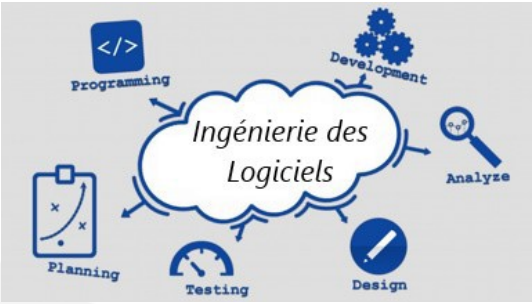
Ingénierie des Logiciels



Dr. Said MEGHZILI

Centre Universitaire de Mila

Département de Mathématique et Informatique

Email: s.meghzili@univ-mila.dz

1.0

Février 2022

Table des matières

I Chapitre 02 : Ingénierie Dirigée par les Modèles

1. Introduction

Exemple : Évolution des technologies

- Les technologies logicielles sont en Évolution permanente.
- Exemple dans les systèmes distribués.
    - Évolution pour faire communiquer et interagir des éléments distants: 1-C et sockets TCP/UDP →2-C et RPC→3-C++ CORBA → 4-Java et RMI→5-C# et Web Service→6-Java et EJB→7- A suivre .....

  Avantages: développement plus rapide et plus efficace.
- Si on veut profiter des nouvelles technologies et de leurs avantages :
    - Nécessite d'adapter une application à ces technologies.
    - Le coût de cette adaptation est très élevé donc on doit réécrire le code d'application.
- Exemple :Application de calculs scientifiques distribués sur un réseau de machines Passage de C/RPC (Paradigme procédural)à Java/EJB (objet/composant):
    - Impossibilité de reprendre le code existant (différence des Paradigmes)
- Nécessité de découpler ***la logique métier*** et de ***la mise en œuvre*** technologique
    - C'est l'un des principes fondamentaux de l'ingénierie dirigée par les modèles : ***Séparation des préoccupations***

Définition : Ingénierie Dirigée par les Modèles

L'Ingénierie IDM est une approche de développement qui met à la disposition de l'utilisateur des outils, des concepts et des langages afin de simplifier et de mieux maîtriser le processus de développement de systèmes. En plus, elle permet aussi d'augmenter la productivité, la qualité, la réutilisabilité et l'évolution de ces systèmes [5]*.

Les modèles sont considérés comme des éléments de base. Les outils permettant de créer et d'exploiter ces modèles sont construits autour des concepts de:

- Méta-modélisation
- Transformation de modèles

Fondamental

Que propose l'IDM ?

- Besoin de modéliser/spécifier
  - A un niveau abstrait ***la partie métier***
  - La plate-forme de ***mise en oeuvre***
  - De projeter ce niveau abstrait sur une plateforme
- Modéliser les applications à un haut niveau d'abstraction
  - Place le ***modèle*** au cœur du processus de conception
  - ***Transformation*** de modèles pour passer d'un niveau à l'autre
  - ***Génère le code*** de l'application à partir des modèles
  - Possibilité de ***contrôler***, ***simuler*** et ***tester*** à différents niveaux

