INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Projet PAMPA

Modèles et outils pour la programmation des architectures parallèles réparties

Rennes

THÈME 1C

Table des matières

1 Composition de l'équipe

Responsable scientifique

Claude Jard [DR CNRS]

Assistante de projet

Marie-Noëlle Georgeault [TR Inria]

Personnel INRIA

Benoît Caillaud [CR]

Thierry Jéron [CR]

Vlad Rusu [CR]

Personnel CNRS

Jean-Marc Jézéquel [CR, jusqu'au 30/9/2000]

Personnel Université de Rennes 1

Jean-Marc Jézéquel [(*) professeur, depuis le 1/10/2000]

Yves Le Traon [(*) maître de conférences]

Noël Plouzeau [(*) maître de conférences, Triskell depuis le 1/9/2000, Adp précédemment]

César Viho [maître de conférences, en délégation à l'INRIA depuis le 1/10/1999]

Personnel Université de Bretagne Sud

Jacques Malenfant [(*) professeur, en délégation INRIA depuis le 1/10/2000]

Post-doctorants

Lydie Du Bousquet [post-doc Inria, jusqu'au 31/8/2000]

Dingy Fang [(*) post-doc Inria, jusqu'au 23/11/2000]

Laurie Ricker [post-doc Ercim, jusqu'au 30/4/2000]

Gerson Sunye [(*) post-doc Inria]

(*) Depuis le 1/9/2000, les membres de PAMPA, notés avec un astérisque (*) ont créé une action de recherche, appelée TRISKELL, dans le but de proposer un nouveau projet INRIA l'année prochaine.

Ingénieurs-experts

Duncan Clarke [ingénieur-expert Inria/Dyade, depuis le 1/9/2000]

Pierre Morel [ingénieur-expert Inria, jusqu'au 11/3/2000]

Solofo Ramangalahy [ingénieur-expert Inria/Dyade]

Séverine Simon [(*) ingénieur-expert Inria]

Ingénieur associé

Katia Vinceller [(*) poste d'accueil Inria, depuis le 1/10/2000]

Chercheurs doctorants

Sébastien Barbin [bourse MENRT, depuis le 1/11/2000]

Benoît Baudry [(*) bourse MENRT, depuis le 1/10/2000]

Hubert Canon [bourse BDI/Dret-CNRS, jusqu'au 30/9/2000]

Eric Cariou [(*) bourse MENRT]

Loïc Hélouët [bourse BDI/Inria-CNRS, jusqu'au 30/9/2000]

Wai Ming Ho [(*) bourse Inria]

Alain Le Guennec [(*) bourse MENRT]

Karine Macedo [(*) bourse Inria, depuis le 1/11/2000]

François Pennaneac'h [(*) bourse BDI/Inria-CNRS]

Lénaïck Tanguy [bourse Université Rennes I]

Hanh Vu Le [(*) bourse Inria]

Elena Zinovieva [bourse Inria/Dyade, depuis le 1/11/2000]

Collaborateurs extérieurs

Daniel Deveaux [(*) maître de conférences, université Bretagne Sud]

Chercheur invité

Simon Pickin [Université de Madrid, depuis le 1/11/2000]

2 Présentation et objectifs généraux

Le développement des réseaux d'ordinateurs permettant l'interconnexion de machines se poursuit. Aussi les questions posées par la construction du logiciel pour ces systèmes sont d'une grande actualité. Même si des progrès spectaculaires ont été accomplis dans les méthodes de génie logiciel, le caractère intrinsèquement parallèle et réparti des logiciels mis en œuvre continue à poser des problèmes ardus de programmation. La question la plus sensible à nos yeux est celle de la maîtrise de la fiabilité du logiciel, c'est-à-dire le contrôle des conditions de son bon fonctionnement. La maîtrise du développement passe par le renforcement des activités de conception, validation et test.

Du point de vue de la conception, la priorité est donnée aux environnements de conception objet et à l'invention de «frameworks» spécialisés pour les systèmes communicants et intégrant des outils de validation. Du point de vue de la validation, l'idée est de renforcer l'impact des méthodes formelles et des outils d'analyse pour permettre la mise au point des spécifications et la génération de tests pour les codes répartis.

Le projet Pampa contribue à l'élaboration de nouvelles technologies logicielles par l'étude de modèles formels des protocoles et l'invention d'outils informatiques associés. Nous privilégions la conception d'outils automatiques permettant d'aider aux tâches de conception, vérification, génération de code et test de programmes réels. Ces outils ont vocation à être diffusés dans le milieu académique et/ou industriel. La mise au point des modèles et outils s'effectue dans le cadre d'applications réparties situées principalement dans le domaine du logiciel pour les télécommunications.

Le fonds scientifique du projet est constitué des méthodes formelles en logiciel de télécommunication, des techniques du type «model-checking» par des parcours de systèmes de transitions, des méthodes de conception objet et des techniques de distribution de code.

L'activité scientifique du projet peut être structurée en deux thèmes de recherche :

- test des programmes répartis,
- conception objet et validation.

2.1 Le test des programmes répartis :

Nous nous concentrons sur les techniques par modèles (dites «model-checking»), et particulièrement sur les aspects algorithmiques (algorithmique à la volée). Nous avons étendu ces techniques pour être capables de générer automatiquement des séquences de test de conformité à partir de spécifications dans des langages comme SDL ou Lotos. Cela a conduit à un outil original par son algorithmique et son architecture (appelé TGV), que nous valorisons auprès de la société Telelogic. Un autre aspect du test est l'évaluation des comportements d'un code réparti à partir des traces qu'il produit. Pour modéliser les phénomènes de causalité et de concurrence, la théorie de l'ordre est notre outil mathématique de base. Nous avons plusieurs applications comme le test d'interopérabilité et le diagnostic de pannes dans les réseaux. Enfin, pour qu'elle ait un véritable impact, la validation (vérification/test) doit s'intégrer dans des environnements et méthodes de conception existantes. Pour cela, le paradigme objet et la notation UML sont maintenant incontournables.

2.2 Conception objet et validation :

L'objectif général est la construction fiable et efficace d'applications réparties par assemblage de composants logiciels. Le cadre objet et la notation UML largement diffusée servent de support. L'approche est de fonder les différentes vues d'UML sur des modèles sémantiques formels, de proposer des méthodes d'assemblage et de validation s'appuyant sur l'expression en UML des propriétés des composants, et de prototyper des outils afférents. Les activités du projet concernent la manipulation formelle de modèles UML (l'outil UMLaut), l'étude d'un modèle sémantique pivot mixte synchrone-asynchrone (BDL), la synthèse d'automates à partir de scénarios et l'intégration de techniques de génération de tests (utilisation de TGV, algorithmes pour le test d'intégration).

3 Fondements scientifiques

3.1 Panorama

Résumé : *le projet élabore de nouvelles technologies logicielles permettant d'aider le développement des logiciels répartis. Les problèmes centraux sont la modélisation des processus et comportements, et le développement d'algorithmes associés pour raffiner la conception, générer du code ou des tests. Ces questions sont examinées dans le cadre des architectures logicielles (à objets répartis). Les techniques de validation utilisées s'appuient sur des simulations complexes des modèles considérés.*

Glossaire :*Logiciel réparti* désigne un programme informatique dont l'exécution met en jeu un ensemble de calculateurs travaillant en réseau. Chaque calculateur évolue à sa vitesse propre. Nous considérons généralement que l'interaction entre ces calculateurs est asynchrone et s'effectue par échange de messages. *Asynchrone* signifie qu'un message peut rester en transit un temps non déterminé, découplant ainsi fortement l'activité des processus s'exécutant sur ces calculateurs. En général, ce type de logiciel est aussi réactif dans le sens où chacun des processus doit réagir aux sollicitations de son environnement et émettre des réponses à ces sollicitations.

3.2 Fondements mathématiques des systèmes réactifs et répartis

Mots clés : système de transitions étiqueté, ensembles partiellement ordonnés.

Résumé : *La structure mathématique qui caractérise le mieux les fondements des travaux de recherche en vérification et génération de tests de programmes répartis sont les systèmes de transitions étiquetés (*labelled transition systems *en anglais, abréviation* LTS*)* [Arn92]*. Cette structure, développée il y a près de cinquante ans est l'un des fondements de l'informatique ; aussi il nous a paru utile de préciser*

[Arn92] A. Arnold, *Systèmes de transitions finis et sémantiques de processus communicants, Études et recherches en informatique*, Masson, 1992, 196 p.

de quelle façon nous utilisons cette structure, notamment sa construction au vol. L'autre aspect fondamental est la notion de causalité entre événements dans les exécutions réparties. C'est le concept central qui permet de parler de l'analyse des comportements des systèmes distribués [Jar94]*. Il est aussi à la base des plus beaux résultats en algorithmique répartie.*

Systèmes de transitions Un LTS est un graphe orienté dont les arêtes, appelées transitions, sont étiquetées par une lettre prise dans un alphabet d'événements. Les sommets de ce graphe sont appelés états.

$M = (Q^M, A, T^M \subset Q^M \times A \times Q^M, q^M_{init})$

Avec : Q^M ensemble des états, q^M_{init} l'état initial, A l'ensemble des événements, T^M la relation de transition.

Il est usuel de parler d'automate d'états finis pour désigner un système de transitions étiqueté dont l'ensemble des états et celui des événements sont finis. Il s'agit en fait du modèle de machine le plus simple que l'on puisse imaginer. Nous employons les LTS pour modéliser des systèmes réactifs le plus souvent répartis. Dans ce cadre les événements représentent les interactions (entrées ou sorties) du système avec son environnement. On parle alors de système de transitions *entrées-sorties* ou de IOLTS (*input-output* LTS).

Ces systèmes de transitions sont obtenus à partir de spécifications de systèmes réactifs répartis décrits dans des langages de haut niveau comme SDL ou Lotos, voire UML. L'association d'un LTS à un programme se fait par l'intermédiaire d'une définition opérationnelle de la sémantique du langage et est en général formalisée sous la forme d'un système de déductions. Pour un langage aussi simple qu'une algèbre de processus (CCS par exemple), la définition de sa sémantique opérationnelle tient en moins de dix axiomes et règles d'inférences ; alors que pour un langage aussi complexe que SDL, cela est plutôt l'affaire d'un document de plus de cent pages.

Pour des raisons de performance, ces sémantiques opérationnelles ne sont jamais mises en œuvre directement ; mais font l'objet de transformations diverses. En particulier, la compacité du codage des états est un facteur déterminant de l'efficacité de la génération des LTS.

Les calculs et transformations opérés sur les LTS se résument à des parcours et calculs de points fixes sur les graphes. L'originalité réside dans la façon de les effectuer : par calcul explicite du LTS ou bien implicitement, sans calcul ou stockage exhaustif du LTS.

Les algorithmes classiques de théorie des langages construisent explicitement des automates d'états finis. Ils sont le plus souvent intégralement stockés en mémoire. Cependant, pour les problèmes qui nous intéressent, la construction (ou la mémorisation) exhaustive des LTS n'est pas toujours nécessaire. Une construction partielle suffit et des stratégies analogues aux évaluations paresseuses des programmes fonctionnels peuvent être employées : seule la partie nécessaire à l'algorithme est calculée.

Dans le même esprit il est possible d'oublier certaines parties précédemment calculées du LTS ; et par recyclage judicieux de la mémoire, il est possible d'économiser l'espace mémoire utilisé par nos algorithmes.

[Jar94] C. JARD, *Vérification dynamique des protocoles*, Habilitation à diriger les recherches de l'université de Rennes 1, décembre 1994.

La combinaison de ces stratégies de calcul sur des LTS implicites permet de traiter des systèmes de taille réelle même en utilisant des moyens de calcul tout à fait ordinaires.

Approches Symboliques Une autre approche pour combattre l'explosion combinatoire est d'utiliser des techniques symboliques. Plutôt que d'énumérer un à un les états du système, on va caractériser des ensembles (finis ou infinis) d'états par des formules logiques, et l'exploration de l'espace d'états se réduit à la manipulation de telles formules. L'avantage de cette approche est qu'elle permet de traiter des espaces d'états très grands, voire infinis. En revanche, l'assistance d'un prouveur intéractif est généralement nécessaire. Nous employons ce type de techniques pour la génération de tests symboliques sur un modèle (appelé IOSTS) qui intègre les aspects symboliques et entrées-sorties.

