Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Centre Universitaire de Mila
Institut des mathématiques et informatique
Département de l'informatique ,
Spécialité: Sciences et technologies de l'information et de la communication. Master 1 : STIC
Laboratoire de Modélisation et simulation des systèmes complexes .

Module: Ingénierie Des Logiciels (IDL)*guidé par les modèles*

« Ingénierie dirigée par les modèles (IDM) »

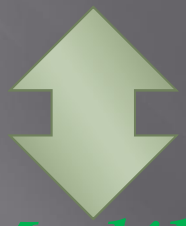
➢**Dr Aouag. M**
➢**Département MI**
aouag.mouna@centre_univ_mila.dz

PLAN

1. Introduction
2. Principes de l'IDM
3. Architecture MDA
4. Ingénierie Dirigée par les Modèles (IDM)
   *** Transformations de Graphes ***
5. Intégration des Méthodes Formelles avec l'IDM
6. La Vérification d'une Transformation
7. Conclusion

INTRODUCTION (1/4)

- Les systèmes d'information développés et déployés en entreprise ne cessent de croître en **complexité**.
- La plupart de ces systèmes sont désormais de **taille conséquente** et basés sur des approches **distribuées** permettant un travail collaboratif.
- Y cohabitent souvent diverses technologies parfois très hétérogènes (plateformes, langages, frameworks, etc).
  - **Ces technologies logicielles sont en évolution permanente**



Modéliser

3

Introduction (2/4)

- « ***Modéliser*** *est le futur, et je pense que les sociétés qui travaillent dans ce domaine ont raison* » ***Bill Gates***

- « *Obtenir du code à partir d'un modèle stable est une capacité qui s'inscrit dans la durée* » **Richard Soley (OMG)**

4

INTRODUCTION (3/4)

- L'ingénierie Dirigée par les Modèles (IdM, ou MDE en anglais pour Model Driven Engineering) apporte des solutions concrètes afin de mieux maitriser cette complexité mais aussi d'augmenter la productivité, la qualité, la réutilisabilité et l'évolution de ces systèmes d'information.

INTRODUCTION (4/4)

- L'IDM est un domaine de recherche en pleine émergence qui considère les **modèles** comme les éléments de base dans la production, le fonctionnement et l'évolution des systèmes d'information.
- L'IDM raisonne entièrement à ce niveau d'abstraction et non plus à celui des langages de programmation classiques.
- Une application sera alors générée en tout ou partie à partir de **modèles**, en utilisant notamment des techniques de **transformation** pour opérer la conversion entre le code et les modèles et vice-versa.

PLAN

1. Introduction
2. Principes de l'IDM
3. Architecture MDA
4. Ingénierie Dirigée par les Modèles (IDM)
   *** Transformations de Graphes ***
5. Intégration des Méthodes Formelles avec l'IDM
6. La Vérification d'une Transformation
7. Conclusion

PRINCIPES DE MDA (1/3)

☞ Une approche pour l'architecture MDA

- L'esprit de l'architecture MDA, découle d'une approche de développement des applications et des systèmes dirigé par les modèles.
- L'MDA 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.

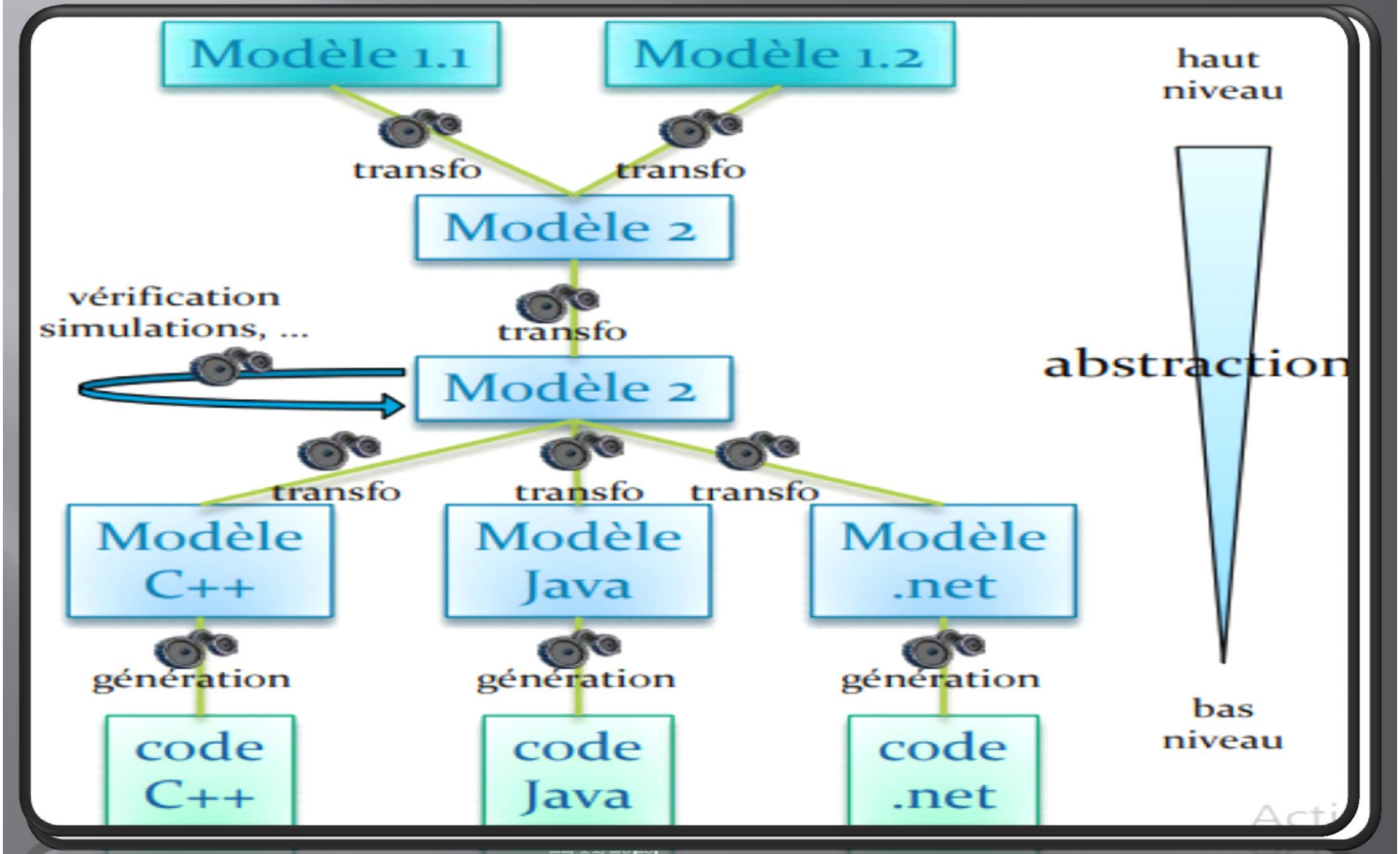
❖ Par conséquent, Le MDA s'investit dans un grand atelier dont le défit relève techniquement de la manipulation des modèles. Dans ce cadre spécifique, la transformation de modèle basé sur les techniques de **Méta-modélisation** et **l'ingénierie de modèles** ouvre un champ d'investigation prometteur pour la recherche

PRINCIPES DE MDA (2/3)

☞Le MDA est perçu comme un processus spécifique de type IDM qui repose sur les principes suivants :

- La maitrise de la complexité
- L'abstraction et la capitalisation des modèles
- La Modélisation:
- La Méta-modélisation
- La transformation de modèles
- Les standards OMG

PRINCIPES DE MDA (3/3)



PLAN

1. Introduction
2. Principes de l'IDM
3. Architecture MDA
4. Ingénierie Dirigée par les Modèles (IDM)
   *** Transformations de Graphes ***
5. Intégration des Méthodes Formelles avec l'IDM
6. La Vérification d'une Transformation
7. Conclusion

ARCHITECTURE MDA (1/13)

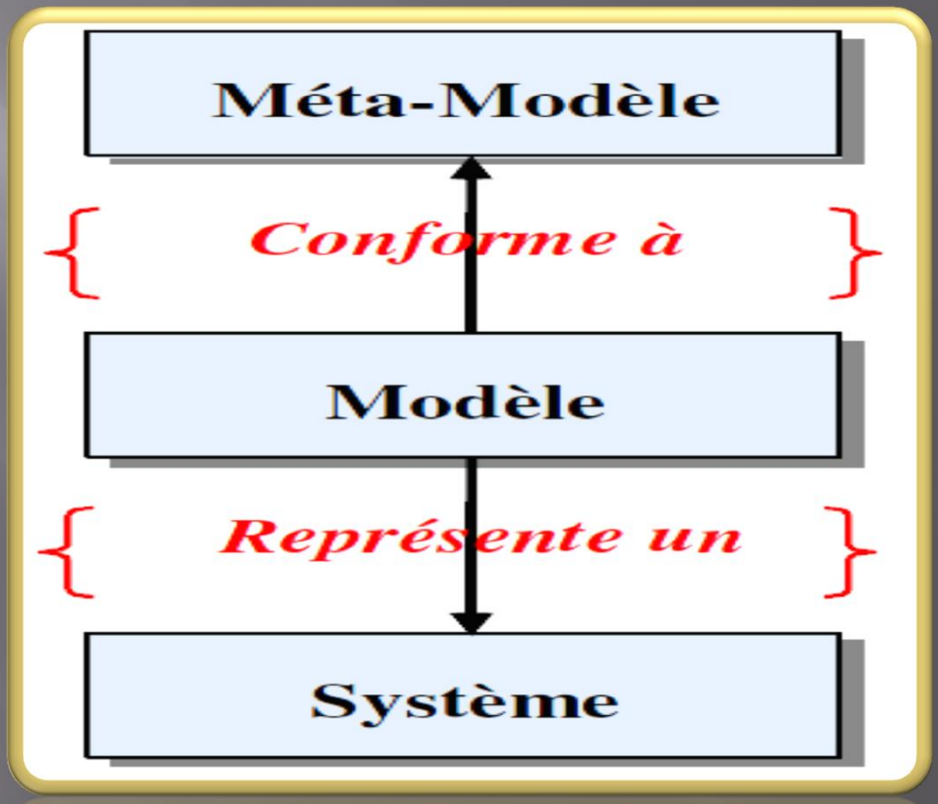
- Pratiques
  - Décomposer en niveaux d'abstraction
  - Automatiser les relations inter/intra niveaux
  - Formaliser les informations contenues dans les niveaux

- Objectifs
  - Élaboration de nouvelles applications
  - Évolution d'applications existantes
  - Maîtriser l'impact des nouvelles technologies

ARCHITECTURE MDA (2/13)

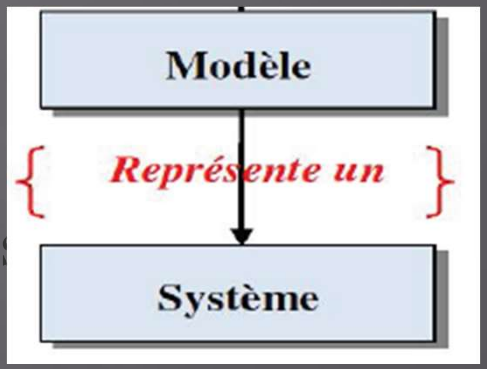
☞ Concepts de bases du MDA

- Modèles
- Méta-modèles



ARCHITECTURE MDA (3/13)

☞ Concepts de bases du MDA



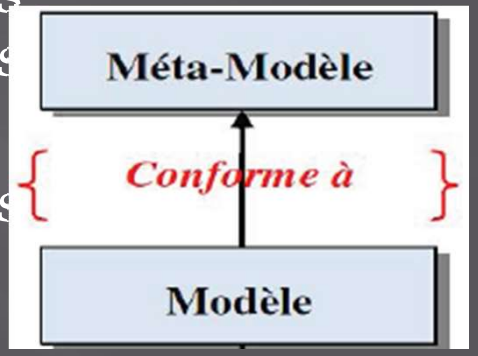
- Un **modèle** est une description, une spécification partielle d'un système
  - Abstraction de ce qui est important dans un contexte et pour un but donné
  - Vue simplifiée d'un système
- **But d'un modèle**
  - Faciliter la compréhension d'un système complexe
  - Simuler le fonctionnement d'un système complexe
- **Exemples de modèles**
  - Un réseau de Petri, Un diagramme UML
  - Modèle météorologique

ARCHITECTURE MDA (4/13)

☞ Concepts de bases du MDA

- 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



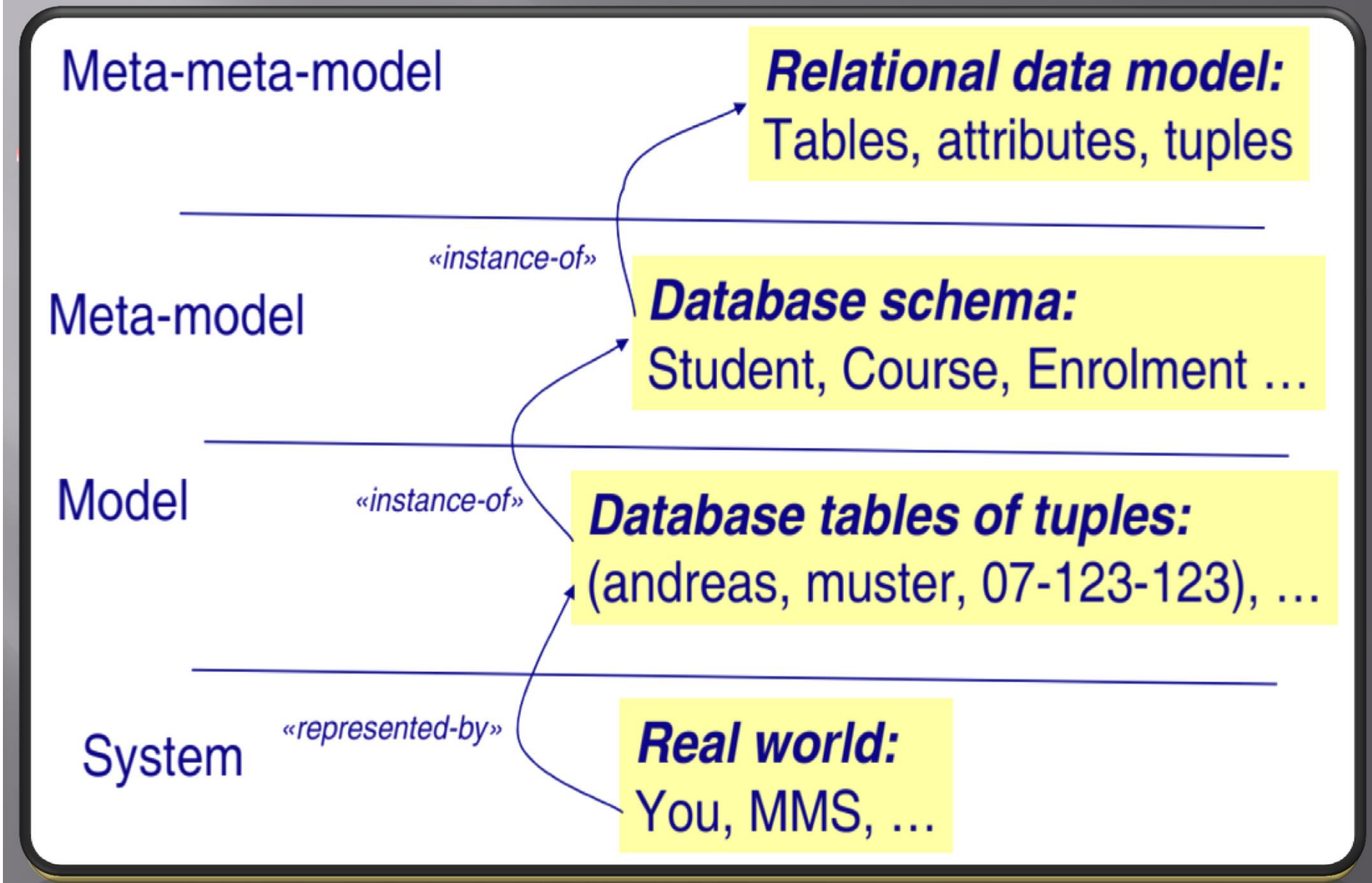
➔ Assurer la correction syntaxiques des modèles

ARCHITECTURE MDA (5/13)

