

## Annexe B: Exemples d'utilisation industrielle

26 novembre 2009

Le projet HI-LITE a comme objectif de fournir des solutions appropriées à des utilisateurs ayant différents besoins de vérification. Pour concrétiser cette vision d'une utilisation plurielle, nous décrivons dans la suite le cas de trois entreprises fictives A, B et C, utilisant la technologie HI-LITE.

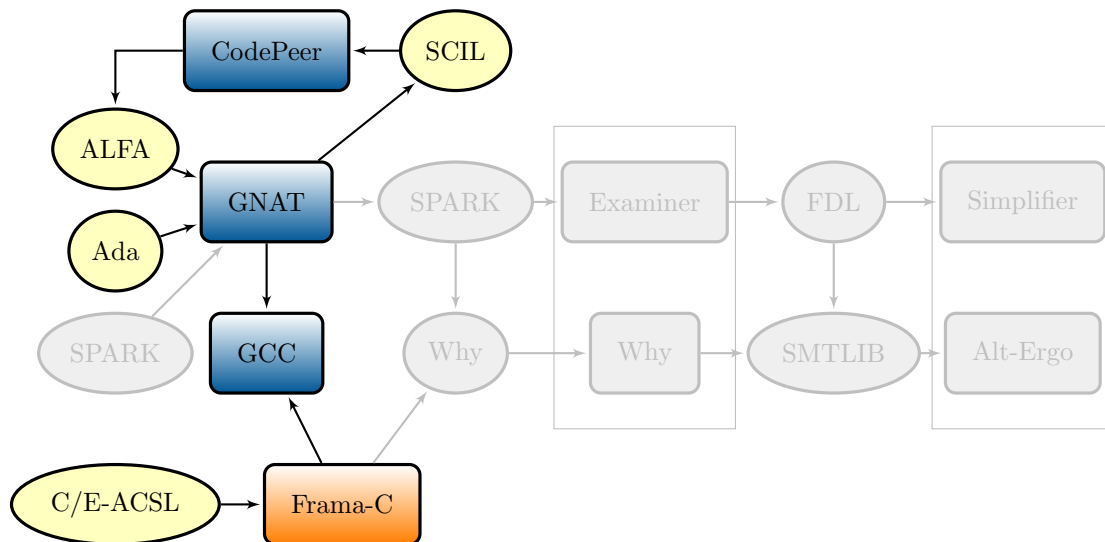


FIG. 1 – Cas de l'entreprise A

**A : Associer spécifications et tests** L'entreprise avionique A fabrique des pièces d'avion, pour lesquelles elle fournit le logiciel de contrôle-commande. Lors des précédents projets, une équipe de conception traduisait les exigences haut-niveau en spécifications semi-formelles, type UML, que les équipes de développement et de test utilisaient pour écrire indépendamment le programme et les tests associés. Les imprécisions de la notation semi-formelle et l'absence d'outils pour vérifier la cohérence de la spécification, du code et des tests ont parfois été la cause d'aller-retours multiples entre les équipes de conception, de développement et de test, avec comme conséquence des retards importants.

Avec la technologie HI-LITE, l'entreprise A a pris la décision de passer à des spécifications plus formelles, avec l'espoir de réduire les incompréhensions entre fournisseurs et consommateurs de spécifications. L'équipe de conception écrit désormais

## HI-LITE

les spécifications en ALFA des paquetages Ada que l'équipe de développement doit implémenter et que l'équipe de test doit tester. Les fichiers de spécification Ada, contenant à la fois la signature des sous-programmes et les contrats vérifiés par ces sous-programmes, sont devenus les documents de référence partagés par toutes les équipes. Au cours du développement, les programmeurs utilisent CodePeer pour détecter d'éventuelles erreurs de codage (qu'ils corrigent) ou de spécification (qu'ils rapportent à l'équipe de conception). Une fois la spécification conforme avec le code, l'équipe de test ajoute aux spécifications les tests unitaires, dans le même langage ALFA. Les erreurs de codage et de spécification sont reportées aux équipes concernées. Pour les parties du code écrites en C, les spécifications sont écrites en E-ACSL et traduites automatiquement en ALFA.

