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

Le projet PAMPA s'est arrêté en juillet 2001 et a très fortement contribué à faire émerger de nouvelles équipes. Il s'agit d'abord de l'action, puis projet Triskell qui fait cette année un rapport d'activité séparé. Depuis juillet 2001, Claude Jard, responsable de Pampa a rejoint ce nouveau projet. Sont concernés ensuite les deux avant-projets Vertecs et S4 qui ont travaillé de façon assez autonome en 2001. L'appartenance des personnels à ces nouveaux projets est indiquée dans la liste. Ce rapport d'activité Pampa 2001 regroupe l'activité des anciens membres de Pampa de ces projets en cours d'officialisation.

Responsable scientifique

Claude Jard [DR CNRS]

Assistante de projet

Marie-Noëlle Georgeault [TR Inria, en congés de mars à Octobre 2001]

Christèle Soulas [CDD Inria, de mars à Octobre 2001]

Personnel INRIA

Benoît Caillaud [CR, responsable de l'avant-projet S4 depuis juillet 2001]

Thierry Jéron [CR, responsable de l'avant-projet Vertecs depuis juillet 2001]

Vlad Rusu [CR, Vertecs]

Personnel Université de Rennes 1

César Viho [maître de conférences, en délégation à l'INRIA jusqu'en octobre 2001, projet Armor depuis septembre 2001]

Ingénieurs-experts

Séverine Simon [ingénieur-expert Inria, jusqu'en février 2001]

Duncan Clarke [ingénieur-expert Inria/Dyade, jusqu'en juillet 2001]

Simon Pickin [ingénieur-expert Inria, à 50% partagé avec le projet Triskell]

Chercheurs doctorants

Sébastien Barbin [bourse MENRT, Armor depuis septembre 2001]

Lénaïck Tanguy [bourse Université Rennes I, jusqu'en octobre 2001]

Elena Zinovieva [bourse Inria/Dyade]

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. 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, et des techniques de distribution de code.

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. La prochaine génération de l'outil est en préparation et prend en compte des aspects symboliques (outil STG). 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.

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 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.

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

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

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.

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.

[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.

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 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.

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

Dans certains cas, nous enrichissons ce modèle d'états accepteurs pour leur faire jouer le rôle d'automate.

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

[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.

considérées comme observables dans la relation de conformité donc par un test. La seconde 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.

4 Domaines d'applications

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

Résumé : *le domaine d'application privilégié du projet est le logiciel pour les télécommunications. Dans ce secteur important, 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 : 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.*

Le test est une facette importante et pragmatique 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 [(correspondant)], 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.

L'année 2000 a vu se développer la connexion UMLAUT/TGV avec le soutien d'Alcatel et de Gemplus.

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.

En 2001, TGV a été connecté avec l'environnement IF de Verimag. Cette connexion a été utilisée sur une étude de cas dans le cadre du contrat IST Agedis. De nouveaux algorithmes sont en cours de réalisation dans TGV permettant de synthétiser des contrôleurs non bloquants sous observation partielle.

5.2 STG : un outil de génération de tests symboliques

Participants : Duncan Clarke, Thierry Jéron [(correspondant)], Vlad Rusu, Elena Zinovieva.

Mots clés : test, symbolique, IF.

Résumé :

STG *(Symbolic Test Generation) est un prototype de générateur de tests symboliques [15]. Cet outil nous sert pour l'instant de plate-forme pour implémenter les résultats de nos recherches sur le test symbolique et représente la génération future de* TGV. *Le logiciel STG a été développé selon les principes définis dans* [RdBJ00]. *Il compte 12000 lignes de C++, lex, et yacc, et utilise les librairies du langage IF développé à Verimag (Grenoble) et du système Omega (Univ. Maryland, USA). A partir d'une spécification et d'un objectif de test,* STG *produit un cas de test symbolique sous la forme d'un système de transitions étendu ou sous la forme d'un programme C++ prêt à être lié à une implantation à tester. Il a été utilisé pour un cas d'étude (un porte-monnaie électronique) [14] et pour une deuxième application carte à puces fournie par CP8-Schlumberger.*

[RdBJ00] V. Rusu, L. du Bousquet, T. Jéron, « An Approach to Symbolic Test Generation », *in: Integrated Formal Methods (IFM'00), Dagstuhl, Allemagne, LNCS, 1945*, Springer Verlag, p. 338–357, novembre 2000.

