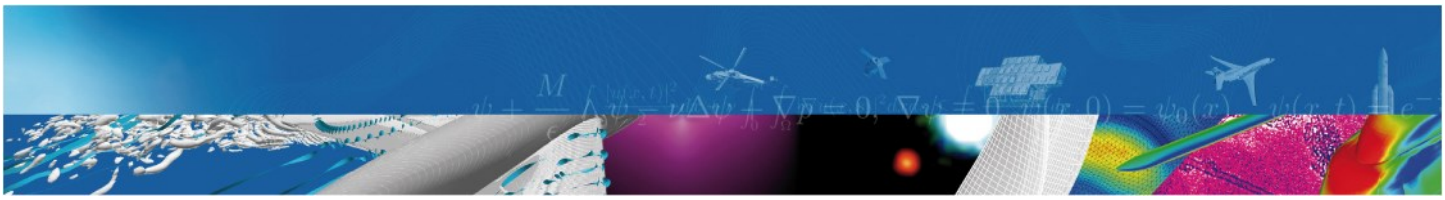

Analyse de la qualité du code via une approche logique et application à la robotique

Soutenance de thèse – David CÔME
25 janvier 2019 à 10h00
Auditorium de l'ONERA

Devant le jury composé de :

Jean-Paul Bodeveix	Université Paul Sabatier	Examinateur
Julien Brunel	ONERA - DTIS	Examinateur
Manuel Alcino Cunha	University of Minho	Examinateur
David Doose	ONERA – DTIS	Examinateur
Karen Godary-Dejean	Université de Montpellier	Examinatrice
Julia Lawall	INRIA	Rapportrice
Jacques Malenfant	Sorbonne Université	Rapporteur
Virginie Wiels	ONERA – DTIS	Directrice de thèse

Résumé

La qualité d'un code informatique passe à la fois par sa correction fonctionnelle mais aussi par des critères de lisibilité, compréhension et maintenabilité. C'est une problématique actuellement importante en robotique où de nombreux frameworks open-source se diffusent mal dans l'industrie en raison d'incertitudes sur la qualité du code. Les outils d'analyse et de recherche de code sont efficaces pour améliorer ces aspects. Il est important qu'ils laissent l'utilisateur spécifier ce qu'il recherche afin pouvoir prendre en compte les spécificités de chaque projet et du domaine. Il existe deux principales représentations du code: son arbre de syntaxe abstraite (Abstract Syntax Tree en anglais, ou AST) et le graphe de flot de contrôle (Control Flow Graph en anglais, ou CFG) des fonctions qui sont définies dans le code. Les mécanismes de spécification existants utilisent exclusivement l'une ou l'autre de ces représentations, ce qui est dommage car elles offrent des informations complémentaires. L'objectif de ce travail est donc de développer une méthode de vérification de la conformité du code avec des règles utilisateurs qui puissent exploiter conjointement l'AST et le CFG. La méthode repose sur une nouvelle logique développée dans le cadre de ces travaux : FO++, qui est une extension temporelle de la logique du premier ordre. Cette logique a plusieurs avantages. Tout d'abord, elle est indépendante de tout langage de programmation et dotée d'une sémantique formelle. Ensuite, elle peut être utilisée comme moyen de formaliser les règles utilisateurs une fois instanciée pour un langage de programmation donné. Enfin, l'étude de son problème de model-checking offre un mécanisme de vérification automatique et correct de la conformité du code. Ces différents concepts ont été implémentés dans Pangolin, un outil pour le langage C++. Étant donné le code à vérifier et une spécification (qui correspond à une formule de FO++, écrite dans le langage utilisateur de Pangolin), l'outil indique si oui ou non le code respecte la spécification. Il offre de plus un résumé synthétique de l'évaluation pour pouvoir retrouver le potentiel code fautif ainsi qu'un certificat de correction du résultat. Pangolin et FO++ ont trouvé une première application dans le domaine de la robotique via l'analyse de la qualité des paquets ROS et la formalisation d'un design-pattern spécifique à ROS. Une seconde application plus générale concerne le développement de programmes en C++ avec la formalisation de diverses règles de bonnes pratiques pour ce langage. Enfin, on montre comment il est possible de spécifier et vérifier des règles intimement liées à un projet en vérifiant des propriétés sur Pangolin lui-même.

Mots-clés

C++, Robotique, Model checking, qualité du code, analyse de code