1ère année Master GLSD

Département d'Informatique

Université de Biskra

Année 2020/2021

Ingénierie des Modèles
Transformations de modèles

Dr. Mohamed Lamine Kerdoudi

Email: l.kerdoudi@univ-biskra.dz

Transformation

Qu'est-ce une transformation?

Une transformation prend en entrée des modèles (**sources**) et fournit en sortie des modèles (**cibles**)

- **Transformation endogène**
  - ➢ Dans le même espace technologique
  - ➢ Les modèles source et cible sont conformes au même méta-modèle
    - ❑ Transformation d'un modèle UML en un autre modèle UML
- **Transformation exogène**
  - ➢ Entre 2 espaces technologiques différents
  - ➢ Les modèles source et cible sont conformes à des méta-modèles différents
    - ❑ Transformation d'un modèle UML en programme Java
    - ❑ Transformation d'un fichier XML en schéma de BDD

Transformation

Dans l'outil de transformation, il y a une **Définition de Transformation.**

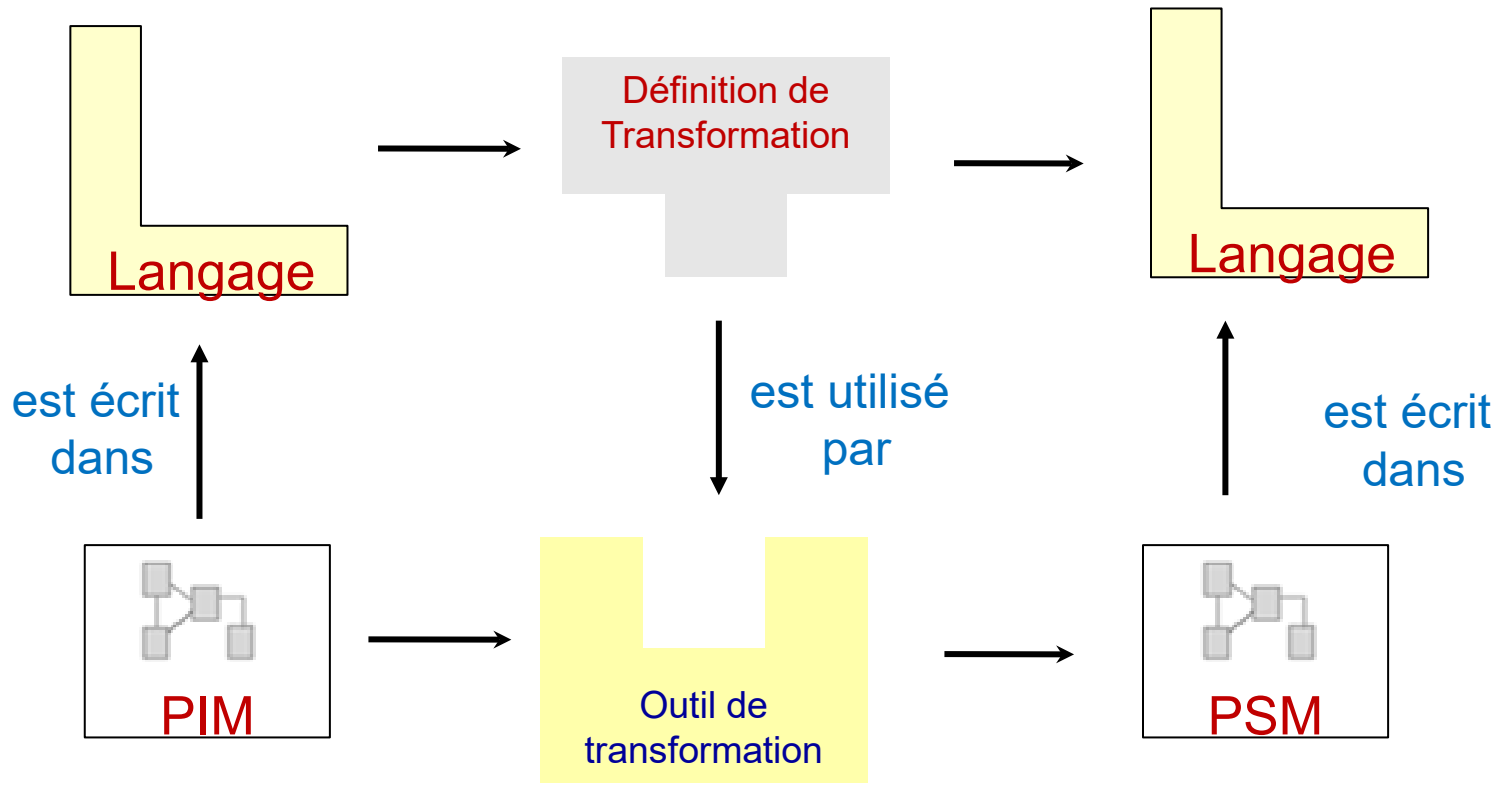
Qu'est-ce qu'une définition de la transformation ?

