Analyse d'un modèle AADL à l'aide de Pola

Pierre-Emmanuel Hladik, Florent Peres, Xiaomu Shi

Analyse d'un modèle AADL à l'aide de Pola

Pierre-Emmanuel Hladik*,**, Florent Peres***, Xiaomu Shi****

*CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse, France
**Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France
***Université de Lille Nord de France - F59000 Lille, INRETS, ESTAS, F59650 Villeneuve d'Ascq
****VERIMAG - Centre Équation, 2 avenue de Vignate, F-38610 Gières, France
pehladik@laas.fr; florent.peres@inrets.fr; xiaomu.shi@doctorant.univ-grenoble.fr

Résumé. Cet article présente un travail mené sur l'intégration d'un langage formel de vérification, Pola, dans le processus de conception des systèmes temps réel critiques en se basant sur le langage AADL (Architecture Analysis and Design Language). Pola est un langage dédié qui permet de décrire le comportement de systèmes temps réel complexes en vue de leur vérification (ordonnançabilité, absence de blocages, etc), laquelle est effectuée par *model checking*. Le travail présenté consiste à utiliser les méthodes et outils de l'ingénierie des modèles pour transformer un modèle AADL vers un modèle Pola.

1 Introduction

La complexité des systèmes embarqués temps réel rend nécessaire de disposer de méthodes et d'outils pour assister les concepteurs lors du processus de développement. L'ingénierie des modèles offre pour cela des solutions intéressantes et prometteuses.

Actuellement, deux langages sont particulièrement étudiés dans le cadre de la modélisation des systèmes embarqués critiques : UML avec le profil MARTE et AADL. Afin d'effectuer la vérification des modèles issus de ces langages, des chaînes d'outils sont développées. Dernièrement, le projet européen SPICES recensait pas moins d'une douzaine d'outils et de méthodes autour du langage AADL permettant de vérifier des propriétés sur l'occupation mémoire, l'ordonnancement, la consommation d'énergie, la génération de code, etc.

Cet article présente une méthode de vérification de la partie temps réel des modèles du langage AADL, consistant à introduire Pola Peres et al. (2009), un langage formel de vérification dédié aux systèmes temps réel, au sein d'une chaîne de transformation et de vérification de modèles, débutant par un modèle exprimé en langage AADL.

2 Présentation des langages AADL et Pola

Le langage AADL (Architecture Analysis and Design Language) est un langage de description d'architecture conçu pour les systèmes temps réel embarqués dans les domaines avionique, spatial, robotique, santé, etc. Une version 2.0 a été publiée en janvier 2009 par le SAE (International Society for Automotive Engineers) et sert de référence pour ce travail.

Le standard SAE AADL fournit des concepts de modélisation par composant d'une architecture logicielle et matérielle. Il permet de décrire le moteur d'exécution d'application en terme de tâches concurrentes ainsi que leurs interactions et leur allocation sur le matériel.

Une architecture est décrite par une hiérarchie de composants typés possédant une interface fonctionnelle. L'interface comprend des propriétés (*properties*) avec les valeurs des attributs et des caractéristiques du composant, ainsi que des *features* qui spécifient comment le composant est interfacé avec les autres. A cela s'ajoute, pour chaque composant, une implémentation qui définit sa structure interne en terme de sous-composants et de connections. L'implémentation autorise aussi la spécification des valeurs des propriétés non-fonctionnelles du composant.

AADL peut être étendu pour introduire des spécificités propre à chaque application en plus des propriétés pré-définies. Elles se présentent sous la forme d'annexes dans lesquelles sont déclarées les extensions du langage sous forme conceptuelle et syntaxique. Notre but étant de rester le plus générique possible, il a été décidé pour ce travail de ne pas introduire de nouvelles propriétés et par conséquent d'utiliser celles fournies dans le standard.

Le langage Pola (Peres et al. (2009)) est un langage spécifique pour les systèmes de tâches temps réel ordonnancées. Son expressivité est limitée par le domaine d'application et par ses possibilités en terme de vérification, c'est-à-dire que toute notion ne pouvant pas être vérifiée en *model checking* est exclue. Cette limitation a cependant un corollaire intéressant : tout modèle du langage est garanti vérifiable, dans les limites des possibilités du *model checking*.

L'aspect spécifique de Pola à un domaine permet la spécification d'un système de manière intelligible pour un utilisateur expert en temps réel, tout en cachant les détails de l'analyse à l'aide d'une chaîne de traduction et vérification automatique : un modèle Pola est traduit dans un langage adapté au *model checking*, puis la vérification proprement dite est effectuée à partir de cette traduction (voir figure 1).