STG est implémenté en C++ et fait appel à l'outil de dessin de graphes DOTTY et, actuellement, au solveur de contraintes Omega (`http://www.cs.umd.edu/projects/omega`). Il prend en entrée des spécifications et des objectifs de test symboliques décrits dans une extension du langage IF de Vérimag. Sous certaines hypothèses sur la spécification, STG permet de calculer un *graphe de test* par manipulation syntaxique, résolution de contraintes locales et extraction de sous-graphe à la façon de TGV. Ce graphe de test n'est pas tout à fait un test, mais représente un ensemble de tests correspondant à l'objectif. La production d'un vrai test nécessitera de supprimer les transitions non tirables et d'améliorer la sélection des tests en utilisant des outils sémantiques tels que l'interprétation abstraite, la preuve, la propagation et résolution de contrainte, le calcul d'invariants.

STG permet aussi l'exécution des tests sur une implémentation. Actuellement, un test exécutable en C++ est produit à partir de sa représentation interne et peut être lié à une implémentation C++ du système à tester (développée par ailleurs). Différentes stratégies sont utilisables pour résoudre le non-déterminisme dans le code exécutable du test. Les paramètres de l'implémentation et du test doivent être fixés à l'exécution pour les rendre exécutables. Lors de l'exécution, la résolution de contraintes (Oméga) est utilisée pour résoudre les contraintes liées aux choix d'émission du testeur.

Nous avons déjà expérimenté STG sur quelques exemples significatifs dont une spécification partielle d'un porte-monnaie électronique (CEPS) de taille conséquente. Les graphes de test produits par manipulation syntaxique et extraction sont déjà extrêmement intéressants et fournissent déjà une aide certaine à la génération de tests. Ces études de cas nous montrent aussi le chemin à faire pour arriver à produire des tests plus proches de ceux qu'on voudrait écrire manuellement. Une deuxième étude de cas a été étudiée dans le cadre d'une collaboration avec CP8-Schlumberger. Il s'agit d'une spécification du système de fichiers de la Java-Card. Cette spécification décrite en NTIF (langage développé par le projet Vasy) a été compilée en IF puis utilisée par STG pour produire des cas de tests. Ces cas de tests ont été exécutés sur une implantation C++ partielle du système.

6 Résultats nouveaux

6.1 Test de conformité et d'interopérabilité

Participants : Sébastien Barbin, Duncan Clarke, Claude Jard, Thierry Jéron, 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 plusieurs 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*

2001, l'accent a été mis sur la génération de tests symboliques par l'utilisation de techniques symboliques, la relation entre génération de tests et vérification et entre génération de tests et synthèse de contrôleurs. Concernant l'exécution des tests, le prototype $\mathcal{D}-\mathcal{PIXIT}$ d'exécution de test répartis intégre à présent la gestion cohérente des timers ainsi que les améliorations des algorithmes de distribution automatique de tests. Notre travail sur le test d'interopérabilité a abouti cette année à la définition d'un cadre méthodologique formel pour le test d'interopérabilité.

Génération de tests symboliques Les algorithmes de TGV sont basés sur des techniques énumératives. Quand une spécification du système à tester contient des paramètres et/ou variables (paramètres de messages, compteurs, files d'attente...), TGV instancie toutes ces variables, qui seront traitées en "énumérant" les états de contrôle et les valeurs possibles des variables dans ces états en un système de transition à entrée/sortie (IOLTS). Ceci peut mener à une explosion combinatoire voire même à la non-terminaison si des variables sont non bornées. Pour résoudre ce problème, nous avons exploré des techniques de génération de tests dites *symboliques*. Les modèles de spécifications et d'objectifs de test sont des *iosts* permettant de décrire des systèmes paramétrés et manipulant des variables. Les cas de test sont produits, non plus par énumération des valeurs des variables (impossible en cas de spécifications paramétrées), mais par des calculs symboliques à partir des modèles de la spécification et de l'objectif de test. Dans [RdBJ00], nous avions défini formellement une méthode de génération de tests symboliques et prouvé que les cas de test produits sont corrects (non-biaisés et non-laxistes). Cette année, ce travail s'est poursuit par la réalisation d'un prototype appelé STG [15], l'étude d'applications cartes à puces dans le cadre d'une collaboration avec CP8-Schlumberger [14] et l'améioration de STG par l'utilisation de techniques de propagation/résolution de contraintes.

Génération de tests et vérification La vérification et le test sont des méthodes complémentaires qui tendent à assurer la correction des systèmes réactifs. Notre activité dans ces domaine comporte différents aspects.

Premièrement, il est possible d'utiliser des techniques de vérification pour aider la génération de tests, comme cela a été fait dans le passé pour TGV. Cette approche est aussi à la base de nos travaux sur le test symbolique dans l'outil STG, dans lequel des techniques de preuve sont utilisées pour simplifier les cas de test.

