

SpoonEMF, une brique logicielle pour l'utilisation de l'IDM dans le cadre de la réingénierie de programmes Java5

Olivier Barais

▶ **To cite this version:**

Olivier Barais. SpoonEMF, une brique logicielle pour l'utilisation de l'IDM dans le cadre de la réingénierie de programmes Java5. 2006. inria-00511393

HAL Id: inria-00511393
https://inria.hal.science/inria-00511393

Submitted on 24 Aug 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

SpoonEMF, une brique logicielle pour l'utilisation de l'IDM dans le cadre de la réingénierie de programmes Java5

Olivier Barais

Projet Triskell/IRISA
campus de Beaulieu.
F - 35 042 Rennes Cedex
barais@irisa.fr

RÉSUMÉ. Ce document présente succinctement SpoonEMF, une brique logicielle pour l'utilisation de l'IDM dans le cadre de la réingénierie d'applications écrites en Java. 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 : le framework EMF pour l'expression du méta-modèle afin de favoriser l'interopérabilité avec d'autres outils et l'intégration au sein du projet Spoon pour la définition du méta-modèle et la réutilisation d'outils comme l'analyseur syntaxique de code Java5 (parser) ou le générateur de code (pretty printer).

MOTS-CLÉS :IDM, réingénierie, Java5, Spoon, Eclipse Modelling Framework,

1. Présentation

a. Motivations

La réingénierie des systèmes consiste à analyser et modifier un système existant en suivant une approche systématique afin d'y apporter des améliorations. La réingénierie nécessite généralement :

1. une modélisation conceptuelle du système existant (qui peut être obtenue par une rétro-ingénierie de représentations opérationnelles du système)
2. une modification de celle-ci qui fournit une représentation conceptuelle d'un système « à construire », puis
3. un mécanisme de génération qui permet de recréer un nouveau système existant à partir de sa représentation conceptuelle.

Dans ce cadre, l'ingénierie dirigée par les modèles offre des langages et des mécanismes efficaces et attrayants pour représenter un système existant de manière abstraite. Les différentes initiatives pour construire des langages de transformation (ATL [1], QVT [2]) peuvent fournir un bon pouvoir d'expression pour déclarer les modifications à apporter à l'application existante. Cependant, il reste généralement à construire le méta-modèle permettant de représenter un système existant et les outils pour générer, mettre à jour, le système existant en fonction des transformations qui lui sont apportées.
La brique logicielle *SpoonEMF* s'inscrit dans cette problématique générale qui vise à utiliser les techniques de transformation issues de l'IDM dans le cadre de la réingénierie. *SpoonEMF* se place alors dans le cas précis de la réingénierie de programmes Java5.

b. Architecture

SpoonEMF fournit un méta-modèle Java5 complet au format *ecore*[1] (70 concepts) construit à partir du méta-modèle Java5 de Spoon[2] [6] représenté sous forme d'un ensemble d'interfaces Java. *SpoonEMF* fournit une implémentation mettant en œuvre la façade de *Spoon* (voir Figure 1). Cette façade est constituée de cet ensemble d'interface Java qui représente le méta-modèle Java5 de Spoon. La mise en œuvre de cette façade permet de réutiliser l'ensemble des outils fournit par le projet *Spoon*, entres autres l'analyseur syntaxique permettant de construire un modèle à partir d'un projet Java et le générateur de code permettant de recréer le code d'une application existante à partir d'un modèle. Le code généré par le *framework* EMF à partir du méta-modèle permet quant à lui d'être compatible avec les outils de l'IDM comme *Kermeta* [3] ou ATL [1] pour l'expression des transformations. Il est important de noter que *SpoonEMF* est construit à partir de *Spoon*. Ainsi, le méta-modèle Java5 en *ecore* a été construit après analyse du méta-modèle représenté comme un ensemble d'interfaces Java dans *Spoon*.

c. Intérêt et atout

Le principal intérêt de *SpoonEMF* est de pouvoir considérer tout programme Java comme un modèle instance de son méta-modèle. Dès lors, *Spoon* considérant toutes les informations d'un programme java5, il est, par exemple, possible d'exploiter les annotations contenues dans un programme Java comme guides pour la réingénierie. Deuxièmement, par rapport à *Spoon* lui-même, le fait de proposer une implémentation utilisant EMF permet d'utiliser ce méta-modèle dans le cadre de nombreux projets existants comme *Kermeta* ou ATL. Finalement, quand *Spoon* propose une représentation du méta-modèle sous forme

[1] (Formalisme de représentation des méta-modèles du *framework* EMF d'Eclipse, http://www.eclipse.org/emf/)
[2] http://spoon.gforge.inria.fr

d'un ensemble d'interfaces Java, *SpoonEMF* propose la représentation du méta-modèle avec un formalisme adapté à savoir *ecore*. Ainsi, les attributs des concepts du méta-modèle sont décrits sous forme d'attributs (non pas sous forme d'accesseurs), les associations entre les concepts du méta-modèle sont identifiées et décrites. Certains patrons de conception comme le patron visiteur sont retirés du méta-modèle et uniquement mis en œuvre au niveau de l'arbre de syntaxe abstraite. Ces points permettent de simplifier la compréhension du méta-modèle comparé à une description du méta-modèle sous forme d'interfaces Java.