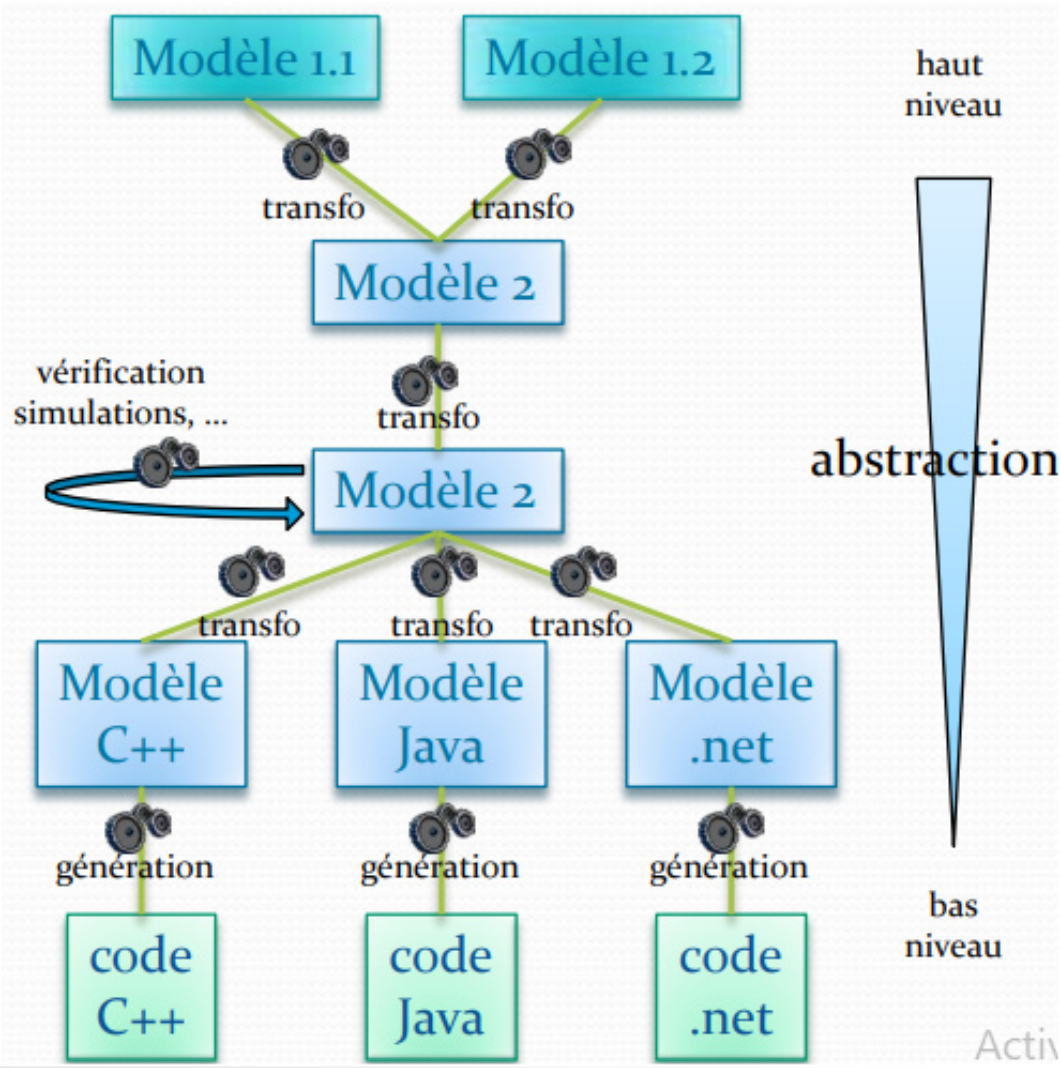


Figure 2.1 : Exemple de transformations de modèles

2. Modèles

Définition : Qu'est-ce qu'un modèle?

- Description / abstraction des objets du monde réel.
- Quelque chose avec une méta description de la façon dont il devrait être structuré.
- Objets et relations (un graphique).
- Vue simplifiée d'un système.

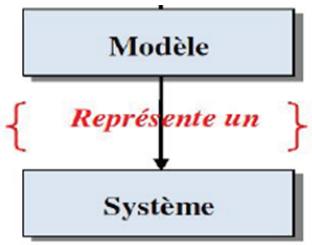


Figure 2.2 : Relation entre système et modèle

Fondamental : But d'un modèle

- Faciliter la compréhension d'un système complexe
- Simuler le fonctionnement d'un système complexe

Exemple : Modéles

- Réseaux de Petri (Rdp)
- Diagrammes UML
- Modèle BPMN
- Modèle Entité/Association
- Modèle météorologique

