ModHel'X, un outil expérimental pour la modélisation multi-paradigmes

Christophe Jacquet, Cécile Hardebolle and Frédéric Boulanger

SUPELEC
3 rue Joliot-Curie
91192 Gif-sur-Yvette Cedex
France
Prenom.Nom@supelec.fr

Résumé

La modélisation multi-paradigmes vise à permettre la modélisation d'un système en utilisant pour chacune de ses sous-parties le formalisme (ou paradigme) de modélisation le plus adapté. Nous nous intéressons plus spécifiquement à la modélisation du *comportement* de systèmes : dans ce cadre, il est nécessaire de pouvoir combiner la sémantique des différents formalismes utilisés dans un modèle de sorte à déterminer le comportement global du système modélisé.

Nous avons conçu un outil appelé ModHel'X qui permet a) de décrire les formalismes de modélisation utilisés grâce au concept de modèle de calcul, b) de décrire l'adaptation sémantique à la frontière entre deux formalismes, c) de construire des modèles faisant appel à plusieurs de ces formalismes et d) de simuler le comportement de tels modèles.

1 Introduction

La modélisation multi-paradigmes [4] vise à permettre la modélisation d'un système en utilisant pour chacune de ses sous-parties le formalisme (ou paradigme) de modélisation le plus adapté. Nous nous intéressons spécifiquement à la modélisation du comportement des systèmes. Dans ce cadre, les formalismes de modélisation utilisés sont par exemple les automates, les flots de données, les systèmes d'équations différentielles en temps continu, etc.

Ce type de modélisation prend une importance croissante dans la conception des systèmes embarqués dans le cadre de l'ingénierie dirigée par les modèles. Actuellement, la plupart des formalismes sont soit décrits de manière informelle en langage naturel, soit décrits par une implémentation de référence qui est généralement l'outil qui supporte le modèle. Pour permettre la spécification de systèmes de façon indépendante des outils utilisés, nous cherchons à décrire les formalismes de façon précise, mais à un niveau d'abstraction plus élevé que celui des langages de programmation.

Nous développons depuis plusieurs années ModHel'X, un outil expérimental pour la modélisation mutli-paradigmes. ModHel'X s'inspire de Ptolemy II [2]. Notamment, il emprunte à ce dernier la description des formalismes de modélisation en tant que *modèles de calculs* (abrégés MoC pour *Model of Computation*).

ModHel'X propose un méta-modèle permettant de décrire la structure de modèles de systèmes, et un moteur d'exécution générique qui s'appuie sur un certain nombre d'opérations primitives pour décrire la sémantique des modèles de calcul, ainsi que les interactions entre modèles de calculs différents. Ceci permet de donner une sémantique d'exécution précise aux modèles multi-paradgimes. ModHel'X a été décrit en détails dans [1].

Cette démonstration s'intéresse aux principaux aspects de ModHel'X : la description de modèles grâce à un méta-modèle générique, indépendant du MoC (section 2), la description

des modèles de calcul eux-mêmes (section 3), la description de l'adaptation sémantique entre MoC (section 4). La section 5 résume la contribution, indique comment obtenir ModHel'X pour expérimenter soi-même, et donne quelques perspectives.

2 Description de modèles : méta-modèle générique

Afin de permettre la construction de modèles multi-paradigmes, ModHel'X introduit un méta-modèle générique, indépendant du modèle de calcul. Tout modèle, quelle que soit sa sémantique, est donc décrit en utilisant un petit nombre de constructions syntaxiques :

Bloc C'est l'unité élémentaire de comportement. Il s'agit d'une boîte noire dont l'interface est spécifiée.

Patte (pin) C'est le mécanisme d'échange d'informations entre blocs. L'ensemble des pattes d'un bloc forme son interface.

Relations Elles connectent entre elles des pattes de différents blocs.

Un modèle est composé d'une structure (assemblage syntaxique de blocs dont les pattes sont connectées par des relations) et d'un MoC qui spécifie sa sémantique. Une même structure peut donc être interprétée différemment, selon le MoC qui lui est associé (voir figure 1).

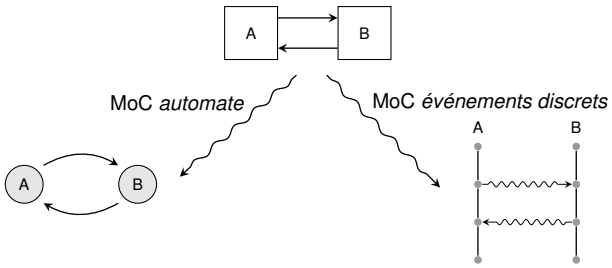

FIGURE 1 – Différents modèles de calcul donnent une interprétation différente de la même structure : états et transitions dans le cas d'un automate, processus s'échangeant des messages dans le cas d'un modèle à événements discrets.

L'hétérogénéité dans les modèles multi-paradigmes est obtenue en structurant hiérarchiquement un modèle : des blocs spéciaux appelés *blocs d'interface* permettent d'embarquer un modèle obéissant à un certain MoC dans un modèle obéissant à un autre MoC (voir figure 2).

3 Description de modèles de calculs

ModHel'X repose sur une approche boîte noire : pour déterminer le comportement global d'un modèle, le moteur d'exécution doit observer les comportements des blocs qui le composent, puis combiner ces observations selon les règles du MoC.

ModHel'X détermine donc une succession d'observations instantanées appelées *snapshots*. Chaque snapshot est calculé grâce à trois étapes :

Schedule Le MoC détermine quel est le prochain bloc à observer.

Update Ce bloc est observé.

