

# Ingénierie des Modèles

## Introduction Générale

Eric Cariou

Master Technologies de l'Internet 2<sup>ème</sup> année

Université de Pau et des Pays de l'Adour  
UFR Sciences Pau – Département Informatique

Eric.Cariou@univ-pau.fr

1

## Évolution des technologies

- ◆ Évolution permanente des technologies logicielles
- ◆ Exemple : systèmes distribués
  - ◆ Faire communiquer et interagir des éléments distants
- ◆ Évolution dans ce domaine
  - ◆ C et sockets TCP/UDP
  - ◆ C et RPC
  - ◆ C++ et CORBA
  - ◆ Java et RMI
  - ◆ Java et EJB
  - ◆ C# et Web Services
  - ◆ A suivre ...

3

## Évolution des technologies

- ◆ Principes de CORBA
  - ◆ Indépendant des langages de programmation
  - ◆ Indépendant des systèmes d'exploitation
  - ◆ Pour interopérabilité de toutes applications, indépendamment des technologies utilisées
- ◆ Mais CORBA ne s'est pas réellement imposé en pratique
  - ◆ D'autres middleware sont apparus : Java RMI
  - ◆ Plusieurs implémentations de CORBA
    - ◆ Ne réalisant pas tous entièrement la norme
  - ◆ Les composants logiciels sont arrivés
    - ◆ OMG a développé un modèle de composants basé sur CORBA : CCM (Corba Component Model)
    - ◆ Mais en concurrence avec les EJB de Java, MS .Net, les Web Services ou services REST

5

## Plan

- ◆ Pourquoi l'ingénierie des modèles (IDM)
  - ◆ Constat sur l'évolution des technologies
  - ◆ Approche MDA de l'OMG
- ◆ Pour quoi faire
  - ◆ But, problématique de l'IDM
- ◆ Définitions générales
  - ◆ Modèles, méta-modèles, transformations

2

## Évolution des technologies

- ◆ Idée afin de limiter le nombre de technologies
  - ◆ Normaliser un standard qui sera utilisé par tous
- ◆ Pour les systèmes distribués
  - ◆ Normaliser un intergiciel (middleware)
  - ◆ C'était le but de CORBA
- ◆ CORBA : Common Object Request Broker Architecture
  - ◆ Norme de l'OMG : Object Management Group
    - ◆ Consortium d'industriels (et d'académiques) pour développement de standards

4

## Évolution des technologies

- ◆ De plus, « guerre » de la standardisation et/ou de l'universalité
  - ◆ Sun/Oracle : Java
    - ◆ Plate-forme d'exécution universelle
    - ◆ Avec intergiciel intégré (RMI)
  - ◆ OMG : CORBA
    - ◆ Intergiciel universel
  - ◆ Microsoft et d'autres : Web Services
    - ◆ Intéropérabilité universelle entre composants
      - ◆ Intergiciel = HTTP/XML
- ◆ Middleware ou *middle war* \* ?

\* citation de Jean-Marc Jézéquel

6

## Évolution des technologies

- ◆ Evolutions apportent un gain réel
  - ◆ Communications distantes
    - ◆ Socket : envoi d'informations brutes
    - ◆ RPC/RMI/CORBA : appel d'opérations sur un élément distant *presque* comme s'il était local
    - ◆ Composants : meilleure description et structuration des interactions (appels d'opérations)
  - ◆ Paradigmes de programmation
    - ◆ C : procédural
    - ◆ Java, C++, C# : objet
      - ◆ Encapsulation, réutilisation, héritage, spécialisation ...
    - ◆ EJB, CCM : composants
      - ◆ Meilleure encapsulation et réutilisation, déploiement ...