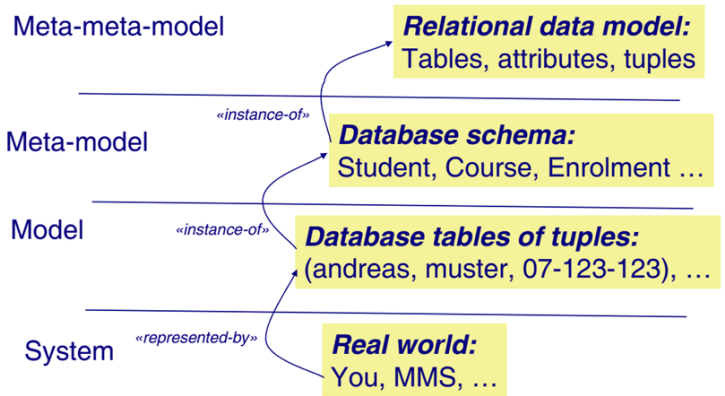


Figure 2.3 : Exemple de l'architecture à quatre niveaux dans les Bases de Données

3. Méta-modèles

3.1. Qu'est-ce qu'un méta-modèle?

- Pour passer à une vision productive, il faut que les modèles soient bien définis.
- La définition d'un langage de modélisation prend la forme d'un modèle, appelé Méta-modèle.
- Un méta-modèle est un modèle qui définit précisément les concepts manipulés dans les modèles ainsi que les relations entre ces concepts.
- Base de l'IDM permettant de développer des outils capables de manipuler les modèles.
- Assurer la correction syntaxiques des modèles.

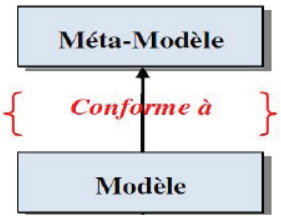


Figure 2.4 : Relation entre modèle et méta-modèle

Fondamental : But de la méta-modélisation

Définir des langages de modélisation ou des langages de manière générale.

3.2. Principales normes de modélisation OMG

- ***MOF*** : Meta-Object Facilities
  Langage de définition de méta-modèles
- ***UML*** : Unified Modelling Language
  Langage de modélisation
- ***CWM*** : Common Warehouse Metamodel
  Modélisation ressources, données, gestion d'une entreprise
- ***OCL*** : Object Constraint Language
  Langage de contraintes sur modèles
- ***XMI*** : XML Metadata Interchange
  Standard pour échanges de modèles et méta-modèles entre outils

3.3. Hiérarchie de Modélisation à 4 niveaux

Définition

L'OMG définit les quatre niveaux de modélisation suivants [4]*:

- M0 : système réel, système modélisé
- M1 : modèle du système réel défini dans un certain langage
- M2 : méta-modèle définissant ce langage
- M3 : méta-méta-modèle définissant le méta-modèle (M3 est le MOF)