Ensembles Partiellement Ordonnés On considère qu'une exécution répartie sur un réseau de processus I est faite d'événements atomiques E, certains étant observables, d'autre ne l'étant pas. Chaque événement est l'occurrence d'une action ou opération (Notons Σ l'alphabet des actions) ; on considère habituellement qu'une action a lieu sur un seul et même processus du réseau. Nous avons donc :

$$\begin{cases} I & \text{ensemble fini de processus} \\ \Sigma & \text{ensemble d'actions} \\ \pi : \Sigma \to I & \text{placement des actions} \\ E & \text{ensemble d'événements} \\ \phi : E \to \Sigma & \text{étiquetage des événements} \end{cases}$$

La relation de causalité décrit le plus petit ordonnancement partiel sur les événements que l'on peut déduire du modèle de fonctionnement du réseau de processus que l'on s'est donné. Il a été présenté sous cette forme pour la première fois dans [Lam78] avec comme hypothèses sur le fonctionnement de l'architecture répartie :

1. Les processus sont séquentiels. Deux événements ayant eu lieu sur le même processus sont ordonnés.

2. Les communications sont asynchrones et points à points. L'émission d'un message précède causalement sa réception.

Ces deux axiomes nous permettent de définir une relation d'ordre $\leq$ sur E : c'est la relation de causalité.

Tout ce que l'on peut dire est que tout ordonnancement aurait respecté l'ordonnancement causal ; autrement dit, le comportement réel du système est une *extension linéaire* de l'ordre de causalité.

Il n'est cependant ni réaliste ni même utile de chercher à savoir quelle extension linéaire s'est réellement produite. Cela n'est pas réaliste car les architectures réparties existantes n'offrent pas les moyens de synchroniser des horloges locales à chaque processus avec une précision

[Lam78] L. LAMPORT, « Time, clocks and the ordering of events in a distributed system », *Communications of the ACM 21*, 7, July 1978, p. 558–565.

suffisante. Cela n'est pas utile car cet ordonnancement dépend de conditions d'exécution qui ne sont pas contrôlables ou répétables. Il faut donc considérer que toute extension linéaire de la relation de causalité est un ordonnancement plausible.

La difficulté est dans la combinatoire en général exponentielle dans le nombre d'événements des extensions linéaires d'une relation d'ordre. Il existe cependant une structure intéressante pour représenter l'ensemble les états dans lequel le système a pu se trouver : c'est le treillis des antichaînes de la relation d'ordre [DP90]. En terme d'exécution répartie ceci correspond à désigner pour chaque processus quel a été le dernier événement qui s'est produit ; cela définit sans ambiguïté un état possible du système. Ce treillis (distributif) peut être représenté sous la forme d'un LTS, avec comme relation de transition, la relation de couverture. Il s'agit du treillis des états globaux.

3.3 Génération automatique de tests

Mots clés : test de conformité, spécification, implantation sous test (IUT), cas de test, objectif de test, point de contrôle et d'observation (PCO), système de transitions à entrées sorties (IOLTS).

Résumé : *Le test de conformité est un test de type boîte noire. On se donne une spécification d'un système ouvert qui sert de modèle de référence et une implantation réelle de ce système dont on ne connaît le comportement que par ses interactions avec l'environnement. Un test consiste à alterner le contrôle de l'*IUT *et son observation par un testeur et en déduire un verdict sur la conformité de l'*IUT *par rapport à sa spécification en fonction d'une relation de conformité. La génération automatique de test consiste à produire automatiquement des cas de tests en fonction de la spécification et de la relation de conformité choisie. La sélection d'un ensemble significatif de cas de tests peut se faire en utilisant des objectifs de test comme cela se fait lors de l'écriture manuelle des tests. Nous abordons la génération de tests par divers moyens dont les fondements sont l'algorithmique des graphes mais aussi les manipulations symboliques et la preuve.*

Modèles

Le modèle de base permettant la modélisation des objets relatifs au test de conformité est le modèle IOLTS. Il distingue deux types d'actions, les actions internes inobservables, les actions observables elles-mêmes séparées en entrées et sorties, permettant ainsi de modéliser l'observation et le contrôle.

Un IOLTS est un système de transitions où l'alphabet est décomposé en $A = \{?\} \times A_I \cup \{!\} \times A_O \cup I$, A_I l'alphabet d'entrées, A_O l'alphabet de sorties, et I l'alphabet des actions internes. Dans certains cas, nous enrichissons ce modèle d'états accepteurs pour leur faire jouer le rôle d'automate.

[DP90] B. Davey, H. Priestley, *Introduction to Lattices and Order*, Cambridge University Press, 1990.

Nous utilisons aussi des modèles de plus haut niveau manipulant des variables. Les transitions sont alors étiquetées par des gardes et des actions composées d'événements (entrée, sorties, affectations).

Ces modèles sont utilisés pour modéliser les comportements des objets intervenant dans l'activité de test et sa génération. En particulier la spécification, l'implantation, les objectifs de test et les tests eux-mêmes utilisent des instances particulières de ces modèles. La génération partant de la spécification et d'un objectif de test implique des transformations de modèles tels que l'abstraction d'actions internes, la déterminisation, la minimisation qui créent des instances de sous-classes du modèle général d'IOLTS.

Une des relations qui lie les modèles d'IOLTS est la relation de conformité. La relation de conformité **ioco** choisie, définie par Jan Tretmans [Tre96] (Université de Twente) stipule qu'une IUT I est conforme à la spécification S si après toute trace (séquence d'actions observables y compris les blocages) de la spécification, les sorties possibles de l'IUT (y compris les blocages) sont incluses dans celles de la spécification.

Algorithmique à la volée

La génération de test implantée dans l'outil TGV utilise une algorithmique de graphe, en particulier des algorithmes de parcours. Ces parcours font aussi de la construction: le graphe parcouru n'est pas connu a priori mais construit pendant le parcours de manière paresseuse. Ceci évite la construction de certaines parties du graphe: c'est ce qu'on appelle la génération à la volée. De plus ces algorithmes font aussi de la synthèse de sous-graphes.

TGV utilise plusieurs parcours en profondeur à plusieurs niveaux dans l'outil et en particulier des adaptations d'un algorithme de Tarjan [Tar72]. Cet algorithme permet de calculer les composantes fortement connexes maximales (CFCs) d'un graphe dirigé. L'algorithme original est récursif mais pour des raisons d'efficacité, nous utilisons une version itérative. Sa complexité est linéaire en temps et en mémoire.

Par définition, les CFCs sont des classes d'équivalence pour la relation d'accessibilité et de co-accessibilité: deux sommets u et v sont dans la même CFC si v est accessible depuis u et u accessible depuis v. On peut aussi remarquer que les CFCs peuvent se caractériser par une autre relation: depuis tous les sommets d'une même composante on peut atteindre exactement les mêmes ensembles de sommets. Cette caractérisation permet de se rendre compte de l'utilité du calcul de CFCs pour la vérification de propriétés d'accessibilité.

C'est cette idée qui est utilisée pour adapter l'algorithme de Tarjan à la génération de tests car justement plusieurs problèmes se réduisent à des problèmes d'accessibilité. La première adaptation est utilisée pour calculer une τ-réduction (abstraction des actions internes). Le problème se traduit en l'accessibilité aux transitions visibles. De plus les boucles d'actions internes τ sont caractéristiques des divergences pour les LTS finis et ces divergences sont considérées comme observables dans la relation de conformité donc par un test. La seconde

[Tre96] J. TRETMANS, « Test Generation with Inputs, Outputs and Repetitive Quiescence », *Software - Concepts and Tools 17*, 1996.

[Tar72] R. TARJAN, « Depth-first search and linear graph algorithms », *SIAM Journal of Computing 1*, 2, juin 1972, p. 146–160.

adaptation est utilisée dans l'algorithme principal de TGV. Ici, le problème est celui de l'accessibilité à un ensemble d'états puisque, pour schématiser, on veut calculer un sous-graphe du produit entre spécification et objectif de test contenant l'ensemble des états (ou seulement une partie) depuis lesquels les états accepteurs de l'objectif de test sont accessibles. Enfin, une troisième adaptation est utilisée pour extraire du sous-graphe précédant une partie dite *contrôlable* c'est à dire n'ayant jamais de choix entre une action contrôlable et d'autres actions (contrôlables ou non). De façon moins immédiate, ceci peut se voir aussi comme un problème d'accessibilité à l'état initial sur le graphe où on a inversé la relation de transition et modifié celle-ci au vol par coupure des conflits de contrôlabilité.

L'algorithme de Tarjan a aussi été adapté par F. Bourdoncle dans le contexte de l'analyse statique. L'idée de son algorithme est d'appliquer récursivement la décomposition en CFCs en enlevant à chaque récursion les racines de CFCs (la racine d'une CFC est le premier sommet atteint dans le parcours). L'algorithme se termine quand il n'existe plus de cycle. Ceci permet alors de décomposer complètement un graphe dirigé quelconque en un ordre topologique faible. La complexité est alors quadratique en temps et en mémoire. Nous avons adapté cet algorithme pour calculer un ordre pour le test d'intégration de composants objet. L'adaptation consiste à modifier le critère de choix du sommet à retirer à chaque itération sans modifier la complexité de l'algorithme: le critère de choix doit donc être calculé en temps constant.

Un des problèmes fondamentaux de la génération de tests est celui de l'explosion combinatoire du graphe d'états de la spécification. Le même problème apparaît dans le cadre de la vérification par modèle et une des techniques utilisées pour l'éviter est le calcul à la volée. Le principe est de vérifier les propriétés pendant la construction par un parcours en profondeur du graphe d'états de la spécification. Quand une violation de la propriété est détectée, la séquence courante donne un contre-exemple. En génération de test, le problème est à la fois plus compliqué car il faut abstraire les actions internes donc considérer plusieurs niveaux de graphes d'états simultanément, et plus simple car les propriétés (ici les objectifs de test) sont moins expressifs. Nous avons donc adapté l'idée de la vérification à la volée à la génération de test. Ceci nécessite un découpage de l'architecture logicielle par des couches utilisant et fournissant des Api composées de fonctions nécessaires au parcours de graphes. La génération de tests à la volée peut alors se voir comme une construction paresseuse du graphe d'état de la spécification et des graphes d'états intermédiaires guidée par l'objectif de test. Même si cela ne change pas la complexité théorique, l'efficacité est nettement meilleure en pratique.

3.4 Technologie objet dans un contexte de génie logiciel réparti

Mots clés : objets, composant logiciel, motifs de conception, framework.

L'approche objet pour le génie logiciel

L'approche objet est aujourd'hui devenue incontournable pour l'analyse, la conception et la réalisation des grands systèmes d'information devant évoluer sur de longues périodes de temps, et pour lesquels l'effort principal en termes de logiciel (parfois jusqu'à 80% ou plus) est consa-

cré à la maintenance [Mey88]. Fondée sur la notion d'objets, c'est-à-dire d'instances de classes modélisant des types d'entités stables d'un système d'information en tant que modules autonomes organisés selon des relations d'héritage (classification) et d'utilisation, cette approche permet en effet de gérer la nature fondamentalement incrémentale, itérative et évolutive du développement de tels logiciels [Jac85,Boo94]. Les diverses phases d'analyse, de conception et de réalisation utilisent le même cadre conceptuel (fondé sur cette notion d'objet) et n'ont pas de frontières rigides entre elles, ce qui fait que le processus de développement objet est parfois qualifié de *continu* [Jéz96]. La notion de continuité est ici empruntée aux mathématiques : il s'agit de la propriété attribuée à la transformation menant du domaine du problème vers l'espace des solutions : informellement une "petite" modification dans la définition des besoins produit une "petite" altération du logiciel.