7

## Évolution des technologies

- ◆ Quelles conséquences en pratique de cette évolution permanente ?
- ◆ Si veut profiter des nouvelles technologies et de leurs avantages :
  - ◆ Nécessite d'adapter une application à ces technologies
- ◆ Question : quel est le coût de cette adaptation ?
  - ◆ Généralement très élevé
    - ◆ Doit réécrire presque entièrement l'application
    - ◆ Car mélange du code métier et du code technique
    - ◆ Aucune capitalisation de la logique et des règles métiers

9

## Évolution des technologies

- ◆ Partant de tous ces constats
  - ◆ Nécessité de découpler clairement la logique métier et de la mise en oeuvre technologique
  - ◆ C'est un des principes fondamentaux de l'ingénierie des modèles
    - ◆ Séparation des préoccupations (*separation of concerns*)
- ◆ 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

11

## Évolution des technologies

- ◆ Conclusion sur l'évolution des technologies
  - ◆ Nouveaux paradigmes, nouvelles techniques
  - ◆ Pour développement toujours plus rapide, plus efficace
  - ◆ Rend difficile la standardisation (désuétude rapide d'une technologie)
    - ◆ Et aussi car combats pour imposer sa technologie
- ◆ Principes de cette évolution
  - ◆ Évolution sans fin
  - ◆ La meilleure technologie est ... celle à venir

8

## Évolution des technologies

- ◆ Exemple
  - ◆ Application de calculs scientifiques distribués sur un réseau de machines
  - ◆ Passage de C/RPC à Java/EJB
    - ◆ Impossibilité de reprendre le code existant
      - ◆ Paradigme procédural à objet/composant
  - ◆ Pourtant
    - ◆ Les algorithmes de distribution des calculs et de répartition des charges sur les machines sont indépendants de la technologie de mise en oeuvre
      - ◆ Logique métier indépendante de la technologie

10

## Model Driven Architecture

- ◆ Approche Model-Driven Architecture (MDA) de l'OMG
  - ◆ Origine de l'ingénierie des modèles
  - ◆ Date de 2000
- ◆ Le MDA est né à partir des constatations que nous venons de voir
  - ◆ Évolution continue des technologies
- ◆ But du MDA
  - ◆ Abstraire les parties métiers de leur mise en oeuvre
  - ◆ Basé sur des technologies et standards de l'OMG
    - ◆ UML, MOF, OCL, CWM, QVT ...

12

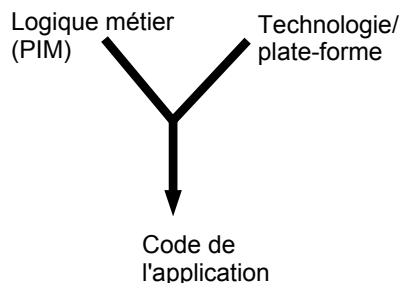
## Model Driven Architecture

- ◆ Le MDA définit 2 principaux niveaux de modèles
  - ◆ PIM : Platform Independent Model
    - ◆ Modèle spécifiant une application indépendamment de la technologie de mise en oeuvre
    - ◆ 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
- ◆ Autres types de modèles
  - ◆ CIM : Computation Independent Model
    - ◆ Spécification du système, point de vue extérieur de l'utilisateur
  - ◆ PDM : Plateform Deployment Model
    - ◆ Modèle d'une plate-forme de déploiement

13

## Model Driven Architecture

- ◆ Cycle de développement d'un logiciel selon le MDA
  - ◆ Cycle en Y
  - ◆ Plus complexe en pratique
    - ◆ Plutôt un cycle en épi