Deuxièmement, nous avons étudié des techniques de simulation symbolique pour une classe d'automates étendus (arithmétique de Presburger et symboles de fonction) [20] qui permettent de coder des classes de programmes avec des données non-bornées (entiers, vecteurs de taille paramétrique, canaux de communication non bornés). En choisissant un critère de couverture syntaxique dans un tel programme (par exemple, exécuter chaque instruction au moins une fois), critère syntaxiquement réalisé par un ensemble de séquences d'instructions, la simulation symbolique permet de détecter des valeurs initiales des variables qui font que ces séquences sont effectivement exécutables, et d'éliminer les séquences qui ne le sont pas. Les séquences

[RdBJ00] V. Rusu, L. du Bousquet, T. Jéron, « An Approach to Symbolic Test Generation », *in: Integrated Formal Methods (IFM'00), Dagstuhl, Allemagne, LNCS, 1945*, Springer Verlag, p. 338–357, novembre 2000.

(instanciées ou non) constituent des objectifs de test qui, fournies en entrée à STG (ou TGV, pour les séquences instanciées) permettront à l'outil de générer des tests qui couvrent la spécification du système sous test selon le critère de couverture qu'on s'est donné.

Troisièmement, nous contribuons par des travaux ponctuels au domaine de la vérification déductive, dans le but de rendre celle-ci plus automatique et en fin de compte utilisable dans le cadre du test [22, 21].

Enfin, nos travaux en cours portent sur l'utilisation de techniques de génération de test en vérification. L'idée est d'utiliser la sélection de sous-comportements de la spécification (des composantes fortement connexes du graphe symbolique) à l'aide d'objectifs de test, et de vérifier ces sous-comportements plutôt que la spécification entière. Sous certaines conditions suffisantes raisonnables, la correction de tels sous-comportements s'étend à la correction de la spécification entière.

Génération de tests et synthèse de contrôleurs Depuis quelques mois, et en relation avec la création du projet Vertecs, nous étudions les rapports entre génération de tests et synthèse de contrôleurs. La synthèse de contrôleur consiste, à partir d'une spécification d'un système réactif et d'un objectif de contrôle décrivant une propriété attendue, de produire un contrôleur qui contraint le système (supposé décrit par la spécification) afin qu'il satisfasse l'objectif. Dans le cadre énuméré pour l'instant, nous tentons d'uniformiser les modèles et de comparer et factoriser les techniques de génération de tests et de synthèse de contrôleurs. Le stage de DEA de Valery Tschaen a constitué un premier pas dans cette direction. Le problème de la synthèse de contrôleurs non bloquants sous observation partielle a été ré-exprimé et les algorithmes ont été implémentés en adaptant les algorithmes de TGV.

Synthèse de tests avec un modèle de vrai parallélisme Il s'agit de revoir la méthode TGV en fondant la représentation des comportements de la spécification non plus sur des automates séquentiels, mais sur des modèles de vrai-parallélisme. Ceci permet de synthétiser des cas de test parallèles utilisant le parallélisme intrinsèque de la spécification (test de systèmes multi-composants) et en définissant des objectifs de test eux-mêmes parallèles. Sous l'hypothèse que les causalités et la concurrence peuvent être observées lors de l'exécution du test (au prix d'une instrumentation des composants), on peut vérifier ainsi une relation de conformité "parallèle" plus fine que celle admise habituellement dans le domaine du test de conformité. Il est prévu de mettre en place une démonstration des concepts dans le cadre d'une modélisation UML. En 2001, nous avons écrit les principes de la synthèse fondés sur un jeu de "puzzle" tel que déjà utilisé dans le contexte du diagnostic réparti (cet article sera présenté lors de la conférence Testcom'2002 à Berlin l'année prochaine). Un prototype de démonstration utilisant l'environnement de dépliage de réseaux de Petri disponible à l'université de Munich est en cours de développement.

Génération de test réparti Nous avions travaillé sur une approche qui consiste à dériver automatiquement des testeurs répartis à partir de test centralisé. Nous avons proposé deux schémas de distribution permettant de produire un ensemble de tests communiquant entre eux de manière asynchrone. Cette année a été consacrée principalement à l'amélioration des algo-

rithmes de distribution et du prototype DaTeC (développé précédemment et qui implémente ces algorithmes, cf. rapport 2000). Ces améliorations concernent d'une part la gestion des asynchronismes et des procédures de coordination des testeurs, et d'autre part une meilleure gestion des timers dans le schéma de distribution basé sur un mécanisme de type round-robin. Nous avons expérimenté ces deux schémas de distribution sur une nouvelle suite de tests fournie par le CELAR [13]. Concernant la phase d'exécution des tests, la nouvelle version de $\mathcal{D}-\mathcal{PIXIT}$ (qui prend en entrée les sorties de DaTEC) intègre les changements indiqués ci-dessus pour DaTEC.

