Méthodologie pour la validation d'exigences globales appliquée à la sécurité*

Virgile Robles[1], Nikolai Kosmatov[1,2], Virgile Prevosto[1], Louis Rilling[3] et Pascale Le Gall[4]

[1]Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France, firstname.lastname@cea.fr
[2]Thales Research & Technology, Palaiseau, France, nikolaikosmatov@gmail.com
[3]DGA, France, louis.rilling@irisa.fr
[4]Laboratoire de Mathématiques et Informatique pour la Complexité et les Systèmes, CentraleSupélec, Université Paris-Saclay, Gif-Sur-Yvette, France, pascale.legall@centralesupelec.fr

Contexte Pour valider un programme, il est courant d'annoter ses fonctions avec des *contrats* : un ensemble de pré-conditions et de post-conditions formalisées au sein d'un langage de spécification tel qu'ACSL [2]. Il est alors possible de vérifier que le programme est correct vis-à-vis de sa spécification via des techniques telles que la vérification déductive.

Nos travaux précédents [3][4] montrent qu'une telle approche basée sur des contrats est limitée lorsqu'il s'agit de spécifier des exigences *haut-niveau*, qui s'appliquent à des composants entiers plutôt qu'à des fonctions individuelles. Ces travaux proposent une technique basée sur les *méta-propriétés*, ou HILARE [1] : une classe de propriétés de programmes décrivant des invariants globaux et des contraintes sur les opérations mémoires. Cette technique est implémentée dans MetAcsl [2], un greffon de la plateforme Frama-C [5], qui permet d'insérer ces propriétés au sein d'un programme C et de vérifier que ce dernier ne les viole pas.

Contributions Nous développons une méthodologie permettant d'utiliser notre approche sur un grand éventail d'exigences haut-niveau, dans le but de fournir des recommandations détaillées aux utilisateurs d'outils de vérification logicielle afin de rendre plus accessible la validation de programmes

*Cette soumission est un résumé étendu d'un article [1], accepté à FormaliSE 2021 (18-21 mai).

1. HIgh-Level Acsl REquirement
2. https://git.frama-c.com/pub/meta

de taille réelle. Cette méthodologie est illustrée sur de petits exemples et appliquée à deux cas d'études : (i) la vérification du chargeur d'amorçage (*bootloader*) du projet de clé USB sécurisée WooKey [6] et (ii) l'ébauche d'une spécification du micronoyau d'un système d'exploitation jouet.

Méthodologie L'expérience montre que le processus de vérification à l'aide d'HILARE de grandes bases de code suit un certain nombre d'étapes récurrentes. Tout d'abord, il est important de commencer par (i) s'assurer que le problème est bien dans le périmètre de MetAcsl [4] et (ii) identifier des points d'ancrages pour les propriétés dans l'état global du programme (c.à.d les variables pertinentes pour exprimer les exigences globales).

Ensuite, la formulation des exigences sous forme d'HILARE peut elle-même se décliner en plusieurs étapes distinctes. Pour chaque exigence, il est nécessaire d'identifier l'ensemble des fonctions concernées, puis d'essayer de la formuler comme un invariant sur l'état global ou bien une *contrainte* sur (i) les modifications de la mémoire, (ii) les accès mémoire ou (iii) les appels de fonction. Cette dernière étape peut être facilitée par la connaissance d'un certain nombre de *motifs* communs couvrant la plupart des exigences courantes, qui sont détaillés et illustrés dans l'article.

La validation d'exigences ainsi formulées peut s'effectuer via les greffons de Frama-C existants tels qu'Eva, E-ACSL ou Wp. Nous nous intéressons à ce dernier outil en particulier, qui permet de faire de la vérification déductive, et donnons un ensemble de recommandations pour réaliser la preuve du programme : comment analyser l'échec d'une obligation de preuve et le résoudre. Enfin, nous listons plusieurs écueils et bonnes pratiques à avoir en tête lors du travail de spécification afin que la vérification repose sur des bases solides : l'absence d'erreurs à l'exécution, la clôture des empreintes mémoires, etc.

Application La formulation d'exigences complexes en HILARE est illustrée de manière détaillée sur un micronoyau à l'architecture générique. Des contraintes telles que l'isolation des tâches ou leur ordonnancement permettent de mettre en œuvre la méthodologie présentée pas à pas.

Cette dernière est ensuite appliquée en situation réelle à un ensemble choisi d'exigences de sécurité sur le bootloader du projet WooKey [6], un prototype de clé USB chiffrante à authentification forte dont le microgiciel (*firmware*) est partiellement écrit en C. Entre autres, nous vérifions que chaque dispositif de sécurité est bien exécuté dans le bon ordre. L'implantation du bootloader comportant plus de 600 fonctions, sa vérification vis-à-vis de plusieurs propriétés complexes démontre la pertinence de l'approche sur des programmes réels.

Remerciements Nous remercions chaleureusement Ryad Benadjila, Arnauld Michelizza, Patricia Mouy, Mathieu Renard, Philippe Thierry et Philippe Trebuchet de l'ANSSI pour nos discussions à propos de WooKey, ainsi que l'équipe de Frama-C pour leur support. Recherche soutenue financièrement par le Ministère des armées - Agence de l'Innovation de Défense. Merci aux reviewers anonymes pour leurs commentaires.

Références

[1] V. Robles, N. Kosmatov, V. Prevosto, L. Rilling et P. Le Gall. « Methodology for Specification and Verification of High-Level Requirements with MetAcsl ». In : *FormaliSE 2021 - 9th International Conference on Formal Methods in Software Engineering*. Mai 2021.

[2] P. Baudin, P. Cuoq, J.-C. Filiâtre, C. Marché, B. Monate, Y. Moy et V. Prevosto. *ACSL : ANSI/ISO C Specification Language*. `https://frama-c.com/acsl.html`. 2018.

[3] V. Robles, N. Kosmatov, V. Prevosto, L. Rilling et P. Le Gall. « MetAcsl : Specification and Verification of High-Level Properties ». In : *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. T. 11427. LNCS. 2019.

[4] V. Robles, N. Kosmatov, V. Prevosto, L. Rilling et P. Le Gall. « Tame Your Annotations with MetAcsl : Specifying, Testing and Proving High-Level Properties ». In : *Tests and Proofs (TAP)*. T. 11823. LNCS. Springer, 2019.

[5] F. Kirchner, N. Kosmatov, V. Prevosto, J. Signoles et B. Yakobowski. « Frama-C : A software analysis perspective ». In : *Formal Aspects of Computing* (2015), p. 573-609.

[6] R. Benadjila, A. Michelizza, M. Renard, P. Thierry et P. Trebuchet. « WooKey : Designing a Trusted and Efficient USB Device ». In : *Annual Computer Security Applications Conference (ACSAC)*. 2019.