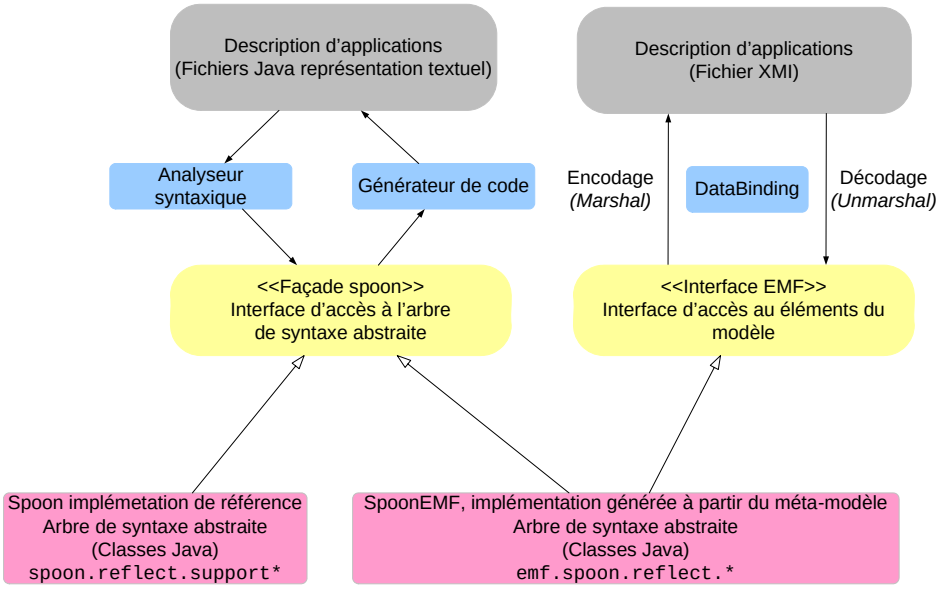


Figure 1 : Architecture de SpoonEMF

2. Perspectives d'utilisation pour des expérimentations scientifiques

SpoonEMF ne se veut pas un résultat scientifique en tant que tel, mais il vise à faciliter l'expérimentation sur des cas de grande taille des propositions autour de l'IDM. Le reste de section présente ainsi un ensemble de cas d'utilisation possibles de *SpoonEMF*.

- En s'inspirant de la proposition du projet COOK[4], *SpoonEMF* peut permettre d'envisager la *fractalisation*[3] de certains codes Java patrimoniaux en identifiant les classes pouvant se représenter sous forme de composants. Le travail de recherche consiste alors à rechercher des heuristiques pour l'identification de composants dans un code objets.
- *SpoonEMF* peut s'intégrer comme une brique élémentaire dans des *frameworks* de migration de code patrimonial. L'objectif scientifique réside alors dans la quantification de l'apport de l'utilisation d'une approche dirigée par les modèles pour la migration de code patrimonial.
- *SpoonEMF* peut aussi être utilisé comme PSM (Platform Specific Model) Java dans des démarches MDA. Ainsi, nous souhaitons par exemple évaluer à l'aide d'une expérimentation la complexité de la construction d'un compilateur Kermeta construit comme une transformation du méta-modèle *Kermeta* vers le méta-modèle Java.
- *L*e méta-modèle Java5 de SpoonEMF est aussi un bon cadre pour la validation empirique de la notion de type de modèle [5]. Cette notion de type de modèle doit permettre entres autres de favoriser la définition de transformation réutilisable pour des modèles ayant des méta-modèles différents. Dans ce cadre, le méta-modèle Java5 de *SpoonEMF* peut être un bon partenaire du méta-modèle UML2 fournit par Eclipse pour cette expérimentation.
- D'une manière générale, pour les approches proposant des mécanismes de transformation ou de composition autour d'EMF, *SpoonEMF* permet de créer rapidement des modèles de grande taille susceptibles de valider les propositions de la communauté scientifique.

SpoonEMF sera distribué dans le cadre d'un MDK (*Model Development Kit*) de Kermeta pour Java5.

[1] F. Jouault and I. Kurtev: Transforming Models with ATL. In: Proceedings of the Model Transformations in Practice Workshop at MoDELS 2005, Montego Bay, Jamaica.
[2] B-K. Appukuttan, T. Clark, A. Evans, G. Maskeri, S. Reddy, P. Sammut, L. Tratt, R. Venkatesh, J. S. Willans -- QVT-Partners revised submission to MOF 2.0 Query/View/Transformations RFP -- OMG Document ad/03-08-08, August 2003
[3] 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.
[4] S. Ducasse, I. Alloui, S. Cimpan, H. Verjus et M-P. Huget -- *COOK: Réarchitecturisation des applications industrielles objets* -- projet GIP-ARN (JC05 42872)
[5] J. Steel and J-M. Jézéquel. -- Model typing for improving reuse in model-driven engineering. -- In S. Kent L. Briand, editor, *Proceedings of MODELS/UML'2005*, volume to be published of *LNCS*, pages --, Montego Bay, Jamaica, October 2005. Springer.
[6] R. Pawlak, C. Noguera and N. Petitprez. Spoon: Program Analysis and Transformation in Java. INRIA Research Report #5901, May 2006.

[3] Construction d'une application à base de composants Fractal, http://fractal.objectweb.org