Un cadre formel pour le test d'interopérabilité Les travaux de cette année ont abouti à la définition d'un cadre méthodologique formel pour le test d'interopérabilité [23]. Nous avons montré que les concepts du test de conformité peuvent être utilisés pour définir un cadre méthodologique pour le test d'interopérabilité. Dans un premier temps, nous avons réalisé un état de l'art présentant les différentes architectures possibles de test d'interopérabilité. Puis nous avons défini plusieurs *relations d'interopérabilité* basées sur les relations d'implémentation/conformité définies pour le test de conformité. Chaque relation d'interopérabilité spécifie formellement les conditions que doivent satisfaire les implémentations afin d'être considérées comme interopérables. Nous avons effectué une comparaison de ces relations en terme de pouvoir de détection de non-interopérabilité. Actuellement, nous étudions des pistes pour la génération automatique de tests d'interopérabilité.

6.2 Modèles d'ordres partiels

Participants : Benoît Caillaud, Claude Jard.

Mots clés : ordres partiels, scénarios, HMSC, synthèse, réseaux de Petri, diagnostic.

Résumé : *Nous développons des modèles fondés mathématiquement sur des familles d'ordres partiels afin de capturer les phénomènes de concurrence et de distribution. Ces travaux ont débouchés sur des algorithmes de réalisation sur des architectures réparties de spécifications hétérogènes (automates, langages, scénarios) et trouvent des applications en particulier sur les langages de scénarios comme les HMSC, langage de spécification normalisé par l'ITU et en partie intégré à la notation UML.*

Application de la synthèse de réseaux de Petri à la génération de squelettes de programmes répartis

Les avancées récentes sur le problème de la synthèse de réseaux de Petri permettent maintenant le développement de techniques nouvelles en matière de synthèse de programmes répartis. Le problème de synthèse est de produire un réseau de Petri borné dont le graphe des marquages est isomorphe à un système de transition donné. Cette relation peut être affaiblie à l'égalité des langages de l'automate et du réseau. Les algorithmes de synthèse de réseaux reposent sur le concept de régions dans les systèmes de transition et ramène la synthèse d'un réseau à la résolution d'un ensemble de programmes linéaires en nombres rationnels.

Nous avons amélioré les algorithmes de synthèse de réseaux de Petri bornés répartissables mis en œuvre dans l'outil SYNET (`http://www.irisa.fr/pampa/logiciels/synet/synet.html`). Deux optimisations on été étudiées : la première consiste à ne considérer que les rayons extrémaux du polyèdre des régions, alors que la seconde repose sur l'utilisation de la bibliothèque de programmation linéaire CPLEX commercialisée par Ilog. L'outil Synet permet la répartition d'automates réactifs en réalisant à la fois une extraction de la concurrence et le traitement de choix non localisés [7]. Les domaines d'applications envisagés sont la synthèse de protocoles de communications, de squelettes de programmes répartis et de contrôleurs de systèmes à événements discrets. Ce dernier sujet a été abordé dans le cadre de l'ARC MARS (décrite au paragraphe 8.1).

Langages de scénarios

L'usage de scénarios pour spécifier les comportements de systèmes est une pratique bien établie. Des notations formelles ont été proposées comme les "High level Message Sequence Charts" (HMSC), principalement utilisés pour décrire l'activité de processus communicants dans le domaine des protocoles. On les retrouve sous une forme proche, les diagrammes de séquence, dans une des vues définies par la notation UML.

Un certain nombre de résultats d'indécidabilité sur les HMSCs (rassemblés dans [10]) indiquent clairement que le seul angle sous lequel on peut raisonnablement considérer et utiliser les HMSCs comme des spécifications de comportements est d'interpréter leurs extensions linéaires comme des langages minimaux à approximer supérieurement dans toute réalisation. Le problème est alors de rechercher un cadre dans lequel on puisse donner une signification précise à ces spécifications incomplètes au moyen d'une opération de fermeture et en particulier la fermeture des langages de HMSCs dans les langages de réseaux de Petri [10]. Cette opération de fermeture correspond à une procédure effective, reposant sur la semi-linéarité des images commutatives des langages de HMSCs, et repose sur les mêmes techniques d'algèbre linéaire que celles utilisées dans l'outil de synthèse de réseaux de Petri SYNET. Cette approche permet également de produire des systèmes d'automates communicants à partir d'un HMSC et a donné sur plusieurs cas de meilleurs résultats que les techniques classiques de synthèse décrites dans la litterature.