La notion d'objet fournit le substrat nécessaire au développement du concept de *composant logiciel*, vu ici comme unité de déploiement. Elle offre en effet des possibilités d'encapsulation et de masquage d'information qui permettent d'établir une analogie avec les composants matériels, avec en plus la notion d'adaptabilité qui permet de les adapter souplement. En effet, au delà de l'idée traditionnelle de la réalisation d'économies d'échelle dans le développement de logiciels en réutilisant des composants plutôt qu'en redéveloppant les applications à partir de zéro, il s'agit aussi aujourd'hui d'offrir la possibilité de modifier radicalement le comportement et les fonctionalités d'une application par substitution ou ajout de composants, même longtemps après son déploiement. La question n'est plus de développer une application répondant à un besoin précis à un moment donné, mais plutôt de construire un canevas d'application, ou "framework", qui va permettre de décliner une gamme d'applications répondant à une famille de besoins évoluant dans le temps.

Un *framework* fournit un ensemble intégré de fonctionnalités spécifiques à un domaine, implanté par une collection de classes liées entre elles par de multiples motifs (*patterns*) de collaboration statiques et dynamiques. Il fournit un modèle d'interaction entre les différents objets instances des classes définies (ou seulement spécifiées pour les classes abstraites) dans le framework. Celui-ci présente en général une inversion du contrôle à l'exécution : alors qu'une application utilisant une bibliothèque s'appuie sur celle-ci, dans le cas d'une application utilisant un framework, c'est le framework qui effectue l'essentiel du travail et appelle "de temps en temps" un composant spécifique réalisé par l'implanteur de l'application. Un framework peut donc être vu comme une application semi-complète. Des applications complètes sont développées en héritant et en instantiant des composants paramétrés de frameworks. Il suffit donc en quelque sorte d'enficher dans un framework les composants spécifiques de son application pour obtenir une application complète.

Dans un contexte de programmation par objets, on s'appuie sur le mécanisme de la *liaison dynamique* pour dissocier la spécification d'une opération (donnée dans une classe du framework) de son implantation dans une sous-classe, qui fait partie du code applicatif fourni par

[Mey88] B. Meyer, *Object-Oriented Software Construction*, Prentice-Hall, 1988.

[Jac85] M. Jackson, *System Development*, Prentice-Hall International, Series in Computer Science, 1985.

[Boo94] G. Booch, *Object-Oriented Analysis and Design with Applications*, édition 2nd, Benjamin Cummings, 1994.

[Jéz96] J.-M. Jézéquel, *Object Oriented Software Engineering with Eiffel*, Addison-Wesley, mars 1996, ISBN 1-201-63381-7.

l'utilisateur du framework.

L'utilisation de frameworks et de composants a un impact majeur sur le cycle de vie du logiciel, qui doit maintenant intégrer des activités de :

- conception d'infrastructures d'accueil de composants ("frameworks")
- conception de composants utilisables comme unités de déploiement
- validation et assemblages de composants (d'origines diverses)
- gestion des composants (maintenance)

Or il est clair que les approches empiriques, sans réel modèle de composition de composant, qui ont présidé à l'émergence d'une véritable industrie du composant (au moins dans le monde Windows) posent des problèmes insurmontables d'intégration et de validation, et ne peuvent donc facilement être transposées à des systèmes plus critiques, comme en témoigne par exemple le crash du vol Ariane 501 [JM97].

Parmi les défis à relever, le besoin de modèles d'assemblages de composants fondés formellement, ainsi que celui d'une qualité vérifiable se signalent tout particulièrement.

Si certains éléments de réponses à ces problèmes existent déjà dans des prototypes de laboratoire, l'adoption massive d'UML (Unified Modeling Language) dans de nombreux secteurs du monde industriel ouvre des perspectives nouvelles pour faire évoluer, passer à l'échelle et ainsi rendre économiquement rentables leurs idées sous-jacentes. En effet, au contraire de ses prédécesseurs (OMTO, Booch, etc.) qui ne définissaient qu'une syntaxe graphique, UML est partiellement formalisé au travers d'un méta-modèle (lui-même exprimé dans un sous-ensemble de UML) et contient un langage sophistiqué de description de contraintes appelé OCL (*Object Constraint Language*), utilisable aussi bien au niveau du modèle que du méta-modèle. Ceci permet de manipuler formellement des modèles UML de composants, de frameworks et d'applications.

4 Domaines d'applications

Mots clés : télécommunication, génie logiciel, test, UML, SDL.

Résumé : *le secteur des télécommunications est en grande expansion, avec la mise en place d'infrastructures mondiales, l'explosion des télécommunications mobiles et le développement de nouveaux services. Le contexte industriel français et européen est par ailleurs assez favorable. Du point de vue du logiciel, la pression est grande pour augmenter la généricité des solutions proposées (aspect méthode) tout en raccourcissant les délais de mise au point (aspect outils). Le projet Pampa, spécialiste de l'ingénierie des protocoles, trouve là un terrain privilégié d'applications. Ceci à condition de ne pas manquer des évolutions majeures du domaine qui sont:*

[JM97] J.-M. JÉZÉQUEL, B. MEYER, «Design by Contract: The Lessons of Ariane», *Computer 30*, 1, janvier 1997, p. 129–130.

la nécessité de considérer globalement le cycle de développement du logiciel, notamment dans le cadre des méthodes de conception objet ; et l'intérêt croissant pour les protocoles des couches hautes exprimées sous la forme de services de communication ou de gestion (avec des données nécessitant des traitements symboliques). L'augmentation de la complexité et les exigences de fiabilité et de réutilisation justifient pleinement les méthodes développées dans le projet. Les sujets abordés sont la validation de conceptions UML*, la génération de tests de conformité pour les protocoles, la conception d'un langage d'interface pour objets communicants, et le suivi de pannes dans les réseaux.*

4.1 Logiciels pour les télécommunications

L'activité de recherche du projet, centrée sur la maîtrise de la fiabilité, est en rapport avec deux types d'applications dans le domaine des télécommunications : la conception fiable de logiciels communicants et le test et diagnostic de systèmes communicants.

Conception fiable de logiciels communicants :

L'exigence de fiabilité des logiciels est facile à comprendre dans un contexte où ils sont présents à de très nombreux exemplaires et dans différentes versions sur un grand réseau de télécommunication. L'accent est porté sur la faculté d'interopérer. Le coût d'apparition d'une faute majeure est considérable dans ces systèmes et la réparation longue et difficile. Il est à noter aussi une exigence d'évolutivité importante liée à la mise en œuvre rapide de nouveaux services. Nous encourageons l'utilisation de méthodes formelles pour faciliter la résolution de ces problèmes.

Mais il ne faut pas oublier que les méthodes formelles doivent de plus en plus s'intégrer à une "approche système" permettant aux ingénieurs de concevoir globalement les systèmes pour prendre en compte tout un ensemble de contraintes et d'objectifs liés aux besoins des utilisateurs. Ces méthodologies formalisées n'en sont qu'à leurs débuts : un bon exemple est l'approche objet qui s'étend rapidement au contexte des télécommunications.

Test et diagnostic des systèmes communicants :

Le test est une autre facette du développement fiable ; il consiste à s'assurer que le système, une fois réalisé, est conforme à ses spécifications. Quel que soit le soin apporté à la conception, cette phase reste de première importance pour vérifier le bon fonctionnement du système dans des environnements complexes et évolutifs. La surveillance et le diagnostic sont aussi des aspects du travail sur les implantations.

Le projet Pampa s'est focalisé sur la génération automatique de tests de conformité à partir de spécifications formelles, ainsi que sur le diagnostic en environnement réparti.

L'obtention d'une bonne suite de tests de conformité est d'un intérêt économique certain. C'est à l'heure actuelle un travail manuel coûteux et répétitif qui est ensuite utilisé à grande échelle. Nous participons au défi de l'automatisation en mettant particulièrement l'accent sur la qualité de la suite de tests (dans sa capacité à détecter les implantations non conformes

et uniquement celles-ci). Sur plusieurs études de cas, nous avons montré la rentabilité de l'approche et le transfert industriel de notre outil TGV en est l'exemple.

Les travaux du projet Pampa s'attaquent actuellement à résoudre deux grandes difficultés :

- l'utilisation de techniques symboliques dans le processus de synthèse de tests, afin de permettre la manipulation explicite de variables dans les spécifications et les tests engendrés.
- l'extension des techniques de génération pour produire des tests répartis contenant du parallélisme, afin d'aborder les questions de test d'interopérabilité entre composants.

5 Logiciels

5.1 TGV : un outil de génération automatique de tests de conformité

Participants : Thierry Jéron, Pierre Morel, Séverine Simon, César Viho, Claude Jard, Vlad Rusu.

Mots clés : test de conformité, TGV, ObjectGéode, CADP, SDL, Lotos, TTCN.

Résumé :

TGV *(Test Generation with Verification technology) est un prototype de générateur automatique de tests de conformité à partir de spécifications formelles (*SDL *ou Lotos). Son originalité tient dans son algorithmique à la volée adaptée du domaine de la vérification par modèle (Model Checking).* TGV *est issu d'un projet commun avec le laboratoire Verimag Grenoble. Il utilise des bibliothèques de la boîte à outils* CADP *de Vérimag et de l'Inria Rhône-Alpes.* TGV *a été déposé par l'Inria à l'Agence de Protection des Programmes en 97.* TGV *est distribué gratuitement dans la boîte à outils* CADP*. Il est déjà utilisé par les membres du projet sur plusieurs études de cas et aussi par quelques utilisateurs extérieurs. Un effort important a été fourni pour le transfert industriel des algorithmes de* TGV *dans l'outil ObjectGéode de Telelogic.*

Une première version de TGV a vu le jour en 95 lors d'un contrat financé par le STEI regroupant Vérilog, Cap Sesa Région Rennes, le Cnet Lannion et le Celar de Bruz, Vérimag et Pampa. Cette première version développée par Vérimag et Pampa était utilisable sur les graphes d'états de spécifications SDL produits par Géode.

Depuis cette première version, TGV a été constamment amélioré du point de vue algorithmique mais aussi par l'interfaçage avec de nouveaux environnements. TGV génère des tests à la volée (sans construire complètement le graphe d'état de la spécification), à la fois sur des spécifications SDL grâce à sa connexion à ObjectGéode, et sur des spécifications Lotos grâce à sa connexion à l'environnement CADP. TGV peut aussi fonctionner sur des graphes explicites décrits au format Aldébaran ou dans le format compressé BCG. En sortie, TGV construit un cas de test dans un format général (Aldébaran ou BCG) qui peut être traduit dans le langage TTCN, langage de description de tests standard de fait dans le monde des télécoms.

TGV est donc relativement indépendant des langages de spécification. Cette indépendance est obtenue par son architecture en couches par l'intermédiaire d'Api fournissant les fonctions de parcours de systèmes de transitions intermédiaires. D'autre part, ces différentes couches utilisent des bibliothèques de stockage de graphes de CADP. La première couche, la seule qui soit spécifique au langage d'entrée, permet de connecter TGV soit à l'Api du simulateur ObjectGéode pour SDL, soit à l'Api fournie par le compilateur Lotos Caesar de la boîte à outils CADP, soit encore aux Api de parcours de graphes explicites. Cette couche fournit une Api de parcours du graphe d'état de la spécification S. Une deuxième couche utilise cette Api et un objectif de test pour fournir l'Api du produit synchrone $PS = TP \times S$ par étiquetage des états de la spécification par les états correspondants de l'objectif (en particulier les états accepteurs de l'objectif). La troisième couche permet le renommage et le masquage d'actions internes fournissant une Api sur l'IOLTS PS_{abs} dans lequel toutes les actions internes sont indifférenciées par τ. La quatrième effectue une réduction des actions internes et une déterminisation et fournit les primitives de parcours du système de transition résultant PS_{vis} ainsi que ses états accepteurs. Enfin la dernière couche implante l'algorithme central qui effectue un parcours de PS_{vis} et synthétise un cas de test TC. Ce dernier algorithme peut être vu comme une extension d'un algorithme de model checking qui synthétise le sous-graphe de tous (ou seulement une partie contrôlable) les comportements menant aux états accepteurs de PS_{vis}, i.e. les témoins de la validité de la propriété exprimée par l'objectif.

