

# Métamodéliser avec Eclipse Modeling Framework

Benoit Combemale

`benoit.combemale@irisa.fr`

Université de Rennes 1

Équipe Triskell (IRISA & INRIA)

`http://perso.univ-rennes1.fr/benoit.combemale/`

2010

## Remerciement

Ce cours s'appuie intégralement sur le cours de Sébastien Mosser (cf. `http://rainbow.i3s.unice.fr/~mosser`). Un grand merci !

- “The object-oriented approach **does not adequately address** the computing requirements of the future.”
- “Object-oriented languages **have lost the simplicity** -some would say purity- that made them special and which were the source of their expressive and development power.”
- “**Objects promised reuse**, and we have not seen much success.”



# Agenda

- 1 Modèles, Métamodèles ...
- 2 Exemple : *Graph – Management*
- 3 Eclipse Modeling Framework
- 4 Outil de Modélisation & Spécificités EMF
- 5 EMF : Un canevas logiciel
- 6 Graphical Modeling Framework
- 7 Prenons un peu de recul ...
- 8 Conclusions

# MDE $\equiv$ *Model Driven Engineering*

## Why do we model ?

— Grady Booch

- To abstract
  - Abstractions are not reality, but intentionally incomplete
- To reason about
  - Compare, synthesize, analyze, generate abstractions, . . .
- To document
  - The code is the truth, but not the whole truth
- To transform
  - Abstractions made manifest as the exec. system itself

# Métamodèles & Profils : deux approches différentes !

## Approche Profils dans UML

- Une *adaptation / spécialisation* d'UML
  - ⇒ **Avantage** : repose sur la grande **puissance** d'UML
  - ⇒ **Inconvénient** : repose sur la **grande** puissance d'UML
- *"The UML is reasonably well-defined"* — Grady Booch
  - ⇒ et pourtant, il fait partie des 3 auteurs initiaux.

## Approche Modèles & Métamodèles

- Construction d'outils spécifique à un domaine
  - ⇒ **Avantage** : expression de concepts **centrés sur le métier**
  - ⇒ **Inconvénient** : une n-ième **roue carrée** à ré-inventer ?
- Mais *modéliser*, c'est un domaine comme un autre non ?
  - ⇒ Justement, l'astuce est là !

# L'IDM : “une (méta) façon de penser”

**Méta** : du grec *μετα* “Au delà de ...”

- (*Métaphysique, Métalinguistique, ...*)
- **Méta**données
  - Systèmes de fichiers, ID3/MP3, Web Sémantique ...
- **Méta**langage
  - Grammaires eBNF, UML, XML, ...

## Modèles, Métamodèles & Transformations

- Un **modèle** est une représentation du domaine d'application
- Un **modèle** est conforme à un **métamodèle**
- Un **métamodèle** est un **modèle** (métamétamodèles)
- On passe d'un **modèle** à un autre par transformation

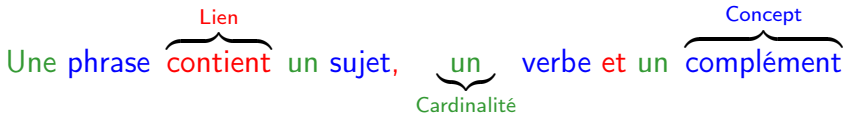
# Mon premier métamodèle ...

(niveau CE1)

*Une phrase contient un sujet, un verbe et un complément*

# Mon premier métamodèle ...

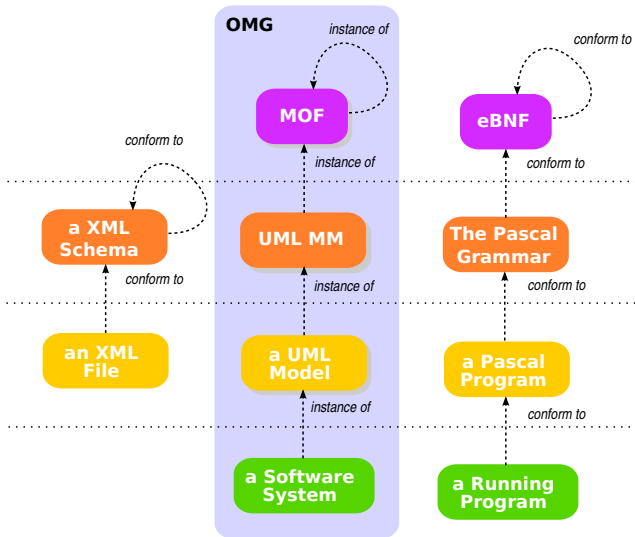
(niveau CE1)