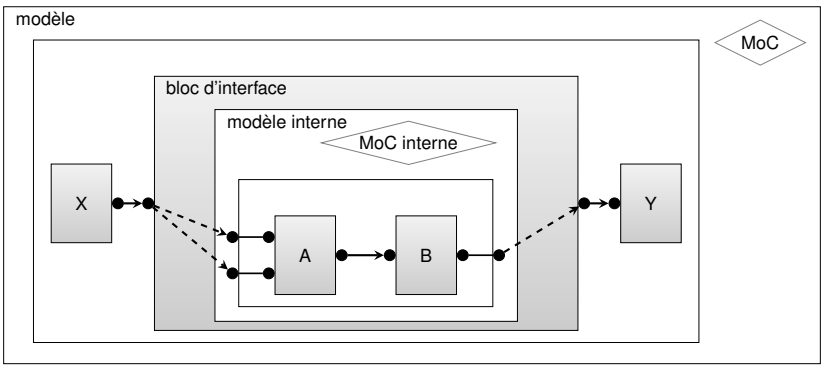

FIGURE 2 – Utilisation de la hiérarchie pour construire un modèle multi-paradigmes.

Propagate Le MoC propage les données dans le modèle.

Chacune de ces étapes permet d'observer une partie du modèle. Cet enchaînement est itéré jusqu'à la détermination complète de l'observation du modèle, c'est-à-dire jusqu'à trouver un point fixe.

L'ensemble de ces trois étapes représente la sémantique abstraite de ModHel'X. Le comportement de chaque bloc vient concrétiser l'opération *update*, et de même chaque MoC concrétise *schedule* et *propagate*. Ainsi, créer un MoC revient à créer des opérations *schedule* et *propagate* pour expliciter sa sémantique. Actuellement, ces opérations sont codées en Java, mais nous envisageons à l'avenir de les décrire dans un langage dédié.

4 Description de l'adaptation sémantique entre MoC

L'adaptation sémantique recouvre trois aspects :

L'adaptation des données Dans le cas général, les données doivent être transformées lorsqu'on passe d'un modèle de calcul à un autre. Par exemple, un automate manipule des événements purs. S'il est intégré dans un modèle de type *événements discrets* (DE), il faut déterminer quelles données faire circuler dans DE suite à l'émission d'événements par l'automate.

L'adaptation du contrôle Les instants d'observation dépendent du modèle de calcul. Par exemple, un automate n'est observé que lorsqu'il réagit, tandis qu'un modèle échantillonné de type *flot de données synchrone* (SDF) doit être observé à chaque instant d'échantillonnage.

L'adaptation du temps Différents modèles peuvent avoir des notions de temps différentes, qu'il est nécessaire de connecter entre elles. Par exemple, différents sous-modèles peuvent avoir des échelles de temps différentes.

Les blocs d'interface sont chargés de mettre en œuvre les trois facettes de l'adaptation. Nous avons créé un langage appelé TESL (Tagged Events Specification Language), qui permet de spécifier déclarativement une partie importante de l'adaptation du temps et du contrôle. Pour le moment, une partie de l'adaptation, notamment l'adaptation des données, est réalisée par du code Java appelé par l'algorithme d'exécution générique de ModHel'X. Cependant, des travaux sont en cours pour remplacer ce code Java par l'utilisation d'un langage dédié à l'adaptation [3].

5 Conclusion et perspectives

Nous avons présenté ModHel'X, un outil expérimental de modélisation multi-paradigmes. Nous avons vu comment sont décrits les modèles, les formalismes de modélisation (MoC), et les adaptateurs entre MoC. En utilisant ces trois éléments, le moteur générique d'exécution de ModHel'X peut simuler des modèles.

ModHel'X est inspiré de Ptolemy II, auquel il ajoute la modélisation explicite et modulaire de l'adaptation sémantique entre modèles de calcul. Les *blocs d'interface* n'ont pas d'équivalents en Ptolemy II, ce qui oblige soit à utiliser une sémantique d'adaptation implicite, soit à modifier les modèles pour y intégrer l'adaptation, ce qui limite fortement la réutilisabilité des modèles.

ModHel'X est actuellement implémenté dans l'écosystème Java. Il est diffusé sous forme de logiciel libre. La page web `http://wwwdi.supelec.fr/software/ModHelX/` permet de télécharger l'outil en lui-même ainsi que différentes démonstrations. Elle donne également accès aux publications sur le sujet.

Pour le moment, nos efforts se sont tournés essentiellement vers l'expression d'une sémantique opérationnelle pour les MoC et les modèles combinant différents MoC. À l'avenir, il serait intéressant de donner aux MoC et aux modèles une sémantique qui soit analysable. Cela permettrait par exemple de prouver des propriétés portant sur un MoC, ou sur un bloc, et d'en déduire des propriétés portant sur un modèle dans son ensemble.

Références

[1] Frédéric Boulanger, Cécile Hardebolle, Christophe Jacquet, and Dominique Marcadet. Semantic Adaptation for Models of Computation. In *Proceedings of ACSD 2011 (Application of Concurrency to System Design)*, pages 153–162. IEEE Computer Society, June 2011.

[2] J. Eker, J. W. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong. Taming heterogeneity – the Ptolemy approach. *Proceedings of the IEEE, Special Issue on Modeling and Design of Embedded Software*, 91(1) :127–144, January 2003.

[3] Bart Meyers, Joachim Denil, Frédéric Boulanger, Cécile Hardebolle, Christophe Jacquet, and Hans Vangheluwe. A DSL for Explicit Semantic Adaptation. In *Proceedings of MPM 2013 (Multi-Paradigm Modeling workshop at Models 2013)*, number 1112 in CEUR Workshop Proceedings, pages 47–56. CEUR, December 2013.

[4] P. J. Mosterman and H. Vangheluwe. Computer Automated Multi-Paradigm Modeling : An Introduction. *SIMULATION*, 80(9) :433–450, 2004.