En 99, TGV a bénéficié d'un lifting complet en vue de sa distribution. L'architecture de TGV a été repensée complètement pour correspondre à la découpe fonctionnelle et permettre une plus grande modularité. Les fonctionnalités ont été améliorées, en particulier pour permettre une plus grande expressivité des objectifs de tests, des fichiers de masquage et de renommage, en généralisant l'utilisation des expressions régulières. Le calcul des temporisateurs a été simplifié et incorporé au cœur de TGV. Le calcul des postambules a été affiné par la notion d'état stable. L'outil prend maintenant en compte le format de graphe BCG à la fois en entrée et en sortie. Un nouveau module appelé VTS et pouvant se substituer au module principal de TGV permet maintenant de vérifier et corriger à la volée des cas de tests. Ceci peut servir à la correction de tests manuels mais sert aussi à tester TGV par la vérification des cas de tests produits. Enfin en plus de la version Solaris, il existe maintenant une version Linux et une version NT.

Une opération de transfert industriel de TGV avec la société Telelogic et le soutien de France Telecom s'est terminée courant 99. Les algorithmes de TGV et son architecture ont été implantés dans l'outil ObjectGéode. Une bonne part des développements relatifs à TGV ont été effectués dans le projet. La version commerciale est vendue sous le nom Test Composer.

Les différentes études de cas dans lesquelles TGV est utilisé nous permettent constamment d'enrichir et d'améliorer TGV sur différents aspects : l'expression des objectifs de test, de l'architecture de test, l'algorithmique de génération de test, de vérification de tests, la connexion avec d'autres outils. L'année 2000 a vu se développer la connexion UMLAUT/TGV avec le soutien d'Alcatel et de Gemplus.

5.2 UMLAUT : un outil de manipulation de modèles UML

Participants : Jean-Marc Jézéquel, Wai Ming Ho, Alain Le Guennec, François

Pennaneac'h, Gerson Sunye, Séverine Simon, Katia Vinceller, Hanh Vu Le.

Mots clés : UML, composant, patterns, validation.

Résumé :

*De nombreux Ateliers de Génie Logiciel (*AGL*) pour* UML *permettent aux ingénieurs logiciels de tracer des diagrammes et de générer des squelettes de code à partir de ceux-ci. Mais souvent les utilisateurs avertis voudraient faire plus de chose avec leurs modèles* UML*, comme par exemple appliquer des design patterns spécifiques, générer du code pour des systèmes embarqués, simuler des aspects fonctionnels ou non fonctionnels du système, ou encore passer des outils de validation sur le modèle ; activités difficiles à mener en utilisant les facilités de script offertes par la plupart des* AGL.

UMLAUT *(*UML *Automatic Universal Transformations) est un framework de transformation de modèles (dont le coeur est développé dans le cadre de la* CTI *Cnet Meta*FOR*) permettant d'appliquer des manipulations complexes à un modèle* UML. *Ces manipulations sont exprimées comme des compositions algébriques de transformations élémentaires réifiées. Elle sont donc extensibles au travers des mécanismes classiques d'héritage et d'agrégation.* UMLAUT *est utilisé en particulier dans le cadre du projet* RNRT *Oural pour transformer le modèle* UML *d'une application répartie en un système de transitions étiquetées ; dans l'objectif de le valider avec des outils avancés de validation de protocoles (e.g.* TGV*).* UMLAUT *est aussi utilisé, dans le cadre du projet* RNRT *Convergence, pour prototyper de futures évolutions d'*UML *telles que discutées actuellement à l'*OMG.

UMLAUT *a été déposé par l'Inria à l'Agence de Protection des Programmes en 1999. Des versions de démonstration pour Linux, Solaris et Windows* NT *sont librement disponibles depuis le site web de l'équipe (http://www.irisa.fr/pampa).*

La notation UML (*Unified Modeling Language*) est issue des travaux de Booch, Rumbaugh et Jacobson, auteurs des méthodes de développement à objets parmi les plus utilisées (les méthodes Booch, OMT et OOSE). UML est le successeur commun de ces trois méthodes, dont elle reprend la plupart des concepts et notations, dans un souci d'unification. Le manque d'uniformisation, les difficultés à communiquer entre outils constituaient en effet un frein au déploiement des méthodes de développement à objets. Il était donc souhaitable de créer un langage commun convenant à la modélisation des systèmes informatiques, mais également suffisamment générique pour prendre en compte des problèmes d'autres domaines. Contrairement à ces prédécesseurs, UML impose un langage (en cours de normalisation par l'OMG), mais laisse libre le choix du processus de développement associé.

La principale particularité de la notation UML réside dans sa base formelle appelée *méta modèle* : la sémantique des concepts manipulés par la notation UML est décrite en utilisant la notation elle-même. Cela a pour conséquence l'amélioration de la correction des modèles (suppression des ambiguïtés et des incohérences), et permet d'envisager l'utilisation de techniques formelles à des fins de vérification/validation, les éléments d'un modèle devant respecter les propriétés et contraintes définies dans le méta-modèle.

L'outil UMLAUT vise à montrer la validité d'une telle approche. Cet outil intègre le méta-modèle UML et fournit un ensemble de facilités permettant de raisonner sur des modélisations UML. Il s'agit en fait d'un véritable framework de transformation de modèles permettant d'appliquer des manipulations complexes à un modèle UML. Ces manipulations sont exprimées comme des compositions algébriques de transformations élémentaires réifiées. Elle sont donc extensibles au travers des mécanismes classiques d'héritage et d'aggrégation.

Les modèles sont fournis à l'outil soit à l'aide d'une interface graphique (implantée en Java avec Swing), soit dans un format d'échange comme CDIF (Case Data Interchange Format) ou XMI (le format normalisé par l'OMG d'échange de modèles UML sur la base de XML), ou encore un format "propriétaire" comme celui de Rational Rose (en cours de développement).

L'outil accepte également en entrée des sources de programmes Java ou Eiffel, afin de reconstruire automatiquement les modèles UML correspondants (*reverse-engineering*).

UMLAUT intégrant le méta-modèle UML, il est possible de réaliser simplement des transformations de modèles définies par un ensemble de règles fournies par l'utilisateur. Dans le cadre du projet RNRT Oural, nous avons commencé à appliquer cette idée pour transformer le modèle UML d'une application répartie en un système de transitions étiquetées ; dans l'objectif de le valider avec des outils avancés de validation de protocoles.

UMLAUT est aussi utilisé, dans le cadre du projet RNRT Convergence, pour prototyper de futures évolutions d'UML en ce qui concerne un langage d'actions (ASL) telles que discutées actuellement à l'OMG.

6 Résultats nouveaux

6.1 Test de conformité et d'interopérabilité

Participants : Sébastien Barbin, Duncan Clarke, Claude Jard, Thierry Jéron, Pierre Morel, Vlad Rusu, Séverine Simon, Lénaïck Tanguy, César Viho, Elena Zinovieva.

Mots clés : test de conformité, test d'interopérabilité, test réparti, test abstrait, test exécutable, test symbolique, abstraction, déterminisation, génération à la volée, vérification, objectif de test, preuve.

Résumé : *L'activité relative au test de conformité revêt deux aspects: la génération automatique de tests de conformité qui produit des cas de tests dits "abstraits" et l'implantation de ces tests abstraits en tests exécutables. Concernant la génération de tests, nous poursuivons notre activité pour améliorer les tests produits, faciliter leur sélection, prendre en compte différentes architectures de test. En 2000, l'accent a été mis sur la génération de tests symboliques par l'utilisation de techniques symboliques. Nous avons également publié une étude de cas sur la vérification de test. Dans le cadre du contrat Van Gogh, et en collaboration avec l'université de Twente, nous avons expérimenté une chaîne complète de la génération à l'exécution des tests pour une étude de cas [22] traitant d'un protocole de conférence (ConfCase). Nous avons aussi collaboré avec le projet Armor et une PME sur l'utilisation de techniques probabilistes pour la génération de test. Sur l'exécution des tests un*

outil prototype appelé $\mathcal{D}-\mathcal{PIXIT}$ a été développé et nous sert de plate-forme de validation expérimentale des méthodes que nous développons.

Génération de tests symboliques Les algorithmes de TGV sont basés sur des techniques énumératives. Lorsque, par exemple, la spécification d'un protocole contient des paramètres et/ou variables (paramètres de messages, compteurs, files d'attente...), TGV instancie toutes ces variables, qui seront traitées en "multipliant" les états de contrôle à explorer par le nombre de valeurs distinctes que les variables peuvent prendre. Ceci peut mener à l'explosion combinatoire ou même à la non-terminaison. Pour résoudre ce problème, nous avons commencé un travail sur la génération de tests *symboliques*: les cas de test sont produits non plus par énumération des valeurs des variables, mais par des calculs symboliques à partir de la spécification et de l'objectif de test. Nous avons défini formellement une méthode de génération de tests symboliques et prouvé que les cas de test produits sont corrects (essentiellement, non-biaisés et non-laxistes). Ce travail a donné lieu à une publication en 2000 [39]. Notre travail actuel consiste à poursuivre ce travail par l'emploi de techniques d'analyse symbolique et de preuve afin de simplifier et d'instancier les cas de test ainsi produits pour les rendre exécutables.

Génération de test et modèles probabilistes Nous avons entamé une collaboration avec le projet Armor et une PME Lavalloise (Alitec) sur la génération de test mixant l'approche de génération de test fondée sur le model-checking de notre outil TGV avec des modèles Markoviens. Il s'agit pour l'instant de produire des séquences à partir d'un modèle markovien et relativement abstrait du système, et d'interpréter ces séquences comme des objectifs de test. Ces objectifs sont ensuite utilisés par TGV pour produire des tests pour une spécification SDL. Cette idée a été expérimentée par Alitec sur des spécifications SDL de l'UMTS.

Génération de test réparti Nous travaillons sur une approche qui consiste à dériver automatiquement des testeurs répartis à partir de test centralisé. Nous avons proposé un schéma de distribution permettant de produire un ensemble de tests communiquant entre eux de manière asynchrone. Cette année,

- nous avons étudié un autre schéma de distribution dans lequel les testeurs parallèles se synchronisent suivant un mécanisme de type round-robin [42].
- nous avons développé un prototype DaTEC qui implémente les deux schémas (le round-robin et celui avec le service de consensus) de distribution.

Nous avons expérimenté la distribution par round-robin sur des cas de test fournis par le CELAR [43].

Notre travail actuel porte sur d'autres méthodes pour la synthèse automatique des procédures de coordination efficaces entre les testeurs parallèles.

Concernant la phase d'exécution des tests, nous avons étendu cette année notre plate-forme expérimentale $\mathcal{PIXIT}$ pour prendre en compte des tests répartis (comme ceux obtenus à la sortie de notre prototype DaTEC de distribution automatique de test). Sur cette nouvelle version appelée $\mathcal{D}-\mathcal{PIXIT}$, nous étudions l'instrumentation qui peut être faite aussi bien du côté des testeurs que des IUTs pour affiner les observations et améliorer le contrôle.

7 Contrats industriels (nationaux, européens et internationaux)

7.1 Diagnostic de pannes dans les réseaux de télécommunications (RNRT Magda)

Participants : Claude Jard, Laurie Ricker.

Mots clés : gestion de réseau, supervision, gestion d'alarmes, diagnostic, système distribué, réseau de Petri, HMM.

Glossaire :
HMM : Hidden Markov Models (Modèles de Markov Cachés), technique consistant à inférer, à partir d'observations "bruitées", l'état interne caché d'un automate stochastique (ou chaîne de Markov). Technique très utilisée en reconnaissance de la parole, et plus récemment dans le diganostic de systèmes dynamiques discrets ou hybrides. Nous étendons cette technique aux réseaux de Petri.

Résumé : *Projet RNRT/Magda associant Cnet, Alcatel, Ilog, Irisa/Pampa, Sigma2 et Aïda, LIPN jusqu'en mi 2001.*

Cette activité est commune avec le projet Sigma2. Il s'agit de développer une approche systématique pour le diagnostic de réseaux de télécommunications, avec les objectifs suivants :

- *prendre en compte explicitement le caractère distribué des réseaux,*
- *suivre une approche "modèle", modèle sur lequel s'appuie l'algorithme de diagnostic,*
- *prendre en compte les aléas (perte d'alarme, confusions possibles,...),*
- *viser une mise en œuvre du logiciel de diagnostic qui soit répartie sur le réseau.*

Une technologie originale, fondée sur une notion nouvelle de puzzle de Viterbi sur des motifs d'ordres partiels. Cette technologie nous permet naturellement de prendre en compte les contraintes qui résultent du contexte "distribué" où nous nous situons. Pour en savoir plus, consulter le rapport de l'équipe Sigma2.