Cette intégration des spécifications et des tests permet à l'entreprise A de fortement réduire les retards sur ses projets. Elle s'intéresse désormais à la preuve de programmes avec les autres outils fournis dans HI-LITE.

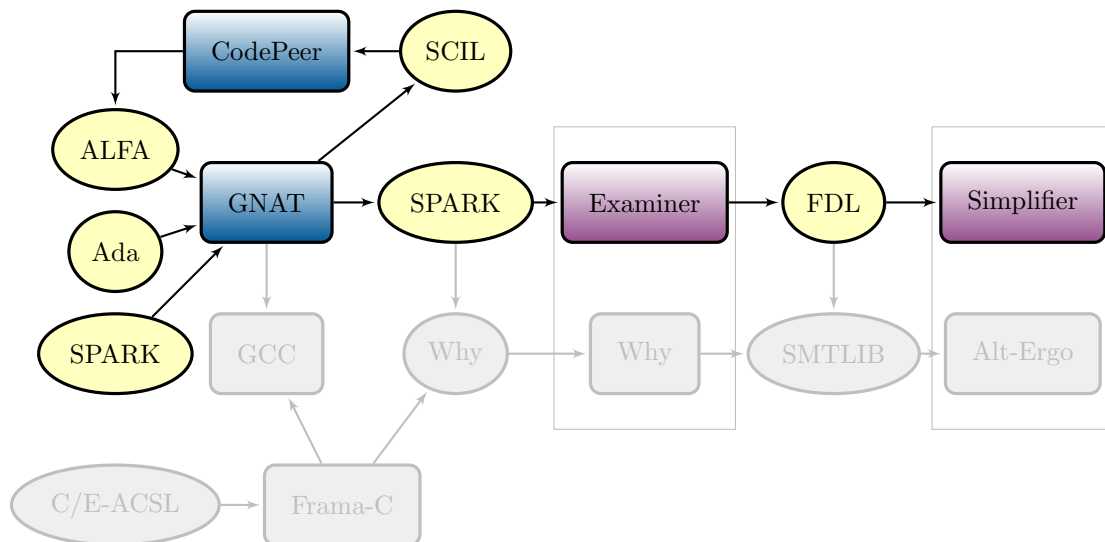


FIG. 2 – Cas de l'entreprise B

**B : Aller plus loin avec SPARK** L'entreprise B fournit des logiciels de contrôle d'accès certifiés au plus haut niveau des critères communs. Autour d'un cœur critique développé en SPARK, les équipes de développement utilisent des programmes écrits dans différents langages, principalement Ada (le code SPARK et Ada est compilé ensemble). Après des années d'utilisation de SPARK pour développer des applications de sécurité et vérifier des propriétés de confidentialité avec l'Examiner, l'entreprise B voudrait aller plus loin et prouver d'autres propriétés de sécurité sur ses programmes.

L'entreprise B a choisi d'utiliser les différents outils de preuve de la technologie HI-LITE pour prouver les propriétés désirées, exprimées sous forme d'annotations SPARK : contrats de sous-programme, *checks* dans le corps des sous-programmes. L'équipe de développement commence donc en général par traduire les propriétés désirées dans des contrats de sous-programme SPARK. Ensuite, elle développe les corps des sous-

programmes, comme précédemment, avec des boucles et des sous-programmes internes quand c'est nécessaire. Une étape clé est la génération d'annotations pour les boucles et les sous-programmes internes. Étant donné que tous les sous-programmes externes sont annotés avec des contrats précis, la génération d'annotations avec CodePeer donne de très bons résultats, qui font gagner un temps précieux à l'équipe de développement. En général, il suffit de relire les annotations générées et de les modifier à la marge avant de les accepter définitivement dans le code, ce qui se fait facilement grâce à l'interface offerte par GPS. Une fois que les annotations définitives sont établies, la nouvelle équipe de preuve applique les différents outils de génération d'obligations de preuve et de preuve automatique pour réduire au maximum le travail résiduel de preuve manuelle.