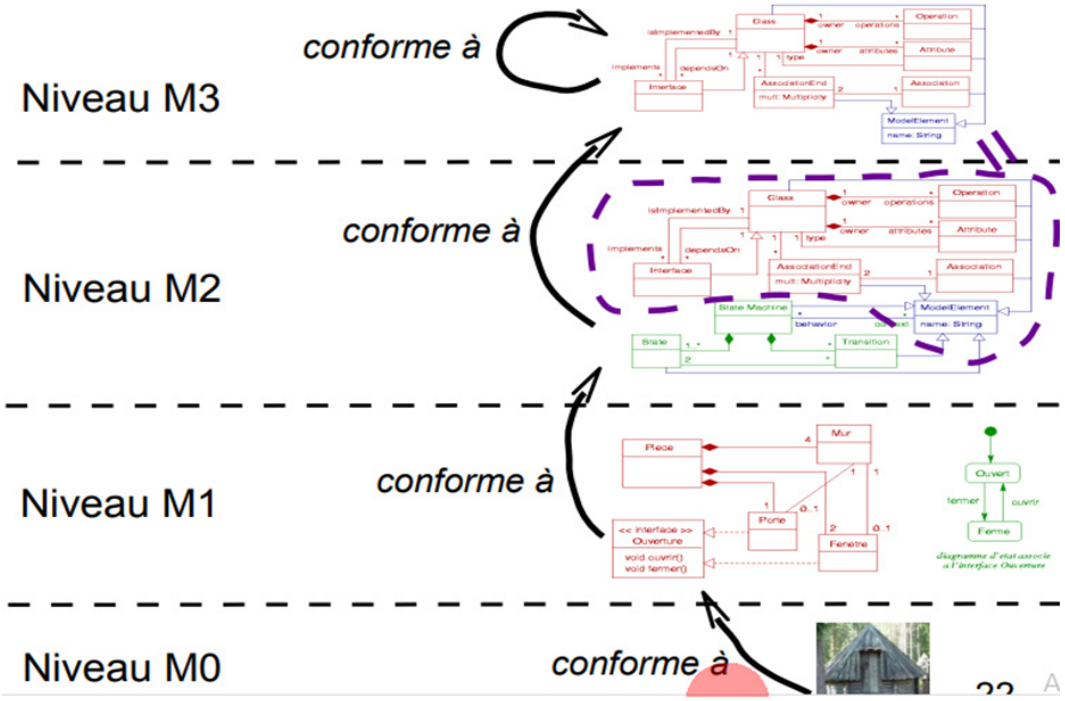


Figure 2.5 : Exemple Hiérarchie de Modélisation

3.4. Syntaxe abstraite et concrète d'un modèle

Définition

Un modèle peut être défini via la syntaxe abstraite ou concrète :

- Syntaxe Abstraite :
  - Les éléments et leurs relations sans une notation spécialisée.
  - Correspond à ce qui est défini au niveau du méta-modèle, à des instances de méta-éléments.
- Syntaxe Concrète :
  - Syntaxe graphique ou textuelle définie pour un type de modèle.
  - Plusieurs syntaxes concrètes possibles pour une même syntaxe abstraite.

