Ingénierie Dirigée par les Modèles

F. Mallet

Ingénierie Dirigée par les Modèles

- Programmation logicielle
  - Construire des logiciels pour que les ordinateurs soient utiles aux utilisateurs pour résoudre des problèmes variés

- Ingénierie Dirigée par les Modèles
  - Construire des modèles pour aider les programmeurs à construire des logiciels plus compliqués et meilleurs

Modularité vs. Composition

- Deux phases pour résoudre un problème
  - Modularité: *“Divide and Conquer”*
    - Décomposer un problème en sous-problèmes plus simples
  - Composition
    - Composer les solutions partielles pour construire la solution complète

- Manuel ou automatique
  - Les approches manuelles ne passent pas à l’échelle
  - Il faut construire des outils pour décomposer et recomposer les solutions partielles

La crise du logiciel est toujours d'actualité !

"The major cause of the software crisis is that the machines have become several orders of magnitude more powerful!" Edsger Dijkstra, 1972

- La programmation structurée puis orientée-objet devaient résoudre la crise

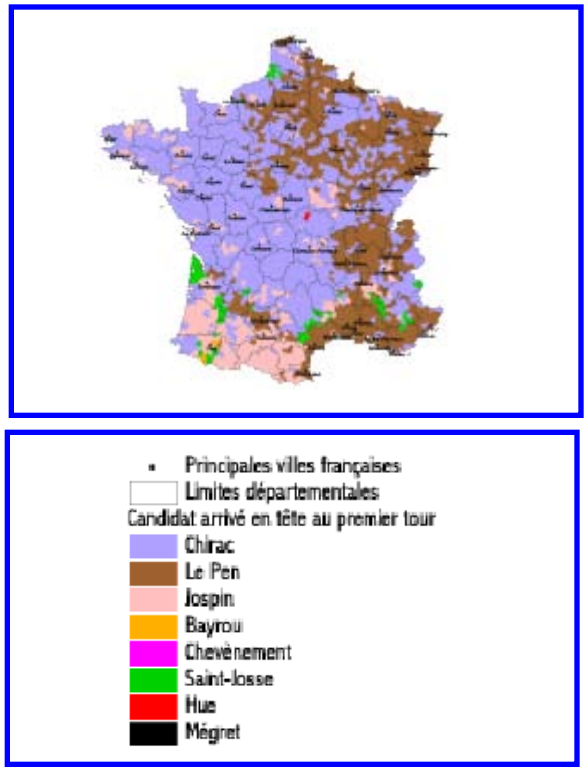
- Le problème persiste car depuis 70s les ordinateurs ont encore beaucoup évolués
  - La distance entre les systèmes à construire et les abstractions disponibles se creuse

Exemple de modèles

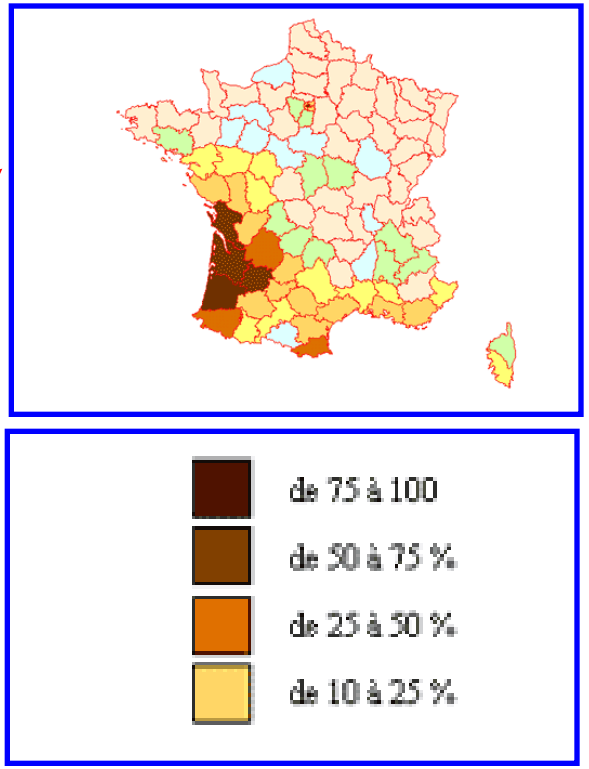
- **Interpréter une carte nécessite une légende**
- **La carte et sa légende utilisent le même formalisme**
  - **ici une représentation graphique**
- **La carte est un modèle**
  - **Une simplification d'une ville**
- **La légende est son méta-modèle**
  - **Définit les éléments utilisés pour construire la carte**

Sans le méta-modèle, que signifie le modèle ?

