La chasse aux bogues grâce à l'analyse statique de code

Jean-Jacques Lévy nous explique comment ces outils de vérification de logiciels ont conquis le monde de l'espace et de l'avionique en 10 ans.

INédit : Tout commence avec l'explosion d'Ariane 5 lors de son premier vol, le 501. Racontez-nous ?

Jean-Jacques Lévy : C'était en juillet 1996. La nouvelle version d'Ariane explosait 39 secondes après son lancement. Les ingénieurs du Cnes ont vite compris qu'il s'agissait d'un problème logiciel, un type d'erreur jamais envisagé jusque là. Le bogue venait du système de navigation qui avait détecté une « exception » logiciel due à un dépassement de capacité lors d'une conversion d'un nombre flottant en nombre entier. Le morceau de programme concerné, hérité d'Ariane 4, était en fait inutile pour Ariane 5 et n'avait pas été protégé... La commission d'enquête, dirigée par Jacques-Louis Lions, président de l'Académie des sciences, a demandé à Gilles Kahn, directeur scientifique de l'INRIA, de faire expertiser tout le code. C'est ainsi que nous avons réuni une petite équipe d'experts de la programmation et de la théorie des langages de programmation : Alain Deutsch, décédé prématurément l'été dernier à 41 ans, Damien Doligez, Robert Ehrlich, Georges Gonthier, François Rouaix, Marcin Skubiszewski et moi-même.

INédit : Vous avez donc passé l'ensemble des logiciels au peigne fin. Avec quels outils ?

Jean-Jacques Lévy : Cela représentait 140 000 lignes de code en ADA, un langage de haut niveau ! Nous avons d'abord mis ce code sous Unix et nous avons réussi à le compiler en entier. Puis, grâce à l'analyseur de code IABC qu'Alain Deutsch développait depuis 10 ans, le meilleur au monde, nous avons vérifié les « conditions de Bernstein », c'est-à-dire les accès aux variables partagées par plusieurs tâches concurrentes, analyse qui repose sur la recherche d'alias. Georges Gonthier a ainsi découvert plusieurs erreurs de logique dans le programme de vol à partir des représentations graphiques que procurait l'analyseur. C'était la première fois qu'une analyse statique de programmes était utilisée à cette échelle. La démonstration de l'intérêt de la démarche pour vérifier la sûreté de logiciels embarqués critiques était faite.

INédit : Êtes-vous du même coup devenu des experts en programmation aérospatiale ?

Jean-Jacques Lévy : En quelque sorte. Quelques uns d'entre nous ont participé à la commission de qualification du vol 502, qui a été un succès. Par la suite, le CNES soumettait chaque nouvelle version de programme au logiciel d'Alain. Avec Daniel Pilaud, ils ont créé une entreprise, Polyspace Technologies, en 1999 pour détecter de façon automatique les erreurs d'exécution des logiciels. Ils ont ainsi vérifié presque tous les codes embarqués des engins spatiaux d'EADS comme l'ATV (*Automated transfer vehicle*), véhicule ravitailleur de la station spatiale internationale (ISS) et l'ARD (*Atmospheric Re-entry Demonstrator*). Le succès s'est étendu aux industries automobile, de défense, de l'énergie… Plus tard, nous avons été chargés par l'agence spatiale européenne (ESA) de la revue du code embarqué du module européen (Columbus) de l'ISS et nous avons contribué à la définition des nouvelles règles de programmation de l'espace.

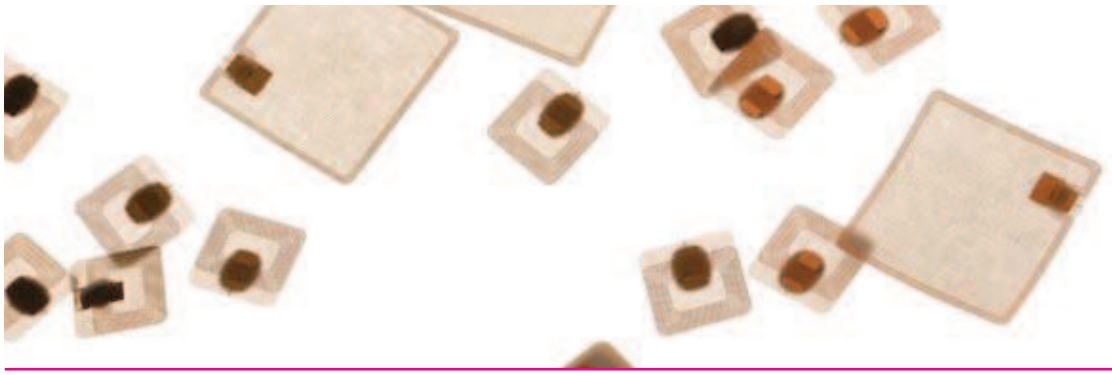
INédit : Autrement dit, l'analyse statique de code a gagné ses galons dans l'industrie ?

Jean-Jacques Lévy : Assurément, ce regard d'informaticiens et ces outils théoriques de vérification de logiciels sont passés dans les moeurs. Ils sont désormais considérés à leur juste valeur pour détecter les vices cachés de conception. Des progrès plus récents ont été faits pour le code embarqué de l'A380. ■

CONTACT

Jean-Jacques Lévy, *équipe-projet Moscova, Centre de recherche INRIA de Paris - Rocquencourt, directeur du centre de recherche commun INRIA-Microsoft Research*

Tél. : +33 1 39 63 56 89, Jean-Jacques.Levy@inria.fr



Le futur est déjà asynchrone

Une page se tourne : l'industrie rompt avec le modèle synchrone.

