# Motif pour la métamodélisation: Flot de contrôle

Mickael Kerboeuf, Alain Plantec, Vincent Ribaud

# Motif pour la métamodélisation

## Flot de contrôle

**Mickaël Kerboeuf, Alain Plantec, Vincent Ribaud**

*LISyC - EA3883, Université de Bretagne Occidentale*
*équipe IDM (Ingénierie Des Modèles)*
*C.S. 93837, 29238 Brest cedex 3*
*{kerboeuf,plantec,ribaud}@univ-brest.fr*

RÉSUMÉ. *Les langages à domaine spécifique (DSL) ne sont pas nécessairement purement déclaratifs. Ils peuvent intégrer des aspects procéduraux dont l'analyse passe par la construction d'un graphe de flot de contrôle. Le motif «Flot de contrôle» permet de construire la syntaxe abstraite d'un DSL procédural disposant d'instructions de branchement.*

ABSTRACT. *Domain Specific Languages (DSL) are not necessarily purely declarative. They can have procedural features in which case their analysis implies the building of a control flow graph. The model pattern "Control Flow Graph" (CFG) describes the way to build the abstract syntax of a procedural DSL provided with branch instructions.*

MOTS-CLÉS : *flot de contrôle, DSL, langage procedural, syntaxe abstraite*

KEYWORDS: *control flow, DSL, procedural language, abstract syntax*

1re soumission à *MP2007 – Deuxième atelier de travail sur les motifs de métamodélisation*, le 5 février 2007

## 1. Motivation

Un DSL[1] (Thibault, 1998) est un langage dédié à l'expression et à la manipulation des concepts d'une même classe d'applications (*i.e.* d'un domaine métier particulier). On retrouve donc naturellement dans le métamodèle d'un domaine les éléments de la syntaxe abstraite d'un DSL de ce même domaine.

Un DSL n'est pas nécessairement purement déclaratif, *i.e.* capable d'exprimer un problème sans précision concernant le calcul de sa résolution. Il peut intégrer des aspect procéduraux qui définissent précisément des traitements sur les données du domaine (Smith, 1982, Spinellis, 2006).

Le type d'un langage détermine les moyens d'analyser les mots reconnus de ce langage. La sémantique d'un langage déclaratif est naturellement dénotationnelle. Un mot reconnu d'un langage déclaratif peut être *évalué*, *i.e.* associé à une valeur dans un espace logique. Par exemple, une requête SQL est naturellement associée à une relation calculée avec les opérateurs de l'algèbre relationnel. *A contrario*, un mot reconnu d'un langage procédural est typiquement associé à une séquence de règles appliquées d'une sémantique opérationnelle. Il peut être *exécuté* comme un programme dans un langage impératif. L'analyse des mots reconnus d'un DSL qui integre des aspects procéduraux implique alors l'analyse de son *graphe flot de contrôle*, lequel reflète l'ordre d'exécution des instructions. On distinguera les instructions de manipulation des concepts du domaine et les instructions de contrôle qui induisent des branchements dans le graphe de flot de contrôle.

## 2. Indications d'utilisation

On utilisera ce motif dans la conception d'un DSL incluant des enchaînements d'instructions de calcul, avec classiquement des itérations, des conditionnelles, et des appels de procédures comme dans le cas de PL/SQL, extension procédurale de SQL.

L'application de ce motif consiste à instancier les concepts d'un graphe (sommets et arcs valués) dans le cas particulier des instructions de contrôle d'un DSL procédural.

Le graphe de flot de contrôle ainsi modélisé permet d'appliquer des techniques d'optimisation (exemple : détection de code mort), de vérification de propriétés (exemple : accessibilité d'une valeur de variable), et de transformation syntaxique (exemple : script textuel vers diagramme d'activité).

## 3. Structure

Voir figure 1.

---

1. *Domain Specific Language*

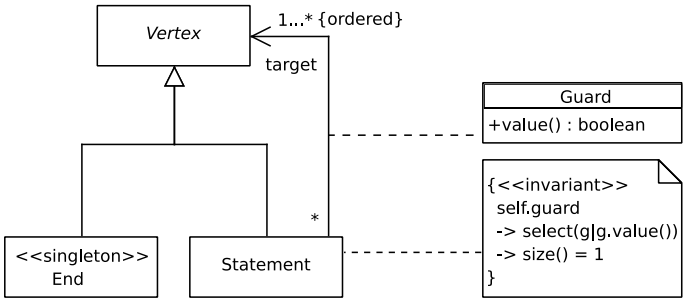


**Figure 1.** *Structure du métamodèle «Flot de contrôle»*

## 4. Constituants

### 4.1. *Vertex*

Un objet de la classe abstraite Vertex représente un sommet dans le graphe de flot de contrôle.

### 4.2. *Statement*

La classe Statement représente les instructions du langage qui doivent être évaluées en séquence. Une instance de cette classe dispose au minimum d'un successeur dans le graphe de flot de contrôle. La liste de ces successeurs, nommée target, est triée selon un ordre d'évaluation.

### 4.3. *End*

La classe End dont l'instanciation est restreinte à un unique type d'instruction est un marqueur de fin d'exécution. Elle représente un état *puit* dans le graphe de flot de contrôle.

### 4.4. *Guard*

La classe-association Guard relie des instructions à un ou plusieurs successeurs. La méthode value de cette association permet de sélectionner l'unique successeur d'une instruction. La contrainte OCL exprime en substance l'existence et l'unicité d'un successeur pour chaque instruction.