# Mon premier méta-métamodèle ...

(niveau CM1)

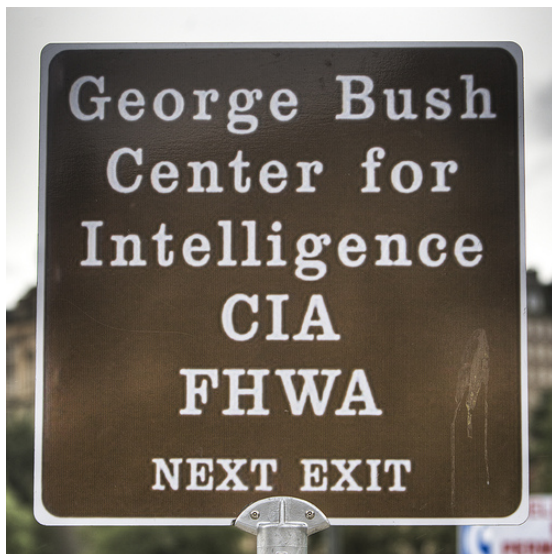




Conformité syntaxique  $\nrightarrow$  conformité sémantique



— Pierre Alain Muller (quoting Magritte)



# Agenda

- 1 Modèles, Métamodèles ...
- 2 Exemple : *Graph – Management*
- 3 Eclipse Modeling Framework
- 4 Outil de Modélisation & Spécificités EMF
- 5 EMF : Un canevas logiciel
- 6 Graphical Modeling Framework
- 7 Prenons un peu de recul ...
- 8 Conclusions

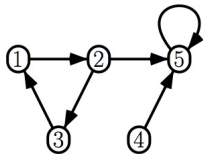
# Notre métier : *Graph – Management*

## Qu'est-ce qu'un graphe orienté (en bref) ?

- Un ensemble de sommet, reliés par des arêtes
- Possède des propriétés : ordre, ...

### Clients potentiels

- Réseaux routiers
- Grilles de calculs
- Bases de donnés
- Systèmes discrets
- Réseaux sociaux
- ...



# Pourquoi métamodéliser ?

(Avantages)

## Une expressivité centrée sur le métier

- Le raisonnement porte directement sur les concepts métiers
  - *“L’ordre d’un graphe est le nombre de ses sommets”*
- Et non pas sur des concepts issus d’un autre métier
  - *“L’ordre d’une instance de la classe Graphe est le nombre d’instances de la classe Sommet référencés depuis la liste sommets, attribut de la classe Graphe”.*

## Exemples de domaines atteignables

- Langage de description : Représentation (GraphViz, ...)
- Algèbre de processus : Vérification ( $\pi$  - calcul, ...)
- ...



# Expressivité métier : “Vérifier un suivant de $s$ ”

En tant qu'expert “*Théorie des Graphes*” :

- On manipule les **arêtes**  $\mathcal{A}$  entre les **nœuds**  $n$  du **graphe**  $\mathcal{G}$
- Et on exprime une **propriété** valable  $\forall g \in \mathcal{G}$  :
  - $n' \in \text{Suivants}(n) \Rightarrow \exists a \in \mathcal{A} \wedge a(n \rightarrow n')$

En tant que qu'expert “*Développement Objet*” :

- On raisonne sur des **Instances** de **Classes**
- On définit une **Méthode** faisant la vérification
  - `public bool estSuivant(Graph g, Node n, Node nPrime)`

# Expressivité métier : “Vérifier un suivant de $s$ ”

En tant qu'expert “*Théorie des Graphes*” :

- On manipule les **arêtes**  $\mathcal{A}$  entre les **noeuds**  $n$  du **graphe**  $\mathcal{G}$
- Et on exprime une **propriété** valable  $\forall g \in \mathcal{G}$  :
  - $n' \in \text{Suivants}(n) \Rightarrow \exists a \in \mathcal{A} \wedge a(n \rightarrow n')$

En tant que qu'expert “*Développement Objet*” :

- On raisonne sur des **Instances** de **Classes**
- On définit une **Méthode** faisant la vérification
  - `public bool estSuivant(Graph g, Node n, Node nPrime)`

Il s'agit simplement de deux métamodèles différents !