Nous avons aussi défini une sémantique d'ordre partiel pour des familles de scénarios représentés par des HMSC. Celle-ci permet d'associer à chaque HMSC une structure d'événements engendrable par une grammaire de graphes finie [12].

Puzzle de Viterbi

(ce travail est mené en collaboration avec l'équipe Sigma2)

Poussés par la question du diagnostic en environnement réparti qui se pose régulièrement dans toutes les études que nous menons dans le projet Pampa, nous avons essayé d'utiliser les automates d'ordres comme un outil pour l'observation des comportements. Dans ce cadre, les dépendances causales attendues sont exprimées par un jeu de pièces (des motifs d'ordre de base) et doivent être confrontées aux dépendances observées.

Le travail est alimenté par l'application au suivi de pannes dans les réseaux de télécommunication (cf. module 7.1, projet MAGDA).

En 2001, Pampa a principalement participé à la définition des algorithmes vus comme la construction guidée de dépliages de réseaux de Petri à partir de jeux de pièces.

7 Contrats industriels (nationaux, européens et internationaux)

7.1 Diagnostic de pannes dans les réseaux de télécommunications (RNRT Magda et Magda2 : Nov. 1999 - Nov. 2003)

Participant : Claude Jard.

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. En 2001, l'apport du projet Pampa a porté principalement sur la définition des algorithles en terme de dépliages guidés de réseaux de Petri. Un deuxième projet exploratoire RNRT dans la suite de Magda (Magda2) démarre en novembre 2001.

7.2 FormalFame : Validation d'architectures multi-processeurs (Mar. 1999 - Mar. 2001)

Participant : César Viho.

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 qui s'est terminée cette année, il s'agissait de fournir des méthodes et outils pour la vérification formelle et le test de protocoles de cohérence de caches pour des architectures multiprocesseurs développées chez Bull. Ce travail a été 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. Dans ce cadre, nous appliquons nos méthodes et outils de vérification/validation et de test 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 a porté ensuite sur une architecture à base de processeurs McKinley (deuxième génération). Nous avons mené 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'expérience acquise dans les expérimentations précédentes est réutilisée [11].

7.3 FormalCard : Validations formelles d'applications embarquées sur cartes à puces (Mar. 2000 - Nov. 2002)

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

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

Résumé : *Cette action commencée avec CP8-Bull dans le cadre de Dyade, avec les équipes Vasy de l'INRIA Rhône-Alpes et Pampa de l'IRISA, se poursuit par un contrat direct avec CP8-Schlumberger. 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, ...). Ce langage appelé NTIF est compilé dans le format IF utilisé par le prototype STG. L'outil STG est employé pour générer des tests de conformité avec des techniques de génération de tests symboliques. Une première étude de cas sur une spécification de porte monnaie électronique (CEPS) a été décrite en IF, des cas de tests ont été produits et exécutés sur une implémentation partielle. Une seconde spécification fournie par CP8 concernant le système de fichier de la Java-Card a été spécifiée en NTIF, compilée en IF. Des cas de tests ont été produits puis exécutés sur une implémentation.

7.4 AEE : Architecture Electroniques Embarquées (Mar. 1999 - Mar. 2003)

Participant : 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 Meïje 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. Nos modèles et l'outil TGV ne sont pas directement applicables mais . nous envisageons une adaptation permettant de contourner ces problèmes.

7.5 Agedis (Mar. 2000 - Nov. 2003)

Participants : Thierry Jéron, Vlad Rusu.

Mots clés : Test, profil UML, TGV..

Résumé : *Ce projet vise à fournir des méthodes et outils pour l'automatisation du test de logiciels, et en particulier pour les logiciels répartis, principalement du domaine des télécommunications. Le projet consiste en la définition d'un profil UML, appelé AML suffisant pour les applications visées, la compilation d'AML dans le langage IF développé par Verimag, la génération de tests basé sur les techniques de Gotcha (IBM) et* TGV *et leur exécution. Nous y collaborons avec IBM Haifa (Israël) et IBM Hursley (G.-B.), France Télécom R& D (Lannion), IntraSoft (Grèce), Imbus (Allemagne), l'université d'Oxford et Vérimag Grenoble. Nous sommes sous-contractants de Verimag dans ce projet.*

AML, le profil UML est en cours de définition. Il comprend essentiellement les diagrammes de classes et d'objets, les statecharts. Une description AML ainsi que des critères de tests sont ensuite compilés dans une extension du langage IF en cours de définition. La description IF est

compilée en un code de simulation utilisé par le générateur de tests. Celui-ci utilise également des critères de test de plusieurs natures : des objectifs de test à la façon de TGV et des critères de couvertures à la façon de GOTCHA. Notre challenge, en collaboration avec Verimag, est d´étendre TGV afin qu'il prenne en compte les modèles et les techniques de GOTCHA.