☞ Concepts de bases du MDA

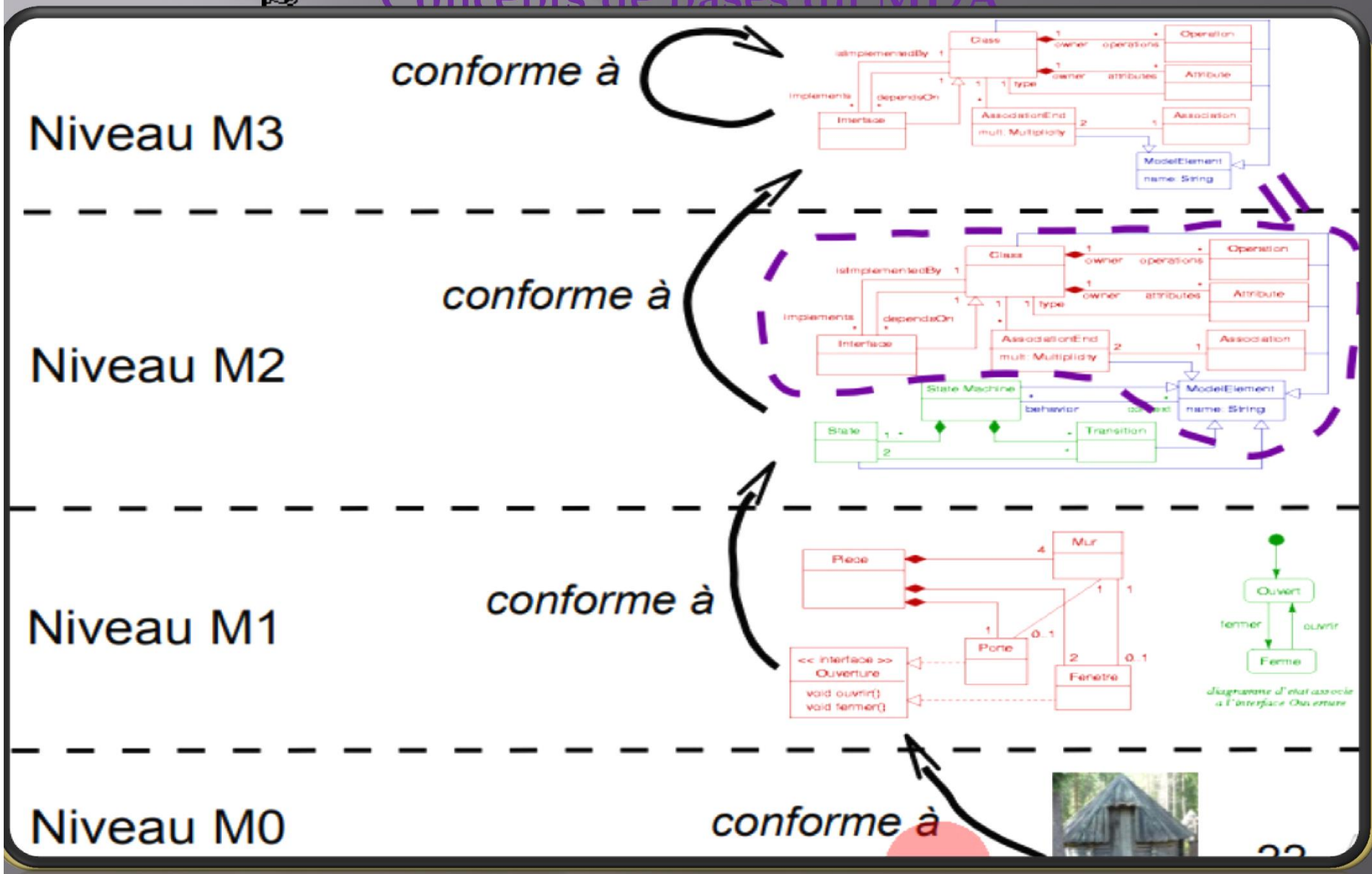
- **But de la méta-modélisation**
  - ➢ Définir des langages de modélisation ou des langages de manière générale
- **Architecture MOF de l'OMG**
  - ➢ 4 niveaux de (méta)modélisation
- **Syntaxes abstraite et concrète**
- **Définition de méta-modèles**
  - ➢ Profils UML
  - ➢ MOF
  - ➢ Ecore

ARCHITECTURE MDA (6/13)



ARCHITECTURE MDA (7/13)

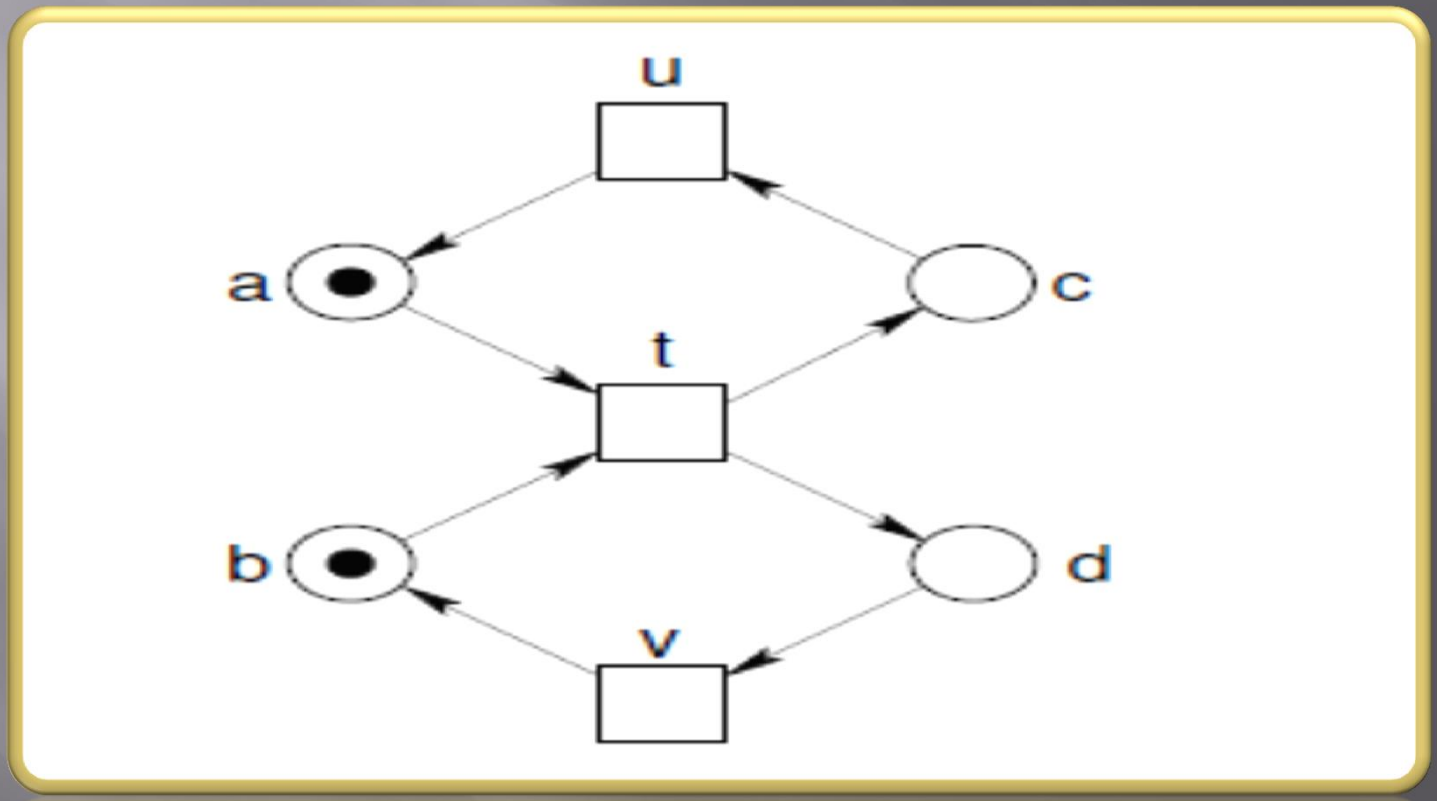
☞ Concepts de bases du MDA



ARCHITECTURE MDA (8/13)

☞ Concepts de bases du MDA

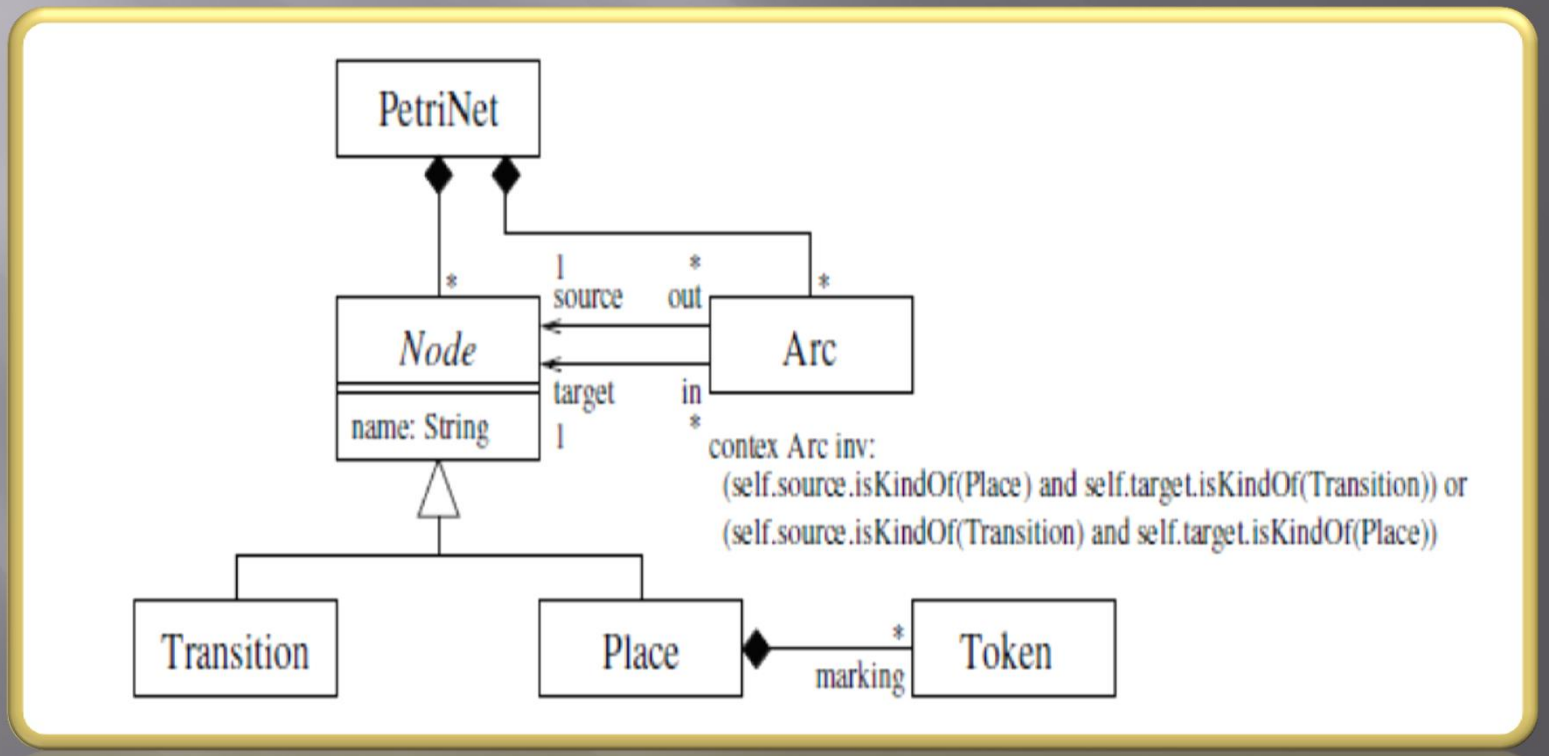
Exemple: **Un modèle** de réseau de Petri Syntaxe Concrète



ARCHITECTURE MDA (9/13)

☞ Concepts de bases du MDA

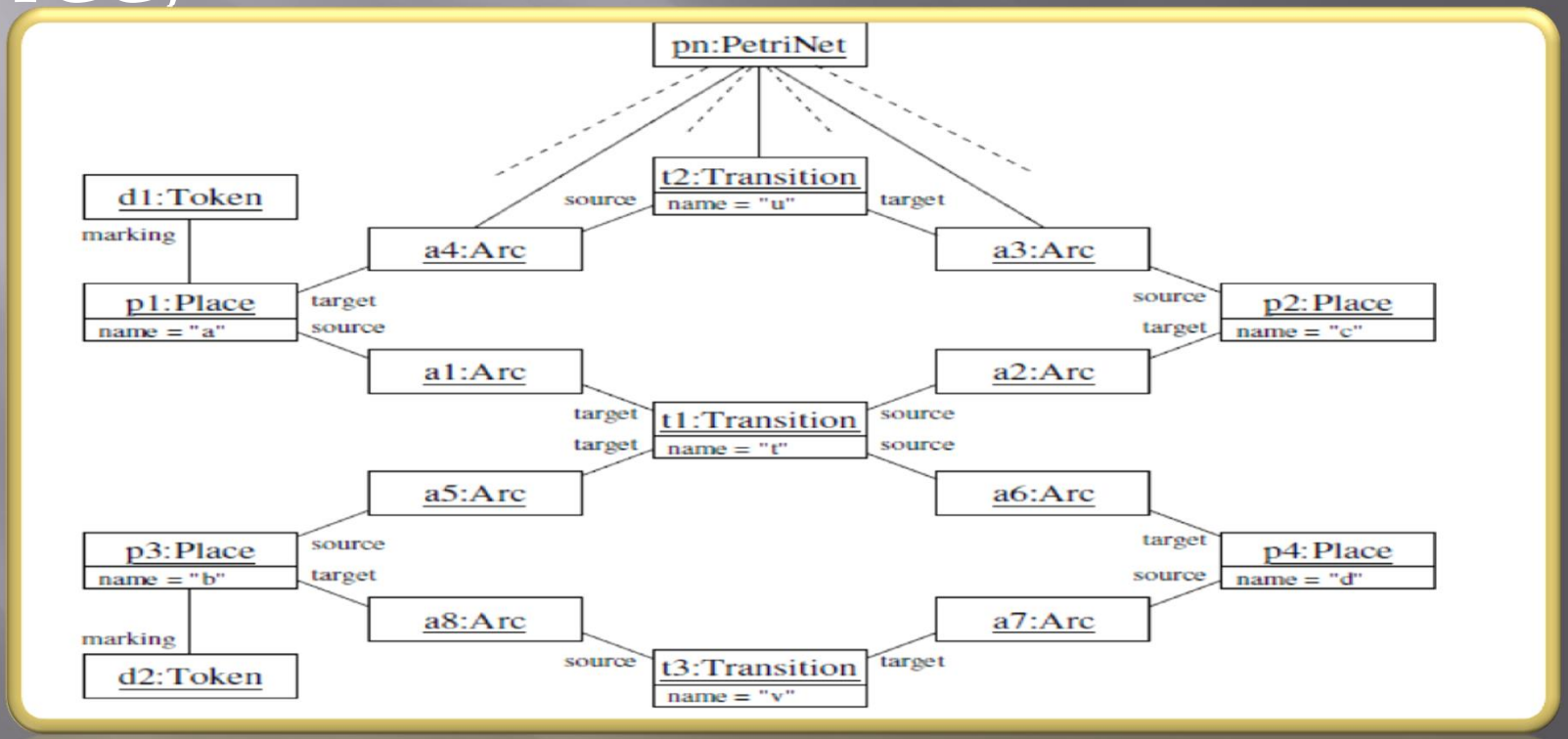
Exemple: **Un meta-modele** de réseau de Petri
Diagramme de Classe UML + OCL



ARCHITECTURE MDA (10/13)

☞ Concepts de bases du MDA

Exemple: Un meta-modèle de réseau de Petri
Syntaxe Abstraite : Diagramme Objet UML (Outil TGG)



MDA (11/13)

Ce niveau définit un langage unique pour la spécification des méta-modèles. Le MOF élément réflexif du niveau M3, définit la structure de tous les Méta-modèles du niveau M2..

est décrit dans le standard UML et qui définit la structure interne des modèles UML, appartient au niveau M2. .

être exprimés dans un langage défini au niveau M2.
UML est un exemple de modèles du niveau M1.

modélisation des objets du monde réel.