15

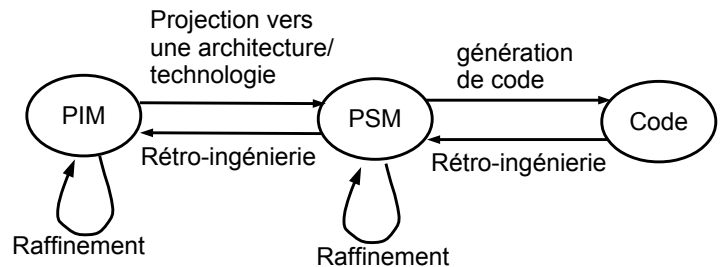
## Model Driven Engineering

- ◆ Limites du MDA
  - ◆ Technologies OMG principalement
- ◆ Ingénierie des modèles
  - ◆ MDE : Model Driven Engineering
    - ◆ IDM : Ingénierie Dirigée par les Modèles
    - ◆ Approche plus globale et générale que le MDA
- ◆ Appliquer les mêmes principes à tout espace technologique et les généraliser
  - ◆ Espace technologique : ensemble de techniques/principes de modélisation et d'outils associés à un (méta)méta-modèle particulier
    - ◆ MOF/UML, EMF/Ecore, XML, grammaires de langages, bases de données relationnelles, ontologies ...
  - ◆ Le MDA est un processus de type MDE

17

## Model Driven Architecture

- ◆ Relation entre les niveaux de modèles



14

## Model Driven Architecture

- ◆ Outils de mise en oeuvre du MDA
  - ◆ Standards de l'OMG
- ◆ Spécification des modèles aux différents niveaux
  - ◆ Langage de modélisation UML
    - ◆ Profils UML
    - ◆ Langage de contraintes OCL
    - ◆ Langages dédiés à des domaines particuliers (CWM ...)
- ◆ Spécification des méta-modèles
  - ◆ Meta Object Facilities (MOF)
- ◆ Langage de manipulation et transformation de modèles
  - ◆ Query/View/Transformation (QVT)

16

## Principes du MDE

- ◆ Capitalisation
  - ◆ Approche objets/composants
    - ◆ Capitalisation, réutilisation d'éléments logiciels/code
  - ◆ MDE
    - ◆ Capitalisation, réutilisation de (parties de) modèles : logique métier, règles de raffinements, de projection ...
- ◆ Abstraction
  - ◆ Modéliser, c'est abstraire ...
  - ◆ Par exemple abstraction des technologies de mise en oeuvre
    - ◆ Permet d'adapter une logique métier à un contexte
    - ◆ Permet d'évoluer bien plus facilement vers de nouvelles technologies

18

## Principes du MDE

- ◆ Modélisation
  - ◆ La modélisation n'est pas une discipline récente en génie logiciel
  - ◆ Les processus de développement logiciel non plus
    - ◆ RUP, Merise ...
  - ◆ C'est l'usage de ces modèles qui change
  - ◆ Le but du MDE est
    - ◆ De passer d'une vision plutôt contemplative des modèles
      - ◆ A but de documentation, spécification, communication
    - ◆ A une vision réellement productive
      - ◆ Pour générer le code final du logiciel pour une technologie de mise en oeuvre donnée

19

## Principes du MDE

- ◆ Pour passer à une vision productive, il faut
  - ◆ Que les modèles soient bien définis
    - ◆ Notion de méta-modèle
  - ◆ Pour que l'on puisse les manipuler et les interpréter via des outils
    - ◆ Avec traitement de méta-modèles différents simultanément
      - ◆ Pour transformation/passage entre deux types de modèles
    - ◆ Référentiels de modèles et de méta-modèles
    - ◆ Outils et langages de transformation, de projection, de génération de code
    - ◆ Langages dédiés à des domaines (DS(M)L : Domain Specific Modelling Language)

21

## Modèle

- ◆ Un modèle est une description, une spécification partielle d'un système
  - ◆ Abstraction de ce qui est intéressant pour un contexte et dans un but donné
  - ◆ Vue subjective et simplifiée d'un système
- ◆ But d'un modèle
  - ◆ Faciliter la compréhension d'un système
  - ◆ Simuler le fonctionnement d'un système
- ◆ Exemples
  - ◆ Modèle économique
  - ◆ Modèle démographique
  - ◆ Modèle météorologique