Candidats à l'élection présidentielle en France en 2002



Pourcentage des villes envahies par les termites

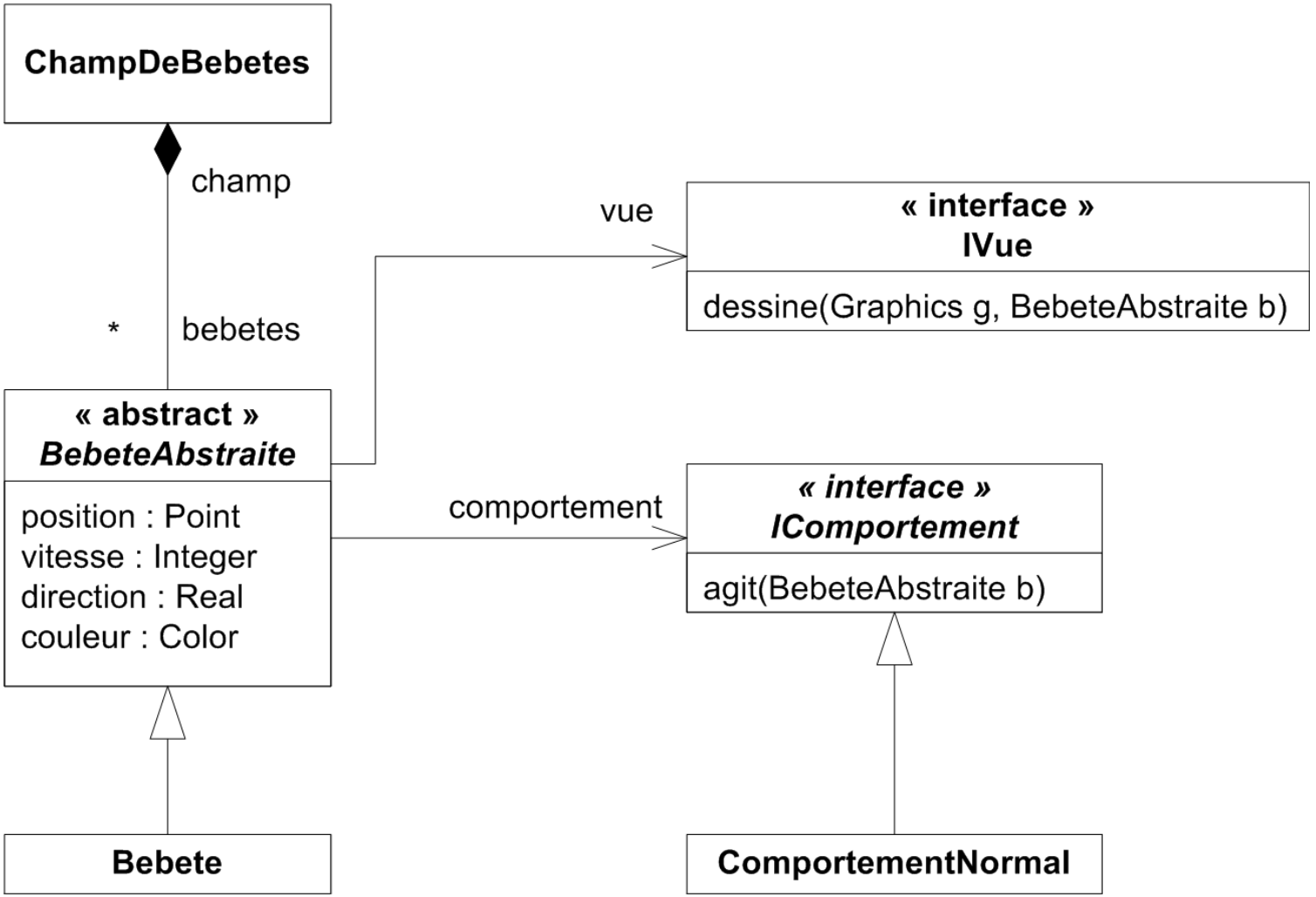


Programme ou modèle

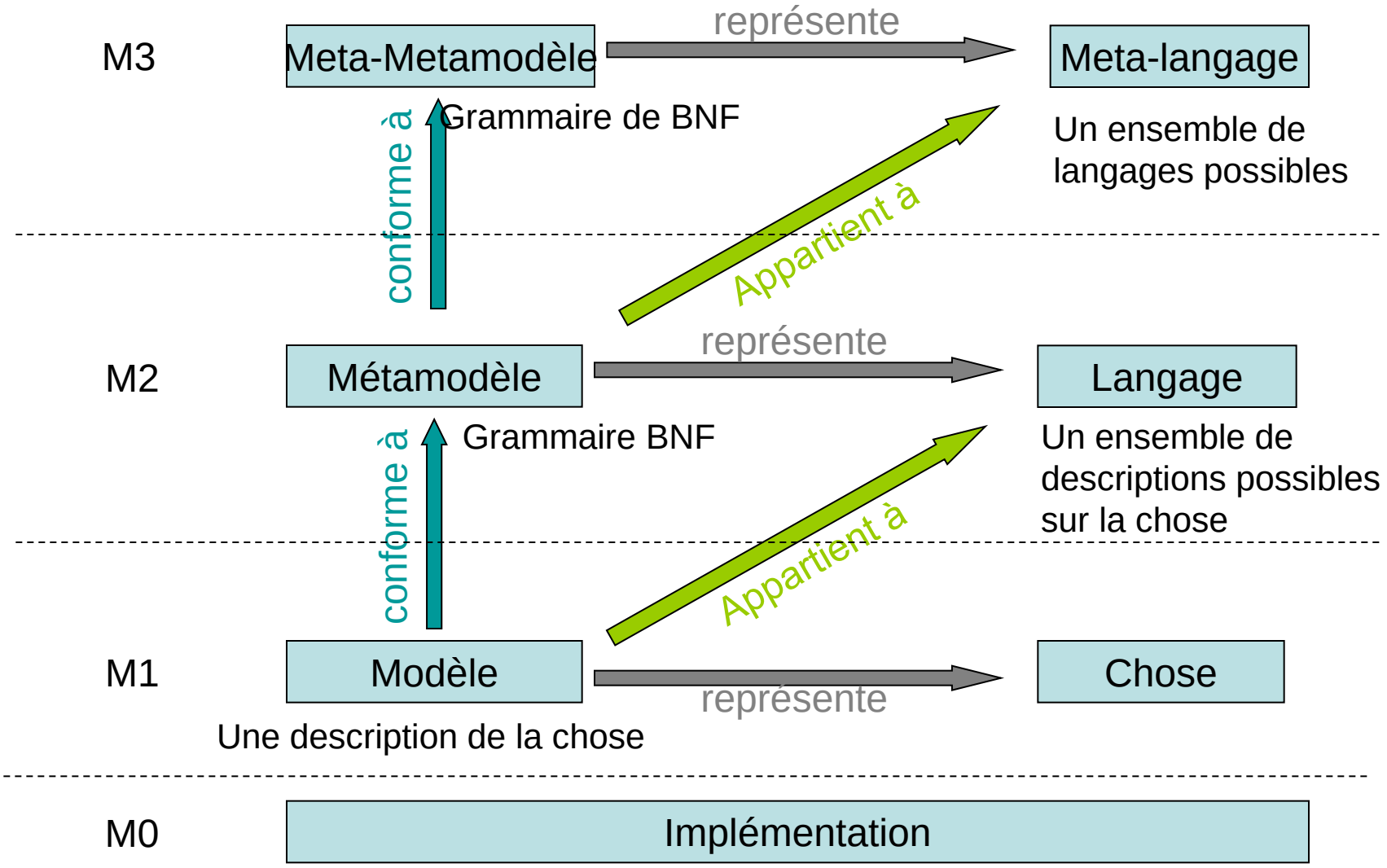
```
abstract class BebeteAbstraite {
    protected int x, y;             // position à l'écran
    int vitesse;                    // vitesse en pixels par
       second
    double direction;               // en radians [0 - 2 PI[
    protected Color couleur;    // Couleur de remplissage
    protected ChampDeBebetes champ;     // Le champ
    static final int TAILLEGRAPHIQUE = 10;

    protected BebeteAbstraite(ChampDeBebetes champ,
       int x, int y, Color couleur) {
       this.champ = champ;
       this.x = x;
       this.y = y;
       this.couleur = couleur;

       Random generateur = new Random();
       direction = generateur.nextFloat() * 2 * Math.PI;
       vitesse = generateur.nextInt(champ.vitesseMax);
    }

    public abstract void agit();

    protected IComportement comportement;
```

```
 public boolean voit(BebeteAbstraite b) {
     double angle = Math.atan2 (b.getY()-y, b.getX()-x);
     double diff = Math.abs(angle-direction)%(2*Math.PI);
     if (diff>Math.PI) diff=2*Math.PI-diff;
     return diff<champDeVue/2;
 }

 public double getDistance(IBebete b) {
     return distanceDepuisUnPoint(b.getX(), b.getY());
 }

 double distanceDepuisUnPoint(double x1, double y1){
       // rend la distance entre la bébête et un point donné
     return Math.sqrt((x1 - x)*(x1-x) + (y1 - y)*(y1 - y));
 }

 protected IVueBebete vue = null;
 void seDessine(Graphics g) {
     if (this.vue != null)
         this.vue.dessine(g, this);
 }

 final public void translate(double dx, double dy) {
     this.x += dx;
     this.y += dy;
 }
}
```