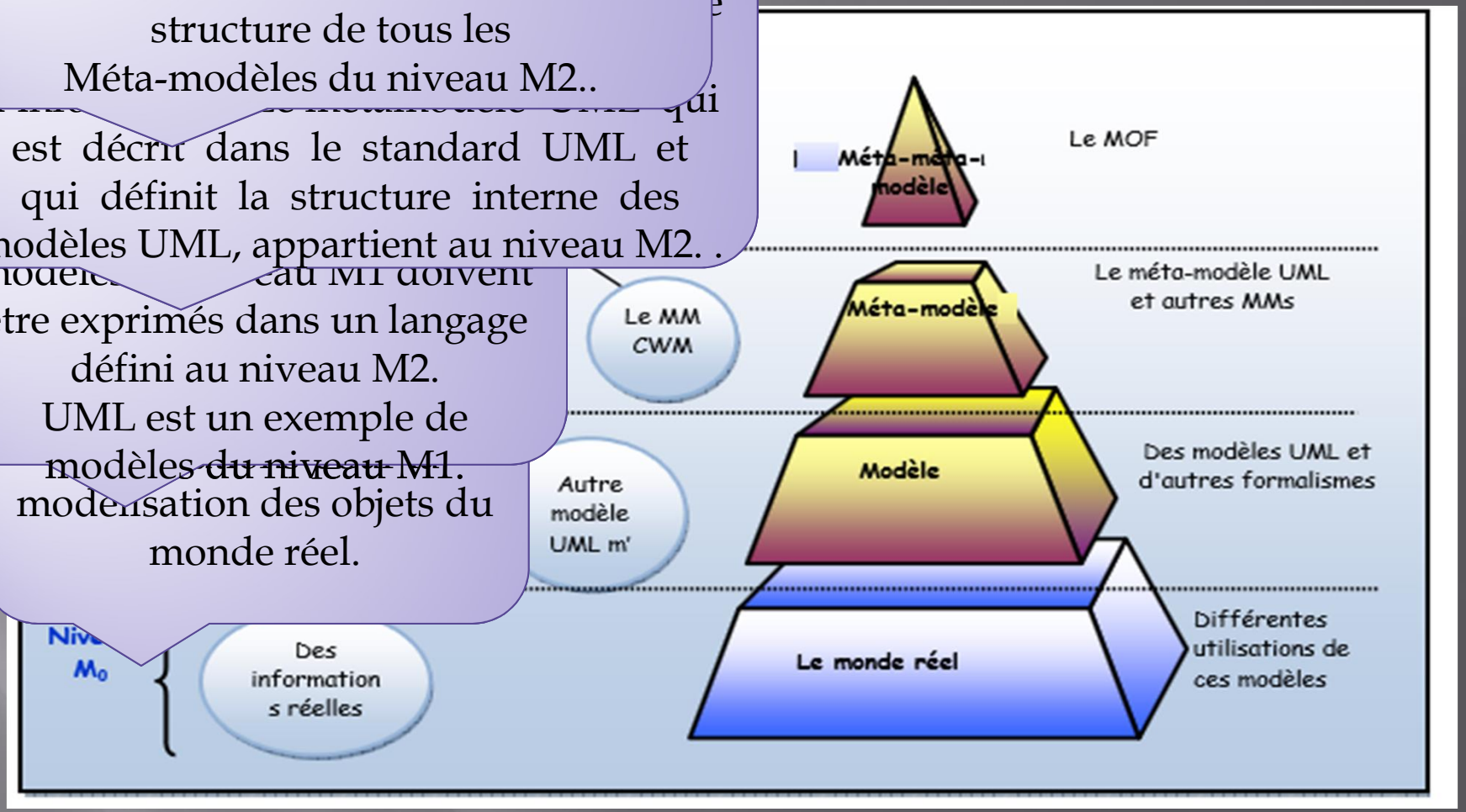


Figure 2. Les quatre niveaux d'abstraction pour MDA.

ARCHITECTURE MDA (12/13)

☞ Les modèles dans l'Architecture MDA

Conceptuellement, le MDA propose trois points de vue, associés à leurs modèles respectifs :

01 • Le modèle CIM

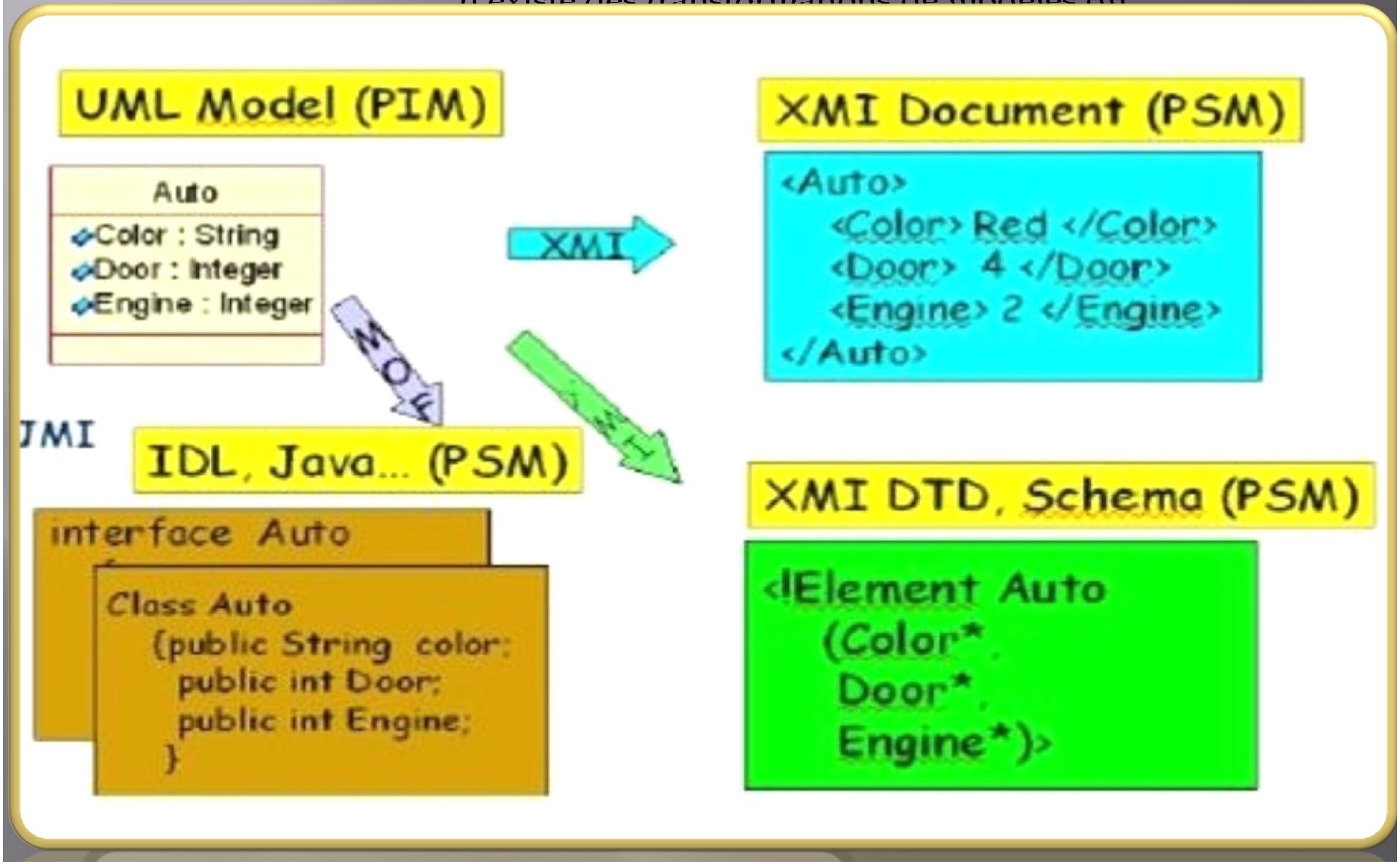
02 • Le modèle PIM

03 • Le modèle PSM

ARCHITECTURE MDA (13/13)

☞ Les modèles dans l'Architecture MDA

Il existe des transformations de modèles du



PLAN

1. Introduction
2. Principes de l'IDM
3. Architecture MDA
4. Ingénierie Dirigée par les Modèles (IDM)
   *** Transformations de Graphes ***
5. Intégration des Méthodes Formelles avec l'IDM
6. La Vérification d'une Transformation
7. Conclusion

Ingénierie Dirigée par les Modèles (IDM)

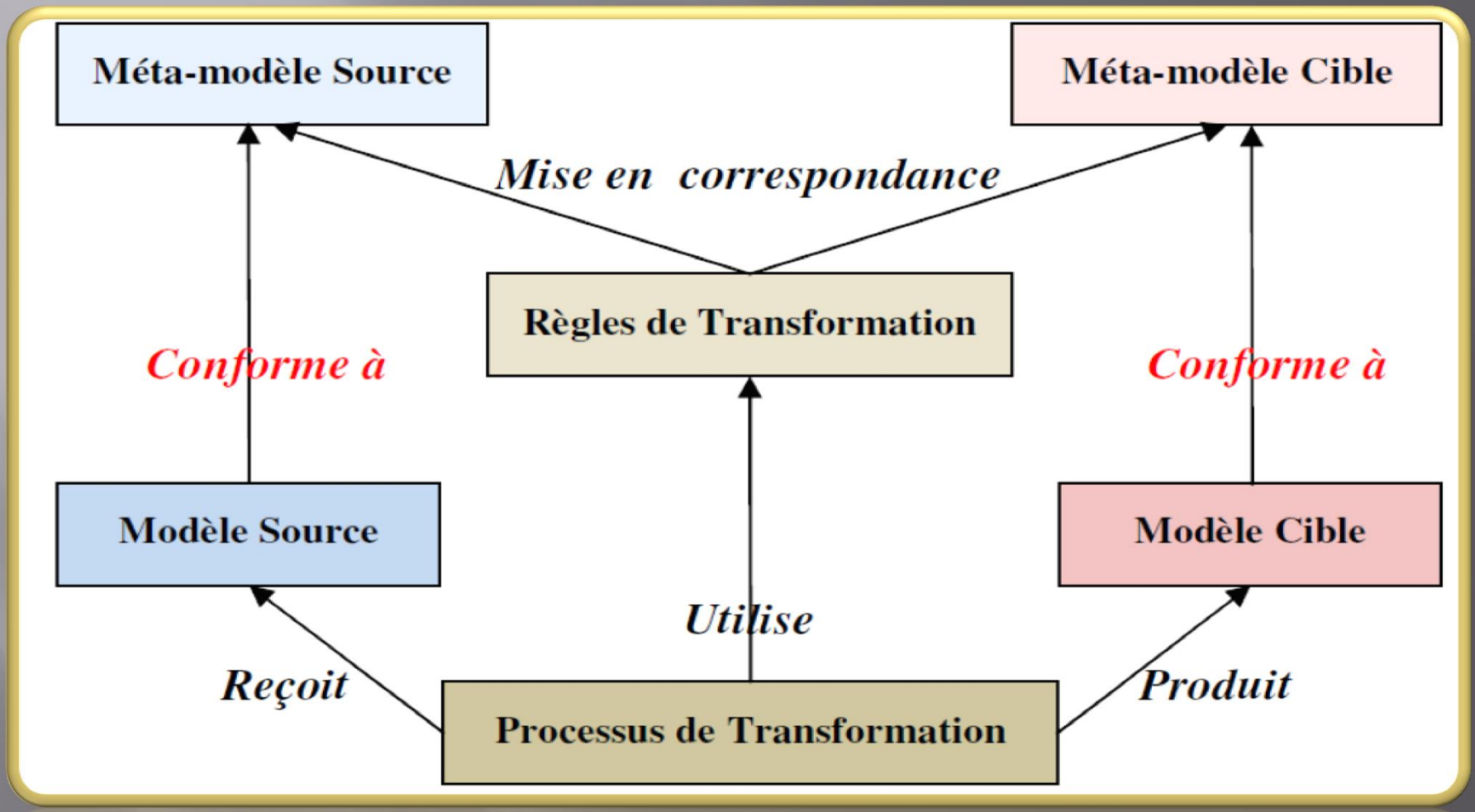
La transformation de modèles

- Principe de transformation
- Types de transformation
- Classification des approches de transformation de modèles
- Transformation de graphes

☞ *L'Ingénierie Dirigée par les Modèles (IDM)*
La transformation de modèles (1/2)

- **Approche de développement mettant à disposition de l'utilisateur des concepts, des langages et des outils**
- **Les modèles sont considérés comme des éléments de base**
- **Raisonnement est entièrement à un haut niveau d'abstraction**
- **L'application sera générée (*en tout ou en partie, automatiquement ou semi-automatiquement*) à partir de modèles**
  - ☛ ***Ingénierie générative***
- **Les outils permettant de créer et d'exploiter ces modèles sont construits autour des concepts de:**
  - **Méta-modélisation**
  - **Transformation de modèles**

L'Ingénierie Dirigée par les Modèles (IDM)
La transformation de modèles (2/2



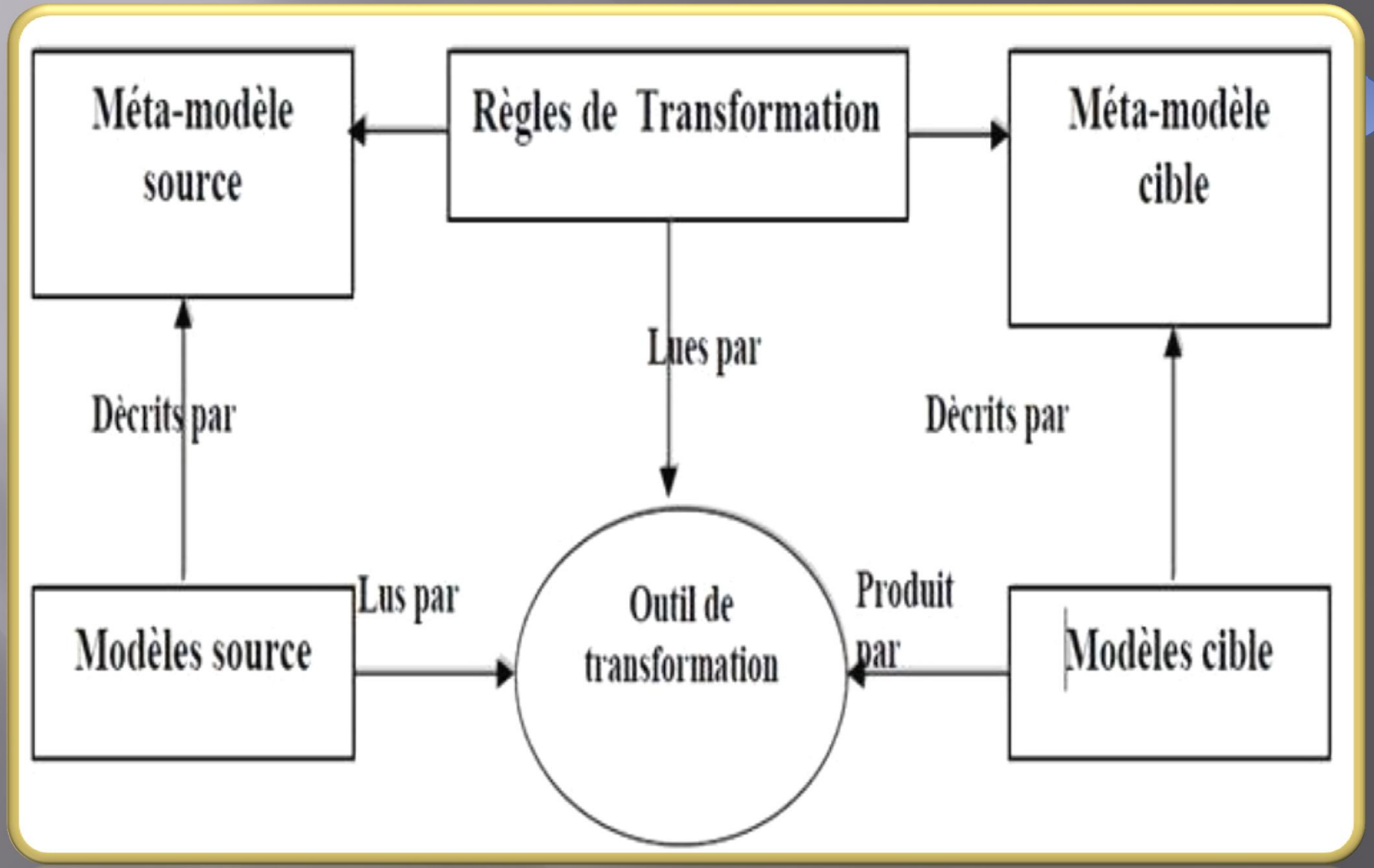
Ingénierie Dirigée par les Modèles (IDM)