Grâce à la génération automatique d'annotations, l'entreprise B a réduit son temps de développement, ce qui lui permet de vérifier de nouvelles propriétés de sécurité plus fines que celles prouvées précédemment.

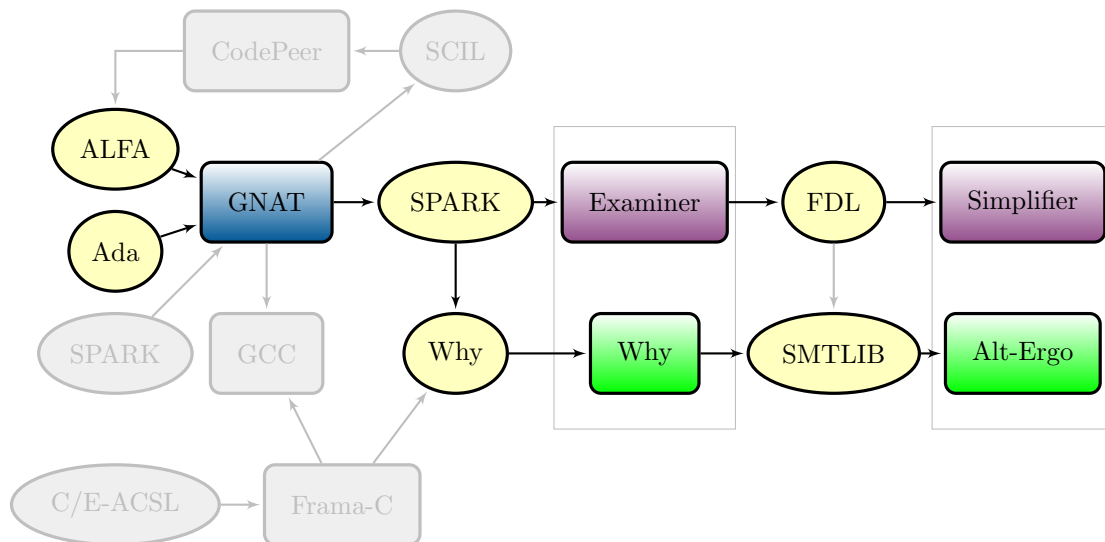


FIG. 3 – Cas de l'entreprise C

**C : Plutôt deux fois qu'une (vérification)** L'entreprise C construit un prototype militaire d'équipement en Ada, pour lequel le plus haut degré de sûreté et de sécurité est exigé. Elle désire donc obtenir le plus possible de garanties formelles sur les programmes développés.

Aucune annotation n'est générée. Ce sont les programmeurs qui développent les annotations en parallèle avec le code. Le développement procède de manière modulaire, de telle sorte que chaque sous-programme est développé, annoté et prouvé avant de continuer. Chaque obligation de preuve doit être prouvée soit par le prouveur automatique de SPARK, le Simplifier, soit par le prouveur SMT, Alt-Ergo ; dans les cas où les deux prouveurs échouent, une inspection manuelle est lancée.

Grâce à la possibilité d'invoquer un prouveur par réécriture (le Simplifier) ou un prouveur SMT (Alt-Ergo) selon les obligations de preuve, l'entreprise C a développé

## HI-LITE

des stratégies différenciées selon les propriétés de sécurité qu'elle cherche à vérifier, ce qui permet à l'entreprise C de relever son niveau d'assurance au-delà de ce qui était précédemment possible. Cela devrait lui permettre d'accéder aux nouveaux standards MILS requis par l'autorité de certification.