Programme ou Modèle



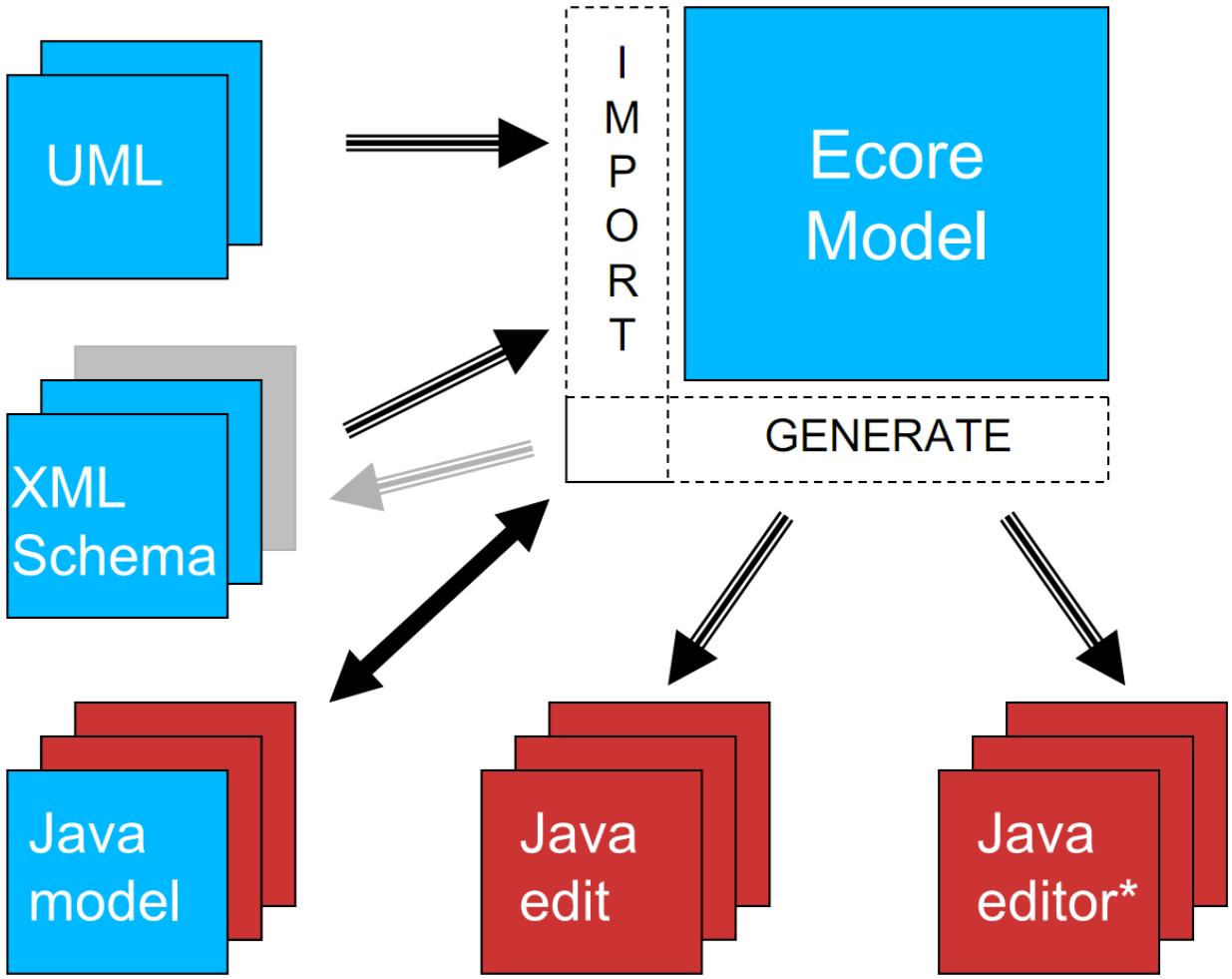
Modèles/Métamodèles/Langages



Eclipse Modeling Framework

- Environnement de modélisation avec génération de code
  - Pour construire des outils basés sur un modèle de données
  - Le modèle est décrit en **XMI** (XML Metadata Interchange)
- Importer du code existant pour construire le modèle
  - Java avec annotations
  - Documents XML (**XSD** – XML Schema Definition)
  - Outil UML (e.g., Rational Rose)
- Génération de code
  - Ensemble de classes et interfaces Java
  - Un environnement Edit/Editor (arbre d'édition)
- Plusieurs (et de plus en plus) d'extensions
  - Générer un éditeur graphique (**GMF** – Graphical Modeling Framework)
  - Générer un parseur – syntaxe concrète (TCS, Sintaks)