23

## Principes du MDE

- ◆ Séparation des préoccupations
  - ◆ Deux principales préoccupations
    - ◆ Métier : le coeur de l'application, sa logique
    - ◆ Plate-forme de mise en oeuvre
  - ◆ Mais plusieurs autres préoccupations possibles
    - ◆ Sécurité
    - ◆ Interface utilisateur
    - ◆ Qualité de service
    - ◆ ...
  - ◆ Chaque préoccupation est modélisée par un ... modèle
  - ◆ Intégration des préoccupations
    - ◆ Par transformation/fusion/tissage de modèles
      - ◆ Conception orientée aspect

20

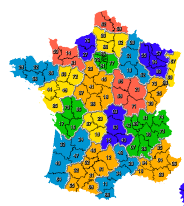
## Définitions

- ◆ Notions fondamentales dans le cadre du MDE
  - ◆ Modèle
  - ◆ Méta-modèle
  - ◆ Transformation de modèle
- ◆ Nous allons donc les définir précisément
  - ◆ En reprenant les notations et exemples des documents réalisés par l'AS CNRS MDA
    - ◆ "L'ingénierie dirigée par les modèles. Au-delà du MDA", collectif sous la direction de JM. Favre, J. Estublier et M. Blay-Fornarino, 2006, Hermes / Lavoisier

22

## Modèle

- ◆ Différence entre spécification et description
  - ◆ Spécification d'un système à *construire*
  - ◆ Description d'un système *existant*
- ◆ Relation entre un système et un modèle
  - ◆ ReprésentationDe (notée  $\mu$ )



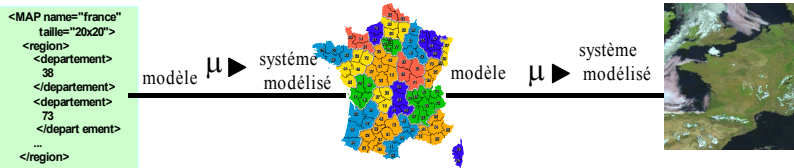
modèle  $\mu$  système modélisé  
Représente ►



24

## Modèle

- ◆ Un modèle représente un système modélisé
  - ◆ De manière générale, pas que dans un contexte de génie logiciel ou d'informatique
  - ◆ Un modèle peut aussi avoir le rôle de système modélisé dans une autre relation de représentation

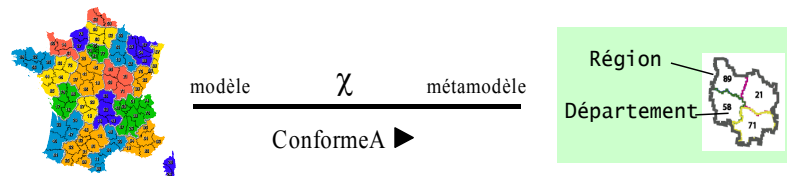


- ◆ Modèle XML de la carte de la France administrative qui est un modèle de la France « réelle »

25

## Méta-modèle

- ◆ Un modèle est conforme à son méta-modèle



### Notes

- ◆ On ne parle pas de relation d'instanciation
  - ◆ Un modèle n'est pas une instance d'un méta-modèle
    - ◆ Instanciation est un concept venant de l'approche objet
    - ◆ Approche objet qui ne se retrouve pas dans tous les espaces technologiques
- ◆ Un méta-modèle n'est pas un modèle de modèle
  - ◆ Raccourci ambigu à éviter
  - ◆ Le modèle XML de la carte administrative de la France n'est pas le méta-modèle des cartes administratives

27

## Méta-modèle et Langage

- ◆ Lien entre méta-modèle et langage
  - ◆ Un méta-modèle est un modèle qui définit le langage pour définir des modèles
  - ◆ Langage
    - ◆ Système abstrait
  - ◆ Méta-modèle
    - ◆ Définition explicite et concrète d'un langage
    - ◆ Un méta-modèle modélise alors un langage
  - ◆ Un méta-modèle n'est donc pas un langage