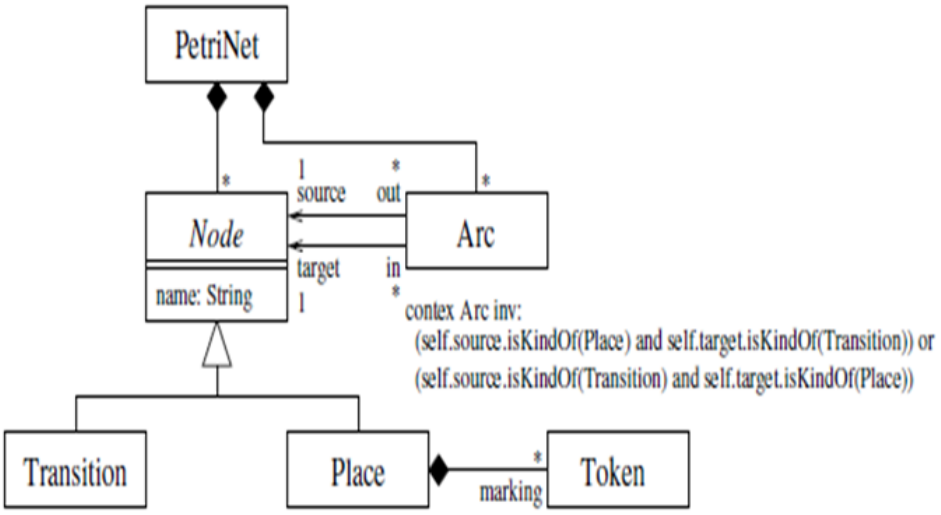


Figure 2.7 : Méta-modèle des Réseaux de Petri

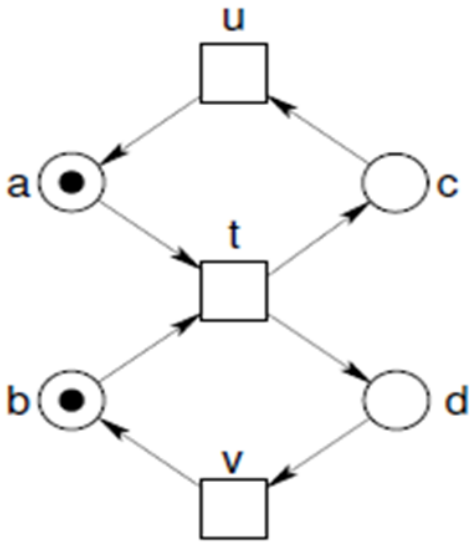


Figure 2.6 : Syntaxe Concrète d'un modèle de réseau de Petri

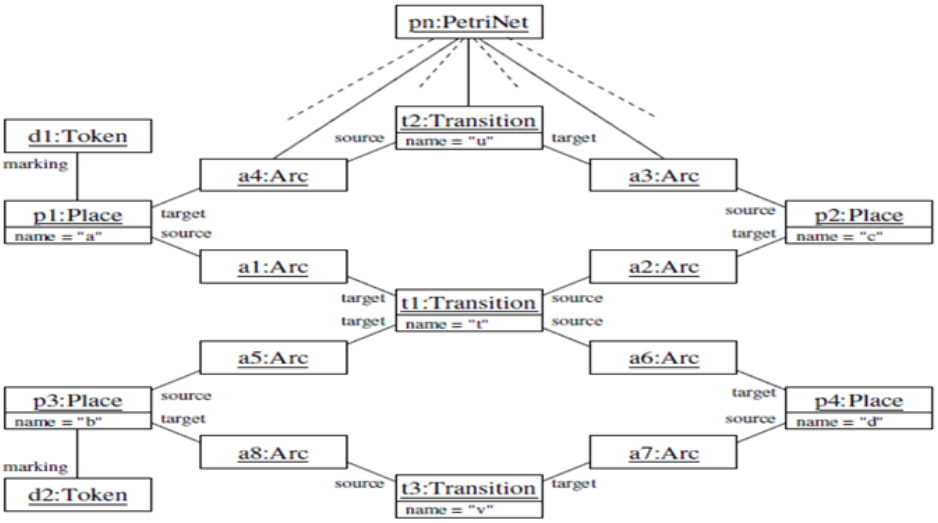


Figure 2.8 : Syntaxe Abstraite d'un modèle de réseau de Petri

4. Transformation de Modèles

Définition : Principe de transformation de modèles

- Une transformation de modèles consiste à passer d'un modèle source à un modèle cible.
- Différentes sémantiques :
    - Optimisation
    - Génération de code
    - Raffinement
- Deux grandes classes de transformation de modèles [24]*:
    - Transformations de type: Modèle vers Code (M2T).
    - Transformations de type: Modèle vers Modèle (M2M).

- Le passage de l'un à l'autre est décrit par des règles de transformation.
- Ces règles sont exécutées sur les modèles sources afin de produire les modèles cibles.
- Une transformation endogène : modèles source et cible conformes au même méta-modèle.
- Une transformation exogène : modèles source et cible conformes à des méta-modèles différents.