# Agenda

- 1 Modèles, Métamodèles ...
- 2 Exemple : *Graph – Management*
- 3 Eclipse Modeling Framework**
- 4 Outil de Modélisation & Spécificités EMF
- 5 EMF : Un canevas logiciel
- 6 Graphical Modeling Framework
- 7 Prenons un peu de recul ...
- 8 Conclusions

# Acronymes & “Vocabulaire”

## MOF : *Meta Object Facility*

(OMG)

- BNF  $\Rightarrow$  Grammaires, MOF  $\Rightarrow$  Métamodèles
- Il existe différentes variantes du MOF :
  - eMOF, cMOF, sMOF
- Standard ISO/IEC19502:2005

## EMF : *Eclipse Modeling Framework*

(IBM)

- Totalement intégré dans *Eclipse*
- Plus ou moins aligné sur EMOF

## XMI, format standard inter-opérable

- Échange de modèles : Encodage XMI
  - Sérialisation XML conforme au métamodèle.

# Acronymes & “Vocabulaire”

## MOF : *Meta Object Facility*

(OMG)

- BNF  $\Rightarrow$  Grammaires, MOF  $\Rightarrow$  Métamodèles
- Il existe différentes variantes du MOF :
  - eMOF, cMOF, sMOF
- Standard ISO/IEC19502:2005

## EMF : *Eclipse Modeling Framework*

(IBM)

- Totalement intégré dans *Eclipse*
- Plus ou moins aligné sur EMOF

## XMI, format standard inter-opérable

- Échange de **méta**modèles : Encodage XMI
  - Sérialisation XML conforme au **méta**métamodèle.

## eMOF (alignement vis à vis d'EMF)

- Construit sur un noyau reflexif
  - Un ensemble de concepts pour décrire des métamodèles
- ⇒ C'est un métamodèle de métamodèle (métamétamodèle)!
- Il en existe  $\neq$  implémentations (ECore, RubyMOF, ...)

## Et le diagramme de classe UML dans tout ça ?

- On en réutilise la syntaxe graphique
- Restriction sémantique (*"small is beautiful"*)
  - Moins expressif (visibilité absente, ...)
  - Trop limité ?

# Concepts manipulés dans EMF

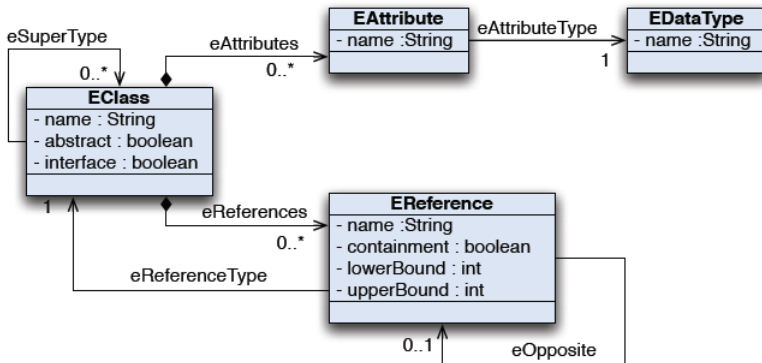


FIG.: Extrait simplifié du méta-modèle Ecore

# Implantation d'ECore ... en ECore !

## Expression du métamodèle ECore dans un métamodèle

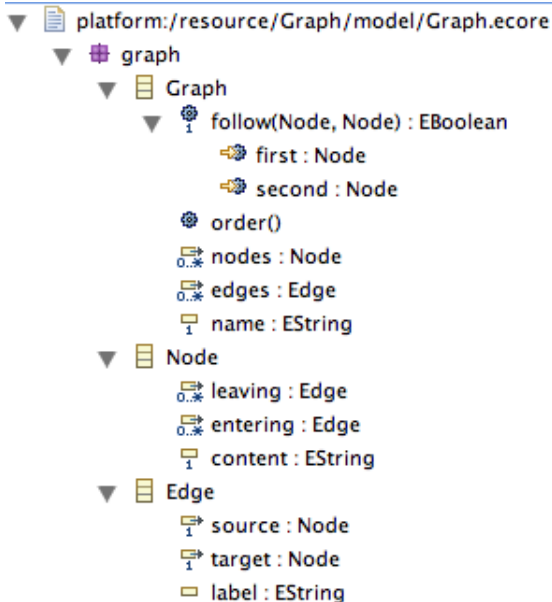
- Rappel : *ECore* est un métamodèle de métamodèle ...  
⇒ On l'implante au même titre qu'un autre métamodèle
  - \$ECLIPSE/plugins/org.eclipse.emf.ecore\_XXX.jar
  - Dans le sous-répertoire models : **Ecore**.ecore
- L'approche est définitivement "méta" :
  - On considère la *modélisation* comme un "domaine métier"
  - Définition des métaoutils pour travailler sur ce domaine
  - Validation naturelle des concepts
- Conséquence : Si EMF l'a fait, on peut aussi le faire !

Tout dans EMF utilise une approche dirigée par les modèles !