La transformation de modèles

- Principe de transformation
- Types de transformation
- Classification des approches de transformation de modèles
- Transformation de graphes

L'Ingénierie Dirigée par les Modèles (IDM)

☞ Principe de Transformation



Ingénierie Dirigée par les Modèles (IDM)

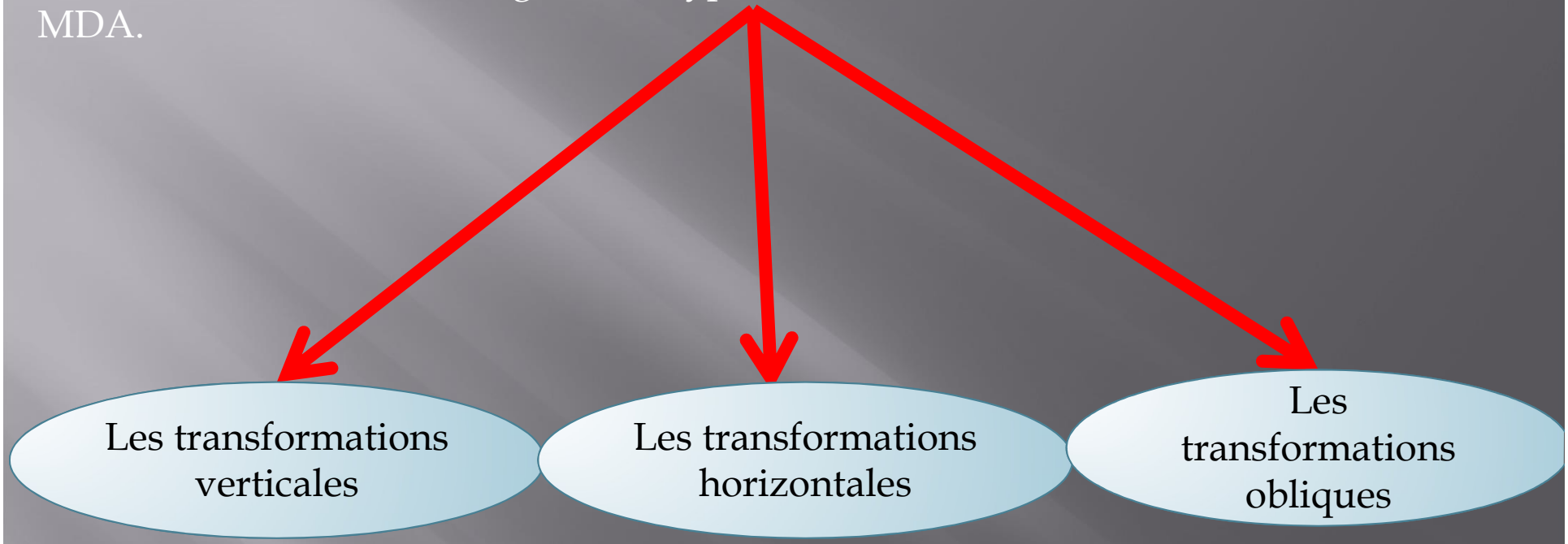
La transformation de modèles

- Principe de transformation
- Types de transformation
- Classification des approches de transformation de modèles
- Transformation de graphes

L'Ingénierie Dirigée par les Modèles (IDM)

☞ Les Types de transformation (1/2)

Dans la littérature on distingue trois types de transformation selon l'architecture MDA.

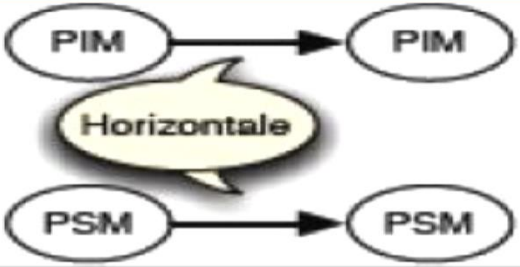
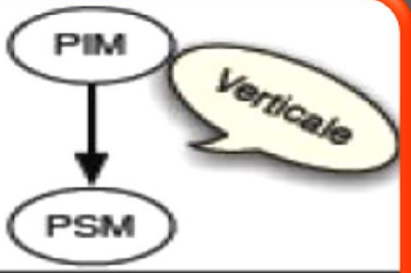
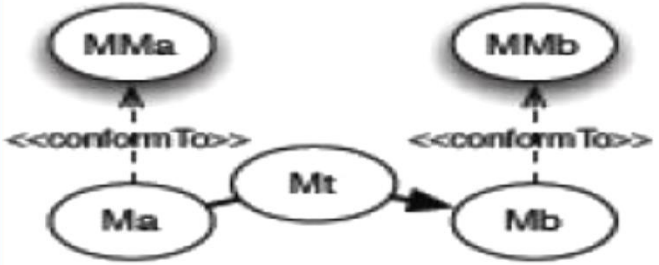


Une transformation oblique combine une transformation horizontale et une verticale. Ce type de transformation est notamment utilisé par les compilateurs, qui effectuent des optimisations du code source avant de générer le code exécutable.

d'informations.

L'Ingénierie Dirigée par les Modèles (IDM)

☞ Les Types de transformation (2/2)

		
 Transformation endogène	Restructuration Normalisation intégration de patrons	Raffinement
 Transformation exogène	Migration de logiciel Fusion de modèles	Génération Rétro-conception

Ingénierie Dirigée par les Modèles (IDM)

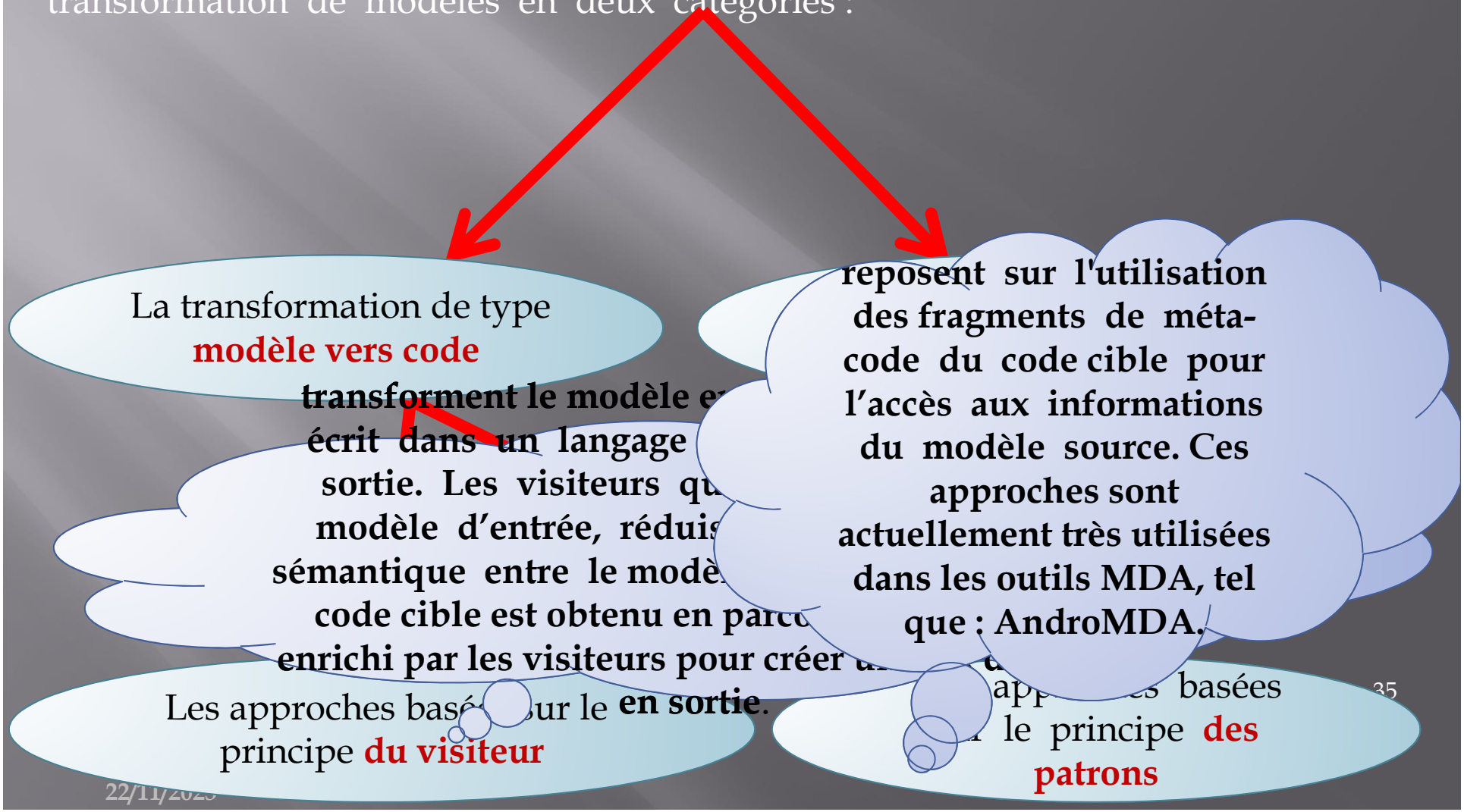
La transformation de modèles

- Principe de transformation
- Types de transformation
- Classification des approches de transformation de modèles
- Transformation de graphes

☞ *L'Ingénierie Dirigée par les Modèles (IDM)*

Classification des approches de transformation de modèles(1/2)

une classification des approches de transformation de modèles, en s'inspirant des travaux de CZarnneki [Czar] où l'on trouve une décomposition de la transformation de modèles en deux catégories :



L'Ingénierie Dirigée par les Modèles (IDM)

Classification des approches de transformation de modèles(2/2)

Quand on trouve un grand espace d'abstraction entre un PIM et un PSM, il serait plus facile de générer des modèles intermédiaires au lieu d'aller directement vers le PSM cible.
Ces modèles peuvent être utiles pour l'optimisation ou bien pour des buts de débogage. Ces transformations sont utiles pour le calcul des différentes vues du système et leur synchronisation et aussi pour des buts de vérification et de validation.

Utili… …es
trans… …ti…
- **Une organisa…**
- **ordonna… orien…**
- **Un mécan… traçab…**

Règles de transformation

- **les règles de transform… la correspondance ent… du méta-modèle sour… du méta-mo…**
- **Le rôle de la sp… description de t… indépendamment de…**
- **MDA prévoit l'auto… complète de cette pha…**

Spécif…ion de… r…es de transformation :

- **les approches par manipulation directe .**
- **les approches relationnelles.**
- **les approches basées sur la transformation de graphes.**
- **les approches basées sur la structure.**
- **les approches hybrides.**

Les approches pour la définition des transformations

Ingénierie Dirigée par les Modèles (IDM)

La transformation de modèles

- Principe de transformation
- Types de transformation
- Classification des approches de transformation de modèles
- Transformation de graphes

L'Ingénierie Dirigée par les Modèles (IDM)

☞ *Transformation de graphes (1/26)*

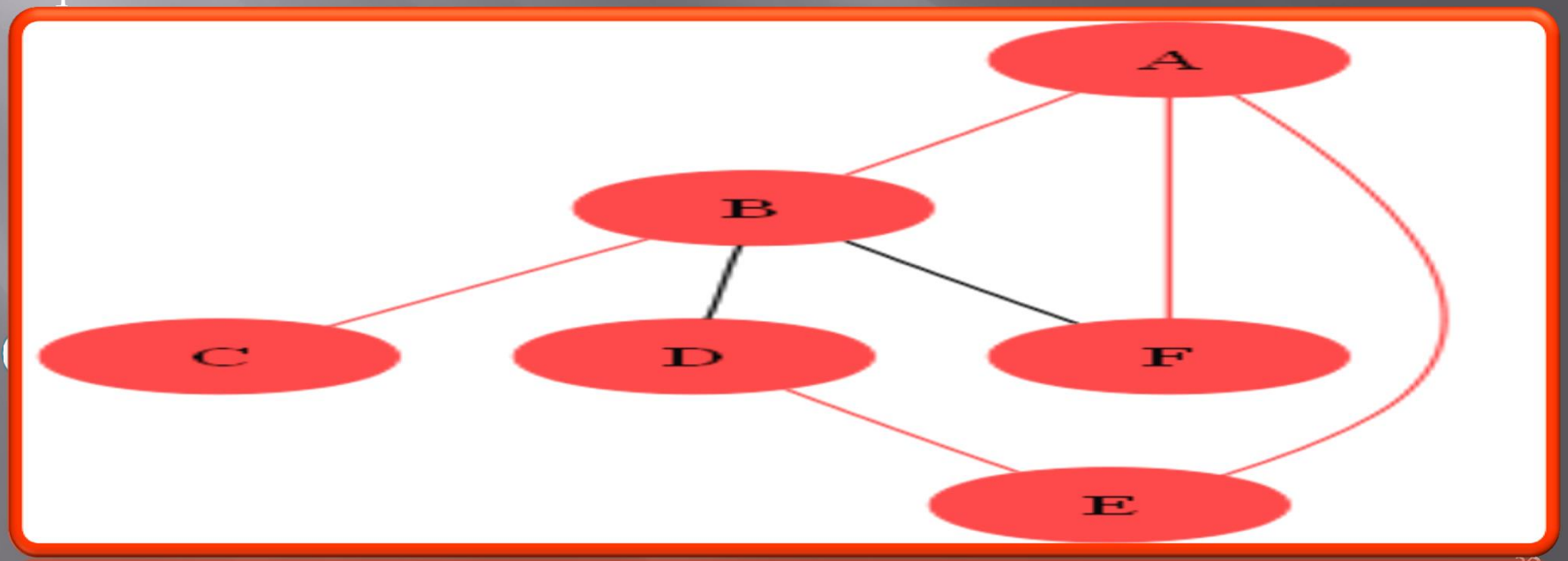
A. Concept de graphe

1. Le Graphe

Un graphe est constitué de sommets ou nœuds qui sont reliés par des arêtes. Plus formellement, on appelle graphe G = {S, A}

- **S** : ensemble fini non vide d'éléments appelés sommets ou nœuds.
- **A** : ensemble fini non vide de paires de sommets appelés arcs.
- $A \subseteq X \times X = \{(s, t) \mid s, t \in S\}$

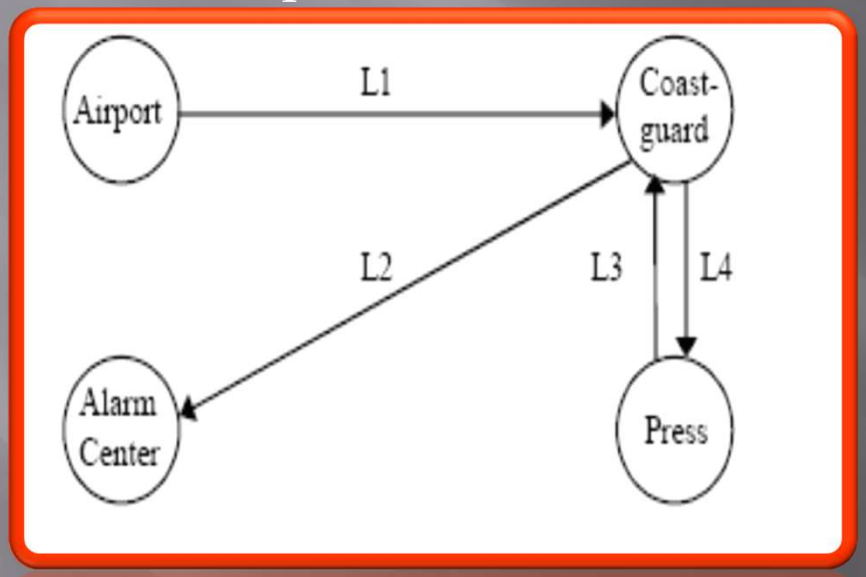
Chaque arc de **A** relie deux sommets de S.



A. Concept de graphe

4. Graphe étiqueté:

Un graphe étiqueté est un graphe orienté dans lequel les arcs possèdent un ensemble non vide d'étiquettes.



5. Le Degré d'un graphe:

le degré d'un sommet dans un graphe est le nombre d'arêtes dans ce sommet. Si le graphe est orienté le degré de sommet se définit en degré entrant et degré sortant, relatifs respectivement au nombre d'arcs entrant et sortants dans ce sommet.

