Intégration du support OCL dans Kermeta
Spécifiez la sémantique statique de vos méta-modèles

Jean-Marie Mottu - Olivier Barais - Mark Skipper – Didier Vojtisek - Jean-Marc Jézéquel

Projet Triskell/IRISA
Campus de Beaulieu.
F - 35 042 Rennes Cedex
prenom.nom}@irisa.fr

Abstract. Ce document présente succinctement l'intégration du support OCL pour le langage de méta-modélisation exécutable Kermeta. L'accent dans cette courte présentation est mis sur les cas d'utilisation possibles de cette brique logicielle et l'intérêt des choix techniques retenus : un support natif du paradigme de Conception par Contrats (Design by Contract) dans Kermeta et une transformation de modèle du méta-modèle OCL au méta-modèle Kermeta pour le support de la syntaxe concrète OCL
Keywords : Ingénierie Dirigée par les Modèles, Sémantique statique, Méta-modélisation, logique du premier ordre, Conception par contrats

1. Motivations

La méta-modélisation dans le domaine de l'informatique se définit comme la mise en évidence d'un ensemble de concepts pour un domaine particulier. Un modèle est une abstraction d'un phénomène du monde réel tandis qu'un méta-modèle est une abstraction d'ordre supérieur mettant en évidence les concepts utilisés pour définir le modèle. Dans ce domaine, on peut parler d'un modèle conforme à son méta-modèle comme on peut parler d'un programme conforme à sa grammaire. L'OMG au travers du MOF [1] propose un langage standardisé afin de permettre la définition de nouveaux méta-modèles. Kermeta [2], développé au sein de l'équipe Triskell, est un langage de méta-modélisation exécutable construit comme une extension du MOF. Il permet de définir un méta-modèle auquel on associe une sémantique opérationnelle pour permettre la simulation des modèles. Le MOF ainsi que Kermeta permettent de définir un certain nombre de contraintes sur le modèle représenté, portant entre autres sur la cardinalité des relations entre les concepts. Ces contraintes peuvent servir à vérifier la cohérence d'un modèle par rapport à son méta-modèle. Cependant, un langage comme le MOF manque, au même titre qu'UML, d'expressivité pour représenter un certains nombre de contraintes dans les futurs méta-modèles. Un exemple consiste à regarder la définition d'un méta-modèle qui permet d'écrire des modèles de composants possédant des ports et des opérations fournies et requises au niveau de ses ports, deux composants étant reliables grâce à un concept de connecteur entre leurs ports. Il est impossible avec un langage comme le MOF ou Kermeta de

définir la sémantique statique de ce méta-modèle, entre autres de définir dans ce méta-modèle ce qu'est un assemblage de composants valides.

OCL (Object Constraint Language) incorpore la notion de conception par contrats [4] en UML. Largement étendu depuis la première version datant de 1999, la version 2 du langage [3] propose actuellement une syntaxe déclarative simple fondée sur une logique du premier ordre permettant à la fois d'associer des contraintes à un méta-modèle et de définir des requêtes sur des modèles. Sa proximité avec UML est, en outre, moins forte autorisant son utilisation dans le cadre de la méta-modélisation. L'utilisation d'OCL dans le cadre de la méta-modélisation offre un moyen simple d'exprimer la sémantique statique d'un méta-modèle dans le cas où celle-ci est exprimable avec une logique du premier ordre. Nous proposons dans la présentation d'exposer le support d'OCL en Kermeta et d'expliquer l'intérêt de ce support pour un futur concepteur de méta-modèles.

```
operation foo(param : String) : String
  pre myprecondition is param != ""
  post apostcondition is result.size > param.size
  is do
     result := param + " world"
  end
```

Fig. 1. Syntaxe Kermeta pour la définition des pré et des post-conditions

2. Architecture

L'intégration d'OCL dans Kermeta est construite à l'aide de deux briques de base.