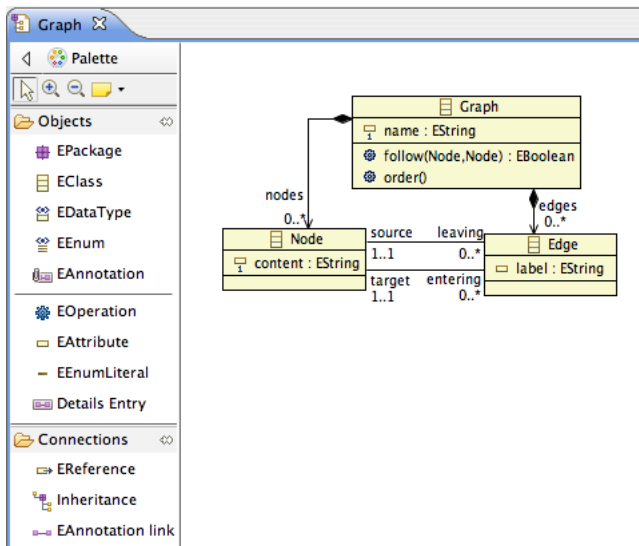
# Agenda

- 1 Modèles, Métamodèles ...
- 2 Exemple : *Graph – Management*
- 3 Eclipse Modeling Framework
- 4 Outil de Modélisation & Spécificités EMF**
- 5 EMF : Un canevas logiciel
- 6 Graphical Modeling Framework
- 7 Prenons un peu de recul ...
- 8 Conclusions

# Fichier ecore : Éditeur arborescent de métamodèles



# Fichier ecorediag : Éditeur graphique de métamodèles



# Définition d'attributs : Nom, Multiplicité & Type

The screenshot shows the Eclipse IDE's Properties window for an attribute named 'content'. The window has several tabs: 'EClass Hierar', 'EClass Refer', 'Console', 'Properties', and 'EMF register'. The 'Properties' tab is active, displaying the following information:

content : EString	
<b>Model</b>	Name: content
<b>Annotation</b>	
Extended Metadata	Lower Bound: 1
GenModel Doc	
<b>Advanced</b>	Upper Bound: 1
	EType: EString [java.lang.String] ...

# Définition de références : Contenance & Opposition

The screenshot shows the EMF editor interface with the following components:

- Toolbar:** EClass Hierar, EClass Refer, Console, Properties, EMF register.
- Header:** source : Node
- Left Panel (Navigation):** Model, Annotation, Extended Metadata, GenModel Doc, Appearance, Advanced.
- Right Panel (Properties):**
  - Name:** source
  - Lower Bound:** 1
  - Upper Bound:** 1
  - Is Containment:**
  - EOpposite:** leaving : Edge

# Définition d'opération : Paramètres

The screenshot displays the Eclipse IDE interface for defining an operation. The top toolbar includes tabs for 'EClass Hierarc', 'EClass Referen', 'Console', 'Properties', and 'EMF registered'. The main window title is 'follow(Node, Node) : EBoolean'. On the left, a sidebar contains a tree view with 'Model', 'Parameter', 'Annotation', 'GenModel Doc', and 'Advanced'. The 'Parameters' table is highlighted with a blue border and contains two entries:

Parameters	
first : Node	
second : Node	

To the right of the table are 'Add' and 'Remove' buttons. Below the table is the 'Parameter Details' section, which includes two input fields:

Name :

Type :

# Fichier xmi : Éditeur graphique de modèle

The screenshot displays the Eclipse IDE interface. The top part shows the 'Graph.xmi' editor with a tree view of the model structure:

- platform:/resource/Graph/model/Graph.xmi
  - Graph Example
    - Node 1
    - Node 2
    - Node 3
    - Node 4
    - Node 5
    - Edge 1 -> 2
    - Edge 2 -> 3
    - Edge 3 -> 1
    - Edge 2 -> 5
    - Edge 5 -> 5
    - Edge 4 -> 5
- platform:/resource/Graph/model/Graph.ecore

The bottom part of the IDE shows the 'Property' view for the selected 'Node 5'. It contains the following table:

Property	Value
Content	5
Entering	Edge 2 -> 5, Edge 5 -> 5, Edge 4 -> 5
Leaving	Edge 5 -> 5

### Attributs :

- Vérifier les types utilisés (EMachin → ??)
- Multiplicités : Dans ECore, \* se lit -1 !

### Références :

- Faire attention à la contenance !
  - Au risque de se retrouver avec des DanglingException !
- Pour définir une relation opposée :
  - Définir la référence et sa réciproque
  - Spécifier pour l'une qu'elle est l'EOpposite de l'autre.