➔ Elle décrit comment le modèle devrait être transformé.

➔ Est un ensemble de règles de transformation.

❑ Une règle de transformation est une **description** de la manière dont une ou plusieurs **constructions** dans un **langage source** peuvent être transformés en une ou plusieurs **constructions** dans le **langage cible**.

Le Framework MDA de base



- Un développeur **focalise sur le développement d'un PIM**, décrivant le système de logiciel à un haut niveau d'abstraction.
- Il choisit un ou plusieurs **outils de transformation**.
- Les **PSMs résultats** peuvent être **transformé en code**

Autre vision des transformations

- Les processus de développement automatisé et basé sur les **transformations** sont utilisés depuis longtemps.

➔ Rien de totalement nouveau

- **Adaptation** à un nouveau contexte

➔ **Exemple** : compilation d'un programme C

- **Programme C** : modèle abstrait
- **Transformation de ce programme** sous une autre forme mais en restant à un niveau **abstrait**
  - **Modélisation**, représentation différente du programme C pour le manipuler : transformation en un modèle équivalent
  - **Exemple** : **arbres décorés** : arbre syntaxique dont les nœuds peuvent être décorés (informations complémentaires)
- **Génération du code** en langage machine
  - Avec optimisation pour une architecture de processeur donnée

Transformations : types d'outils

- 3 grandes familles de modèles et outils associés

1) Données sous forme de séquence

   - **Ex : fichiers textes** (AWK: le langage script)

2) Données sous forme d'arbre

   - **Ex : XML** (XSLT)

3) Données sous forme de graphe

   - **Ex : diagrammes** UML
   - **Outils**
     - Transformateurs de graphes déjà existants
     - Nouveaux outils du MDE et des Atelier de Génie Logiciel (**QVT, J, ATL, Kermeta** ...)

Techniques de transformations

3 grandes catégories de techniques de transformation

- **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'élément
  - Ecriture de la transformation « assez » simple mais ne permet pas toujours d'exprimer toutes les transformations facilement
- **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
  - Ecriture transformation peut être plus lourde
- **Approche hybride** : à la fois déclarative et impérative
  - La plupart des approches déclaratives offrent de l'impératif en complément car plus adapté dans certains cas

Transformations en EMF

- Plusieurs langages pour le framework EMF/Ecore
  - Directement via le framework **Java** EMF
    - Chaque élément du méta-modèle Ecore correspond à une classe Java avec ses interfaces d'accès aux éléments de modèles
    - **Pb** : reste de la progammation en Java, pas totalement adapté à la manipulation de modèles
- Langages de plus haut niveau,
  - **ATL** : approche déclarative
    - Une **règle de transformation** sélectionne un élément dans le source et génère un (ou plusieurs) élément(s) dans le cible
    - La sélection se fait par des **query OCL**
  - **Kermeta** : approche impérative
    - Langage de programmation **orienté-objet**
    - **Primitives dédiées** pour la manipulation de modèles
    - **Ajout** d'opérations/attributs sur méta-éléments par **aspect**
    - **Intègre** aussi **un langage de contraintes similaire à OCL**
      - **Invariant** sur méta-élément et pré/post sur leurs opérations

Ajout de proxy : comparaison

- **ATL**
  - **Purement déclaratif**
  - A un type d'élément source, on associe un (ou deux) type(s) d'élément cible
    - **Ne se préoccupe pas de savoir comment les éléments sont retrouvés dans le modèle**
    - Accède à des éléments crées dans d'autres règles sans de préoccuper de l'ordre de leurs créations
- **Kermeta**
  - **Approche impérative**
  - **Doit explicitement naviguer sur les modèles**
    - **sources** : pour récupérer les éléments à dupliquer avec modifications requises
    - **cibles**: pour en créer les éléments un par un

Questions ?