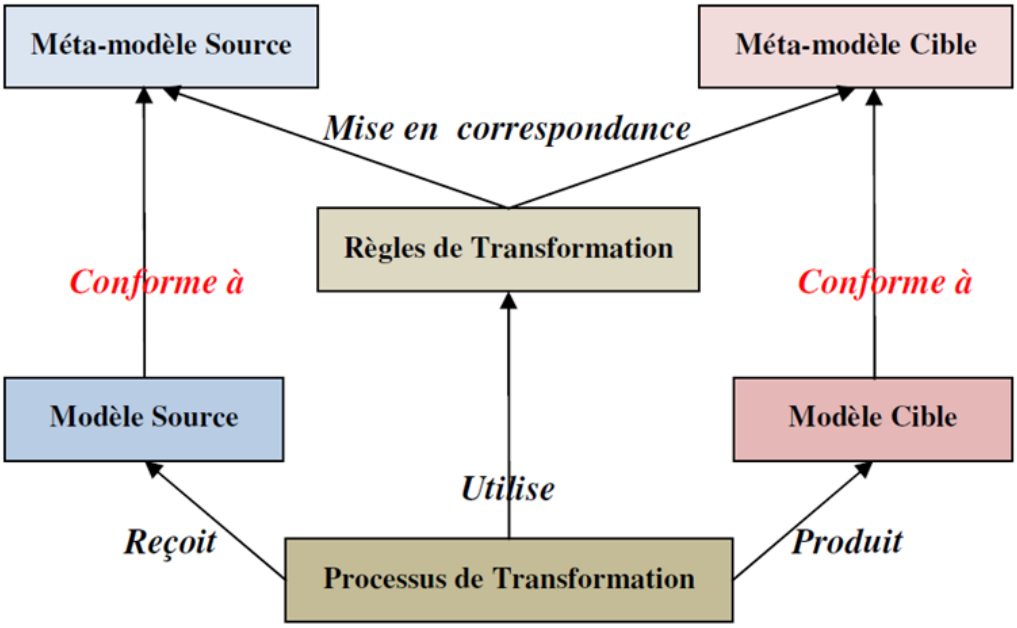


Figure 2.9 : Concepts de base de la transformation de modèles

Exemple : Transformations de modèles

- M2M exogène : Transformation des diagrammes UML vers les Réseaux de Petri.
- M2T :Génération de code Java à partir des diagrammes UML.
- M2M exogène : Transformation d'un diagrammes UML vers un schéma de BDD.
- M2M endogène Réduction des Réseaux de Petri.

5. Architecture MDA

Définition : Principe de la vision MDA

- MDA [3]*, [6]*, [7]* vise à modéliser des applications et des systèmes indépendamment de l'implémentation cible (niveau matériel ou logiciel), ce qui permet la réutilisation des modèles développés.
- Les modèles ainsi créés (PIM :Platform Independant Model) seront transformés pour obtenir des modèles d'applications spécifiques à la pateforme cible (PSM : Platform Specific Model).
- Des outils de génération automatique du code, permettent par la suite de générer les programmes directement à partir des modèles.
- Cette approche, permet en plus de faire évoluer aisément les applications et leurs architectures à partir des modèles.
- Par conséquent, Le MDA s'investit dans un grand atelier dont le défit relève techniquement de la manipulation des modèles. Dans ce cadre spécifique, la transformation de modèle basé sur les techniques de Méta-modélisation et l'ingénierie de modèles ouvre un champ d'investigation prometteur pour la recherche.

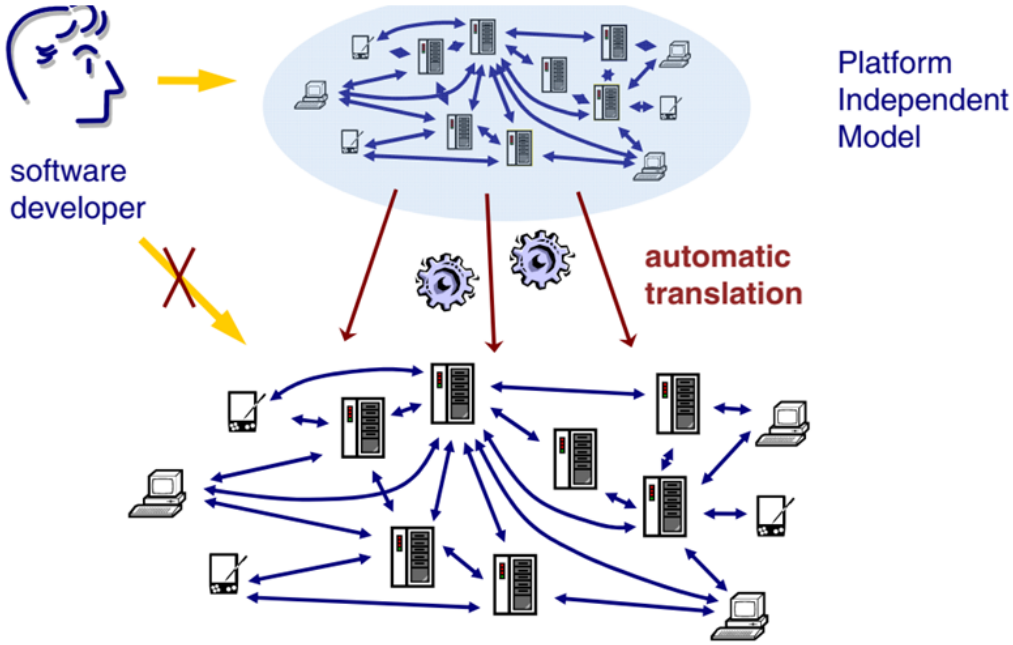


Figure 2.10 : La vision MDA

Complément : Principaux niveaux de modèles MDA

Le MDA définit deux principaux niveaux de modèles [4]*:

- PIM : Platform Independent Model
  - Modèle spécifiant une application indépendamment de la technologie de mise en œuvre.
  - Uniquement spécification de la partie métier d'une application.
- PSM : Platform Specific Model
  - Modèle spécifiant une application après projection sur une plate-forme technologique donnée.
  - Relation entre les niveaux de modèles.

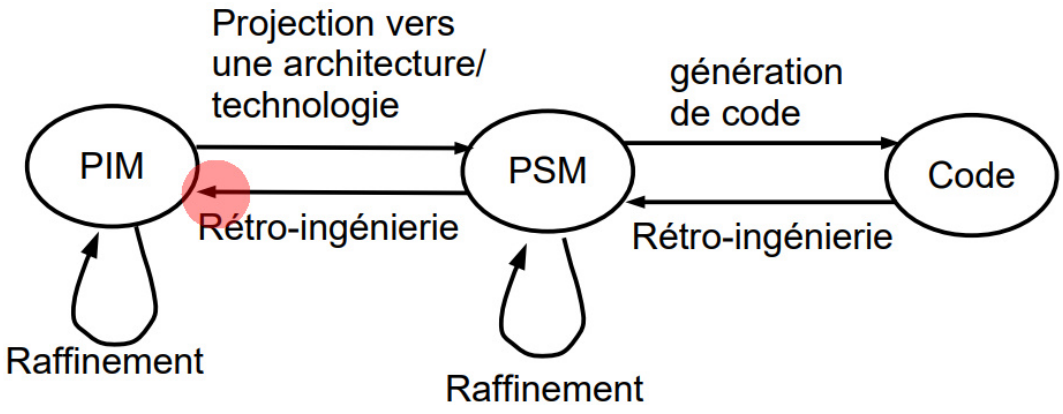


Figure 2.11 : Les modèles et les transformations dans l'approche MDA

6. Modélisation Multi-Paradigme

Définition : Les trois directions

La Modélisation Multi-Paradigme contient trois directions de recherche orthogonales [1]*:

- Modélisation Multi-Abstraction: la relation entre des modèles à des niveaux d'abstraction différents.
- Modélisation Multi-Formalisme: le couplage et la transformation entre des modèles décrits dans plusieurs formalismes.
- Méta-Modélisation: la description des formalismes et des langages de modélisation.