## 5. Exemple

Le programme de la figure 2 est un exemple de script dans un dialecte de C constitué d'instructions basiques interprétées en séquence. Les instructions `while` et `if` introduisent des branchements dans le flot de contrôle.

```
int i;
int x;
i = 0;
x = 10;
while ( i < x ) {
   if ( i%2 == 0 )
      printf("even");
   else
      printf("odd");
   i++;
}
printf("\n");
```

**Figure 2.** *Exemple de code procédural*

En instanciant le métamodèle (niveau M2 du MOF (Object Management Group, 2006)), on obtient la syntaxe abstraite de ce langage en précisant la nature des branchements dans le flot de contrôle (figure 3, niveau M1 du MOF en gris).

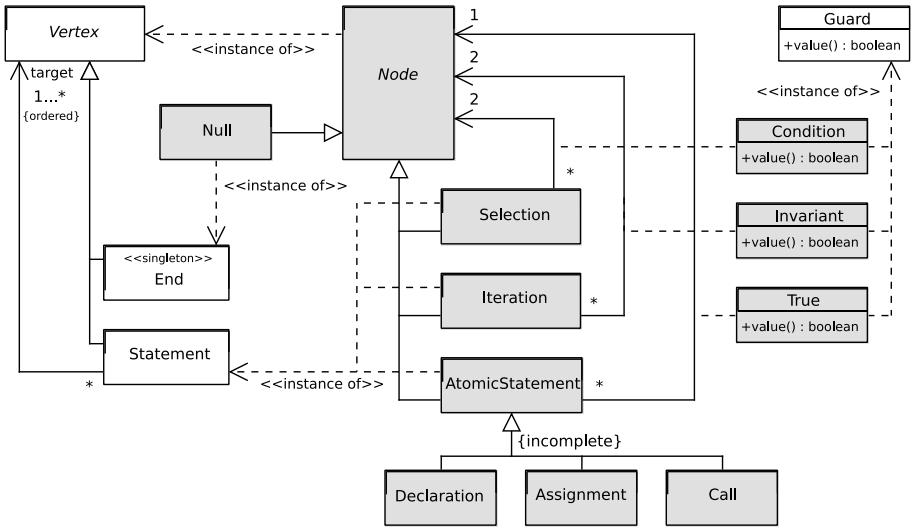


**Figure 3.** *Modèle de DSL procédural*

Le branchement de l'instruction `if` (classe Selection) dépend de l'évaluation d'une condition dénotée par la classe Condition. Le branchement de l'instruction `while` (classe Iteration) dépend d'un invariant de boucle dénoté par la classe Invariant. Le branchement d'une instruction basique (classe AtomicStatement) n'est pas soumis à condition : la méthode value de l'association reliant ce type d'instruction à son unique successeur renvoie toujours la valeur *vrai* (classe True).

Cette version très simple n'intègre pas les notions de *bloc* d'instructions et d'*appel* de procédure qui devraient dériver dans les deux cas de la classe Statement. Dans le cas d'un bloc d'intructions, il faudrait préciser en plus qu'il est constitué d'objets ordonnés, eux-mêmes de la classe Statement.

Le diagramme d'objet de la figure 4 est le graphe de flot de contrôle du programme de la figure 2. Pour simplifier les notations, seules les expressions evaluées par les différentes méthodes value sont affichées sur les liens entre les instances d'instructions.

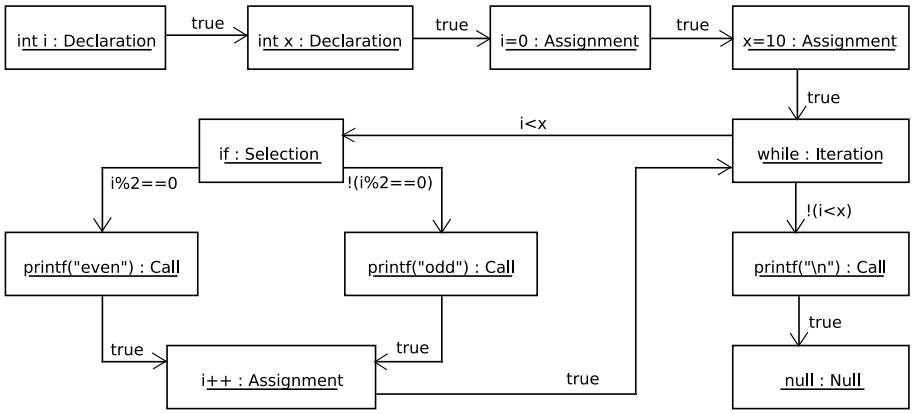


**Figure 4.** *Diagramme d'instances du programme de la figure 2*

## 6. Bibliographie

Object Management Group, *Meta-Object Facility*, `http://www.omg.org/technology/documents/formal/MOF_Core.htm`, 2006.

Smith B. C., Reflection and Semantics in a Procedural Language, PhD thesis, Massachusetts Institute of Technology, January, 1982.

Spinellis D., « Choosing a Programming Language », *IEEE Software*, vol. 23, n° 4, p. 62-63, July/August, 2006.

Thibault S., Langage Dédiés : Conception, Implémentation et Application, Thèse de doctorat, Université de Rennes 1, France, October, 1998.