick (University of Maribor), Septembre 2001 dans le cadre de PROTEUS.

9 Diffusion de résultats

9.1 Animation de la Communauté scientifique

- Isabelle Attali
  - a co-présidé (avec Thomas Jensen, Irisa) le Comité de Programme de la conférence eSmart 2001, tenue à Cannes, les 19,20, et 21 Septembre 2001 (plus de 150 participants) (voir `http://www-sop.inria.fr/eSmart2001/`).
  - est membre du comité de direction du GDR ALP : Algorithmique, Langages et Programmation.
  - a été *rapporteur* dans le cadre des Jurys de Doctorat de :
    Rémi Forax, 5 Décembre 2001, Université de Marne la Vallée,
    Marc Eluard, 10 Décembre 2001, Université de Rennes I.
- Françoise Baude
  - est membre de la commission des spécialistes, 27^{e} section de l'UNSA.
  - a participé au comité de programme de RenPar 2001.
  - 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 début 2002.
- Denis Caromel
  - est co-président avec John Reynders (Los Alamos National Laboratory, New Mexico) du comité d'organisation de la conférence internationale "ACM Java Grande - ISCOPE

2001", Stanford University, California, Juin 2001. (voir `http://www.inria.fr/JGI2001/`).
- est Président du comité de sélection des tutoriels de la conférence IPDPS 2001 IEEE Computer Society and ACM SIGARCH, San Francisco, Avril 2001 (voir `http://www.ipdps.org/`).
- est membre du comité d'organisation du workshop "Java for Parallel and Distributed Computing", IEEE IPDPS'2001 dont les actes sont édités par IEEE CS Press (ISBN 0-7695-0990-8) (voir `http://www.ipdps.org/ipdps2002/2002_workshops.html`).
- est membre du Comité de Programme de la conférence IEEE International Symposium on High Performance Distributed Computing, HPDC-10, San Francisco, California, August 7-9, 2001.
- 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 paraître, Hermès, 2000.
- a été *rapporteur* dans le cadre des Jurys de Doctorat de :
  Ousmane SY, 16 Mars 2001, Université de Toulouse I,
  Fabien Dagnat, 28 mai 2001, ENSEEIHT,
  Pennaneac'h François, 20 Décembre 2001, Université de Rennes I.
- a été *examinateur* dans le cadre des Jurys de Doctorat de :
  Gabriel Antoniu, 21 Novembre, École Normale Supérieure de Lyon,

- Didier Parigot a organisé le 1er Workshop sur "Language Descriptions, Tools and Applications" (LDTA'01), le 7 avril Gènes, Italie, dans le cadre de ETAPS'2001 (voir `http://www-sop.inria.fr/oasis/LDTA/ldta.html`).
- Isabelle Attali et Denis Caromel ont participé au jury de thèse de doctorat de Marjorie Russo, 13 Juillet 2001, UNSA, en tant que directeurs.
  Plusieurs membres du projet ont évalué des articles soumis aux :
  - conférences suivantes :
    eSmart 2001, RenPar'01, ICS, SuperComputing'01, LDTA'01
  - et revues suivantes : Software Practice and Experience.

9.2 Enseignement

- Isabelle Attali
  - intervient au DEA Informatique UNSA, dans le cours de tronc commun "Méthodes formelles et fiabilité du logiciel" organisé par Robert de Simone (module de 36h).
  - 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 2001, Isabelle Attali a co-organisé l'Ecole à Cargèse, du 4 au 16 Juin 2001, avec son successeur, Thomas Jensen (Irisa) (voir `http://www-sop.inria.fr/EJC2001/`).
  - est responsable de la commission Formation Emploi de l'association Télécom-Valley.
- 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 du module de mise à niveau en Systèmes, du DESS Télécommunications de l'UNSA,
  - est responsable de l'enseignement de Licence d'informatique de l'UNSA "Concepts des systèmes d'exploitation et gestion de la concurrence"
  - participe à l'enseignement de Java en Licence Informatique,
  - participe au module de tronc commun du DEA RSD "Algorithmique parallèle et distribuée".
- Denis Caromel
  - est responsable du module de Maitrise d'informatique "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.
- Ludovic Henrio intervient dans les cours suivants :
  - TDs de Java Card, DESS Télécomunication de l'Université de Nice-Sophia Antipolis, 24 heures
- Marjorie Russo est intervenue dans les cours suivants :
  - TD et TP d'algorithmie et structures de données en Java, en Deug MASS (UNSA) pour les 2^{es} années, 19 heures 30 de TD, 39 heures de TP,
  - TPs d'Analyse et Compilation, Licence informatique UNSA, 20 heures,
  - Cours, TD et TP de Prolog, Licence informatique UNSA, 25 heures,
  - Encadrement de 6 groupes de Travaux d'étude en Licence informatique UNSA, 12 heures.
- Carine Courbis intervient dans les cours suivants :
  - Cours de Programmation en C, IUT Génie des Télécommunications et des Réseaux de Sophia-Antipolis, 1ère année traditionnelle, par alternance et année spéciale, 15 heures, octobre 2000 à janvier 2001
  - TDs sur machine en Architecture/Système, Université de Nice Sophia-Antipolis, Licence Informatique, 12 heures octobre 2001 à novembre 2001
  - TPs en Systèmes Informatiques, Université de Nice Sophia-Antipolis, DEUG Mathématiques/Informatique 2ème année, 39 heures octobre 2001 à janvier 2002
- Laurent Baduel est intervenu en MASS1 (UNSA) pour 15h de TDs et 15h de TPs d'Algorithmique et Programmation.
- Julien Vayssière est intervenu pour l'année universitaire 2000-2001 dans les cours suivants :
  - TDs du cours *Internet et Bureautique* du DEUG MIAS SM (1ere année) de l'Université de Nice Sophia Antipolis (UNSA), 20 heures.
  - TDs du cours de *Architecture des ordinateurs et langage d'assemblage* de la licence d'informatique de l'UNSA, 36 heures.
  - Cours et TDs de *Systèmes distribués* du DESS Telecom de l'UNSA, 30 heures