EMF import/export

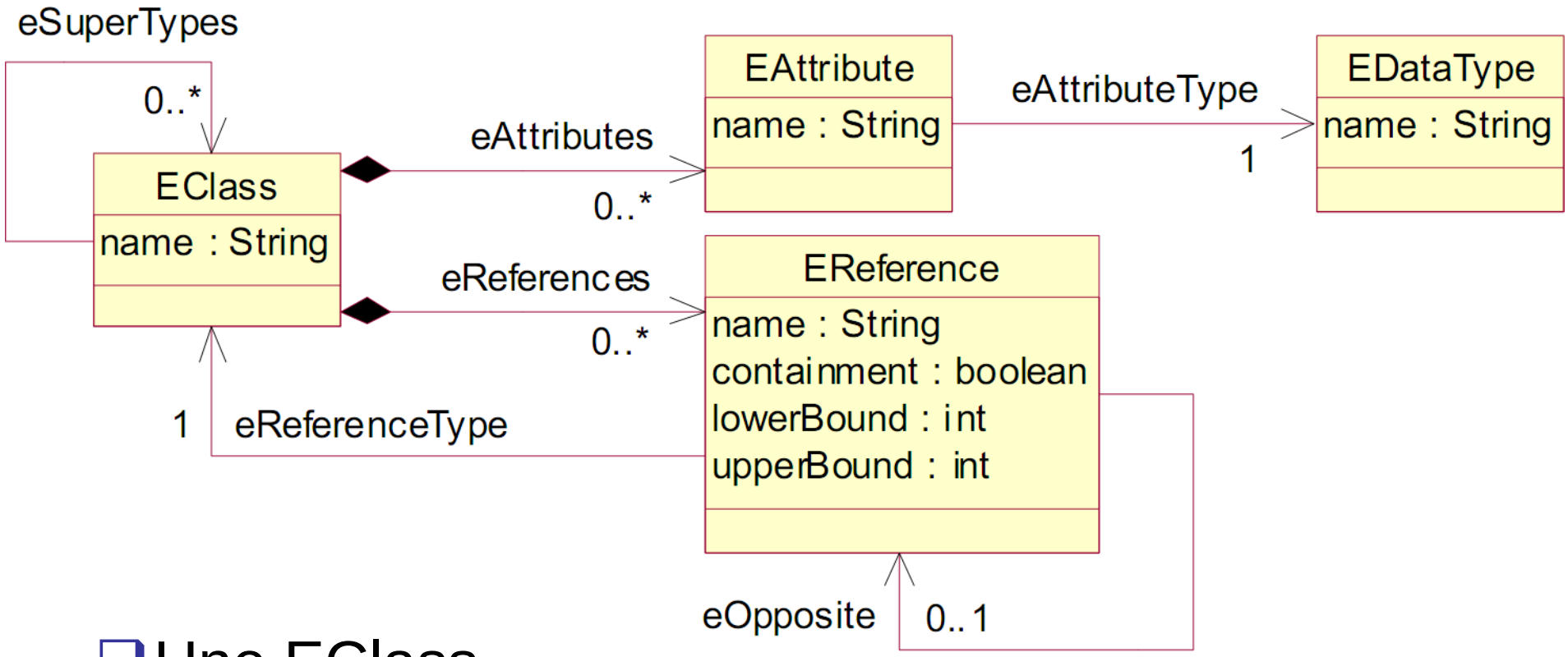


IBM, Ed. Merks & D. Steinberg, EclipseCon 2005

EMF et UML ?

