De l'hypergénéricité aux modèles métiers exécutables

Pierre Crescenzo

Projet ***Objets et Composants Logiciels*** (OCL)
Laboratoire I3S

jeudi 8 avril 2004

Un modèle hypergénérique (1)

Pourquoi faire ?

- pour modéliser les langages de programmation orientés objets

➔ pour les comparer sur des critères précis

➔ pour offrir des capacités de modification ou d'adaptation aux besoins des programmeurs

Un modèle hypergénérique (2)

La démarche, pragmatique :

- étudier les langages de programmation orientés objets les plus répandus
- → en extraire les principaux concepts et les relations qui les lient
- → déterminer ainsi des critères de comparaison pertinents et simples

Un modèle hypergénérique (3)

S'approcher du but :

- définir un modèle qui met en évidence les *sortes* de *classes* et de *relations interclasses*
- permettre la définition de concepts, relations et comportements nouveaux
- tirer parti des critères de comparaison déterminés

Un modèle hypergénérique (4)

Dans les faits, ça donne quoi ?

le modèle *Open Flexible Languages* (OFL)

C'est un modèle métaobjet.

Chaque entité *(classe, méthode, attribut, objet,...)* dispose d'une métaentité qui définit sa structure *(état)* et son comportement *(sémantique opérationnelle)*.

Un modèle hypergénérique (5)

Dans les faits, ça donne quoi encore ?

le modèle *Open Flexible Languages* (OFL)

C'est un modèle hypergénérique.

Chaque métaentité est paramétrée en fonction des critères de comparaison : il suffit de changer la valeur d'un paramètre pour modifier le comportement des entités décrites et, par conséquent, de tout le système (d'où le terme *paramètres hypergénériques*).

Un modèle hypergénérique (6)

Un exemple ! Un exemple ! (le début)

- Chaque classe est instance d'une métaclasse.
  Pour Java, on a, entre autres :
  - `MétaClasseConcrète`
  - `MétaClasseAbstraite`
  - `MétaInterface`
- Chaque relation est instance d'une métarelation.
  Pour Java, on a, entre autres :
  - `MétaHéritageInterClasses`
  - `MétaHéritageInterInterfaces`
  - `MétaImplémentation`
  - `MétaAgrégation`

Un modèle hypergénérique (7)

Un exemple ! Un exemple ! (la suite)

- Chaque métaclasse est définie par des paramètres.
  Les paramètres de `MétaClasseConcrète` indiquent notamment :
  - que les classes concrètes de Java peuvent créer des instances,
  - qu'elles acceptent la surcharge,…
- Chaque métarelation a aussi ses paramètres.
  Les paramètres de `MétaHéritageInterClasses` indiquent :
  - que ces héritages sont simples (non multiples),
  - que ces héritages ne peuvent pas être cycliques,…

Un modèle hypergénérique (8)

Un exemple ! Un exemple ! (la fin)

Pour rendre l'héritage de Java multiple (comme dans C++ ou Eiffel), il doit suffire de modifier la valeur du paramètre correspondant.

Présent et futur : des modèles métiers exécutables

- Idée : généraliser le travail fait sur les langages à objets à d'autres métiers.
- Généralisation de : métadescription, paramètres hypergénériques, actions exécutables, assertions, capture de la sémantique opérationnelle, plate-forme logicielle…
- Technologies proches : UML exécutable, langage de (méta) modélisation, plate-forme de composants, protocoles métaobjets, modèles de contrat, MDA, programmation générative…
- Premiers modèles cibles : une centrale nucléaire (sur un contrat avec INRIA/EDF) et les langages orientés objets (comme modèle de référence)