7.2 Méta-Framework pour Objets Répartis

Participants : Jean-Marc Jézéquel, Wai Ming Ho.

Mots clés : framework, design patterns, composant, système distribué.

Glossaire : Framework Un *framework* fournit un ensemble intégré de fonctionnalités spécifiques à un domaine, implanté par une collection de classes liées entre elles par de multiples (*patterns*) de collaboration statiques et dynamiques.

Résumé : *Contrat CTI-Cnet ref. 98 1B 070, mars 1998 - mars 2001*

L'objectif de cette étude est d'étudier, concevoir et valider des modèles et des méthodes pour la conception et le développement d'architectures d'objets distribués.

Le modèle Metafor intègre plusieurs patrons de conception et s'appuie sur le langage UML *pour mettre en pratique les techniques de construction et de réutilisation de Frameworks. Les prototypes sont développés sur la base d'*UMLAUT *l'outil de manipulation de modèles* UML *en cours de développement dans l'équipe. Une étude de cas représentative, permettant d'expérimenter et de valider l'approche, est fournie par le Cnet.*

Nous explorons depuis quelques années le domaine de la construction de frameworks d'objets répartis, avec des domaines d'application variés comme le calcul scientifique intensif, les télécommunications, ou le travail coopératif. L'objectif est d'étudier, concevoir et valider des modèles et des méthodes de construction de logiciels pour architectures distribuées par composition de composants logiciels dans un contexte de programmation par objets. Ceci nous a conduit à distinguer dans les frameworks applicatifs (ou métier) certaines classes de problèmes qui reviennent de manière répétitive lorsqu'on veut les utiliser dans un contexte distribué. Il s'agit en particulier de :

- Modèles de conception s'abstrayant des particularités de l'architecture répartie sous-jacente. Les travaux que nous avons menés autour de la notion d'*opérateurs parallèles* sont un exemple d'utilisation de ce type de modèle de conception.
- Gestion de versions et de configurations. Nous avons commencé à explorer une voie consistant à réifier la notion de variantes, et à doter les composants logiciels de moyens d'auto-configuration leur permettant de s'adapter automatiquement à des environnements matériels ou logiciels variés.
- Adaptations d'interfaces. Il s'agit de prendre en compte la sémantique du composant adapté en s'appuyant sur la notion de contrat, i.e. la spécification formelle du composant.

L'objectif général de ce projet est l'exploration de ces problèmes, et l'étude d'un modèle de conception d'architectures d'objets distribuées permettant de les régler. Il s'agit en quelque sorte d'un *Méta-Framework* encadrant le développement de frameworks d'objets métiers distribués. Les défis scientifiques à relever concernent à la fois la définition du modèle (ou plus probablement du méta-modèle) de conception d'architectures d'objets distribuées Metafor intégrant un certain nombre de patrons de conception —*Design Patterns*—, mais aussi la définition et la réalisation efficace des composants génériques nécessaires et des outils de validation associés.

7.3 Convergence SDL-UML (RNRT Convergence)

Participants : Jean-Marc Jézéquel, François Pennaneach.

Mots clés : UML, SDL, objets, protocole.

Résumé : *Projet exploratoire* RNRT *(Contrat 298C4990031318011, 1/12/98 - 1/12/2001).*

SDL *est un standard utilisé dans les télécommunications pour la conception de systèmes temps réel.* UML *est un nouveau standard pour l'analyse et la conception orientées objet d'applications de tous types. Des sociétés américaines étudient des évolutions de* UML *pour le temps réel en réinventant de nombreux concepts sans tenir compte du fait que* SDL *les normalise déjà.*

Ce projet étudie et prototype des extensions temps-réel de UML *s'appuyant sur les solutions* SDL. *Il va également enrichir* SDL *avec des constructions* UML. *Il aboutit sur des propositions d'évolutions de ces normes à l'*ITU *et à l'*OMG *pour arriver à une approche unifiée permettant à* UML *d'être plus puissant et aux dizaines de milliers d'utilisateurs* SDL *dans les télécoms de préserver leur acquis et de bénéficier des avantages de* UML.

UML s'impose aujourd'hui comme notation universelle pour l'analyse et la conception. Certains des concepts présents dans UML visent plus particulièrement les systèmes temps réel, réactifs ou distribués : StateCharts, notion de classe active, de thread, etc.

La diffusion large et rapide d'UML fait qu'il est souvent considéré comme une alternative crédible à SDL et MSC, en dépit de quelques faiblesses dans le domaine du temps réel.

Les objectifs de ce projet sont :

1. Prise en compte dans UML des concepts nécessaires au développement de systèmes temps réel ou distribués, dans un standard reconnu qui garantisse l'indépendance du client vis-à-vis du vendeur.

2. Définition d'une correspondance formelle entre UML et SDL, permettant par exemple :

   - de faciliter la migration de l'existant SDL vers UML,
   - de clarifier, voire de définir formellement les concepts flous d'UML en bénéficiant du travail important effectué sur SDL,
   - de faire cohabiter des descriptions SDL et UML.

3. Intégration dans SDL de descriptions UML (utilisé dans ce cas pour la description des données, mal couverte par SDL aujourd'hui).

Le projet se déroule sur une durée de trois ans en cherchant à influencer les standards UML et SDL pendant cette période. Nos partenaires sont Telelogic et Objectif Technologie.

7.4 FormalFame : Validation d'architectures multi-processeurs

Participants : César Viho, Solofo Ramangalahy, Claude Jard, Thierry Jéron.

Mots clés : test de hardware, protocole de cohérence de caches, génération de test, vérification et validation, exécution.

Résumé : *Dans le cadre de l'action Vasy/FormalFame (Validation de Systèmes) du* GIE *(Groupement d'Intérêt économique) Bull-Inria Dyade, il s'agit de*

fournir des méthodes et outils pour la vérification formelle et le test de protocoles de cohérence de caches pour des architectures multi-processeurs développées chez Bull. Ce travail est effectué en collaboration avec l'Inria Rhône-Alpes (projet Vasy de H. Garavel).

Bull développe des architectures multiprocesseurs pour les prochaines générations de processeurs 64 bits Intel qui vont être mis sur le marché prochainement. Dans ce cadre, nous appliquons nos méthodes et outils de vérification et validation ; plus particulièrement aux composants ayant pour charge de maintenir la cohérence des caches de ces machines multiprocesseurs.

Après avoir travaillé sur une architecture à base de processeurs Merced (première génération de processeurs 64 bits Intel), notre cas d'étude est désormais une architecture à base de processeurs McKinley (deuxième génération). Nous menons des activités de vérification et validation grâce aux outils de CADP. Ces activités complètent celles utilisées dans le cycle de développement de l'architecture à Bull.

L'objectif de ces travaux est de pouvoir intégrer les méthodes formelles dans le cycle de développement utilisé. Par exemple, des séquences de simulations sont réutilisées pour valider grâce au *model checking* des exigences du cahier des charges.

L'expérience acquise dans les expérimentations précédentes [14] est réutilisée.

7.5 FormalCard: Validations formelles d'applications embarquées sur cartes à puces

Participants : Vlad Rusu, Elena Zinovieva, Duncan Clarke, Thierry Jéron.

Mots clés : logiciel pour carte à puce, vérification, génération de tests, techniques symboliques..

Résumé : *Cette nouvelle action Dyade associe la société Bull CP8 et les équipes Vasy de l'INRIA Rhône-Alpes et Pampa de l'IRISA. L'objectif est de réaliser un environnement de validation formelle pour logiciels embarqués sur cartes à puces.*

L'équipe Vasy est chargée de définir un langage de spécification prenant en compte les spécificités des applications visées (sécurité maximum, peu d'espace mémoire, ...). A partir de ce langage, les techniques de vérification enumérative de la boite à outils CADP permettront de faire de la vérification formelle. L'outil TGV sera employé pour générer des tests de conformité. Les techniques de génération de tests symboliques, en cours de développement dans l'equipe, seront également explorées.

7.6 AEE : Architecture Electroniques Embarquées

Participants : Benoît Caillaud, Thierry Jéron.

Mots clés : Architecture embarquée, automobile, composants, test..

Résumé :
Cette Action de Développement a pour but de concevoir et valider un processus rapide et sûr pour la définition de l'Architecture Système et le développement des logiciels associés embarqués. Ce processus est fondé sur l'indépendance entre matériel et logiciel, sur l'utilisation de méthodes, outils et composants standards, et sur la contractualisation des échanges entre les acteurs. Ce projet vise les applications de transport, notamment l'automobile. Les partenaires industriels de ce projet sont PSA, Renault, EDA (ex Aérospatiale Matra), Sagem, Siemens, Valéo et du coté académique l'IRCyN (Ecole Centrale de Nantes), Le LORIA-INPL et l'Inria (projets Meije à Sophia, Sosso, Hipercom à Rocquencourt, Ecoo à Nancy , Ep-Atr et Pampa à l'Irisa) . L'action est soutenue par le ministère de l'industrie.

Nous participons à ce projet dans une tâche qui a pour but de développer des méthodes et outils de validation d'une architecture distribuée constituée de logiciels de sources diverses et de matériels hétérogènes communiquants. L'entrée de cette tâche est un langage de description d'architecture (AIL) défini dans le projet. Nous intervenons plus particulièrement dans le cadre de la génération automatique de test à partir de descriptions d'architectures en AIL et envisageons l'utilisation de notre outil TGV.

7.7 Méthodologie de tests d'interopérabilité

Participants : César Viho, Lénaïck Tanguy, Claude Jard.

Mots clés : test d'interopérabilité, test réparti, asynchronisme, concurrent-TTCN, exécution de tests.

Résumé : *Contrat Université Rennes I/*KBKN *marché 98 42.561, novembre 1999 - octobre 2001*
L'objectif de cette collaboration Irisa/Pampa avec le Celar-Bruz (Centre Electronique et de l'armement) de la DGA *est de proposer un cadre méthodologique - analogue à ce qui existe pour le test de conformité - pour le test d'interopérabilité des protocoles de communication.*

Le test d'interopérabilité va au-delà du test de conformité : il vérifie l'interaction cohérente d'un ensemble de composants. La génération automatique de tests d'interopérabilité est plus complexe car cela revient à générer un ensemble de testeurs répartis et interagissant entre eux de manière asynchrone. Les problèmes d'asynchronisme et de nouveaux problèmes de contrôle et surtout d'observation répartie des entités en interaction se posent. Le test d'interopérabilité considére que :

- les différentes entités à tester sont réparties sur une architecture répartie,
- le testeur est également implanté de façon répartie sous la forme d'un ensemble de testeurs parallèles connectés aux entités de protocole et interconnectés entre eux et échangeant des informations de synchronisation de manière asynchrone.

Ceci engendre de nombreux problèmes à résoudre simultanément :

- l'observation répartie des comportements par les testeurs répartis,
- la gestion de l'asynchronisme testeur/testé et entre testeurs,
- la synthèse de la procédure de coordination entre les testeurs répartis.

7.8 Oural

Participants : Jean-Marc Jézéquel, Alain Le Guennec, Séverine Simon.

Mots clés : UML, TGV, validation, objets.