Les premiers ordinateurs étaient naturellement asynchrones, conduisant aux problèmes d'interblocages, de famine, de conflits d'accès aux ressources partagées... Pour éviter ces problèmes, les architectes ont privilégié une approche plus simple dans laquelle toutes les parties d'un circuit devaient se synchroniser sur une horloge unique. Ce compromis a longtemps paru acceptable : la quasi-totalité des circuits était synchrone, ainsi que les outils de CAO utilisés pour les concevoir ; l'augmentation continue des fréquences d'horloge compensait la perte de performance due à l'horloge unique.

La situation est en passe de changer radicalement. Au niveau macroscopique, on assiste à l'émergence des infrastructures distribuées (Internet, grilles, systèmes embarqués communicants) qui structurent le monde de manière asynchrone. Au niveau microsco-

pique, la course aux performances impose l'abandon du modèle synchrone dans les circuits à cause de ses inconvénients : consommation élevée, surface de silicium requise pour l'horloge, émissions radioélectriques permettant d' « espionner » le fonctionnement du circuit... Ne pouvant augmenter indéfiniment les fréquences d'horloge, les constructeurs s'orientent vers davantage d'asynchronisme : processeurs multi-cœurs, architectures GALS (Globalement Asynchrones, Localement Synchrones), logique asynchrone... Les différentes parties des systèmes fonctionnent à des vitesses différentes et peuvent même être arrêtées lorsque leur utilisation n'est pas nécessaire.
La même évolution se retrouve dans d'autres domaines traditionnellement acquis à l'approche synchrone. Dans les voitures, la multiplicité des calculateurs embarqués rend impossible ou trop coûteux de les synchroniser pas à pas. C'est aussi le cas dans les avions où l'avionique modulaire distribuée (DMA) tend à remplacer l'avionique modulaire intégrée (IMA).
Cette évolution vers plus de décentralisation est toutefois freinée par des considérations de fiabilité. En effet, les systèmes asynchrones sont complexes à concevoir à cause de la présence d'acteurs et d'événements simultanés, ce qui bouleverse les habitudes de programmation et requiert des outils spécifiques d'aide à la conception asynchrone qui font encore défaut.
L'équipe Vasy s'attaque à ce problème en utilisant des méthodes formelles pour modéliser les systèmes asynchrones et des techniques de vérification énumérative. Pour limiter le nombre de cas dont l'importance excèderait la mémoire de l'ordinateur ou prendrait trop de temps, ces chercheurs combinent « force brute » — explorer toutes les combinaisons possibles — et stratégies « intelligentes ». Leurs résultats sont mis en œuvre dans la boîte à outil CADP (voir Inédit n°58) utilisée, entre autres, dans le projet Multival du pôle de compétitivité Minalogic et le projet Topcased du pôle AESE. ■

CONTACT
Hubert Garavel, *équipe-projet Vasy, Centre de recherche de l'INRIA Grenoble - Rhône-Alpes*
Tél. : + 33 4 76 61 52 24, Hubert.Garavel@inria.fr

Conjuguer les approches pour générer des tests adaptés

Les méthodes de test basées sur des modèles de spécifications* (model-based testing) *et celles basées sur le code (test structurel) se rejoignent pour créer de nouveaux outils de génération de test.

S'assurer qu'un système informatique fait bien ce que son concepteur avait prévu qu'il fasse est une étape du génie logiciel d'autant plus décisive que les systèmes concernés sont embarqués ou critiques (voir article Ariane 5). Dans un système embarqué, les difficultés liées à la validation et au test sont décuplées : les critères temps-réel sont beaucoup plus contraignants que dans des réseaux de télécommunication par exemple, et le nombre et la taille des domaines des variables à prendre en considération est beaucoup plus grand. Impossible, dans ces conditions, de tester exhaustivement des programmes par l'énumération des valeurs possibles des variables. Une solution, développée ces dix dernières années, consiste à traiter les variables de manière symbolique en représentant des ensembles de vecteurs de valeurs de variables par des formules. Il est possible ensuite d'appliquer des techniques de propagation et résolution de contraintes et, plus récemment, des méthodes d'interprétation abstraite des modèles capables de guider la génération de tests.

Pour les chercheurs de l'équipe Vertecs, l'avenir est sans aucun doute à la combinaison de ces méthodes. Ils ont ainsi conçu un outil original, STG (*Symbolic Test Generator*) qui sélectionne des tests à partir de modèles de systèmes réactifs et d'objectifs de tests décrivant des comportements abstraits à tester. La sélection des tests s'appuie sur une analyse approchée par interprétation abstraite pour renforcer les contraintes sur les entrées du modèle afin de satisfaire l'objectif. Ces contraintes sont ensuite résolues à l'exécution des tests sur le système réel. La démarche est complémentaire de techniques basées sur la couverture de code (instructions, branches, etc) issues du test structurel. Elle peut en effet servir de module de base pour la sélection de tests basée sur la couverture, et compléter celle-ci en ciblant des comportements à risques. Les évolutions à venir portent sur la complexité des modèles (taille, structure de contrôle, hétérogénéité des données), et sur l'adaptation des critères de couverture au test basé sur des modèles qui, contrairement au test structurel, n'ont pas toujours la même structure que le code.
Pour les chercheurs, STG est un exemple de la faisabilité de cette approche dans le cas de modèles de programmes réactifs manipulant des variables booléennes et numériques. Il est actuellement distribué et, s'il donne satisfaction, les communautés du test basé modèle et du test structurel auront tout à gagner à travailler ensemble malgré les différences de culture et de langage. ■

CONTACT
Thierry Jéron, *équipe-projet Vertecs, centre de recherche INRIA Rennes - Bretagne Atlantique*
Tél. : +33 2 99 84 74 64, Thierry.Jeron@inria.fr