A. Concept de graphe

6. L'ordre d'un graphe :

l'ordre d'un graphe G se définit par le nombre de sommets présents dans ce graphe .

7. Le Chemin dans un graphe:

le chemin dans un graphe se définit par la succession des arcs parcourus dans un sens définit dans le graphe, il possède les propriétés et les caractéristiques suivantes :

- ✓ la longueur du chemin est égale au nombre d'arcs parcourus .
- ✓ le chenin est dit une chaîne si l'on ne tient pas compte de la direction des arcs .
- ✓ le chemin est dit circuit s'il revient à son point de départ .
- ✓ la distance entre deux sommets dans un graphe se définit comme étant la longueur du plus court chemin entre ces deux sommets, si cette longueur est de 1 les sommets sont dits adjacents.
- ✓ le diamètre d'un graphe est la plus grande distance séparant deux sommets dans ce graphe.

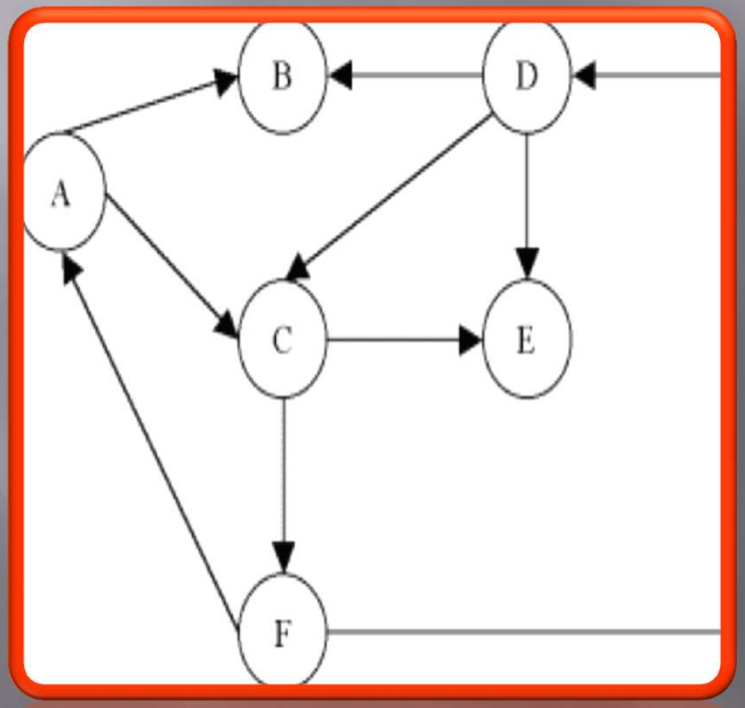
L'Ingénierie Dirigée par les Modèles (IDM)

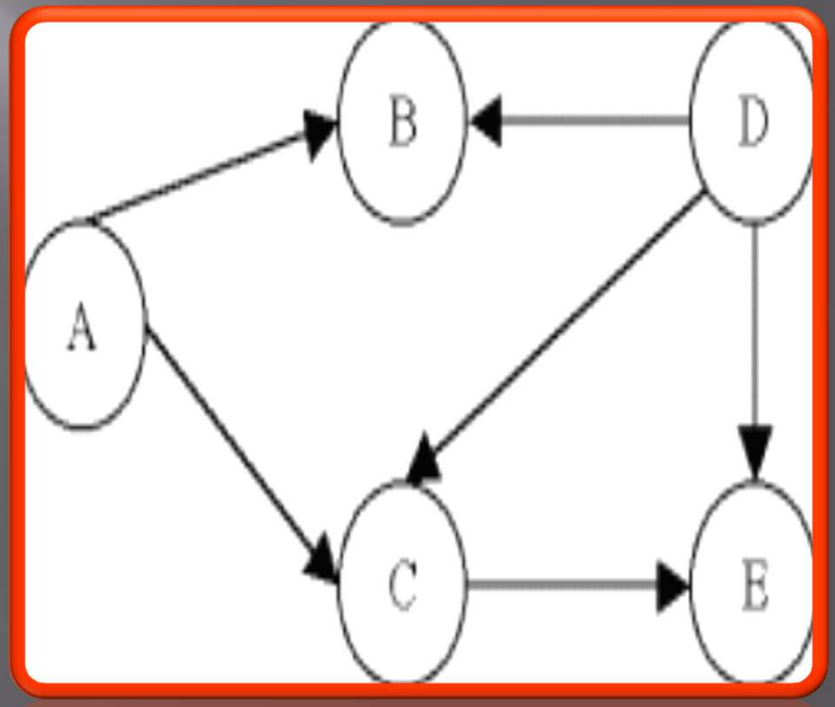
☞ *Transformation de graphes (4/26)*

A. Concept de graphe

8. Le sous-graphe:

un sous graphe G'(S',A') d'un graphe G(S,A) est un graphe composé d'un sous ensemble de sommet S'∈ S et d'un sous ensemble d'arêtes A'∈A tel que A' représente les arêtes reliant les sommets S' dans le graphe G'

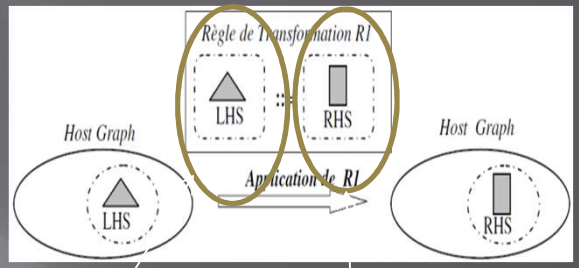


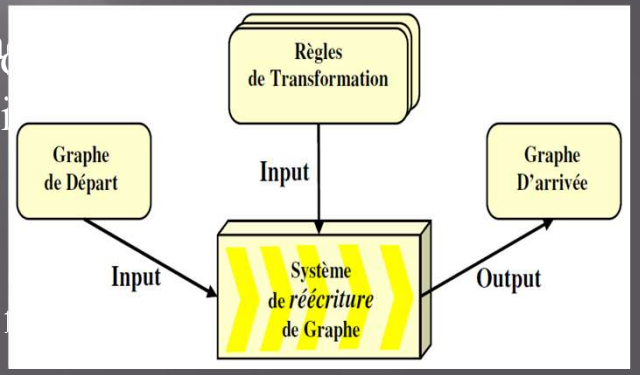


L'Ingénierie Dirigée par les Modèles (IDM)
Transformation de graphes (5/26)
B. Système de transformation de graphes

- Des techniques directement applicables à la transformation de modèles
  - La transformation est:
    - Spécifiée sous forme d'un modèle de ***Grammaires de Graphes***
    - Effectuée par un ***Système de Réécriture de Graphes***
  - Grammaires de graphes
    - Généralisation d'une grammaire de Chomsky
    - Un ensemble de règles
    - Une règle est constituée de deux parties:

      La partie gauche (LHS): décrit les a… La partie… modificati… du modèle source
  - Système de réécriture de graphes
    - Application itérative des règles





- ***Avantage:*** approche formelle fondée sur des bases …
  - ✓ La théorie des graphes
  - ✓ Les grammaires sont formelles
  - ✓ La réécriture de termes

☞ *Transformation de graphes (6/26)*

B. Système de transformation de graphes

1. Principes des règles

Une règle de transformation de graphe est définie par : r = (L, R, K, glue, emb, cond) Elle consiste en:

- Deux graphes **L** graphe de côté gauche et **R** graphe de côté droit.
- Un sous graphe **K** de **L.**
- Une occurrence g**lue** de **K** dans **R** qui relie le sous graphe avec le graphe de côté droit.
- **emb,** Une relation d'enfoncement qui relie les sommets du graphe de côté gauche et ceux du graphe du côté droit.
- Un ensemble **cond** qui spécifie les conditions d'application de la règle.

2. L'application des règles

L'application d'une règle r= (L, R, K, glue, emb, cond) à un graphe G produit un graphe résultant H. Le graphe H fourni, peut être obtenu depuis le graphe d'origine G en passant par les cinq étapes suivantes :

A. Choisir une occurrence du graphe de côté gauche L dans G.

B. Vérifier les conditions d'application d'après cond.

C. Retirer l'occurrence de L (jusqu'à K) de G ainsi que les arcs pendillé, c.-à-d. tous les arcs qui ont perdu leurs sources et/ou leurs destinations. Ce qui fournit le graphe de contexte D de L qui a laissé une occurrence de K.

B. Système de transformation de graphes

D. Coller le graphe de contexte D et le graphe de côté droit R suivant l'occurrence de K dans D et dans R. c'est la construction de l'union de disjonction de D et R et, pour chaque point dans K identifier le point correspondant dans D avec le point correspondant dans R.

E. Enfoncer le graphe du côté droit dans le graphe de contexte de L suivant la relation d'enfoncement emb .

L'application de r sur un graphe G pour fournir un graphe **H est appelée une dérivation directe depuis G vers H à travers r,** elle est dénotée par G ⇒ H ou simplement par G ⇒ H.

En donnant les notions de règle et de dérivation directe comme étant les concepts élémentaires de la transformation de graphe, on peut définir les systèmes de transformation de graphe, les grammaires de graphe et **la notion de langages engendrés**

3. Langage engendré

Soit G 0 un graphe initial, G n est le graphe final et une séquence de transformations successives de graphes : G 0 → G 1 → ... → G n est une dérivation successive à partir de G 0 vers G n en appliquant les règles de R qui sont étiquetées par les symboles de T, est dit langage engendré par R, G 0 et T et on écrit L(R, G 0 , T)

L'Ingénierie Dirigée par les Modèles (IDM)
Transformation de graphes (8/26)
B. Système de transformation de graphes

Outils de transformation de graphes

Plusieurs outils de transformation de graphes existent actuellement, parmi
lesquels :

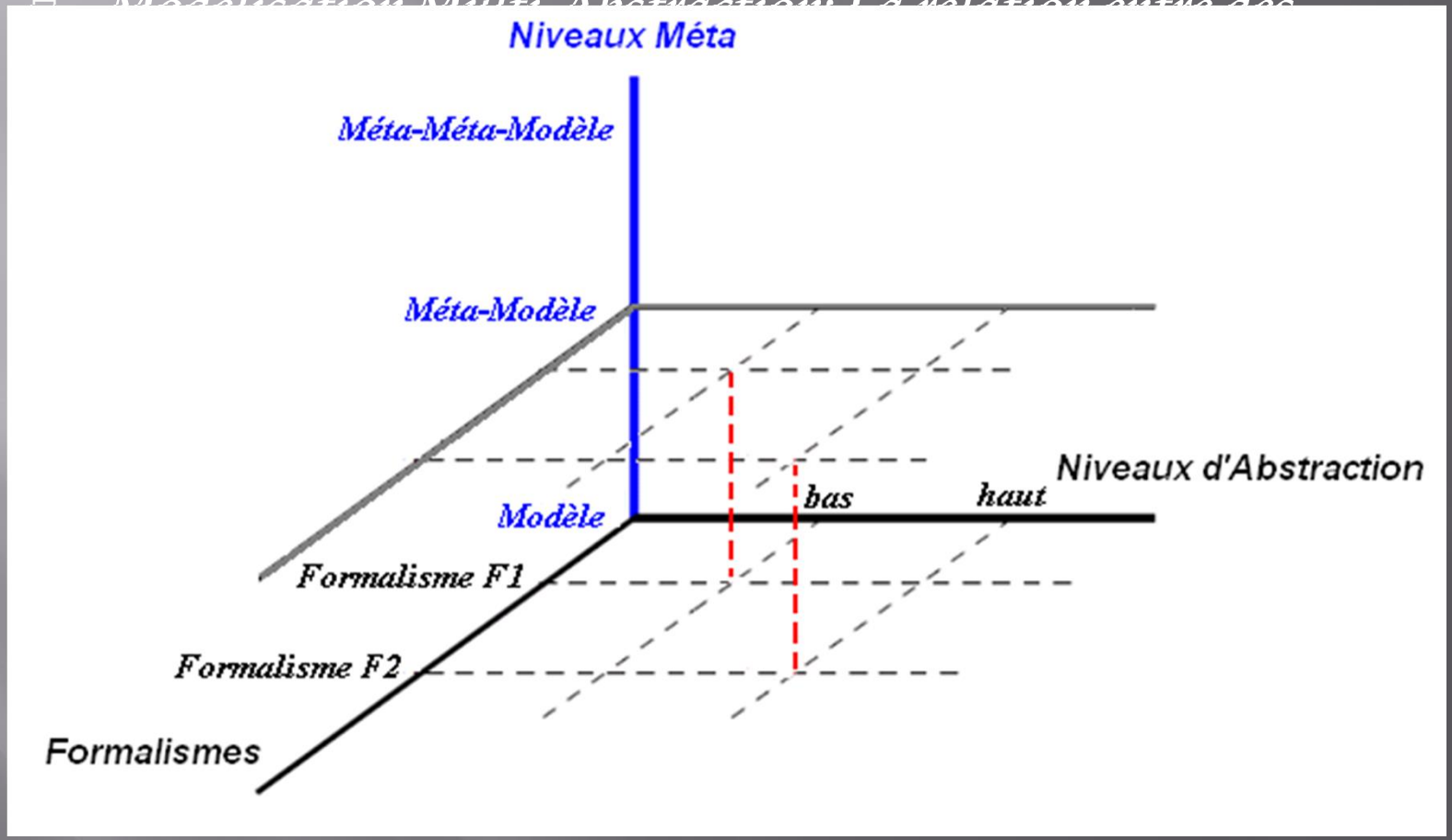
- **AGG** [AGG]:The **A**ttributed **G**raph **G**rammar System
- **AToM 3** [ATOM3]:**A** Tool for **M**ulti-formalism and **M**eta-Modelling
- **VIATRA** [Viatra]:**VI**sual **A**utomated model **TRA**nsformations.
- **FUJABA** [Fujaba]: **F**rom **U**ML to **J**ava **a**nd **b**ack **a**gain.
- AToMPM : A Tool for Multi-Paradigm Modeling.
- **GreAT** [Great]: The **G**raph **Re**write And **T**ransformation tool suite.

Parmi ces outils nous avons choisi l'outil AToMPM à cause des avantages qu'il présente. parmi ces avantages :

- outil simple .
- disponible.
- il est multi paradigmes
- permet la méta-modélisation et la transformation des modèles (modèle vers code et modèle vers modèles).

L'Ingénierie Dirigée par les Modèles (IDM)
Transformation de graphes (9/26)
C. Modélisation Multi-Paradigme

- Modélisation Multi-Abstraction: La relation entre des



L'Ingénierie Dirigée par les Modèles (IDM)
Transformation de graphes (10/26)
C. Modélisation Multi-Paradigme

- 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'éfforts
- L'utilisation de nouveaux environnements de simulation et d'analyse
- 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

L'Ingénierie Dirigée par les Modèles (IDM)
Transformation de graphes (11/26)
C. Modélisation Multi-Paradigme

- 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***

☞ *Transformation de graphes (12/26)*

C. *AToMPM: A Tool for Multi-formalism Meta-Modeling*

AToMPM:signifie « A Tool for Multi-Paradigm Modeling ».

- Outil pour la modélisation multi-paradigme.
- Successeur de AToM3.
- Il basé complètement sur le Web.
- Il fonctionne sur le nuage.
- Effectuer des transformations de modèles, et manipuler et gérer des modèles.

AToMPM possède :

- Une couche de Méta-modélisation qui permet La modélisation graphique d'un formalisme ainsi que la génération automatique d'un outil pour manipuler les différents modèles décrits dans le formalisme spécifié.
- Un système de réécriture de graphes qui permet Les manipulations de modèles par application itérative des règles d'une Grammaire de Graphes.