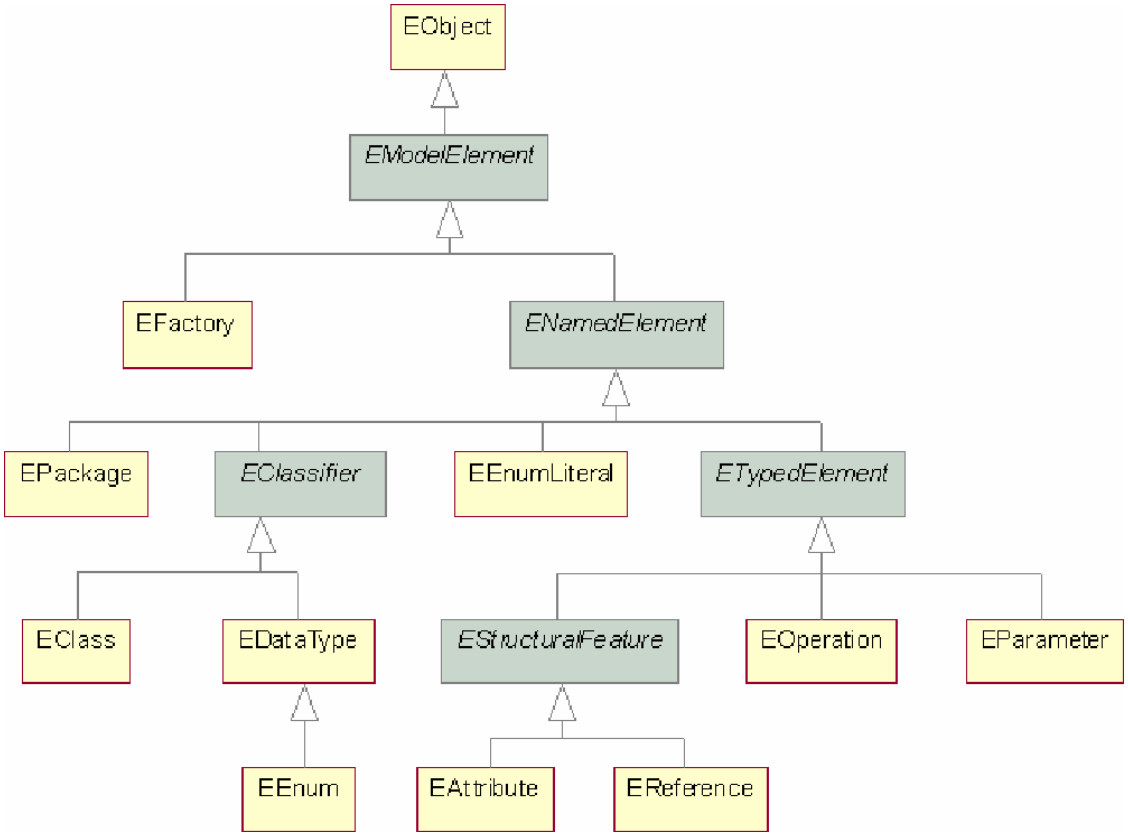
- **MOF**: Méta-métamodèle (M3) de l'OMG
  - **M**eta-**O**bject **F**acilities
  - EMOF (Essential MOF) est un sous-ensemble
- EMF
  - Etait à l'origine une simple implémentation du MOF
  - A évolué séparément pour donner **ECORE**
- Deux modèles différents
  - OMG # Eclipse Foundation

Ecore: EMF méta-métamodèle (M3)



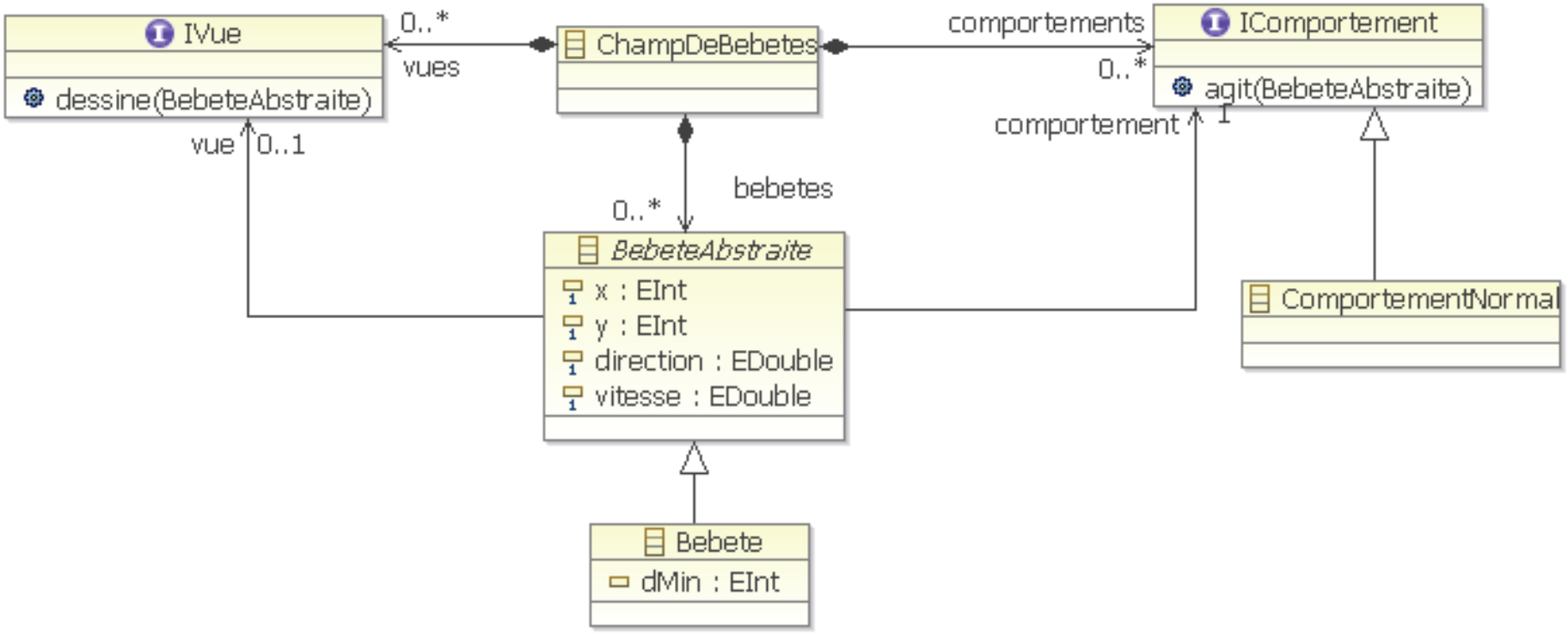
- Une EClass
  - Possède des attributs et références typés
  - Peuvent avoir des super types (autres EClass)