7.6 Méthodologie de tests d'interopérabilité (Nov. 1998 - Nov. 2001)

Participants : César Viho, Lénaïck Tanguy, Sébastien Barbin, 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 considère 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.

Les travaux menés dans ce cadre ont abouti à :

– un état de l'art complet présentant les différentes architectures et méthodes possibles de test d'interopérabilité,
– la définition d'un cadre méthodologique formel pour le test d'interopérabilité autorisant la génération et l'exécution des tests de manière répartie.

7.7 COTE (Nov. 2000 - Nov. 2002)

Participants : Claude Jard, Simon Pickin, Thierry Jéron.

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

Résumé :

Ce projet fait collaborer deux laboratoires de recherche (Irisa/Pampa, Triskell 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 et Triskell amènent leurs compétences sur UMLAUT/TGV et participent 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. L'année 2001 a été consacrée à définir un langage d'expression des tests à l'intérieur de la notation UML[19] et spécifier l'intégration des outils.

8 Actions régionales, nationales et internationales

8.1 Actions nationales

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. L'outil SYNET a notamment été utilisé par Xiaolan Xie, Nydhal Reszg et Asma Ghaffari de l'INRIA-Lorraine pour synthétiser des contrôleurs assurant la vivacité de réseaux de Petri bornés. Cette action s'est clôturée par l'organisation du workshop SCODES'2001 sur le contrôle de supervision des systèmes à événements discrets [7, 6] — Paris, 23 juillet 2001, organisé par Benoît Caillaud et Xiaolan Xie, workshop satellite de la conférence CAV'01.

8.1.1 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.

Résumé : *Notre objectif est d'étendre notre savoir-faire en matière de tests de conformité au contexte du test d'interopérabilité des nouveaux protocoles des réseaux*

informatiques, avec la prise en compte des aspects quantitatifs liés à la QoS. Il s'agit de développer de nouvelles méthodes, de nouveaux algorithmes et outils appropriés, et de les implémenter dans des plate-formes d'expérimentation.

La première année a été consacrée à la génération de 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 en 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. Cette année, en collaboration avec le projet Armor, nous avons principalement étudié les protocoles de mobilité IPv6 (MIPv6) et la QoS au dessus de IPv6 tels que Diffserv et RSVP. Nous avons adapté des outils existants pour IPv4 de façon à les rendre opérationnels au dessus de IPv6. Nous avons ainsi généré des tests de conformité et d' interopérabilité pour ces protocoles. Ces tests seront utilisés au cours de l'exposition (appelé maintenant plugtest) 2001 organisé par l'ETSI Sophia du 19 au 23 novembre 2001. Par la suite, nous proposons d'étendre le savoir-faire en matière de génération de tests de conformité et d'interopérabilité 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.

8.1.2 Groupe de travail test et objets

Participants : Claude Jard, Thierry Jéron.

L'équipe Pampa a animé 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 (Triskell) en est le secrétaire. Une dizaine d'équipes en France participent. La page Web `http://www.irisa.fr/testobjets` peut être consultée. En fin 2001, ce groupe a permis de faire reconnaître une "action spécifique" du CNRS sur le test des systèmes complexes.

8.1.3 Action coopérative FISC : formalisation et instrumentation des scénarios

L'ARC FISC (`http://www.irisa.fr/pampa/arc-fisc/`) est une action de recherche coopérative de l'INRIA lancée en janvier 2001 et d'une durée de deux ans. Elle regroupe un ensemble d'équipes autour d'un thème commun : les langages de scénarios pour les télécommunications et l'automatique. Quinze chercheurs répartis dans quatre laboratoires participent à l'action. Les laboratoires impliqués sont : l'IRISA (Rennes), le LIAFA (université Paris 7), l'INRIA Rocqencourt et l'INRIA Rhône-Alpes (Grenoble). Benoît Caillaud en est le coordinateur.

8.2 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 un séjour de un mois au SRI dans ce cadre.

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). Il a effectué une visite en Pologne en novembre 2000 et y a donné deux séminaires, l'un à l'IPIPAN, Varsovie, et l'autre à l'université de Gdansk.

9 Diffusion de résultats

9.1 Animation de la communauté scientifique

Claude Jard a co-édité un livre de la série LNCS Tutorial [8], construit à partir de révisions des cours donnés lors de l'école Movep'2002 à Nantes. Il co-organise l'édition 2002.

