Ingénierie des modèles

Transformations de modèles

Master : Systèmes d'Information et Données

A. Mebarki - Maître de Conférences

USTO - MB

abdelkrim.mebarki@univ-usto.dz

http://amebarki.visiondz.info/teach.php#MDE

2020 - 2021

Plan du chapitre

L'ingénierie dirigée par les modèles

Transformations

- Opération générale de manipulation de modèles
- Prend un ou plusieurs modèles en entrée et fournit un ou plusieurs modèles en sortie.

Transformation endogènes

- Dans le même espace technologique.
- Comme transformer un modèle UML en un autre modèle UML.

Transformations exogènes

- Entre deux espaces technologiques différents.
- Comme transformer un modèle XML en un schéma BDD.

Transformation de modèles

Exemple : Transformation d'un programme en C

- Transformation en un autre programme C.
- Compilation de ce programme en générant le code machine.
- Optimisation du code machine pour une architecture de processeur donné.

Transformation de modèles

Pour réaliser des transformations

- Définir un langage de transformation via un méta-modèle de transformation.
- Disposer d'un repository référentiel ou dépôt pour stocker les modèles manipulés.
- Disposer de méta-modèles pour représenter les structures des modèles et les stoker également dans des repository.

Transformation de modèles

Les familles de modèles

- Données sous forme de séquences : Fichiers texte.
- Données sous forme d'arbre : XML (Transformations par XSLT)
- Données sous forme de graphes : UML (Transformation par QVT)

Transformation de modèles

Techniques de transformation

- Approche déclarative
- Approche impérative
- Approche hybride

Transformation de modèles

Approche déclarative

- Recherche de certains patrons (d'éléments et de leurs relations) dans le modèle source.
- Chaque patron trouvé est remplacé dans le modèle cible par une nouvelle structure d'éléments.
- Écriture de la transformation simple mais ne permet pas toujours d'exprimer toutes les transformations facilement.

Transformation de modèles

Approche impérative

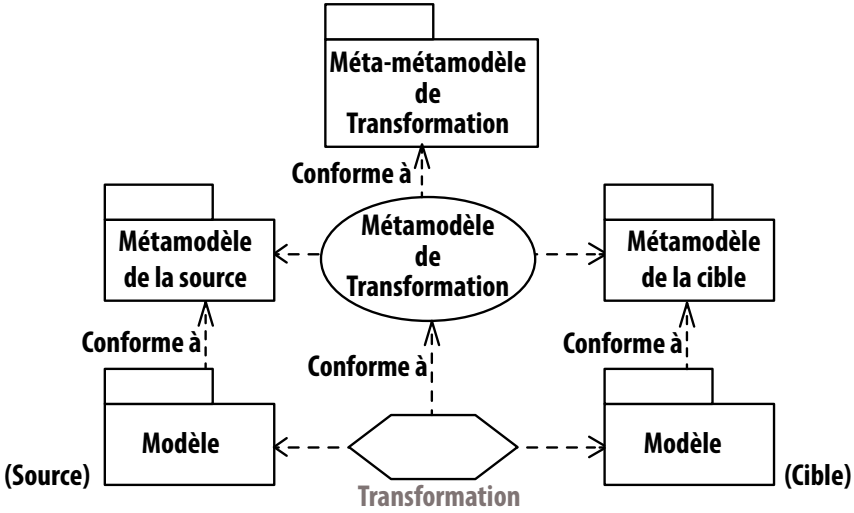
- Proche des langages de programmation usuels.
- On parcourt le modèle source dans un certain ordre et on génère le modèle cible lors de ce parcours.
- Écriture de la transformation peut être plus lourde mais permet de tout définir, notamment les cas algorithmitiquement complexes.

Transformation de modèles

Le Repository

- C'est un référentiel pour stocker les modèles et les méta-modèles.
- La forme de stockage dépend de l'outil et du formalisme.
- Nous stockons : Les métamodèles (conformes à MOF), les modèles (conformes aux métamodèles), et le métamodèle de la transformation.