Résumé : *Projet pré-compétitif* RNRT. *Le projet se déroule sur une durée de deux ans, à partir de Février 1999. Nos partenaires sont Softeam, Alcatel et Thomson LCR.*

*L'objectif de ce projet est de fournir un ensemble d'outils destinés à accélérer et améliorer le processus industriel de développement de produit par le biais de réutilisation de composants logiciels. Dans ce processus, nous nous focalisons essentiellement sur les phases "amont" que sont: la définition et analyse d'architecture, la validation de l'architecture, l'exploitation des informations en vue de la génération de scénarios exécutables et la génération de squelettes de code. Sur une échéance de deux ans, le projet a pour objectif la conception et le développement d'un environnement autour d'*UML *permettant à la fois:*

- *de décrire une application du point de vue de son architecture dans le formalisme standard* UML,
- *de valider fonctionnellement cette architecture*
- *de valider, notamment par la simulation, des aspects non fonctionnels de l'architecture*
- *de dériver cette description sur une plate-forme générique basée sur l'architecture de Corba*

Le projet a pour ambition de fédérer autour du langage UML *les différents outils des partenaires, et de promouvoir un processus de développement industriel basé sur une modélisation unique de l'application. Pour ce qui concerne Pampa, Oural est la partie pré-compétitive de la recherche menée dans Reutel.*

L'objectif de ce projet est de fournir un ensemble d'outils destinés à accélérer et améliorer le processus industriel de développement de produit par le biais de réutilisation de composants logiciels. Dans ce processus, nous nous focalisons essentiellement sur les phases "amont" que sont : la définition et analyse d'architecture, la validation de l'architecture, l'exploitation des informations en vue de la génération de scénarios exécutables et la génération de squelettes de code. Le projet se déroule sur une durée de deux ans, à partir de Février 1999. Nos partenaires sont Softeam, Alcatel et Thomson-LCR.

Le projet est articulé autour de l'outil d'édition UML Objecteering fourni par Softeam. Thomson-CSF apporte son expérience en validation non fonctionnelle d'architecture. L'Inria apporte ses compétences en validation fonctionnelle du logiciel, et Alcatel-CIT réalise le lien entre la modélisation sous UML et la génération de code exécutable sur une plate-forme générique Corba.

Le sous projet outil de description a pour but de définir autour de l'outil Objecteering les extensions requises pour permettre l'utilisation d'UML, tant du point de vue de l'ingénierie système que de l'ingénierie logicielle. Dans ce sens, une évolution de la sémantique d'UML sera prototypée (UML/T) et les outils exploitant cette sémantique seront intégrés autour de l'éditeur d'Objecteering.

Le sous projet outils d'ingéniérie fonctionnelle a pour but de définir et fournir un outillage destiné a faire une validation formelle d'une architecture logicielle décrite en UML/T. Ces travaux s'articulent autour des extensions d'UML destinés à enrichir la description fonctionnelle du comportement d'une classe. L'environnement propose un simulateur de comportement et un générateur de tests. L'utilisation de tels outils nécessitent une modélisation fonctionnelle de la plate-forme d'exécution générique.

Le sous projet outils d'ingénierie non fonctionnelle a pour but de fournir un environnement de validation par la simulation d'une architecture de composants. Ces composants possèdent des propriétés aussi diverses que les performances temps-reel, l'utilisation de ressources, le degrés de fiabilité,.. qui seront extraites des descriptions UML/T. L'environnement de validation comprends un simulateur de comportement non fonctionnel et un générateur de tests. Afin de mener a bien les simulations, une modélisation non fonctionnelle de la plate-forme générique sera étudiée.

Le sous projet outils de déploiement a pour but de fournir les outils destinés a déployer une architecture de composants sur la plate-forme générique Alcatel/Thomson pour en faire une architecture exécutable. Cet outil exploite les informations de déploiement décrites dans UML pour générer le code destiner à piloter les services de la plate-forme (ORB, tolérance au défaillances, autres middleware). D'autre part, un environnement de test en ligne sera étudié à partir de l'exploitation des scénarios de tests résultants des précédents outils.

7.9 Alcatel/Reutel

Participants : Claude Jard, Jean-Marc Jézéquel, Benoît Caillaud, Hubert Canon, Loïc Hélouet.

Mots clés : méthode, UML, langage d'interface, IDL, BDL, synchrone, objets.

Résumé : *Contrat Inria 1 97 A 93600 000MC012, novembre 1995 - décembre 2000*

La collaboration avec Alcatel a commencé en novembre 1995 dans le cadre de l'appel Artica du ministère de la recherche et la mise à disposition d'un post-doctorant Inria chez Alcatel en 1996. En septembre 1997, elle a pris un nouvel essor avec un contrat direct avec Alcatel permettant le démarrage de plusieurs thèses. Claude Jard en est le coordinateur à l'Inria. L'objectif de la collaboration

> *est la maîtrise du développement logiciel d'applications de télécommunication par la conception d'outils de manipulations formelles à l'intérieur d'une chaîne de développement définie par Alcatel. En 2000, cinq équipes de l'Inria participent au projet (*ADP*, Compose,* EPATR*, Vasy et Pampa) autour de quatre actions scientifiques. Pampa est principalement engagé avec* EPATR *et Vasy dans la définition et l'outillage d'un langage d'interface appelé* BDL*.*

Le défi pour Alcatel est de réduire les coûts de développement tout en améliorant la qualité du logiciel. Il s'agit aussi d'assurer une flexibilité et réactivité élevées en regard des évolutions du marché et des technologies naissantes. Précisément, il s'agit d'utiliser et de contrôler au mieux la mise en œ uvre d'une application sur une plate-forme donnée (par génération de code quand c'est possible).

Le cadre de développement d'Alcatel est défini dans l'esprit Corba (the Common Object Request Broker Architecture) et met l'accent sur l'écriture de schémas de programmes dans des langages d'interface comme IDL (Interface Definition Language). Il doit être compatible en amont avec une méthodologie de conception objet type UML. Il doit aussi pouvoir s'interfacer avec un choix libre de plate-formes d'exécution : il faut donc bien veiller au rôle tampon du langage d'interface qui doit abstraire correctement la plate-forme et permettre aussi l'utilisation de langages variés pour la programmation des objets logiciels.

Pour faire en sorte qu'il ne reste pas un simple guide de structuration du logiciel, et puisse vraiment répondre aux objectifs généraux de la maîtrise du développement, il est nécessaire de lui adjoindre une batterie d'outils pour assister le concepteur d'applications. C'est ici que se situe l'offre de l'Inria proposant l'utilisation de méthodes formelles. Par méthode formelle, nous entendons un formalisme fondé sur une sémantique mathématique et un ensemble de techniques associées permettant la manipulation (en général automatique) des programmes, à savoir la génération de code, des transformations, des vérifications, des optimisations, de la génération de tests,... Le travail consiste à proposer des extensions aux langages d'interface pour faire apparaître des informations non fonctionnelles (comportementales) et à concevoir des outils de manipulation formelle traitant les schémas de programme définis par les interfaces. L'Inria participe aussi à l'extension de la chaîne pour la prise en compte de la résistance aux défaillances.

Les équipes Inria engagées dans Reutel en 1999 sont :

- EPATR : modélisation synchrone, vérification, génération de code optimisé.
- Pampa : modélisation asynchrone, vérification, génération de tests, méthodologie de conception objet.
- Compose : optimisation de code, spécialisation, évaluation partielle.
- ADP : services et algorithmes pour réaliser la fonction de tolérance aux défaillances.
- Vasy : technologie de vérification, ouverture de l'outil CADP.

et se sont regroupées sur quatre actions de recherche :

1. Corba "fault-tolerant" (M. Hurfin)

2. Spécialisation et analyse de codes (G. Muller)

3. Modélisation et validation UML. Extensions à Oural (JM. Jézéquel)

4. Outils de validation et de génération de code pour UML (JP. Talpin, B. Caillaud)

Le travail du projet Pampa se concentre principalement sur les actions 3 et 4 en coopération étroite avec le projet EPATR. Nous concevons un nouveau langage d'interface : BDL (pour Behavioural Description Language). Le rôle de BDL est d'accompagner le développeur depuis les tâches préliminaires de spécification jusqu'au travaux de codage.

BDL intervient dans cette chaîne comme support à cet accompagnement et comme interface à l'intégration d'outils. Il est tout autant, pour son utilisateur, un langage de spécification, riche, expressif et structuré, avec une forte orientation objet, qu'un moyen de représentation intermédiaire simple, flexible et efficace pour l'utilisation des logiciels d'analyse, de test, de vérification et d'optimisation envisagés.

BDL se présente techniquement comme un formalisme permettant d'abstraire un objet et ses méthodes sous la forme d'un ensemble de graphes orientés étiquetés par les états de l'objet et de son environnement. Ces graphes permettent de coder les communications et leur ordonnancement, ainsi que les dépendances de données et de contrôle. Du point de vue de l'utilisateur, BDL se situe entre les MSC et les StateCharts. Du point de vue sémantique, ils correspondent aux automates d'ordres partiels.

Lorsque ces ordres partiels sont synchronisés, on se retrouve dans le monde des langages synchrones à la Signal, ce qui permet de récupérer les techniques de preuve et de génération de code associés. Ces ordres partiels peuvent également être interprétés de manière asynchrone, donnant un fondement sémantique à la communication, et permettant aussi une connexion avec les outils de preuve et test CADP et TGV. En 2000 a été réalisé une plate-forme de démonstration intégrant BDL à UMLAUT, CADP et TGV. L'éditeur graphique de BDL a été sous-traité à la société TNI.

7.10 Gemplus

Participants : Lydie Dubousquet, Jean-Marc Jézéquel, Claude Jard.

Mots clés : tests application de commerce électronique, UML, test symbolique.

Résumé : *Contrat Gemplus-Inria. Septembre 1999 - Septembre 2000.*

L'étude a consisté à comparer le processus manuel de validation utilisé par la société Gemplus à un processus outillé proposé par Pampa, et sur une application de porte-monnaie électronique, couplé à des applications de fidélité. A l'Irisa, nous avons écrit une spécification UML de cette application, puis généré des cas de test en utilisant le couplage UMLAUT/TGV. Plusieurs outils complémentaires, mettant en œuvre une méthodologie particulière de conceptions de tests ont été spécifiés et validés. En parallèle avec cette étude de cas, un travail sur le test symbolique en UML a été commencé, travail qui va se poursuivre dans le cadre du projet RNTL COTE 7.11.

7.11 COTE

Participants : Claude Jard, Jean-Marc Jézéquel, Simon Pickin, Thierry Jéron.

Mots clés : Tests en UML, TGV, test symbolique.

Résumé : *Contrat RNTL/COTE CNRS. Novembre 2000 - Novembre 2002.*
Ce projet démarre en fin d'année 2000. Il fait collaborer deux laboratoires de recherche (Irisa/Pampa et Lande, et le LSR/Imag), deux grands industriels (France Telecom et Gemplus) ayant un intérêt stratégique pour le test de composants, et un industriel outilleur Softeam, développant un atelier UML. COTE a pour objectif de définir des techniques de modélisation de test en UML, pour pouvoir modéliser les modes d'exploitation d'un composant, et en dériver des moyens de test automatique, ainsi que des moyens de "labellisation". Pampa amène ses compétences UMLAUT/TGV et va participer à une intégration dans l'atelier Objecteering, tout en effectuant de la recherche en collaboration avec le projet Lande et le LSR sur le mixage des aspects contrôle et données dans le processus de génération de tests.

8 Actions régionales, nationales et internationales

8.1 Actions régionales

ITR-Softeam

Participants : Yves Le Traon, Jean-Marc Jézéquel, Hanh Vu Le.

Mots clés : tests d'intégration, UML.

Le but de l'ensemble du projet ITR en collaboration avec SOFTEAM et Pampa est :

- de fournir des moyens de planifier les tests dès la conception du logiciel (à partir de modèles UML, en particulier des diagrammes de classes),
- de fournir des stratégies en amont du développement les plus efficaces possibles pour minimiser l'effort de test,
- et de produire le plus automatiquement une synthèse de ces tests intégrée à l'outil Objecteering.

Les travaux accomplis portent sur :

1. Un état de l'art permettant de situer l'intérêt d'un tel projet et de servir de base de comparaison expérimentale pour mesurer les gains obtenus par les algorithmes proposés. Cet état de l'art révèle qu'aucun travail scientifique n'a cherché à modéliser et à planifier les tests d'intégration des composants à partir de UML. Il existe par contre des travaux non spécifiques à UML qui portent sur le test d'intégration dans le domaine OO. Les algorithmes d'intégration étudiés servent de base de comparaison lors de l'évaluation de l'efficacité de nos propres algorithmes.

2. La mise en place d'une méthodologie basée sur la notion classe-spécification-test comme canevas et la modélisation de la notion de dépendances de test à partir de UML. Les algorithmes développés visent à minimiser les coûts et délais de la phase de test d'intégration (minimisation des stubs et optimisation de la répartition des tâches de test). Ils traitent de manière efficace le problème des cycles de dépendances et produisent un plan d'intégration dès la conception.