29

## Modèle

- ◆ Un modèle est écrit dans un langage qui peut être
  - ◆ Non ou peu formalisé, la langue naturelle
    - ◆ Le français, un dessin ...
  - ◆ Formel et bien défini, non ambigu
    - ◆ Syntaxe, grammaire, sémantique
    - ◆ On parle de méta-modèle pour ce type de langage de modèle
- ◆ Pour les modèles définis dans un langage bien précis
  - ◆ Relation de conformité
    - ◆ Un modèle est conforme à son méta-modèle
    - ◆ Relation EstConformeA (notée  $\chi$ )

26

## Méta-modèle

- ◆ Cette relation de conformité est essentielle
  - ◆ Base du MDE pour développer les outils capables de manipuler des modèles
  - ◆ Un méta-modèle est une entité de première classe
- ◆ Mais pas nouvelle
  - ◆ Un texte écrit est conforme à une orthographe et une grammaire
  - ◆ Un programme Java est conforme à la syntaxe et la grammaire du langage Java
  - ◆ Un fichier XML est conforme à sa DTD ou son schéma
  - ◆ Une carte doit être conforme à une légende
  - ◆ Un modèle UML est conforme au méta-modèle UML

28

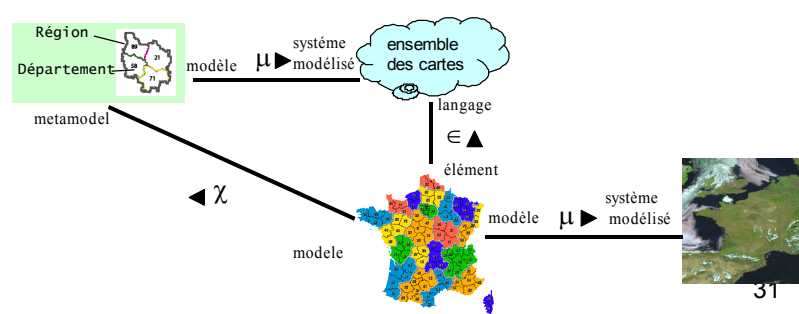
## Méta-modèle et Langage

- ◆ En linguistique
  - ◆ Un langage est défini par l'ensemble des phrases valides écrites dans ce langage
  - ◆ Une grammaire est un modèle de langage
    - ◆ Une grammaire est un méta-modèle
- ◆ Relation entre langage et modèle
  - ◆ Un modèle est un élément valide de ce langage
    - ◆ Une phrase valide du langage en linguistique
  - ◆ Relation d'appartenance
    - ◆ AppartientA, notée  $\epsilon$

30

## Relations générales

- ◆ Exemple de la carte
  - ◆ Une carte modélise un pays selon un point de vue
  - ◆ Le méta-modèle de la carte est sa légende
  - ◆ La légende définit un ensemble de cartes valides
  - ◆ Une carte conforme à une légende appartient à cet ensemble



31

## Les concepts de l'IDM en une image ...



« One and three chairs », Joseph Kosuth, 1965

33

## Transformation

- ◆ Opération générale de manipulation de modèle
  - ◆ Prend un (ou plusieurs) modèle(s) source en entrée
  - ◆ Fournit un (ou plusieurs) modèle(s) cible en sortie
- ◆ Transformation endogène
  - ◆ Les modèles source et cible sont conformes au même méta-modèle
    - ◆ Transformation d'un modèle UML en un autre modèle UML
- ◆ Transformation exogène
  - ◆ 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