Ecore: EMF méta-métamodèle (M3)



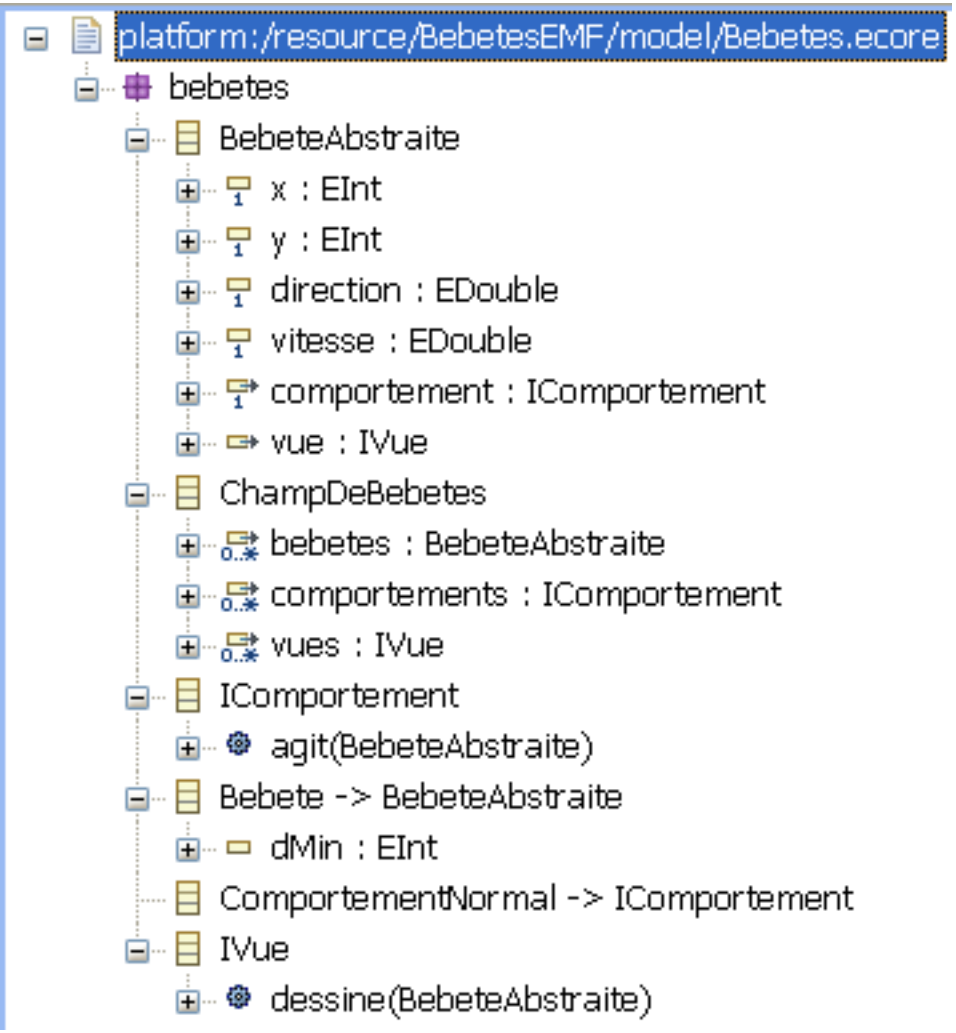
- Petit méta-métamodèle
  - Améliore le modèle d'introspection de Java
  - Seulement des éléments structurels

