**Nom du Projet** : Tests Automatiquement Générés pour Ada
**Acronyme** : TAGAda
**Porteur** : AdaCore (Claire Dross, dross@adacore.com)
**Partenaire** : Thales Research & Technology
**Financement** : Dispositif RAPID - DGA
**Dates** : Du 01/09/2021 au 31/08/2023
**TRL** : De 3 à 4-5 selon les lots

Le logiciel tient une place prédominante au sein des systèmes critiques, qu'ils soient civils ou militaires. Les défaillances et vulnérabilités dans ces logiciels peuvent avoir des conséquences dramatiques en termes humains, matériels ou environnementaux. Le langage Ada est largement utilisé dans le développement de ces logiciels. Il est donc primordial d'être en mesure d'assurer un haut niveau de qualité et de sécurité aux logiciels développés en Ada. Quelles que soient les techniques de validation et vérification utilisées, la méthode incontournable pour révéler des fautes à l'exécution reste le test.

Tester un programme consiste à l'exécuter sur un ensemble fini de valeurs, de manière à détecter des divergences entre son comportement réel et le comportement attendu. Bien que le test soit la méthode de validation du logiciel la plus utilisée, les tests sont écrits à la main la plupart du temps. L'écriture de ces tests, ainsi que leur maintien, est l'un des points difficiles et coûteux pour les équipes de vérification et de validation. Des avancées récentes dans la génération automatique de tests ont montré leur efficacité en augmentant la couverture du code et la capacité des tests à détecter des fautes et des vulnérabilités.

On s'intéressera principalement dans ce projet aux trois approches suivantes : le property-based testing, qui génère des données de test valides pour un programme en fonction des types de ses arguments, et vérifie que les sorties satisfont les propriétés attendue ; le fuzz testing, principalement utilisé pour détecter des failles de sécurité, qui produit de très nombreuses données de test par mutations aléatoires de données valides, de façon à maximiser la proportion du code exécutée par les tests ; l'exécution symbolique, qui s'appuie sur une analyse mathématique du code pour produire des données de test ciblées sur certaines instructions ou branches spécifiques. Ces trois techniques, parce qu'elles suivent des approches différentes, peuvent être utilisées de façon complémentaire.

Ces techniques existent, de manière séparée, pour plusieurs langages. Le code Ada, largement utilisé dans l'industrie critique, n'en bénéficie pas. Nous voulons étudier, parmi ces avancées, celles qui sont les plus pertinentes dans le contexte d'un développement logiciel industriel basé sur ce langage. Comme les types de données du langage Ada sont très contraints, la génération de données guidée par ces types peut rapidement produire des tests pertinents. De plus, la possibilité d'écrire les propriétés attendues des programmes directement dans le langage, sous forme de contrats, permettra d'affiner le verdict des tests. Ainsi, il sera possible de détecter à la fois des vulnérabilités du code et des non conformités par rapport aux contrats, qui décrivent une partie des exigences de bas niveau.

Nous pensons que la combinaison de différentes stratégies de génération permettra d'atteindre une large couverture des codes sources étudiés avec un effort minimal des équipes de vérification et de validation. Le but du projet est d'améliorer la qualité, la sûreté et la sécurité des logiciels critiques écrits en Ada utilisés dans l'industrie civile et militaire, en maximisant la détection d'erreurs dans le code source. La réalisation de ce but passera par : l'identification des meilleures stratégies de génération de données de test existantes ; la mise en œuvre de ces techniques dans un démonstrateur intégré aux outils dédiés à Ada ; l'automatisation de la génération de tests exécutables ; la fourniture de mesures de couverture par les tests générés.