  - Cours et TDs de *Sécurité* du DESS Telecom de l'UNSA, 10 heures
- Fabrice Huet intervient dans les cours suivants :
  - TPs de Systèmes Informatiques, Université de Nice pour les 2es années de Mathématiques-Informatique, 26 heures,
  - TPs d'Algorithmique et Programmation, IUP Nice pour les 2es années MASS, 39 heures.

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

[4] 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.

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

[6] 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`.

[7] 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.

[8] E. Madelaine, « Verification Tools from the Concur project », *EATCS Bulletin*, 47, 1992.

[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] I. Attali, T. Jensen (éditeurs), *Java on Smart Cards : Programming and Security, Lecture Notes in Computer Science 2041*, Cannes, Springer Verlag, septembre 2001. revised papers of the First International Java Card Workshop.

[12] I. Attali, T. Jensen (éditeurs), *Smart Card Programming and Security, Lecture Notes in Computer Science 2140*, Cannes, Springer Verlag, septembre 2001.

[13] D. Caromel, S. Chaumette, G. Fox, P. Graham (éditeurs), *Java for Parallel and Distributed Computing*, IPDPSP'01, San Francisco, Workshop in IEEE IPDPS'2001 Proceedings, IEEE CS Press, mai 2001. ISBN 0-7695-0990-8.

Thèses et habilitations à diriger des recherches

[14] M. Russo, *Java et ses aspects concurrents : sémantique formelle, visualisation et propriétés*, thèse de doctorat, Université de Nice - Sophia Antipolis, juillet 2001.

Articles et chapitres de livre

[15] I. Attali, D. Caromel, Y.-S. Chen, J.-L. Gaudiot, A. L. Wendelborn, « Enhancing Functional and Irregular Parallelism : Stateful Functions and their Semantics », *International Journal on Parallel Programming 29*, 4, 2001.

[16] I. Attali, D. Caromel, C. Courbis, L. Henrio, H. Nilsson, « An Integrated Development Environment for Java Card », *Special issue on Smart Cards of the Journal Computer Networks 36*, 4, 2001, p. 391–405.

[17] I. Attali, D. Caromel, M. Russo, « Graphical Visualization of Java Objects, Threads, and Locks », *IEEE Distributed Systems Online 2*, 1, 2001, `http://computer.org/dsonline`.

Communications à des congrès, colloques, etc.

[18] I. Attali, C. Courbis, P. Degenne, A. Fau, J. Fillon, D. Parigot, C. Pasquier, C. S. Coen, « SmartTools : a Development Environment Generator based on XML Technologies », *in : in XML Technologies and Software Engineering, ICSE'2001*, Lecture Notes in Computer Science, Toronto, Canada, 2001.

[19] I. Attali, C. Courbis, P. Degenne, A. Fau, D. Parigot, C. Pasquier, « SmartTools : a Generator of Interactive Environment Tools », *in : In Reinhard Wilhelm, editor, proceedings of the 10th International Conference, CC 2001. Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2001, 2027*, Lecture Notes in Computer Science, Genova, Italy, avril 2001, `http://www-sop.inria.fr/oasis/SmartTools/publications/ETAPS01/smartetap01.ps`.

[20] F. Baude, A. Bergel, D. Caromel, F. Huet, O. Nano, J. Vayssière, « IC2D : Interactive Control & Debug for Distribution », *in : 3ème Int. Conference on "Large-Scale Scientific Computations", LNCS, 2179*, 2001. Invited paper.

[21] F. Baude, D. Caromel, N. Furmento, D. Sagnol, « Optimizing Metacomputing with Communication-Computation Overlap », *in : 6th International Conference PaCT 2001*, V. Malyshkin (éditeur), *2127*, LNCS, p. 190–204.