- L'ajout de la notion de conception par contrats « à la Eiffel » dans Kermeta. La définition des invariants et des pré-post conditions se fait alors à l'aide de la syntaxe concrète Kermeta. (voir exemple de la Figure 1)
- Le support de la syntaxe concrète d'OCL en fournissant une transformation de modèles entre l'AST d'OCL et l'AST de Kermeta. Cette transformation est écrite en Kermeta. Le modèle résultant de la transformation est automatiquement associé avec le méta-modèle auquel il s'applique grâce à un mécanisme de composition appelé aussi merge de modèles. Le schéma de la figure 2 illustre l'architecture générale de la transformation.

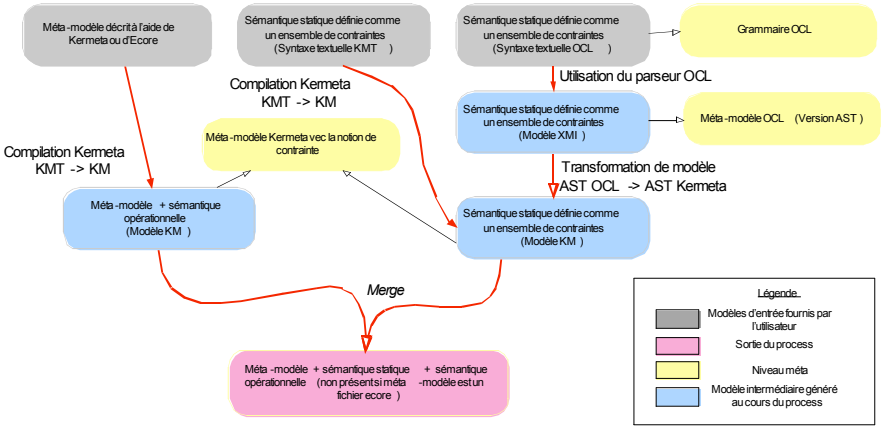

Fig 1. Vue globale de l'architecture

3. Intérêts et atouts

Le support d'OCL dans Kermeta apporte au moins trois principaux avantages pour un concepteur de méta-modèle.

- Pour les concepteurs de méta-modèle qui utilisent déjà Kermeta, ce support améliore la compatibilité avec les standards existants et permet l'intégration de contraintes définies en OCL. Ce mécanisme permet alors dans tous les cas de définir simplement la sémantique statique d'un méta-modèle. Kermeta offre ainsi un moyen de vérifier à la demande la correction d'un modèle avec sa sémantique statique au cours d'une simulation.
- D'un point de vue méthodologique pour le concepteur de méta-modèle, cette architecture permet une bonne séparation des préoccupations entre la description du méta-modèle, la description de la sémantique statique et la description de la sémantique opérationnelle. Chacune de ces sémantiques étant en effet définies à l'aide d'une syntaxe dédiée dans une unité dédiée : avec un fichier ecore par exemple pour le méta-modèle, un fichier Kermeta pour la sémantique opérationnelle et un fichier OCL pour la sémantique statique.
- Enfin, pour les utilisateurs d'OCL, l'implémentation d'OCL en Kermeta offre un moyen très simple de mettre en œuvre une extension d'OCL. En effet, OCL offre la possibilité d'appeler des méthodes définies dans des méta-classes du méta-modèle à partir du moment où ces méthodes n'ont pas d'effet de bord sur le modèle. La définition en Kermeta de la sémantique opérationnelle de ces méthodes permet une mise en œuvre et une évaluation de ces nouvelles primitives du langage.

[1] OMG, MOF 2.0 Core Final Adopted Specification. 2004.
[2] P-A. Muller, F. Fleurey, and J-M Jézéquel. -- Weaving executability into object-oriented meta-languages. -- In S. Kent L. Briand, editor, Proceedings of MODELS/UML'2005, volume 3713 of LNCS, pages 264--278, Montego Bay, Jamaica, October 2005. Springer.
[3] OMG, UML 2.0 Object Constraint Language (OCL) Final Adopted specification. 2003.
[4] B. Meyer. Applying "Design by Contract". Computer 25, 10 (Oct. 1992), 40-51.