3. Des études de cas sur des modèles UML correspondant à des logiciels réels et comparaison de notre approche avec les autres approches existantes.

8.2 Actions nationales

8.2.1 Test d'interopérabilité et de QoS des protocoles Internet Nouvelle Génération

Contrat CNRS-Programme Telecommunications 2K0008. Durée : 24 mois à partir de décembre 1999.

Participants : César Viho, Claude Jard, Sébastien Barbin.

Mots clés : test d'interopérabilité, protocoles Internet nouvelle génération, QoS.

Il s'agit d'anticiper les problèmes liés à l'interconnexion des nouveaux réseaux Internet IPv6 (dits de nouvelle génération). Ceci passe notamment par le développement de méthodes et outils appropriés pour le test d'interopérabilité des nouveaux protocoles mis en œuvre dans ces réseaux. Or, il n'existe pas à l'heure actuelle de méthodologie bien définie pour les tests d interopérabilité et les recherches en la matière sont à un stade de balbutiement. Cette année, nous avons principalement étudié les normes des protocoles IPv6 et généré des tests de conformité et d' interopérabilité pour les protocoles de base tels que Neighbor Discovery, Path MTU Discovery, Stateless Address Autoconfiguration, mobile-IPv6, RIPng, PPPv6. Ces tests ont servi pour le "bake-off" organisé par l'ETSI Sophia du 2 au 6 octobre 2000. Ils ont permis de mettre en évidence des erreurs dans les implantations de piles protocolaires IPv6 testées, ainsi que des ambiguïtés/incohérences/imprécisions dans les normes.

Du point de vue des utilisateurs de ces réseaux, il faudra également garantir une qualité de service (QoS) de plus en plus forte : garantie de débit minimum, de performance, d'accès aux ressources, etc. Au test d'interopérabilité, il faudra donc rajouter des tests de QoS. Or, les travaux combinant les aspects qualitatifs et quantitatifs sont inexistants. Généralement, ces deux types de tests sont effectués de manière totalement indépendante de façon ad hoc, "à la demande" et "à la main". Par la suite, nous proposons d'étendre le savoir-faire TGV à la prise en compte d'aspects quantitatifs liés à la QoS tels que le temps d'acheminement, les débits minimum garantis, etc. Il s'agit d'un sujet original, qui nous semble maintenant abordable et comportant des aspects modèles, algorithmiques et expérimentaux. Techniquement, nous envisageons de démarrer avec le système KRONOS d'analyse d'automates temporisés, développé à Vérimag-Grenoble et connectable dès maintenant à l'outil TGV.

8.2.2 Groupe de travail test et objets

Participants : Claude Jard, Jean-Marc Jézéquel, Thierry Jéron, Yves Le Traon, Alain Le Guennec.

L'équipe Pampa anime un groupe de travail national sur le thème du test et des objets. Ce groupe est affilié au PRC/GDR ALP. Claude Jard en est co-responsable (avec MC. Gaudel et P. Thevenod). Yves Le Traon en est le secrétaire. Une dizaine d'équipes en France participent. La page Web http://www.irisa.fr/testobjets peut être consultée.

8.2.3 Action coopérative MARS

Participant : Benoît Caillaud.

L'action de recherche coopérative de l'INRIA "MARS" (`http://www.loria.fr/\%7Exie/Mars.html`) porte sur la modélisation, la vérification et la synthèse à l'aide de réseaux de Petri de commande des systèmes à événements discrets : systèmes de production, systèmes de flux de tâches, etc. Benoît Caillaud contribue à cette action en montrant l'applicabilité des techniques de synthèse de réseaux de Petri (et de l'outil SYNET) au problème de la synthèse de contrôleurs de systèmes à événements discrets répartis.

8.3 Réseaux et groupes de travail internationaux

L'équipe Pampa participe à une action de coopération concertée permettant l'échange de chercheurs avec l'université de Twente (E. Brinskma) sur la génération automatique de tests.

Nous avons aussi défini un accord de collaboration avec le SRI (Stanford Research Institute, J. Rushby), associant aussi le laboratoire Vérimag sur l'utilisation de techniques symboliques. Vlad Rusu a effectué deux séjours de un mois au SRI dans ce cadre.

Jean-Marc Jézéquel est membre de Nice (*Non-Profit International Consortium for Eiffel*), qui a en charge la standardisation du langage Eiffel et de ses bibliothèques. Il participe au groupe de travail international *Trusted Components* (http://www.trusted-components.org/), qui a pour objectif l'étude et le développement de technologies permettant l'utilisation de composants logiciels dans des systèmes critiques.

Benoît Caillaud participe à la coopération franco-polonaise CATALYSIS entre le CNRS et l'IPIPAN de Gdansk sur la synthèse de réseaux de Petri et la théorie des régions (responsables scientifiques M. Bednarczyk et P. Darondeau).

8.4 Accueils de chercheurs étrangers

Dans la lignée d'une série d'échange avec l'université de Tokyo depuis 1996, Naohito Sato (université de Tokyo) a effectué un séjour de trois semaines en septembre 2000 dans Pampa sur le thème de la programmation parallèle et répartie par objets avec UML.

L'équipe a accueilli L. Ricker (venant de l'université de Toronto) pour neuf mois dans le cadre du programme de post-doctorat Ercim.

L'équipe accueille S. Pickin (de l'université de Madrid) pendant 2 mois en 2000 dans le cadre du projet COTE.

9 Diffusion de résultats

9.1 Animation de la communauté scientifique

Claude Jard a co-organisé l'école d'été internationale MOVEP'2000 sur la modélisation et vérification de processus parallèles (une centaine de participants à Nantes). A sa suite, il co-édite un livre de la série LNCS Tutorial.

Claude Jard a co-présidé la conférence internationale SAM'2000 à Grenoble. Loïc Hélouet était dans le comité d'organisation et Benoît Caillaud dans le comité de programme.

Claude Jard a participé en 2000 aux conférences suivantes en tant que membre du comité de programme :

- Tacas'2000 : Tools and Algorithms for the Construction of Systems, Etaps, Berlin, avril 2000,
- Testcom'2000 : Test of communicating systems, Ottawa, octobre 2000,
- Afald'2000 : Atelier sur l'assistance formalisée au développement logiciel, Grenoble, janvier 2000,
- CFIP'2000 : Ingénierie des protocoles, Toulouse, septembre 2000,
- RENPAR12 : Rencontre du parallélisme, Besançon.

Claude Jard est membre du comité de pilotage de la série de colloques MSR.

Claude Jard est président de la commission personnel de l'Irisa, chargée d'instruire les recrutements de personnels scientifiques temporaires. Claude Jard est membre du conseil de laboratoire et du comité des projets de l'Irisa, membre des commissions de spécialistes à l'université de Rennes 1 et l'Insa. Il est membre du bureau de la section 07 du comité national de la recherche scientifique.

Claude Jard est membre du comité de pilotage de l'association Goétic qui rassemble plusieurs sociétés industrielles et publiques dans l'objectif de veille technologique dans le domaine des télécommunications.

Claude Jard est membre du comité d'expertise du programme canadien (NSERC) sur les réseaux d'excellence en informatique et télécommunication.

Thierry Jéron est membre de la commission de spécialistes de l'ENS Cachan.

Thierry Jéron a fait partie du comité de programme de Lfm'2000 : Fifth Nasa Langley Formal Methods Workshop, juin 2000, Williamsburg, Virginie, USA.

Thierry Jéron est membre élu au Conseil de Laboratoire de l'Irisa. Il est responsable du Séminaire Irisa, Marie Noëlle Georgeault se chargeant des aspects administratifs et diffusion.

Jean-Marc Jézéquel a présidé la conférence Tools Europe 2000 (Mont St Michel, Juin 2000), le Workshop "Objects and Patterns in Telecom" associé à ECOOP2000, et animé la journée "Composants Fiables" organisée à Paris par le groupe "Zéro Défauts" de la SEE le 25 mai 2000, ainsi que la journée "développement par objets et UML" dans le cadre des séminaires IrisaTech le 10 novembre 2000.

Jean-Marc Jézéquel a fait partie des comités de programmes des conférences :

- ICSSEA 2000 : Génie logiciel, Paris, 2000.
- NOTERE 2000 : Colloque Francophone sur les NOuvelles TEchnologies de la REpartition, Paris, Novembre 2000.
- OCM'2000 (Objets, Composants et Modélisation). Nantes, 2000.
- LMO2000 (Langages et Modèles à Objets), Montréal, janvier 2000.
- Tools Pacific 2000 : Technologies Objets, Sydney, Novembre 2000.
- UML'2000 : 3rd International Conference on UML. York, 2000.

César Viho participe à l'association GOETIC (Groupe d'observation et d'évaluation des technologies de l'information et de la communication). Il est responsable du projet MTI (Méthodologie de tests d'interopérabilité).

César Viho est membre du G6test, le sous-groupe du G6 (Groupe francophone des utilisateurs de IPv6) qui s'occupe des problèmes de test de conformité et d'interopérabilité des nouveaux protocoles IPv6. Il a co-organisé le "bake-off" de tests d'interopérabilité des protocoles IPv6 qui a eu lieu à l'ETSI Sophia du 2 au 6 octobre 2000.

César Viho est membre de la commission des spécialistes de l'Ifsic/université Rennes I, de l'Insa/Rennes et de l'université d'Angers.

9.2 Enseignement universitaire

Depuis septembre 1999, Benoît Caillaud est responsable de la filière "génie logiciel et méthodes formelles" du DEA d'informatique de l'université de Rennes 1. Il est également responsable du module "analyse comportementale des systèmes réactifs et répartis" de cette filière du

DEA.

Claude Jard et Yves Le Traon sont responsables du module "Composants et Test" du DEA informatique de l'Ifsic.

Thierry Jéron intervient dans le module "Effectivité et Efficacité" du DEA informatique de l'Ifsic.

Claude Jard et Thierry Jéron enseignent la validation de protocoles (Vérification, model-checking, test, observation répartie) à l'ENSTB Rennes et en Diic à l'Ifsic.

César Viho anime un cours sur les "mécanismes des réseaux informatiques" dans la filière DIIC et sur la "validation et le test de protcoles" dans la filière Réseau du DESS-ISA de l'Ifsic.

Jean-Marc Jézéquel anime un cours de méthodologie de conception par objets de logiciels en DESS/CCI, en Diic $3^{\text{ème}}$ année à l'Ifsic, et à l'ENSTB (Rennes) ; il assure les mêmes cours plus un cours d'architectures logicielles pour réseaux à haut débit à Supélec (Rennes).

Yves Le Traon a créé la filière Génie Logiciel du DESS-Isa.

Vlad Rusu a donné un cours-conférence en DEA sur l'utilisation du prouveur PVS.

L'équipe Pampa a accueilli en 2000, deux stagiaires de DEA de l'Ifsic, un stagiaire de Supelec, deux mini-projet Diic et deux stagiaires étrangers.

9.3 Participation à des colloques, séminaires, invitations

Claude Jard a été invité à donner les séminaires suivants :

- "Distributed Testing", Université de Madrid, Espagne, mars 2000.
- "Executing High-level Message Sequence Charts", Dagstuhl, Allemagne, Novembre 2000.

Thierry Jéron a été invité à donner les séminaires suivants :

- "Automatic test synthesis using model-based verification techniques" à à l'Université Juan Carlos de Madrid en Mars 2000.
- "On the fly test synthesis with TGV" dans le cadre du Workshop *Specification and Validation of Reactive Systems* organisé par le projet LTR Vires à Autrans en juin 2000
- "Génération automatique de tests de conformité" dans le cadre de l'Ecole CEA-EDF-INRIA *Méthodes Formelles : Spécifications et vérification de systèmes dynamiques* en juillet 2000,

Jean-Marc Jézéquel a été invité à donner les séminaires suivants :

- "Testing OO Software", Université de Madrid, Espagne, mars 2000.

– "Trusted Components", Microsoft Research, Cambridge, UK, mars 2000.