☞ *Transformation de graphes (13/26)*

C. *AToM³ : A Tool for Multi-formalism Meta-Modeling*

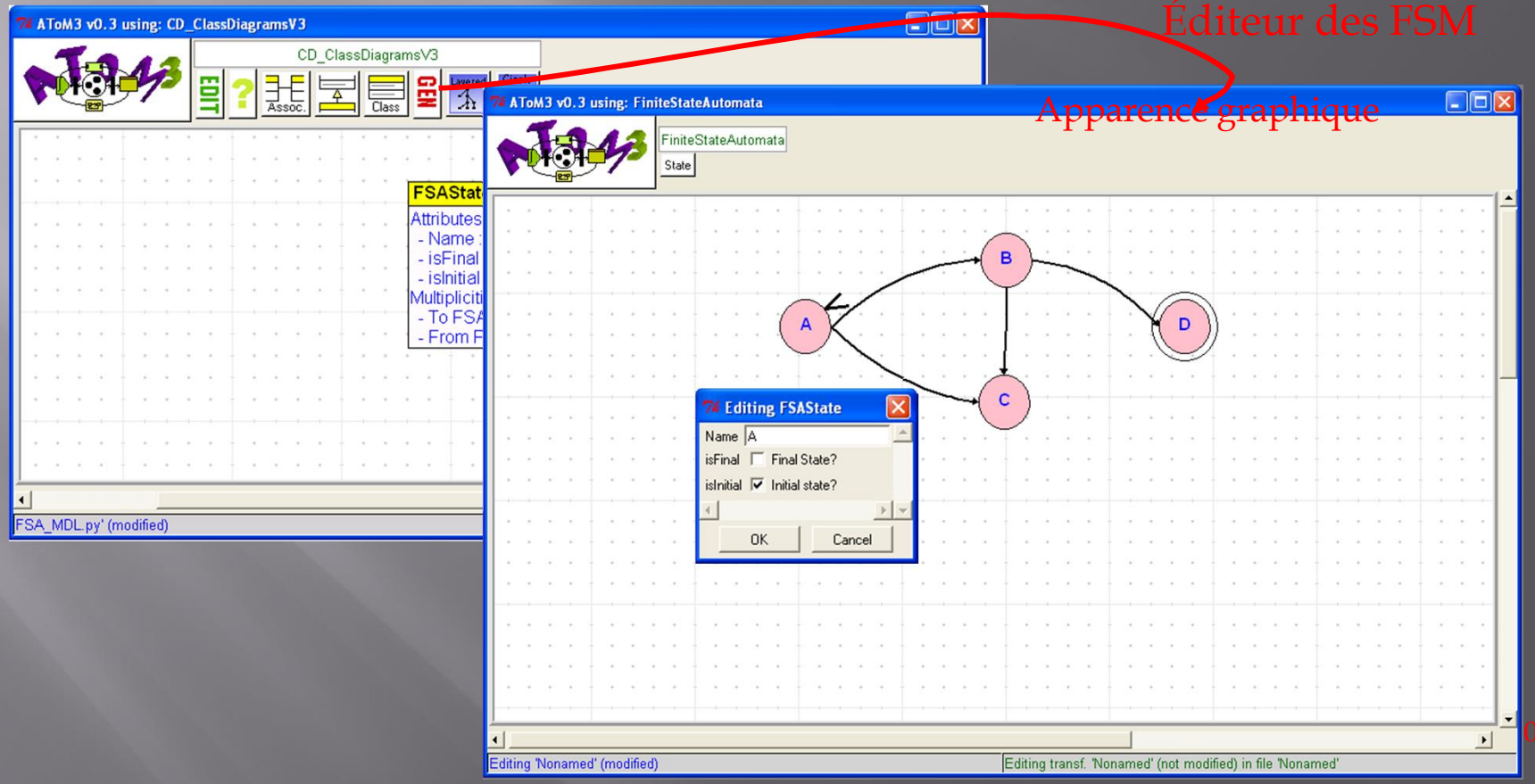
Exemple: les Automates d'Etats Finis (FSM)

Méta-modèle des FSM

Méta-Modèle

Éditeur des FSM

Apparence graphique



☞ Transformation de graphes (14/26)

C. AToM³ : A Tool for Multi-formalism Meta-Modeling

Exemple: les Automates d'Etats Finis (FSM)

Transformation des FSM en des Réseaux de Petri

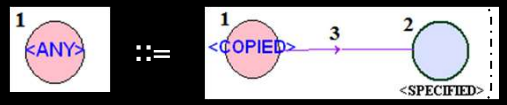
Grammaire de Graphes: FSM → Rdp

Action Initiale: *Node in ListOf_FSM_State: Node .Visited == 0*

Règle ***State2Place***. *Priorité 1*

Condition: Node(1).Visited == 0

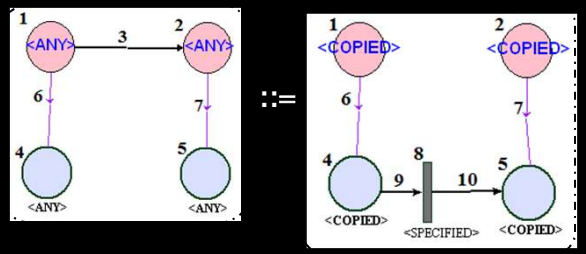
LHS ::= **RHS**



Action: Node(1).Visited = 1

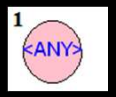
Règle ***Trans_Transition***. *Priorité 2*

LHS ::= **RHS**



Règle ***Del_State***. *Priorité 3*

LHS ::= **RHS**

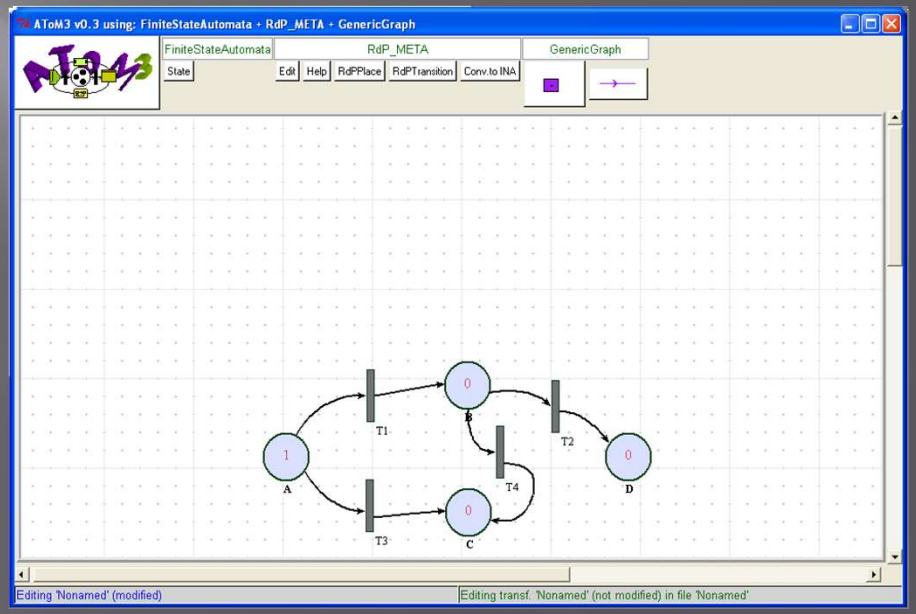


Action Finale: *Node in ListOf_FSMState: Del Node .Visited*

Transformation

Etape N°: 0

Règle N°: 3



D. *Une Approche Intégrée UML/CPN pour la Modélisation et la Vérification*

- UML est devenu quasiment incontournable dans le domaine du développement des systèmes complexes
- Il permet de décrire les différents aspects (statiques ou dynamiques) à travers ses différents diagrammes
- Les aspects dynamiques sont largement modélisés avec les diagrammes d'états-transitions (les Statecharts) et les diagrammes de collaboration
  - ***Les diagrammes d'états-transitions*** modélisent le comportement des objets du système en fonction des occurrences d'événements
  - ***Les diagrammes de collaboration*** définissent les échanges de messages (ou d'événements) inter-objets
- ***Inconvénient:*** UML souffre du manque de sémantique formelle

➔ ce qui rend impossible la vérification du comportement des systèmes décrits par plusieurs diagrammes

D. *Une Approche Intégrée UML/CPN pour la Modélisation et la Vérification*

- Les réseaux de Pétri colorés (CPNs) permettent aussi la description de l'aspect dynamique des systèmes complexes mais de façon mathématique précise et rigoureuse
- Leur principal intérêt est que leur sémantique est bien définie et permet donc de vérifier le comportement des systèmes
- En contrepartie, leur difficulté d'apprentissage et d'utilisation leur a souvent été reprochée.

D. *Une Approche Intégrée UML/CPN pour la Modélisation et la Vérification*

- L'utilisation conjointe des langages de spécification semi-formelle et formelle permet de tirer profit de ces deux types d'approches en termes de convivialité, d'expressivité et de rigueur

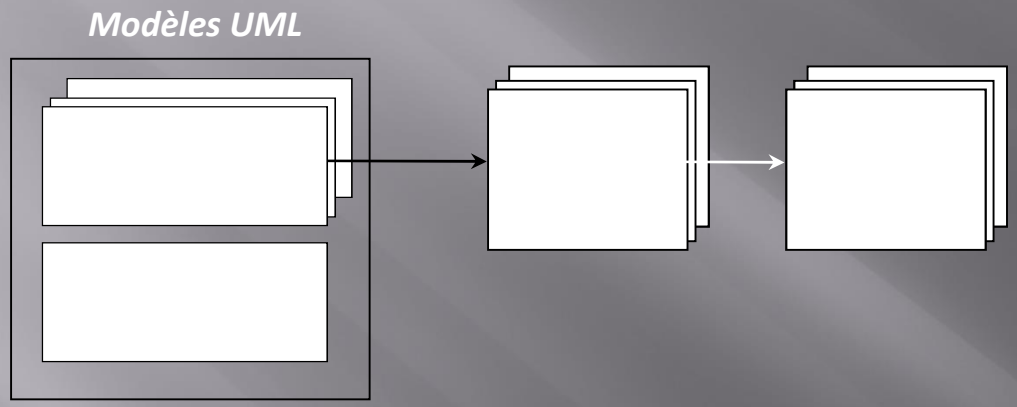
- Proposition d'une approche intégrée UML/CPN qui permet:

  - ✓ De modéliser les aspects dynamiques des systèmes complexes avec un langage convivial et graphique: UML

  - ✓ De vérifier de façon formelle les propriétés dynamiques de ces systèmes: CPN.

D. *Une Approche Intégrée UML/CPN pour la Modélisation et la Vérification*

Architecture de l'approche proposée



- Le comportement du système est spécifié par un ensemble de diagrammes d'états-transitions et un diagramme de collaboration
- Les diagrammes d'états-transitions sont d'abord convertis en automates d'états finis (Flat State Machine (FSM)) qui ne contiennent que des états simples et des transitions
- Les modèles FSMs sont ensuite transformés en un type de réseaux de Petri à objets appelé (Object Net Modèles (ONM))

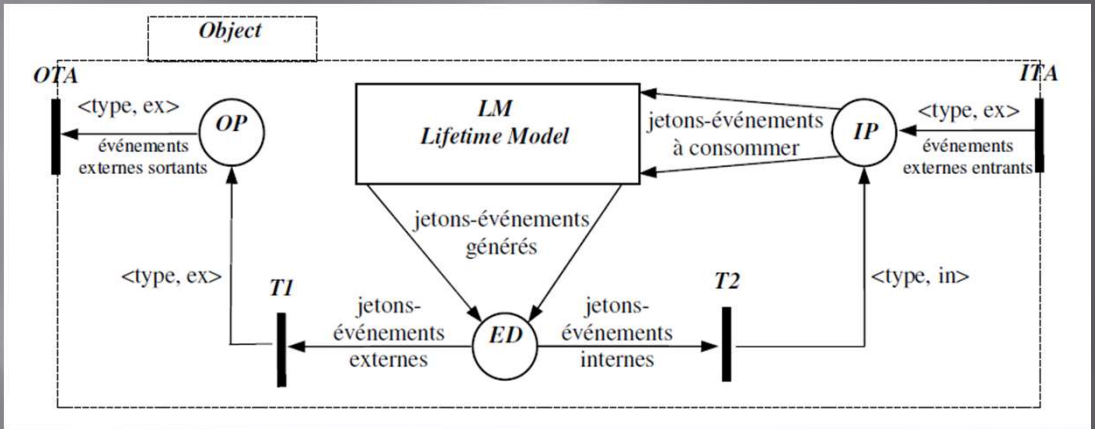
L'Ingénierie Dirigée par les Modèles (IDM)

☞ Transformation de graphes (19/26)

D. Une Approche Intégrée UML/CPN pour la Modélisation et la Vérification

Object Net Modèles (*ONM*)

- Un modèle ONM est dérivé d'un diagramme d'états-transitions "aplatis"
- Il décrit le ***comportement*** et le ***mécanisme de routage des jetons***
- La structure d'un modèle ONM est constituée de deux parties:



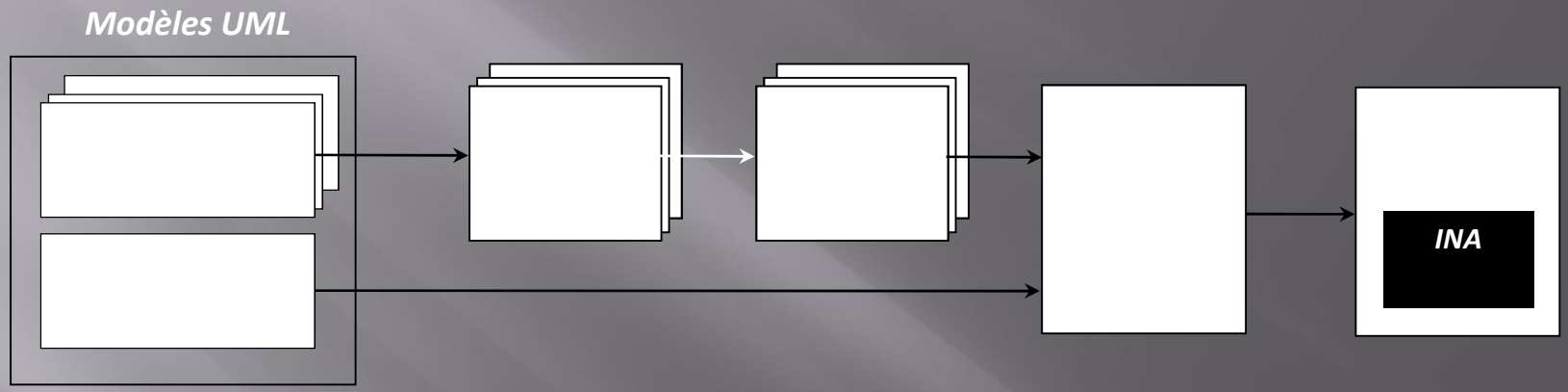
- un modèle du cycle de vie de l'objet (*LM*) représente le modèle CPN équivalent au diagramme d'états-transitions de l'objet
- une structure de routage des jetons-événements qui remplace la sémantique des événements dans les diagrammes d'états-transitions
  - Les jetons remplacent les événements dans les CPN

L'Ingénierie Dirigée par les Modèles (IDM)

☞ Transformation de graphes (20/26)