Claude Jard a participé en 2001 aux conférences suivantes en tant que membre du comité de programme :

- Tacas'2001 : Tools and Algorithms for the Construction of Systems, Etaps, Genova, avril 2001,
- Afald'2001 : Atelier sur l'assistance formalisée au développement logiciel, Nancy, mars 2001,
- MSR'2001 : Modélisation des Système Réactifs, Toulouse, Octobre 2001.

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. A ce titre, président de la commission locale des postes d'accueils Inria et membre de la commission nationale des détachements Inria. Il est membre de l'équipe de direction de l'Irisa. Claude Jard est membre 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. Il est récemment membre du conseil scientifique du département STIC du CNRS.

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.

Claude Jard fait partie du conseil stratégique de la société TNI-Valiosys.

Thierry Jéron est membre de la commission de spécialistes de l'ENS Cachan et depuis 2001 de celle de l'Ifsic.

Thierry Jéron a fait partie du comité de programme de Fates2001 : Formal Approaches for Testing, workshop satellite de Concur2001, Aalborg, Danemark, août 2001. Il co-organise Fates2002.

Thierry Jéron est membre élu au Conseil de Laboratoire de l'Irisa.

Benoît Caillaud a co-organisé le workshop SCODES'2001 sur le contrôle de supervision des systèmes à événements discrets [7]. Ce worskhop était un événement satellite de la conférence CAV'01 (18–23 juillet 2001, Paris) et il a rassemblé plus de trente participants.

Benoît Caillaud est co-éditeur d'un livre sur la synthèse et le contrôle de systèmes à événements discrets [6]. Ce livre a été édité à partir de présentations à deux workshops organisés pendant l'été 2001 : le workshop SCODES'2001 [7] et le *worskhop on synthesis of concurrent systems*, satellite de la conférence ICATPN'01, Newcastle upon Tyne, Royaume Uni.

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 coordinateur du G6test, le sous-groupe du G6 (Groupe francophone des utilisateurs de IPv6) qui s'occupe de test de conformité et d'interopérabilité des protocoles IPv6. Il co-organise le "bake-off" (plugtest 2001) de tests d'interopérabilité des protocoles IPv6 qui a lieu à l'ETSI Sophia du 19 au 23 novembre 2001.

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.

Vlad Rusu a donné un séminaire invité a France Télecom R& D, intitulé "Verifying that Invariants are Context-Inductive"

9.2 Enseignement universitaire

Benoît Caillaud est responsable du module "analyse comportementale des systèmes réactifs et répartis" (A2R) de la filière "génie logiciel et méthodes formelles" du DEA d'informatique de l'université de Rennes 1. Il a également été responsable de cette filière de septembre 1999 à août 2001.

Claude Jard est responsable 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.

Thierry Jéron enseigne la validation à l'ENSTB Brest.
Vlad Rusu a donné un cours-conférence en DEA sur l'utilisation du prouveur PVS et est responsable d'un nouveau module sur le même sujet à la rentrée 2001.

César Viho anime un cours sur les "mécanismes des réseaux informatiques" dans les filières LSI, ARC, INC et TST du DIIC ainsi qu'en formation continue (FUTE/THOMSON). Il est également responsable du module "Validation et test de protocoles" dans la filière Réseau du DESS-ISA de l'Ifsic.

9.3 Participation à des colloques, séminaires, invitations

Claude Jard a été invité à donner les séminaires suivants :
- "Executing High-level Message Sequence Charts", Dagstuhl, Allemagne, Novembre 2000.
- "La manipulation formelle de scénarios pour les systèmes répartis", MSR'2001, Toulouse, Octobre 2001.
- "Using true-concurrency models to synthetize tests from formal specifications", CPWC-SE'2001, Montréal, Septembre 2001.

Claude Jard et Simon Pickin ont animé un stand au colloque national RNTL en avril 2001.

Benoît Caillaud a donné une présentation invitée intitulée *SYNET : P/N synthesis at work* dans le *workshop on synthesis of concurrent systems*, juin 2001, Newcastle upon Tyne, Royaume Uni.
Benoît Caillaud a donné les séminaires suivants en novembre 2000 :
- *Distributing finite automata through P/N synthesis*, à l'université de Gdansk, Pologne.
- *BDL : a semantics backbone of UML dynamic views*, à l'IPIPAN, Varsovie, Pologne.

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] 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.

[5] 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.

Livres et monographies

[6] B. Caillaud, P. Darondeau, L. Lavagno, X. Xie (éditeurs), *Synthesis and Control of Discrete Event Systems*, Kluwer Academic Press, 2001, To appear.