– "Design Patterns with Contracts", Université de Brighton, UK, novembre 2000.

Yves Le Traon a été invité à donner un séminaire sur le thème "Trusted components" à la conférence Ericson, Rome, décembre 1999.

Vlad Rusu a donné en juillet 2000 un séminaire intitulé "An approach to Symbolic Test Generation" à l'université de Twente.

10 Bibliographie

Ouvrages et articles de référence de l'équipe

[1] J. Fernandez, C. Jard, T. Jéron, L. Mounier, « On-the-fly Verification of Finite Transition Systems», *Formal Methods in System Design 1*, 1993, p. 251–273.

[2] J.-C. Fernandez, C. Jard, T. Jéron, C. Viho, « An Experiment in Automatic Generation of Test Suites for Protocols with Verification Technology», *Science of Computer Programming*, 29, 1997, p. 123–146.

[3] E. Fromentin, C. Jard, G.-V. Jourdan, M. Raynal, « On-the-fly Analysis of Distributed Computations», *Information Processing Letters*, 54, 1995, p. 267–274.

[4] F. Guidec, J.-M. Jézéquel, J.-L. Pacherie, « An Object Oriented Framework for Supercomputing», *Journal of Systems and Software Special Issue on* Software Engineering for Distributed Computing, juin 1996.

[5] C. Jard, R. Groz, J. Monin, « Development of VEDA: a prototyping tool for distributed algorithms», *in : IEEE Trans. on Software Engin., 14,3*, p. 339–352, mars 1988.

[6] C. Jard, J.-M. Jézéquel, «ECHIDNA, an Estelle-compiler to prototype protocols on distributed computers», *Concurrency Practice and Experience 4*, 5, août 1992, p. 377–397.

[7] J.-M. Jézéquel, *Object Oriented Software Engineering with Eiffel*, Addison-Wesley, mars 1996, ISBN 1-201-63381-7.

Livres et monographies

[8] R. Mitchell, J.-M. Jézéquel, J. Bosch, B. Meyer, A. C. Wills, M. Woodman (éditeurs), *Technology of object-oriented languages and systems (TOOLS Europe), 33*, IEEE Computer Society, juin 2000.

Thèses et habilitations à diriger des recherches

[9] L. Hélouët, *Analyse des exigences des systèmes répartis exprimées par des langages de scénarios*, thèse de doctorat, Ecole doctorale MATISSE, Universite de Rennes 1, Octobre 2000.

[10] P. Morel, *Une algorithmique efficace pour la génération automatique de tests de conformité*, thèse de doctorat, Université de Rennes I, France, février 2000.

Articles et chapitres de livre

[11] A. Benveniste, B. Caillaud, P. Le Guernic, « Compositionality in dataflow synchronous languages: specification and distributed code generation », *Information and Computation*, To appear.

[12] M. Bozga, J.-C. Fernandez, L. Ghirvu, C. Jard, T. Jéron, A. Kerbrat, P. Morel, L. Mounier, « Verification and test generation for the SSCOP protocol », *Journal of Science of Computer Programming, Special Issue on The Application of Formal Methods in Industrial Critical Systems 36*, 2000, p. 27–52.

[13] B. Caillaud, P. Darondeau, L. Hélouët, G. Lesventes, «HMSCs as specifications... with PN as completions », *in : Selected papers from the summer school MOVEP'2k: Modelling and verification of parallel processes*, *Lecture Notes in Computer Science*, Springer, to appear.

[14] H. Garavel, C. Viho, M. Zendri, «System design of a CC-NUMA multiplrocessor architecture using formal specification, model-checking, co-simulation and test generation and execution », *International Journal on Software Tools for Technology Transfer - Special Issue To be published*, novembre 2000.

[15] C. Jard, T. Jéron, « An educational case study in protocol verification and distributed observation», *Journal of Computer Science Education, ECASP Special Issue 10*, 3, 2000, to appear.

[16] J.-M. Jézéquel, *Traité IC2 : Langages à objets*, Hermès Science Publications, Paris, To be published 2000, ch. Eiffel.

[17] S. Ramangalahy, P. Le Gall, T. Jéron, «Une application de la théorie des jeux au test de conformité», *Revue Electronique sur les Réseaux et l'Informatique Répartie 9*, mai 2000, p. 3–23.

[18] Y. L. Traon, T. Jéron, J.-M. Jézéquel, P. Morel, «Efficient OO Integration and Regression Testing», *IEEE Trans. on Reliability 49*, 1, mars 2000, p. 12–25.

Communications à des congrès, colloques, etc.

[19] B. Baudry, Y. L. Traon, J.-M. Jézéquel, H. V. Le, «Trusatble Components: Yet Another Mutation-Based Approach», *in : 1st Symposium on Mutation Testing (Mutation'2000)*, p. 69–76, San Jose, CA, octobre 2000.

[20] B. Baudry, Y. L. Traon, H. V. Le, J.-M. Jézéquel, «Building Trust into OO Components using a Genetic Analogy», *in : International Symposium on Software Reliability Engineering 2000 (ISSRE'2000)*, San Jose, CA, octobre 2000.

[21] B. Baudry, Y. L. Traon, H. V. Le, «Testing-for-Trust: the Genetic Selection Model applied to Component Qualification», *in : Technology of object-oriented languages and systems (TOOLS Europe)*, *33*, IEEE Computer Society, p. 108–119, juin 2000.

[22] A. Belinfante, L. Du Bousquet, S. Ramangalahy, S. Simon, C. Viho, R. De Vries, « Formal test automation: the conference protocol with TGV/TorX », *in : IFIP TC6/WG6.1 13th International Conference on Testing of Communicating Systems, TestCom 2000*, Kluwer Academic, p. 221–228, août 2000.

[23] B. Caillaud, P. Darondeau, L. Hélouët, G. Lesventes, «HMSCs as specifications... with PN as completions», *in : Proceedings of the summer school MOVEP'2k: Modelling and verification of parallel processes*, F. Cassez, C. Jard, B. Rozoy, M. Ryan (éditeurs), p. 87–103, Nantes, June 2000.

[24] E. Cariou, A. Beugnard, «Specification of Communication Components in UML», *in : The 2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2000), session Coordination in Parallel and Distributed Applications and Activities*, June 2000, `http://www.ashland.edu/~iajwa/Conferences/PDPTA2000/pdpta.html`.

[25] E. Cariou, «Spécification de Composants de Communication en UML», *in : Objets, Composants, Modèles (OCM'2000)*, May 2000.

[26] L. Du Bousquet, H. Martin, « Automatic test generation for Java-Card applets », *in : 4th Workshop on Tools for System Design and Verification*, juillet 2000.

[27] L. Du Bousquet, H. Martin, « Automating Java Card applet testing », *in : Java Card workshop*, septembre 2000.

[28] T. Duval, F. Pennaneac'h, « Using the PAC-Amodeus model and design patterns to make interactive an existing object-oriented kernel», *in : Technology of Object-Oriented Languages and Systems (TOOLS Europe)*, *33*, IEEE Computer Society, p. 407–418, 2000.

[29] E. Fabre, A. Benveniste, C. Jard, L. Ricker, M. Smith, «Distributed State Reconstruction for Discrete Event Systems», *in : 39th IEEE Conf. on Detection and Control (CDC2000*, décembre 2000.

[30] A. L. Guennec, G. Sunyé, J.-M. Jézéquel, « Precise Modeling of Design Patterns », *in : Proceedings of UML 2000*, *LNCS*, *1939*, Springer Verlag, p. 482–496, 2000.

[31] A. L. Guennec, «Méthodes formelles avec UML», *in : CFIP'2000 : Colloque Francophone sur l'Ingénierie des Protocoles*, Hermes, oct 2000.

[32] L. Hélouët, C. Jard, B. Caillaud, « An effective equivalence for sets of scenarios », *in : Proceedings of GRATRA'2000 (joint appligraph and getgrats workshop on graph transformation systems), ETAPS 2000*, G. T. H. Ehrig (éditeur), Berlin, Germany, March 2000.

[33] L. Hélouët, C. Jard, « Conditions for synthesis of communicating automata from HMSCs », *in : 5th International Workshop on Formal Methods for Industrial Critical Systems (FMICS)*, A. R. E. Stefania Gnesi, Ina Schieferdecker (éditeur), GMD FOKUS, Berlin, avril 2000, `http://www.gmd.de/publications/report/0091`.

[34] L. Hélouët, P. Le Maigat, «Decomposition of Message Sequence Charts», *in : Proceedings of SAM2000(2nd conference on SDL and MSCs)*, Grenoble, Juin 2000.

[35] W. Ho, F. Pennaneac'h, N. Plouzeau, «UMLAUT: A Framework for Weaving UML-based Aspect-Oriented Designs », *in : Technology of object-oriented languages and systems (TOOLS Europe)*, *33*, IEEE Computer Society, p. 324–334, juin 2000.

[36] C. Jard, T. Jéron, P. Morel, « Verification of test suites », *in : IFIP TC6/WG6.1 13th International Conference on Testing of Communicating Systems, TestCom 2000*, p. 3–18, août 2000.

[37] P. Le maigat, L. Hélouët, «A (max,+) approach for time in Message Sequqnce Charts», *in : Proceedings of WODES'2000 (workshop on Discrete Event Systems)*, Gent, Belgique, Aout 2000.

[38] L. Ricker, E. Fabre, «On the construction of modular observers and diagnosers for discrete-event systems», *in : 39th IEEE Conf. on Detection and Control (CDC2000*, décembre 2000.

[39] V. Rusu, L. du Bousquet, T. Jéron, «An Approach to Symbolic Test Generation», *in : Integrated Formal Methods (IFM'00), Dagstuhl, Allemagne*, Novembre 2000. à paraitre dans LNCS.

[40] N. Sato, J.-M. Jézéquel, «Implementing and Evaluating an Efficient Load Balancer for Distributed Molecular Dynamics Simulation», *in : IEEE Workshop on High-Performance Scientific and Engineering Computing with Applications*, IEEE Press, août 2000.

[41] G. Sunyé, A. Le Guennec, J.-M. Jézéquel, «Design Pattern Application in UML», *in : ECOOP'2000 proceedings*, E. Bertino (éditeur), 1850, Lecture Notes in Computer Science, Springer Verlag, p. 44–62, juin 2000.

[42] L. Tanguy, C. Viho, C. Jard, «Synthesizing coordination procedures for distributed testing of distributed systems», *in : ICDCS Workshop Distributed System Validation and Verification (DSVV'2000)*, T. H. Lai (éditeur), IEEE Computer Society, p. E67–E74, Taipei, Taiwan, ROC, avril 2000.

[43] L. Tanguy, C. Viho, C. Jard, «Un outil pour la parallélisation automatique de l'exécution répartie de cas de test», *in : CFIP'2000 : Colloque Francophone sur l'Ingénierie des Protocoles*, J.-P. Courtiat, M. Diaz, P. Sénac (éditeurs), Hermes, p. 333–348, Toulouse, France, oct 2000.

[44] Y. L. Traon, F. Ouabdesselam, C. Robach, «Analyzing Testability on Data Flow Designs», *in : International Symposium on Software Reliability Engineering 2000 (ISSRE'2000)*, San Jose, CA, octobre 2000.

Rapports de recherche et publications internes

[45] B. Caillaud, P. Darondeau, L. Hélouët, G. Lesventes, «HMSCs en tant que spécifications partielles et leurs complétions dans les réseaux de Petri», *rapport de recherche n° RR-3970*, Irisa/INRIA Rennes, juillet 2000, `http://www.inria.fr/rrrt/rr-3970.html`.

[46] B. Caillaud, J.-P. Talpin, J.-M. Jézéquel, A. Benveniste, C. Jard, «BDL: A Semantics Backbone for UML Dynamic Diagrams», *rapport de recherche n° RR-4003*, Irisa/INRIA Rennes, septembre 2000, `http://www.inria.fr/rrrt/rr-4003.html`.

[47] J.-P. Talpin, A. Benveniste, B. Caillaud, P. Le Guernic, «Hierarchic Normal Forms for Desynchronization», *rapport de recherche n° RR-3822*, Irisa/INRIA Rennes, décembre 1999 1999, `http://www.inria.fr/rrrt/rr-3822.html`.