Le langage en entrée du *model-checking* est basé sur les réseaux de Petri temporels étendus par une notion de priorité, ou plus généralement une notion de contraintes sur les quantités temporelles du réseau avec les relations d'inhibitions et de permissions, ainsi que l'extension des chronomètres pour modéliser la préemption (voir Peres et al. (2009)). Un ensemble de formules de logique temporelle est également défini pour spécifier les contraintes à vérifier, telles que « les tâches doivent respecter leurs échéances ».

Le langage Pola est composé de deux parties distinctes : une partie déclarative et une partie comportementale. La partie déclarative du langage permet de modéliser les ressources d'exécution et les tâches d'un système temps réel, ainsi que les éléments qui contrôle l'exécution.

La partie comportementale permet d'ajouter des comportements non génériques au système. Bien que s'éloignant de la démarche des langages spécifiques, l'utilisateur peut décrire les comportements à l'aide d'un réseau de Petri dans une partie dédiée.

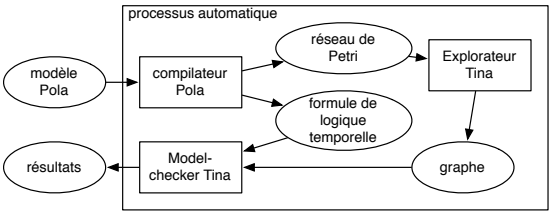

FIG. 1 – *Processus de vérification avec Pola*

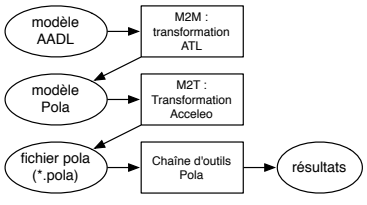

FIG. 2 – *Processus de vérification d'un modèle AADL avec Pola*

3 Travail réalisé

Motivation du travail. Dans le domaine de la vérification avec AADL, deux approches existent : la première consiste à extraire du modèle AADL que les informations qui sont relatives à un type de propriétés pour ensuite utiliser un outil dédié à l'analyse. Par exemple, les outils Cheddar (Singhoff et al. (2004)) et MAST (Harbour et al. (2001)) utilisent cette méthode pour l'analyse d'ordonnançabilité. La limite d'une telle approche est imposée par l'expressivité des outils de vérification employés, ce qui contraint généralement les modèles AADL à certaines classes d'applications.

La seconde approche, plus générique, consiste à transformer un modèle AADL vers un modèle formel à la sémantique bien définie, comme dans l'approche utilisant Fiacre (Berthomieu et al. (2009)) ou IF (Abdoul et al. (2008)). Cette méthode est généralement exhaustive du point de vue comportemental et permet de modéliser un grand nombre d'applications, mais est rapidement confrontée à la complexité des techniques de vérification et leur explosion combinatoire.

Le choix fait ici est d'employer les modèles formels en exploitant la spécialisation du langage Pola pour l'analyse comportementale. Cela permet de modéliser efficacement un large ensemble de comportements, en particulier au niveau de l'ordonnancement, et d'en faire l'analyse. Par rapport à la démarche basée sur des outils dédiés, cette méthode n'est pas contrainte à des modèles standards et permet de composer des comportements complexes.

L'intérêt de modéliser avec Pola et non pas directement avec des réseaux de Petri, est de montrer la correction de la transformation d'un modèle AADL vers un modèle formel grâce à la proximité sémantique des concepts de Pola et de AADL. En effet, AADL ne possèdant pas de sémantique formelle, il n'est pas possible de *vérifier* cette transformation. Au final, la transformation vers Pola donne *une interprétation formelle* de la sémantique de AADL.

Bien que le passage par l'écriture d'un modèle Pola peut sembler introduire de la complexité pour assurer la *traçabilité* entre les modèles de la chaîne de traduction, il nous semble au contraire diminuer la complexité de la transformation et faciliter ainsi la traçabilité.

Etude sémantique. Pour formaliser le cadre du travail, nous employons les processus issus de l'ingénierie des modèles pour transformer un modèle AADL vers Pola. Cependant, une chaîne de vérification basée sur des transformations de modèles n'est possible et pertinente que si elle garantit que le comportement exprimé par le modèle d'origine est identique à celui de la cible. Pour cela, il a été nécessaire d'étudier les sémantiques de chaque langage et la possibilité de traduire chaque élément d'un langage vers l'autre. Pour une raison de place, nous ne présentons pas ce travail ici, le lecteur peut se référer à Hladik et al. (2010) pour de plus amples détails.