D. Une Approche Intégrée UML/CPN pour la Modélisation et la Vérification

Architecture de l'approche proposée



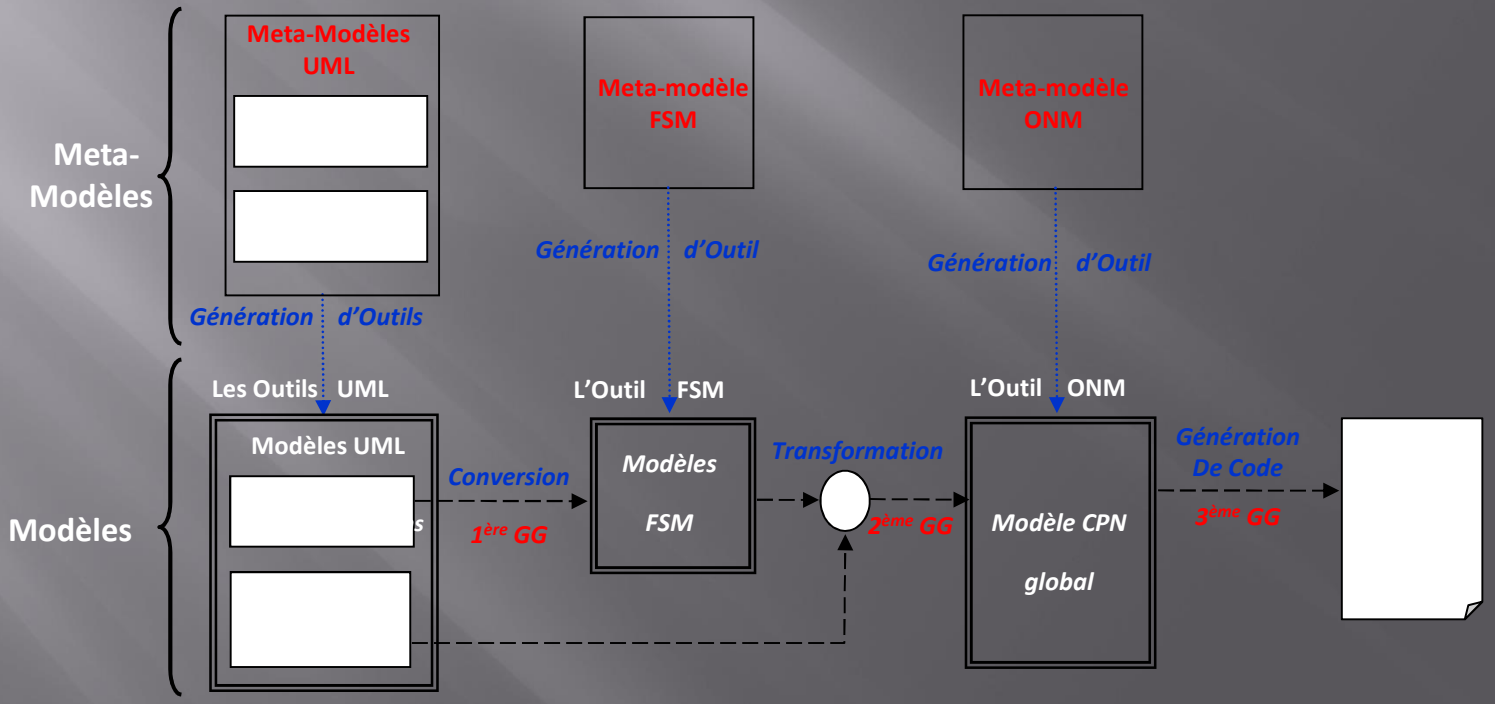
- Enfin, le diagramme de collaboration est utilisé pour interconnecter l'ensemble des modèles ONMs afin d'obtenir un seul modèle CPN
- A partir du modèle CPN global résultat, n'importe quel outil d'analyse peut être utilisé pour vérifier le comportement du système
  - ➔ On a opté pour l'analyseur INA (*Integrated Net Analyzer*)

L'Ingénierie Dirigée par les Modèles (IDM)

☞ Transformation de graphes (21/26)

D. Une Approche Intégrée UML/CPN pour la Modélisation et la Vérification

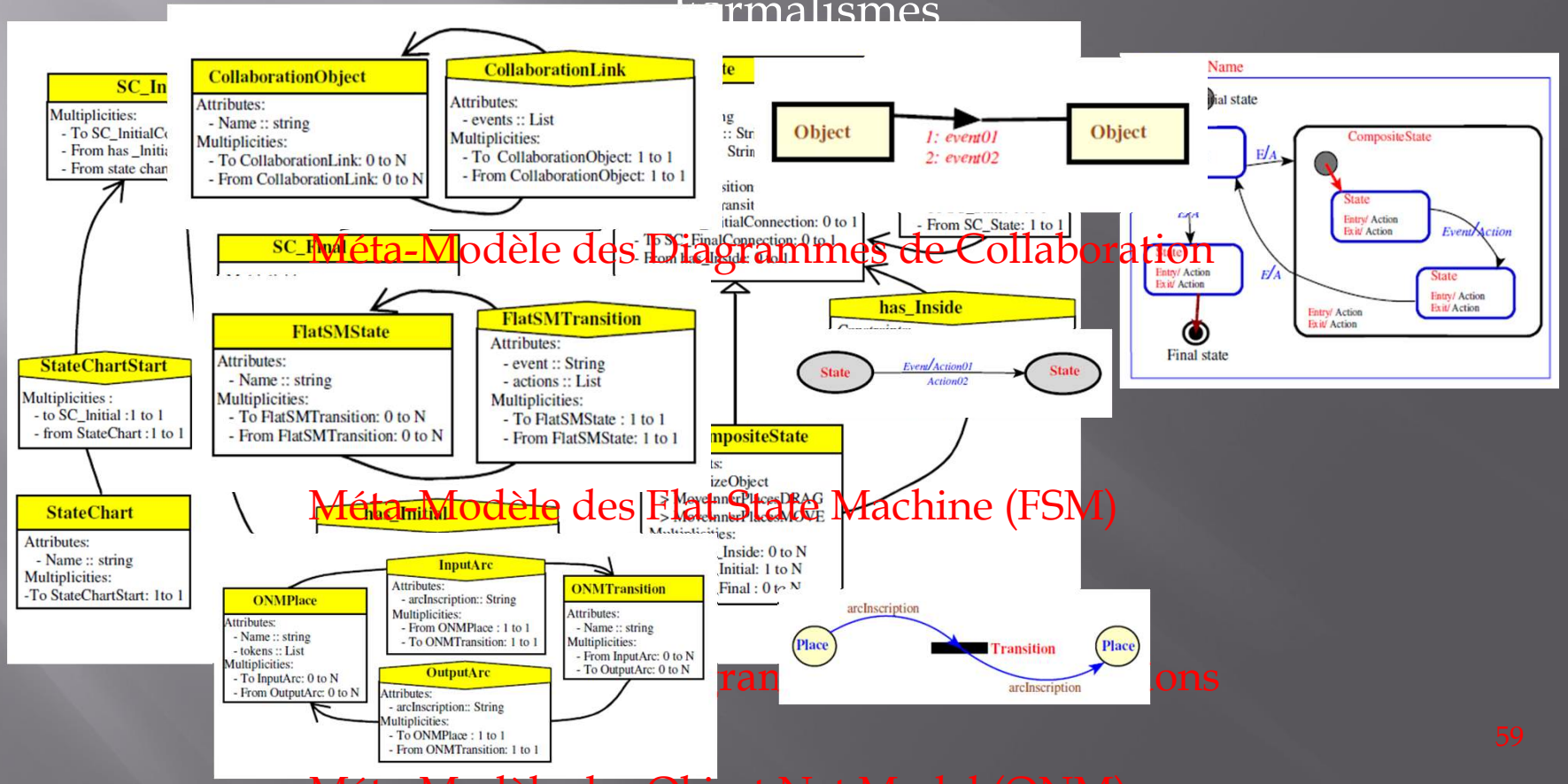
- Notre démarche comprend deux étapes:
  - ➔ *1ère étape*: La Méta-modélisation des diagrammes UML et des formalismes
  - ➔ *2ème étape*: La définition des grammaires de graphes



☞ *Transformation de graphes (22/26)*

D. *Une Approche Intégrée UML/CPN pour la Modélisation et la Vérification*

1ère étape: Méta-Modélisation des Diagrammes UML et des Formalismes



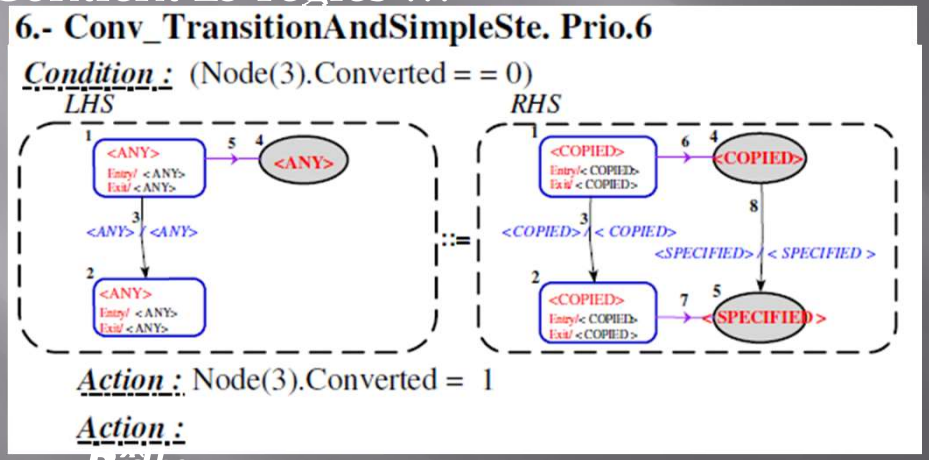
L'Ingénierie Dirigée par les Modèles (IDM)

☞ *Transformation de graphes (23/26)*

D. *Une Approche Intégrée UML/CPN pour la Modélisation et la Vérification*

2ème étape: 1ère GG, Conversion des diagrammes d'états-transotions en des modèles FSMs

- Produit un modèles FSM équivalent à un diagramme d'états-transotions
- Contient 23 règles …



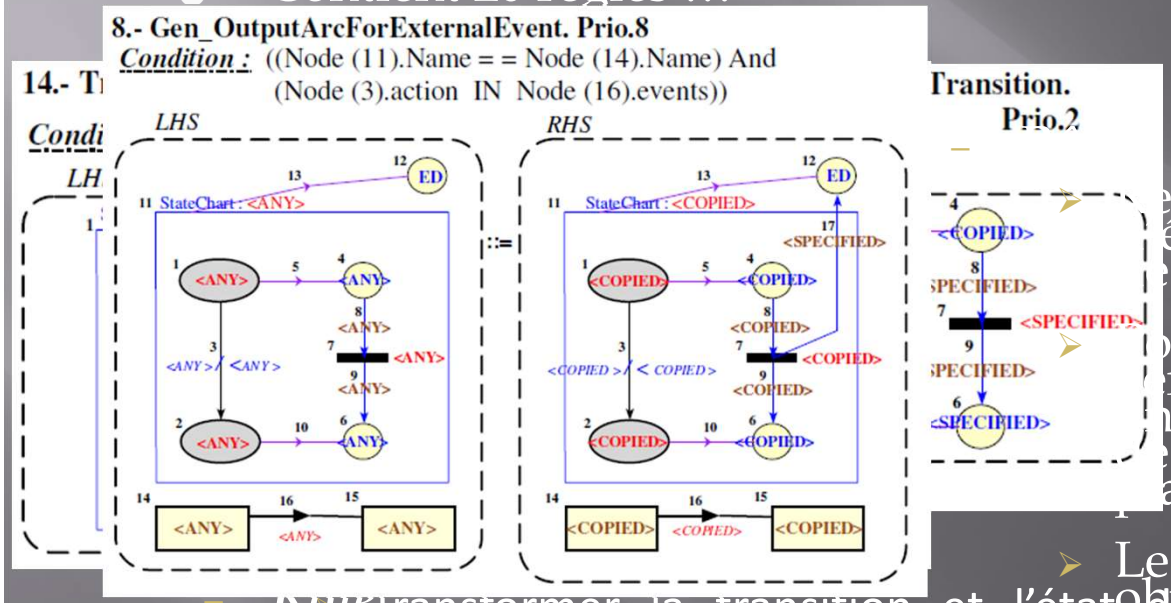
- *Rôle*:
  - Convertir la transition et l'état initial du diagramme d'états-transitions par leurs équivalents dans le modèle FSM
  - Convertir la transition à l'état initial du diagramme d'états-transitions par leurs équivalents dans le modèle FSM de relier les éléments appertenants aux formalismes différents

☞ *Transformation de graphes (24/26)*

D. *Une Approche Intégrée UML/CPN pour la Modélisation et la Vérification*

2ème étape: 2ème GG, Transformation des FSMs et diagramme de Collaboration vers un modèle CPN

- Produit un modèles CPN représentant le comportement global du système
- Contient 20 règles …



- *Rôle*: Transformer la transition et l'état du modèle FSM en leurs équivalents place et transition dans le diagramme de collaboration
  - Générer une structure de routage des jetons (les places (IP, ED et OP), les transitions (OTA, ITA, T1 et T2)) et une place équivalente à l'état initial du diagramme d'états-transitions

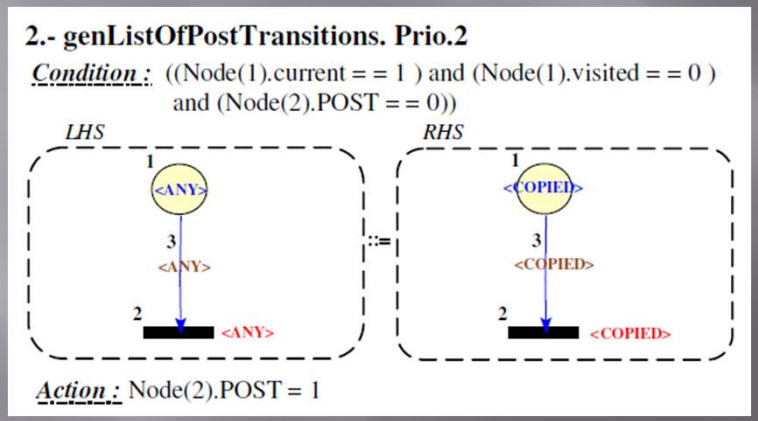
L'Ingénierie Dirigée par les Modèles (IDM)

☞ Transformation de graphes (25/26)

D. Une Approche Intégrée UML/CPN pour la Modélisation et la Vérification

2ème étape: 3ème GG, génération de la spécification INA

- Produit un fichier de spécification INA équivalent à un modèle CPN
- Contient 10 règles …



  - **Rôle**:
    - Générer la liste des sous-transitions pour une sous-place de la place courante

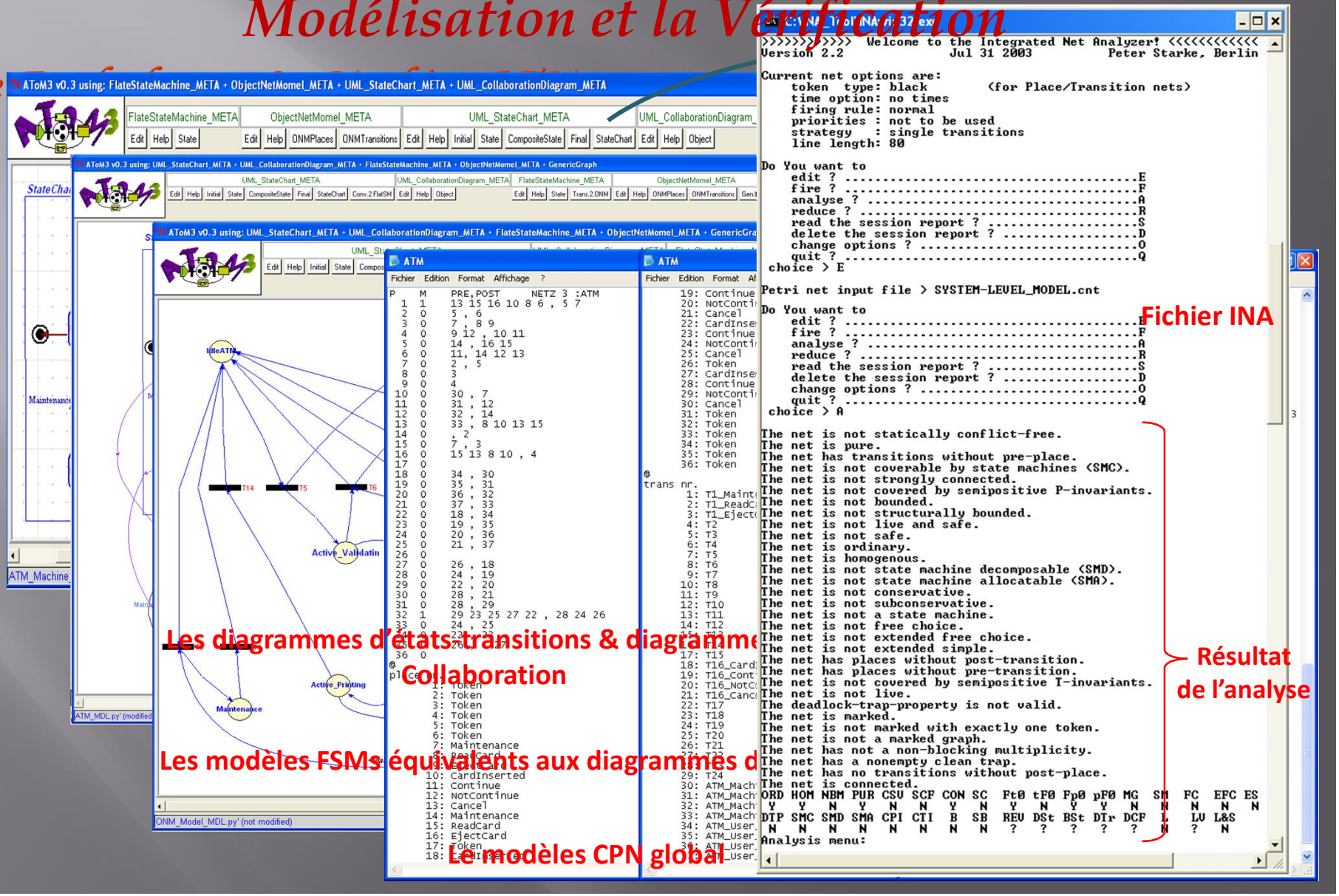
- La grammaire génère la structure dépliée du modèle CPN et les informations de dépliage