Exemple – Les bébêtes

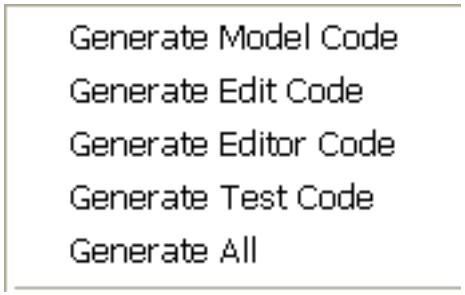


- Diagramme Ecore
  - Bebetes.ecorediag + Bebetes.ecore

Modèle de génération: .genmodel



- Importer l'Ecore
- Créer un .genmodel
  - Quoi générer
  - Où générer
  - Comment générer



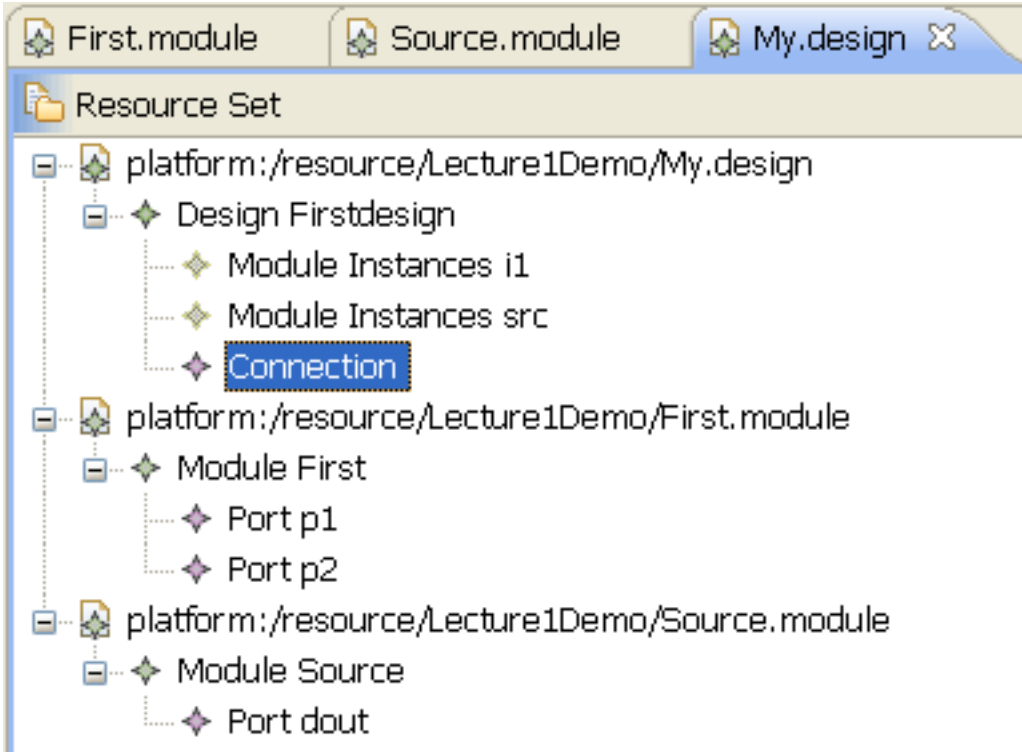
Générer code/Edit/Editor

- Modèle abstrait
  - Interfaces Java

```
public interface BebeteAbstraite
    extends EObject {
 /** @generated */
  int getX();
  void setX(int value);
  double getDirection();
```

- Une implantation
  - Avec écouteurs

```
public abstract class
    BebeteAbstraiteImpl extends
    EObjectImpl implements
    BebeteAbstraite {
  protected static final int
    X_EDEFAULT = 0;
  protected int x = X_EDEFAULT;
```



- Editeur arborescent
- (Dé)Sérialization en XML

BebeteFactory

- **Une Usine à éléments est générée automatiquement**
  - **Pas besoin de connaître les classes concrètes utilisées pour l'implantation**

```
public interface BebetesFactory extends EFactory {
  BebetesFactory eINSTANCE =
   bebetes.impl.BebetesFactoryImpl.init(); /* SINGLETON */

/** @generated */
  ChampDeBebetes createChampDeBebetes();

/** @generated */
  Bebete createBebete();

/** @generated */
  ComportementNormal createComportementNormal();
}
```