[22] D. Caromel, L. Henrio, B. Serpette, « Context Inference for Static Analysis of Java Card Object Sharing », *in : in eSmart 2001 Conference Proceedings, Lecture Notes in Computer Science 2140*, Springer Verlag, Cannes, France, 2001.

[23] D. Caromel, F. Huet, J. Vayssière, « Combining Security with Meta Programming in Java », *in : Proceedings of the third International Conference, Reflection 2001, Kyoto, Japan, September 25-28, 2001*, A. Yonezawa, S. Matsuoka (éditeurs), *LNCS, 2192*, Springer, p. 118–125, 2001.

[24] D. Caromel, J. Vayssière, « Reflections on MOPs, Components, and Java Security », *in : Proceedings of the European Conference on Object-Oriented Programming, ECOOP'2001, Budapest, Hungary, June 18-22*, J. L. Knudsen (éditeur), *LNCS, 2072*, Springer, p. 256–274, 2001.

[25] J. Vayssière, « Transparent Dissemination of Adapters in Jini », *in : Proceedings of DOA'2001*, Z. T. Gordon Blair, Douglas Schmidt (éditeur), IEEE Computer Society, p. 95–104, Rome, Italy, September 2001.

[26] J. Woo, I. Attali, D. Caromel, J.-L. Gaudiot, A. L. Wendelborn, « Alias Analysis On Type Inference For Class Hierarchy In Java », *in : Proceedings of the Twenty-Fourth Australasian Computer Science Conference*, Bond University, Australie, février 2001.

Rapports de recherche et publications internes

[27] R. Boulifa, « Modélisation de propriétés de comportements de programmes ProActive », *rapport de recherche*, 2001, Rapport Technique INRIA, à paraitre.

[28] J. G. Variamparambil, « Getting SmartTools and VisualStudio.NET to talk to each other using SOAP and web services », *rapport de recherche*, INRIA, 2001, `http://www-sop.inria.fr/oasis/SmartTools/publications/Joseph/report.ps`.

Divers

[29] S. Alouf, F. Huet, P. Nain, « Forwarders vs. Centralized Server : An Evaluation of Two Approaches for Locating Mobile Agents », 20 pages, soumis à publication, 2001.

[30] I. Attali, C. Courbis, P. Degenne, A. Fau, J. Fillon, C. Held, D. Parigot, C. Pasquier, « Aspect and XML-oriented Semantic Framework Generator SmartTools », 10 pages, soumis à publication, 2001.

[31] I. Attali, C. Courbis, P. Degenne, A. Fau, J. Fillon, C. Held, D. Parigot, C. Pasquier, « L'apport des technologies XML et Objets pour un générateur d'environnements : SmartTools », 36 pages, soumis à publication, 2001.

[32] L. Baduel, *Communication de Groupes Efficace pour Objets Actifs Répartis*, Mémoire, Université de Nice - Sophia Antipolis, juin 2001, `ftp://ftp-sop/oasis/publications/2001/rapportDEA_LaurentBaduel.ps`.

[33] F. Baude, A. Bergel, D. Caromel, F. Huet, O. Nano, J. Vayssière, « Graphical Visualisation and Control of Distributed and Metacomputing Applications : IC2D », 1st EuroGlobus Workshop, June 2001, Invited talk, `http://www.euroglobus.unile.it/`.

[34] F. Baude, A. Bergel, D. Caromel, F. Huet, O. Nano, J. Vayssière, « IC2D : Interactive Control & Debug of Distribution », Grappes 2001, May 2001, Invited talk, `http://www.univ-ubs.fr/valoria/grappes2001`.

[35] F. Baude, D. Caromel, F. Huet, J. Vayssière, « Design and Performance Evaluation of a Reflective Mobile-Object Library », 12 pages, soumis à publication, 2001.

[36] F. Baude, D. Caromel, F. Huet, J. Vayssière, « Objets actifs mobiles et communicants », 25 pages, soumis à publication, 2001.

[37] R. Boulifa, E. Madelaine, « Proof of behaviour properties for distributed Java programs », soumis à publication, 2001.

[38] A. Contes, *Programmation à objets distribués : Sécurisation de collecticiels*, Mémoire, Université de Nice - Sophia Antipolis, juin 2001, `ftp://ftp-sop/oasis/publications/2001/contes.ps.gz`.

[39] C. Courbis, A. Fau, D. Parigot, « Programmation par visiteurs et par aspects dynamiques », 13 pages, soumis à publication, 2001.

[40] H. KALLA, *Conception d'un langage de spécification sémantique prouvable et exécutable*, Mémoire, Université d'Orléans, septembre 2001, `ftp://ftp-sop/oasis/publications/2001/FSem_kalla.pdf`.

[41] E. REUTER, F. BAUDE, « Plate-forme d'administration de systèmes et de réseaux bâtie sur la bibliothèque Java ProActive », soumis à publication, 2001.