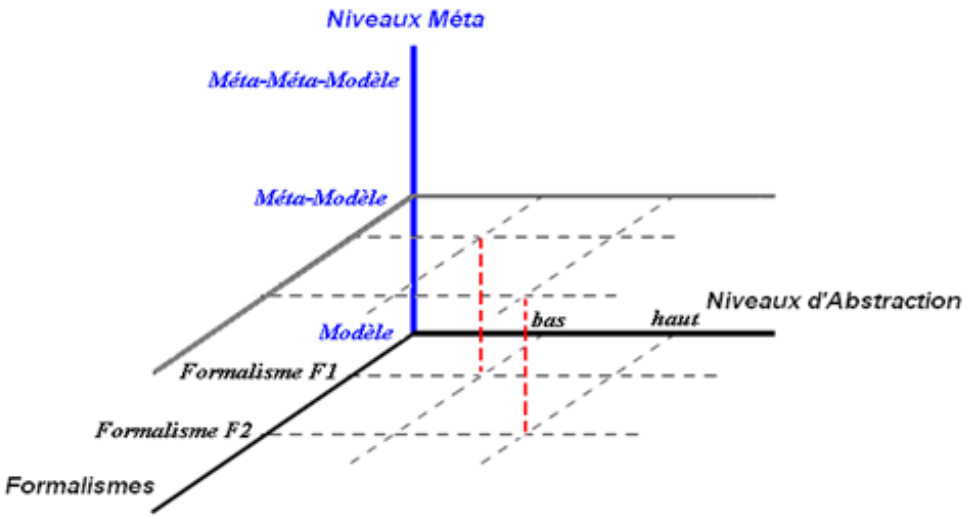


Figure 2.12 : Modélisation Multi-paradigme: les trois directions

Complément : Modélisation Multi-Abstraction

- Un système peut être décrit par plusieurs modèles, et dans plusieurs niveaux d'abstraction ou de détail.
- Chaque niveau d'abstraction est mieux adapté à une tâche particulière.
- Le processus de changement d'abstraction est un type de transformation de modèle.
- La dérivation automatique de modèles à différent niveaux d'abstraction augmente la qualité des modèles pour mieux maîtriser et comprendre tous les détails du système.

Complément : Modélisation Multi-Formalisme

- La complexité croissante des systèmes exige l'introduction de plusieurs formalismes.
- Chaque composant/aspect du système est modélisé en utilisant le formalisme et l'outil d'analyse approprié.
- Le comportement global du système est évalué en transformant les modèles des composants/aspects vers un formalisme unique.
- L'automatisation de ces transformations par des supports outillés augmente la productivité de la modélisation.
    - L'ajout de nouveaux formalismes sans fournir beaucoup d'efforts.
    - L'utilisation de nouveaux environnements de simulation et d'analyse.

Complément : Méta-Modélisation

- Besoin de multiples éditeurs
- La réalisation de ces éditeurs est relativement complexe et coûteuse
  Solution: principe de la Méta-modélisation
    - Modéliser les formalismes
    - Générer automatiquement des éditeurs pour ces formalismes

- La seule information à fournir est le méta-modèle du langage sans se préoccuper des détails d'implémentation de l'éditeur
- Avantages: préservation des acquis et flexibilité
    - Adaptation aux nouveaux besoins de modélisation
    - Ajout de contraintes spécifiques à un domaine

Complément : Mettre en relation les trois directions

- Transformation de modèles permet de relier ces trois dimensions pour cumuler leurs avantages
    - Combiner et transformer les différents formalismes
    - Utiliser des formalismes et des outils spécifiques au domaine d'application
    - Vérifier la cohérence entre les différentes vues/aspects du système
- Différentes manipulations de modèles
    - La transformation de formalismes
    - L'optimisation de Modèle
    - La simulation
    - La génération de code
- Les modèles et les méta-modèles sont des graphes
    - les transformations entre modèles peuvent être décrites et réalisées par des Transformations de Graphes

7. Conclusion

L'ingénierie dirigée par les modèles (IDM) est une approche de développement de logiciels qui met à la disposition de l'utilisateur des concepts, des langages et des outils pour réduire la complexité du développement. Dans ce chapitre, nous avons présenté les concepts de base de l'ingénierie dirigée par les modèles: le modèle, le méta-modelé, la transformation de modèles, l'approche MDA et la modélisation multi-paradigmes.

Références

1
Kerkouche Elhillali, Modélisation multi-paradigme, Thèse de Doctorat en sciences , Université Mentouri Constantine, 2012.

24
Da Silva, Alberto Rodrigues. "Model-driven engineering: A survey supported by the unified conceptual model." Computer Languages, Systems & Structures 43 (2015): 139-155.

3 Davide Buscaldi, OMG's Model Driven Architecture.

4 Cours, «Ingénierie des modèles », Alain Cariou, Université de Pau, France, 2020/2021.

5
Cours «Ingénierie Dirigée par des les Modèles », Alloua Chaoui, Université de Constantine 2, 2020/2021.

6 Livre de Xavier Blanc , MDA en action, Editions Eyrolles

7 Oscar Nierstrasz, OMG's Model Driven Architecture.