Ce qui est ressorti globalement de cette étude est une grande similitude entre les éléments de AADL et Pola et la possibilité de traduire quasiment chaque élément de AADL vers Pola — au pire, en écrivant un patron en réseaux de Petri dans la partie comportementale du langage Pola. Cependant, certaines imprécisions sur la sémantique de AADL, en particulier sur les propriétés, ne permettent pas de garantir une équivalence sémantique de la traduction. De plus, l'utilisation à terme de l'annexe comportementale Franca et al. (2007) de AADL semble une nécessité pour décrire correctement le comportement des entités exécutables.

Implémentation. Du point de vue de la mise en œuvre, nous nous sommes appuyés sur les méthodes de l'ingénierie dirigée par les modèles. Pour cela, nous avons utilisé le langage de transformation ATL (ATLAS Transformation Language) avec le métamodèle Ecore de AADL

de l'atelier TOPCASED. Le métamodèle du langage Pola a été écrit en Ecore de manière ad-hoc. La réécriture des modèles sous une forme textuelle (*model-to-text* (M2T)) en entrée à la chaîne de traitements Pola a été réalisée à l'aide d'Acceleo de la société Obeo. Son intégration dans TOPCASED a permis de développer une chaîne complète de transformation dans cet environnement (fig. 2).

Une première implémentation de la transformation a été réalisée sur un sous-ensemble de AADL. Cette implémentation comprend : la réécriture des composants de contrôle de l'exécution ; la prise en compte des politiques d'ordonnancement *Rate Monotonic* et à priorité fixe ; l'activation périodique, sporadique et apériodique ; la prise en compte des données partagées avec une politique d'accès de type sémaphore. Ce prototype bien que limité, nous a permis de démontrer la faisabilité d'une telle approche qui devra être étudiée plus en détail en s'attachant particulièrement à la sémantique des langages.

4 Conclusion

L'étude menée sur les correspondances sémantiques entre les modèles de AADL et Pola, a permis de réaliser un prototype implémentant une transformation entre ces deux modèles pour en permettre une vérification formelle. Les travaux futurs s'attacheront à l'intégration de cet outil dans l'atelier TOPCASED et à une extension des propriétés prises en considération, en particulier celles de l'annexe comportementale AADL pour pouvoir affiner l'analyse au niveau Pola. De plus, des travaux sont en cours sur la prise en charge de politiques d'ordonnancement dynamiques avec Pola, ce qui permettra, à terme, de vérifier des systèmes avec de tels ordonnancements.

Références

Abdoul, T., J. Champeau, P. Dhaussy, P.-Y. Pillain, et J.-C. Roger (2008). AADL execution semantics transformation for formal verification. In *13th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2008)*, pp. 263–268.

Berthomieu, B., J.-P. Bodeveix, C. Chaudet, S. Dal-Zilio, M. Filali, et F. Vernadat (2009). Formal verification of AADL specifications in the topcased environment. In *Ada-Europe*, pp. 207–221.

Franca, R. B., J.-P. Bodeveix, M. Filali, J.-F. Rolland, D. Chemouil, et D. Thomas (2007). The AADL behaviour annex – experiments and roadmap. In *12th IEEE International Conference on Engineering Complex Computer Systems (CECCS '07)*.

Harbour, M. G., J. J. G. García, J. C. P. Gutiérrez, et J. M. D. Moyano (2001). MAST : Modeling and analysis suite for real time applications. In *13th Euromicro Conference on Real-Time Systems (ECRTS 01)*, pp. 125–134.

Hladik, P.-E., F. Peres, et X. Shi (2010). Analyse d'un modèle aadl à l'aide de pola. Technical Report (à paraître), LAAS-CNRS.

Peres, F., P.-E. Hladik, et F. Vernadat (2009). Specification and verification of real-time system using the POLA tool. In *Third International Workshop on Verification and Evaluation of Computer and Communication Systems (VECoS 2009)*.

Singhoff, F., J. Legrand, L. Nana, et L. Marçé (2004). Cheddar : a flexible real time scheduling framework. In *ACM Special Interest Group on Ada (SIGAda'2004)*.

Summary

These article presents the integration of the formal language Pola in a verification process for real-time systems modeled in AADL. Pola is a Domain Specific Language to model real-time scheduled critical tasks and aims to proceed formal verifications based on model checking. The work presented here uses the techniques and the methods of model-driven engineering to transform a AADL model in a Pola model.