Transformation de modèles



QVT de l'OMG

Query View Transformation

- Query : sélectionner des éléments sur un modèle.
  - Le langage utilisé pour cela est OCL légèrement modifié et étendu avec une syntaxe différente et simplifiée.
- View : Une sous-partie d'un modèle.
  - Une vue est un modèle à part, avec éventuellement un métamodèle restreint spécifique à cette vue

QVT de l'OMG

Les 3 langages de QVT

- QVT-Relation (Mode déclaratif : de haut niveau)
- QVT-CORE (Mode déclaratif : de bas niveau)
- QVT-Operational (Mode impératif ou hybride)

La syntaxe

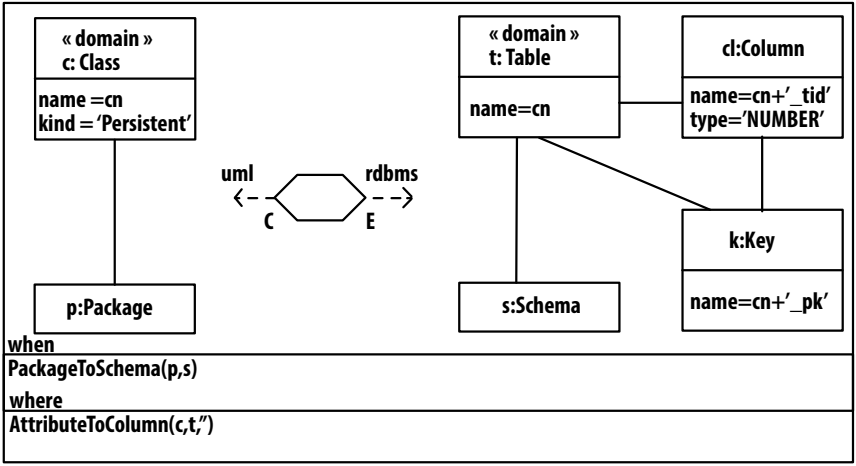
- Textuelle ou graphique selon le langage

QVT de l'OMG

Exemple de QVT-Relation

- Transformation UML, SGBDR
- Transformer un modèle de données UML en un schéma de données relationnel (et inversement)
- Chaque package UML correspond à un schéma de BDD, chaque classe persistante à une table, chaque attribut de classe à une colonne de table . . .

La syntaxe graphique



Transcription et contraintes

QVT

- `transformation umlRdbms (uml: SimpleUML, rdbms: SimpleRDBMS) { ...`
- Une transformation a un nom : umlRdbms
- Elle porte sur des modèles candidats. Ici deux modèles candidats *uml* et *rdbms*
- Chaque modèle candidat est typé par un méta-modèle : *SimpleUML* et *SimpleRDBMS*
- Une transformation contient des relations qui doivent être vraies sur les modèles candidats pour que la transformation soit réussie.
- Une direction de transformation peut être précisée.

Transcription et contrainte

QVT

- `relation PackageToSchema { domain uml p:Package { name = pn } domain rdbms s:Schema { name = pn } }`
- Une **relation** spécifie des contraintes qui doivent être satisfaites sur les éléments de modèles candidats d'une transformation.
- **Domaine** = motif typé qui correspond à une partie d'un modèle candidat :
  - Un paquetage p de *uml* qui a pour nom pn
  - Un schéma s de *rdbms* de nom pn

Transcription et contrainte

QVT

- Les différents domaines (2 ici) doivent pouvoir être mis en correspondance :
  - Pour chaque paquetage, il doit y avoir un schéma de même nom
  - Pour chaque schéma, il doit y avoir un paquetage de même nom
- Clause **when** : condition sous laquelle la relation doit être vérifiée.
- Clause **where** : condition que doivent vérifier tous les éléments de modèle participant à la relation. Permet de contraindre les variables libres de la relation et ses domaines.

Transcription et contrainte

Contrat de transformations de modèles

- Contraintes OCL :
  - Contraintes sur le modèle source
  - Contraintes sur le modèle cible
  - Contraintes sur les relations entre éléments des deux modèles (Contraintes du passage)