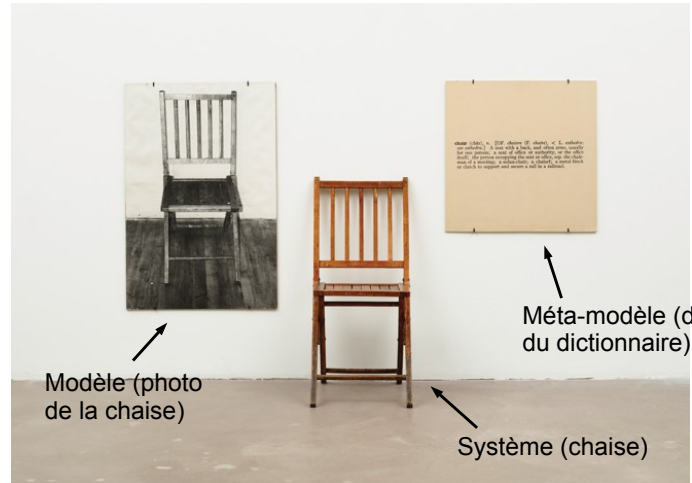
35

## Relations générales

- ◆ Exemple avec un langage de programmation
  - ◆ Un programme Java modélise/simule un système (le programme à l'exécution)
  - ◆ Un programme Java est conforme à la grammaire du langage Java
  - ◆ La grammaire de Java modélise le langage Java et donc tous les programmes valides
  - ◆ Le programme Java appartient à cet ensemble
- ◆ Exemple avec UML
  - ◆ Un diagramme UML modélise un système
  - ◆ Un diagramme UML est conforme au méta-modèle UML
  - ◆ Le méta-modèle UML définit l'ensemble des modèles UML valides
  - ◆ Un modèle UML appartient à cet ensemble

32

## Les concepts de l'IDM en une image ...



« One and three chairs », Joseph Kosuth, 1965

34

## Conclusion

- ◆ Le MDE est une nouvelle approche pour concevoir des applications
  - ◆ En plaçant les modèles et surtout les méta-modèles au centre du processus de développement dans un but productif
    - ◆ Les modèles sont depuis longtemps utilisés mais ne couvrait pas l'ensemble du cycle de vie du logiciel
    - ◆ Les méta-modèles existent aussi depuis longtemps (grammaires, DTD XML, ...) mais peu utilisés en modélisation « à la UML »
    - ◆ Nouvelle vision autour de notions déjà connues : le méta-modèle devient le point clé de la modélisation
      - ◆ Automatisation de la manipulation des modèles
      - ◆ Création de DSL : outillage de modélisation dédié à un domaine
  - ◆ Automatisation du processus de développement
    - ◆ Application de séries de transformations, fusions de modèles

36

## Conclusion

- ◆ Problématiques, verrous technologiques, besoin en terme d'outils dans le cadre du MDE
- ◆ Définition précise de modèles et de méta-modèles
- ◆ Langage et outils de transformations, fusion de modèles, tissage de modèles/aspects
- ◆ Traçabilité dans le processus de développement, reverse-engineering
- ◆ Gestion de référentiels de modèles et méta-modèles
  - ◆ Y compris à large échelle (nombre et taille des (méta)modèles)
- ◆ Exécutabilité de modèles
  - ◆ Connaissance et usage du modèle à l'exécution, modèle intégré dans le code
- ◆ Validation/test de modèles et de transformations

37

## Contenu du module

- ◆ Module couvre un nombre important de points clés du MDE avec application concrète via des outils
- ◆ Définition de modèles/méta-modèles
  - ◆ Ecore/EMF, profils UML
  - ◆ Définition de DS(M)L
- ◆ Transformations de modèles
  - ◆ ATL, Kermeta
- ◆ Exécutabilité de modèles
  - ◆ Exécution de statecharts avec PauWare
- ◆ Vérification & Validation
  - ◆ Application aux transformation de modèles
- ◆ Création d'éditeurs de modèles
  - ◆ Xtext

39

## Conclusion

- ◆ Ingénierie des modèles ou ...  
... ingénierie des langages ?
- ◆ Désormais le cœur de l'IDM est souvent vu comme
  - ◆ La définition de langages spécifiques via des méta-modèles : DSL
  - ◆ La création de tout un outillage autour de ces langages
    - ◆ Transformateurs de modèles, générateurs de code/texte, éditeurs textuels ou graphiques, ...
    - ◆ Grace à des frameworks comme Eclipse/EMF par exemple qui intègre beaucoup d'outils et de plugins
  - ◆ Une discipline permettant donc de faire de l'ingénierie des langages

38