L'Ingénierie Dirigée par les Modèles (IDM)

☞ Transformation de graphes (26/26)

D. Une Approche Intégrée UML/CPN pour la Modélisation et la Vérification

Une



D. *Une Approche Intégrée UML/CPN pour la Modélisation et la Vérification*

➢ Discutions

- L'approche présentée dans ce travail vise à réduire l'écart entre les notations semi-formelles (les diagrammes UML) et les notations formelles (les réseaux de Petri colorés)

- Les outils proposés permettent de faciliter l'utilisation de multiples modèles pour modéliser et vérifier les aspects dynamiques des systèmes complexes
  - les diagrammes UML : conviviales et expressifs
  - les réseaux de Petri : analysables

PLAN

1. Introduction
2. Principes de l'IDM
3. Architecture MDA
4. Ingénierie Dirigée par les Modèles (IDM)
   *** Transformations de Graphes ***
5. Intégration des Méthodes Formelles avec l'IDM
6. La Vérification d'une Transformation
7. Conclusion

☞Intégration des Méthodes Formelles avec l'IDM (1/5)

Processus de Vérification de Modèles en IDM

- Recouvre plusieurs propriétés: la consistance, la sûreté, l'atteignabilité , la vivacité, …
- Différentes techniques sont utilisées:
  - La simulation
  - Les tests
  - Les méthodes formelles
- Toutes ces techniques s'appuient sur l'utilisation de représentations formelles du système (à base de logique, d'automates, de Réseaux de Petri, …)
- Dans le cas de systèmes complexes, ces tâches deviennent impossibles à réaliser sur un modèle préexistant

  ➢ La tendance actucelle vise à vérifier ces propriétés par construction de plusieurs modèles adaptés

☞ Intégration des Méthodes Formelles avec l'IDM (2/5)

Processus de Vérification de Modèles en IDM

- Les méthodes formelles (MFs) sont des techniques basées sur les mathématiques permettant à la fois de ***modéliser*** le système et de ***vérifier*** les propriétés attendues
- Les méthodes formelles reposent sur l'utilisation de langages formels
- Un langage formel est un langage doté d'une sémantique mathématique rigoureuse
- Principaux obstacles d'utilisation des MFs dans les activités de développement des systèmes sont liés à :
  - La difficulté réelle de manipuler les concepts théoriques et les méthodes d'analyse associées
  - La difficulté pour les développeurs d'exprimer les propriétés du système d'une façon aisée

☞ *Intégration des Méthodes Formelles avec l'IDM (3/5)*

Processus de Vérification de Modèles en IDM

- La vérification des modèles comptent aujourd'hui parmi les enjeux les plus importants de l'IDM
- L'IDM joue un rôle essentiel dans l'introduction des MFs dans les activités de développement des systèmes
- Celle-ci repose sur :

  ➔ La définition de langages formels par le biais de la ***méta-modélisation***

  ➔ L'utilisation de ***transformations de modèles*** pour générer des modèles décrits dans ces langages formels

☞ Intégration des Méthodes Formelles avec l'IDM (4/5)

Combinaison d'IDM avec les MFs

- Chacune des deux approches a des points forts et des points faibles
- Les deux approches peuvent être combinées dans le sens où les inconvénients de l'un peuvent être surmontés grâce aux apports de l'autre

	Avantages	*Inconvénients*
IDM	√ Notations conviviales et souvent graphiques √ Génération automatique d'outils de développement √ Transformations automatiques de modèles	✗ Manque de sémantiques formelles ✗ Analyse de modèles impossible
MFs	√ Fondements mathématiques rigoureux √ Analyse formelle de modèles	✗ Notations complexes ✗ Absence d'outils de développement ✗ Manque d'intégration

- La combinaision permet de tirer profit de ces deux approches

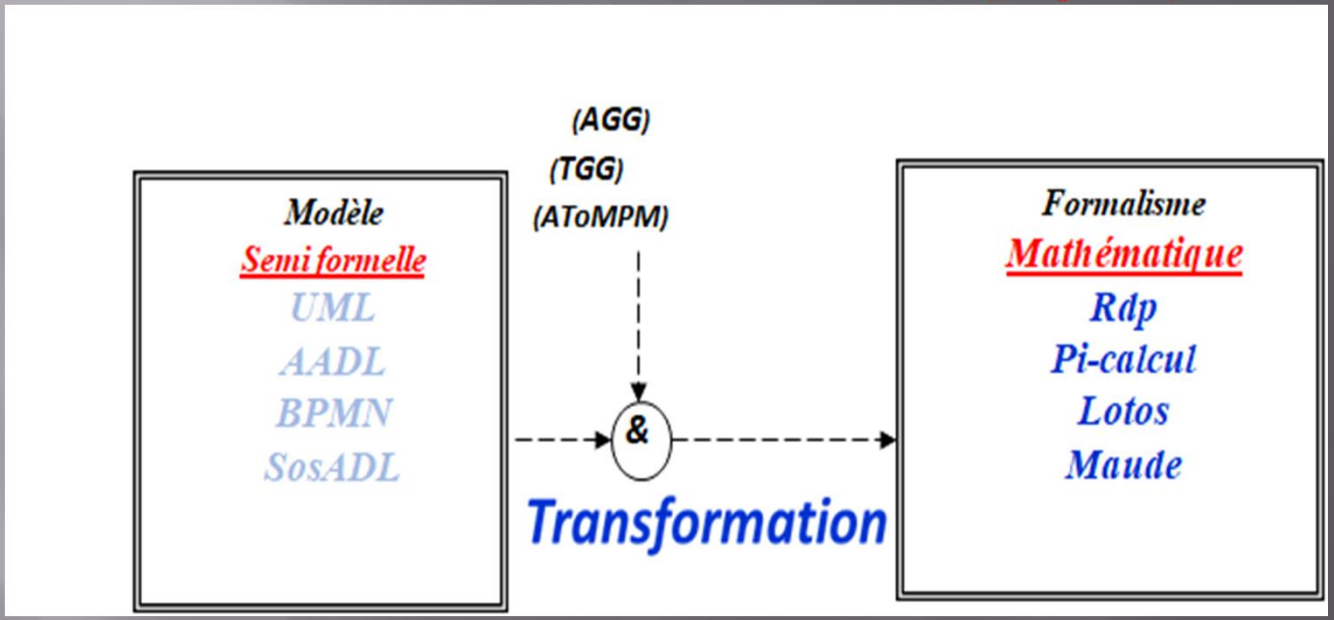
Les avantages et les inconvénients de l'IDM(5/5)

Les Avantages	Les inconvénients
l'automatisation du processus de développement des systèmes. La représentations du système à un haut niveau d'abstraction qui sont les modèles. Le processus de développement, dans cette approche, revient à raffiner, maintenir, et éventuellement de transformer les modèles en d'autres modèles ou de générer le code exécutable. Les différentes activités se font d'une manière automatique. La méta-modélisation est un concept clé de l'approche IDM, permet de donner à un langage de modélisation, une notation abstraite, ce qui permet de générer automatiquement son éditeur. La méta modélisation de langages est de plus en plus adoptée pour des domaines spécifiques afin de réduire la complexité et d'exprimer efficacement les concepts du	la définition de la sémantique de ces langages reste une question ouverte et cruciale l'IDM n'apporte rien de nouveau sur le plan théorique aux concepts d'analyse formelle, mais elle peut permettre une meilleure utilisation afin de profiter pleinement de leurs avantages.

PLAN

1. Introduction
2. Principes de l'IDM
3. Architecture MDA
4. Ingénierie Dirigée par les Modèles (IDM)
   *** Transformations de Graphes ***
5. Intégration des Méthodes Formelles avec l'IDM
6. La Vérification d'une Transformation
7. Conclusion

Vérification d'une Transformation A.Pourquoi la vérification d'une transformation ?(1/6)



Plusieurs outils de la transformation sont développés : TGG, AGG, AToMPM, . . . etc.

Plusieurs transformations sont réalisées notamment des modèles semi-formelles (UML, BPMN, AADL) vers des modèles formels.

Mais ces *transformations et leurs outils souffrent de manque de vérification*

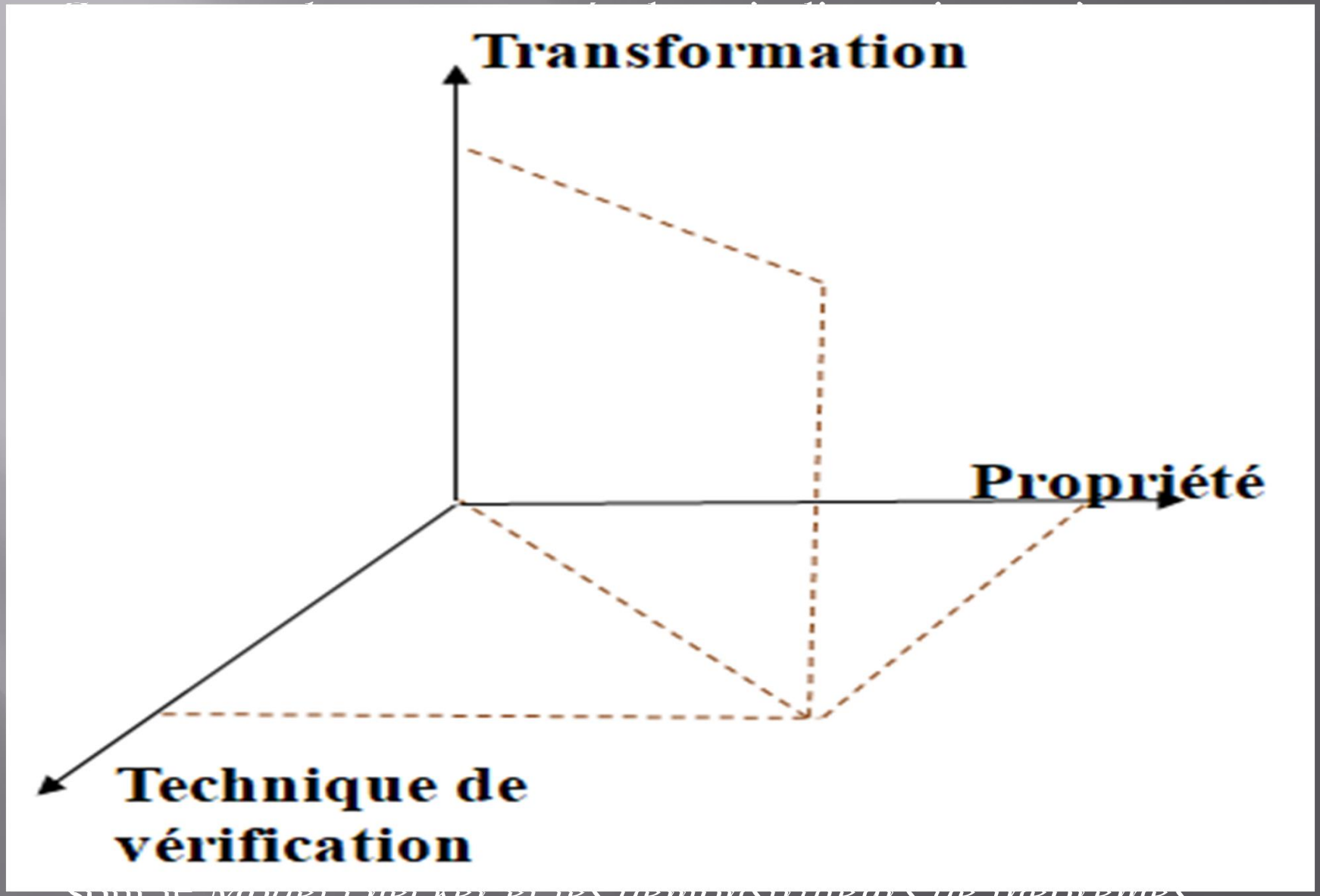
Vérification d'une Transformation A.Pourquoi la vérification d'une transformation ?(2/6)

Propriétés à Vérifier:

- Terminaison de la transformation
- Préservation de la sémantique du modèle source
- Confluence
- Invariants
- Complétude
- Absence d'interblocage & boucle infinie ...

- Donc, il est intéressant d'appeler à de nouvelles méthodes, techniques et outils pour le développement des transformations de modèles.

B .Approche de trois dimensions(3/6)



sont le *Model checker* et les démonstrateurs de théorèmes.

B .Approche de trois dimensions(4/6)

Il existe deux classes de propriétés :

- Propriétés fonctionnelles de la transformation elle-même, il existe 2 types:

1.1.Terminaison: Assurer que la transformation termine après un nombre des étapes finie .Assurer l'existence d'un modèle cible.

1.2.Confluence: l'ordre d'application des règles de transformation n'est pas important. Assurer que la transformation toujours produire le même résultat.

- Propriétés liées aux modelés source et cible, il existe trois types:

2.1.Conformité: Un modèle est valide s'il est conforme à son méta modèle. Est vérifié automatiquement dans des Framework de modélisation.

2.2. Relation syntaxique : Certains éléments(source)seront transformé en d'autres éléments (cible)

2.3.Relation sémantique : Liée la signification des deux modèles (source et cible). Construire les LTS de la sémantique des deux modèles.

B .Approche de trois dimensions(5/6)
Méthode : Model Checker

Le principe de fonctionnement du modèle Checker consiste en trois phases. La première phase est la modélisation du comportement du système. La deuxième phase est la spécification des propriétés attendues du système dans la logique temporel LTL ou CTL. Finalement, il répond avec oui si la propriété est satisfaite sinon un contre-exemple est généré automatiquement qui montre un scénario possible d'erreur. Cependant, modèle Checker souffre de problème de l'explosion combinatoire de graphe d'états.

Dans le cas du démonstrateur de théorèmes, le système et les propriétés attendues sont exprimés en utilisant des descriptions dans une logique mathématique par exemple Coq utilise le langage Galina. La preuve d'une propriété consiste en une ou plusieurs étapes et chaque étape peut faire appel aux axiomes, aux règles, aux définitions ou aux lemmes qui ont été déjà prouvés. L'avantage est qu'ils peuvent spécifier et effectuer des preuves sur des systèmes infinis à travers des techniques comme l'induction. Cependant,les démonstrateurs de théorèmes sont très coûteux, nécessitent des compétences avancées en matière de preuves ainsi que

C. Approches de Vérification formelle de transformations de modèles

Méthode : Vérification d'une transformation basée sur le Model checker(6/6)

Cette approche considère la transformation comme une fonction mathématique qui compose de plusieurs fonction récursives. Elle consiste en deux parties:

En Isabelle/HOL:

- Description de modèle source
- Description de modèle cible
- Description de la transformation
- Spécification des propriétés attendues
- Preuve de correction des propriétés attendues

En EMF/Scala:

- Définition des méta-modèles en utilisant EMF.
- Génération de code Scala de la fonction Trans.
- Définition des fonctions Trans1 et Trans2.
- Composition des fonctions :Trans2 (Trans (Trans1(modèle source)))-->modèle cible.

PLAN

1. Introduction
2. Principes de l'IDM
3. Architecture MDA
4. Ingénierie Dirigée par les Modèles (IDM)
   *** Transformations de Graphes ***
5. Intégration des Méthodes Formelles avec l'IDM
6. La Vérification d'une Transformation
7. Conclusion

Conclusion

☞ Les travaux réalisés dans ce cours s'inscrivent dans le cadre de la Modélisation Multi-paradigme (l'ingénierie dirigée par les modèles).

☞ Le point clé dans ce cours est la possibilité de transformer les modèles d'un espace de modélisation ou d'un niveau d'abstraction vers un autre adapté à une préoccupation particulière.

☞ Le cours est basé sur :

- La Méta-modélisation
  - Modélisation de formalismes et génération automatique de leurs éditeurs
- Les Grammaires de Graphes
  - Spécification des transformation de modèles
- L'outil AToMPM: outil de Modélisation Multi-paradigme

Automatiser l'utilisation conjointe de multiples modèles décrits dans différents paradigmes

?

VOS QUESTIONS