[7] B. Caillaud, X. Xie (éditeurs), *Proceedings of the Symposium on the Supervisory Control od Discrete Event Systems*, INRIA, Paris, July 2001.

[8] F. Cassez, C. Jard, B. Rozoy, M. Ryan (éditeurs), *Modeling and Verification of Parallel Processes, revised Tutorial Lectures*, *Lecture Notes in Computer Science, 2067*, Springer Verlag, 2001.

Articles et chapitres de livre

[9] E. Badouel, P. Darondeau, B. Caillaud, « Distributing Finite Automata through Petri Net Synthesis », *Journal on Foundamental Aspects of Computer Science*, 2001, to appear.

[10] B. Caillaud, P. Darondeau, L. Hélouët, G. Lesventes, « HMSCs as specifications... with PN as completions », *in : Modeling and Verification of Parallel Processes*, F. Cassez, C. Jard, B. Rozoy, et M. Dermot (éditeurs), *Lecture Notes in Computer Science, 2067*, Springer, 2001, p. 125–152.

[11] H. Garavel, C. Viho, M. Zendri, « System design of a CC-NUMA multi-processor architecture using formal specification, model-checking, co-simulation and test generation and execution », *International Journal on Software Tools for Technology Transfer - Special Issue Vol. 5/Issue 2*, juillet 2001.

[12] L. Hélouët, C. Jard, B. Caillaud, « An Event Structure Semantics for Message Sequence Chart », *Mathematical Structures in Computer Science*, 2001, To appear.

Communications à des congrès, colloques, etc.

[13] S. Barbin, L. Tanguy, C. Viho, « Distributed testing using a round-robin mechanism », *in : AFRICOM'2001, 5th International Conference on Communication Systems*, K. Koen, P. Kritzinger, P. Aspinall (éditeurs), IFIP, p. 9–25, Cape Town, South Africa, may 2001.

[14] D. Clarke, T. Jéron, V. Rusu, E. Zinovieva, « Automated Test and Oracle Generation for Smart-Card Applications », *in : e-Smart 2001, International Conference on Research in Smart Cards, LNCS 2140*, 2001.

[15] D. Clarke, T. Jéron, V. Rusu, E. Zinovieva, « STG : A Tool for Generating Symbolic Test Programs and Oracles from Operational Specifications », 2001. Poster at 8th European Software Engineering Conference (ESEC).

[16] L. Hélouët, C. Jard, « La manipulation formelle de scénarios pour les systèmes répatis : l'exemple des HMSC », *in : Modélisation des Systèmes Réactifs*, G. Juanole, R. Valette (éditeurs), Hermes, Toulouse, Octobre 2001.

[17] C. Jard, « How to simulate HMSC », *in : SDL'01 : Meeting UML, 10th SDl Forum*, LNCS, Springer Verlag, Copenhagen, June 2001. Tool demo session.

[18] C. Jard, « Using True-concurrency Models to Synthetize Tests from Formal Specifications », *in : Proceedings of the Concordia Prestigious Workshop on Communication Software Engineering*, R. Dssouli, F. Khendec (éditeurs), Concordia university, Montréal, September 2001.

[19] S. Pickin, C. Jard, T. Heuillard, J.-M. Jézéquel, P. Desfray, « A UML-integrated test description language for component testing », *in : Practical UML-based Rigorous Development Methods*, A. Moreira (éditeur), Toronto, October 2001.

[20] V. Rusu, E. Zinovieva, « Analyzing Automata with Presburger Arithmetic and Uninterpreted Function Symbols », *in : Workshop on Verification of Parameterized Systems (VEPAS'01)*, *ENTCS*, *50*, 4, 2001.

[21] V. Rusu, « Verifying a Sliding-Window Protocol using PVS », *in : Formal Description Techniques, FORTE'2001*, IFIP, Corée, 2001.

[22] V. Rusu, « Verifying that Invariants are Context-Inductive », *in : Theorem Proving in Higher-Order Logics (TPHOLs'01)*, p. 337–351, University of Edinburgh, Informatics Research Report EDI-INF-RR-0046, 2001.

[23] C. Viho, S. Barbin, L. Tanguy, « Towards a formal framework for interoperability testing », *in : 21st IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems*, M. Kim, B. Chin, S. Kang, D. Lee (éditeurs), p. 53–68, Cheju Island, Korea, aug 2001.

Rapports de recherche et publications internes

[24] V. Rusu, « Analyzing Automata with Presburger Arithmetic and Uninterpreted Function Symbols », *rapport de recherche n°4100*, INRIA, 2001, `http://www.inria.fr/rrrt/rr-4100.html`.