### Création de modèles xmi conforme à un métamodèle ecore :

- Create Dynamic Instance sur le concept dans le .ecore
- Load Resource pour charger un modèle dans un autre
  - EMF est un adepte du *chargement paresseux*.

### Prise de position sur les EOperations :

- Dans une approche **implémentation**
  - On déclare les opérations dans les méta-classes
  - On implémente leur comportement dans le langage cible
- Dans une approche **modélisation**
  - Les métamodèles sont des structures supports au raisonnement
  - On ne se soucie pas de leur exécution au *design*
    - Mais on s'en occupera plus tard, ...
    - Et peut-être avec d'autres outils (e.g. Kermeta)

### Quoi qu'il en soit ...

- La multiplicité d'une EOperation concerne son type de retour
- Les paramètres se définissent dans un *sous-onglet*.

# Agenda

- 1 Modèles, Métamodèles ...
- 2 Exemple : *Graph – Management*
- 3 Eclipse Modeling Framework
- 4 Outil de Modélisation & Spécificités EMF
- 5 EMF : Un canevas logiciel**
- 6 Graphical Modeling Framework
- 7 Prenons un peu de recul ...
- 8 Conclusions

# Intégration d'ECORE dans EMF : Manipulation & Génération

## EMF est un (E)canevas logiciel ...

- Dirigé par les **modèles**, mais implémenté dans un **langage**
- ⇒ Processus de réification *IDM* → *Java* :
- Les métamodèles sont représentées par des *EClass*
  - Les modèles sont représentés par des *EObject*

## Génération du code des métamodèles via un ... modèle !

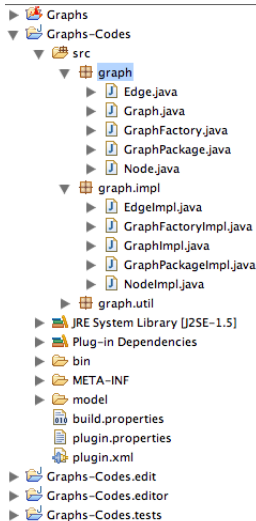
- Les outils de génération ne travaillent pas sur le `.ecore`
- *Eclipse* définit un fichier `.genmodel` en parallèle :
  - New/ Other/ Eclipse Modeling Framework/ EMF Model
  - On peut *customiser* le générateur de code !
  - L'IDE se charge du maintien de la cohérence (**ou pas**)

# Générateurs de code

## Action disponibles sur le métamodèle :

- 1 Generate Model Code :
  - Classes Java reposant sur EMF
- 2 Generate Edit Code :
  - Plugin supportant l'édition
- 3 Generate Editor Code :
  - Plugin d'édition arborescente
- 4 Generate Test Code :
  - Plugin de test unitaire

Actions disponibles à partir du `.genmodel`,  
et dans un EMF Project.



# Génération de code : Opening the box

[A. Gaignard]

## JET : “Java Emitter Template”

- Configurable (*cf.* vue *Properties* du `.genmodel`)
- Approche “template” : définition de patrons de classe
  - Vers *Java* : Support de l’héritage multiple (injection)
  - Vers *X* : implantation des templates ad’hoc.

## Remarques sur les codes générés automatiquement

- Fonctionnalités EMF intégrée dans les codes :
  - Persistance, Notification, Réfexivité, Contraintes
- Re-génération  $\Rightarrow$  Fusion avec le code existant
- Éviter l’écrasement : `@generated NOT` (*Javadoc*)
- *Tips&Tricks* : **Bijection ECore  $\Leftrightarrow$  Code augmenté fragile**

# Chargement d'un modèle XMI dans un EObject Java

```
public Graph load(File f) {
    ResourceSet rs = new ResourceSetImpl();
    Resource.Factory.Registry registry =
        rs.getResourceFactoryRegistry();
    Map<String, Object> m = registry.getExtensionToFactoryMap();
    m.put("xmi", new XMIResourceFactoryImpl());
    rs.getPackageRegistry().put(GraphPackage.eNS_URI,
                                GraphPackage.eINSTANCE);
    URI uri = URI.createFileURI(f.getAbsolutePath());
    Resource resource = rs.getResource(uri, true);
    if ( resource.isLoaded() &&
        resource.getContents().size() > 0) {
        return (Graph) resource.getContents().get(0);
    }
    return null;
}
```

# Stockage d'un EObject Java dans un modèle XMI

```
public void save(Graph g, File f) {
    ResourceSet rs = new ResourceSetImpl();
    Resource.Factory.Registry registry =
        rs.getResourceFactoryRegistry();
    Map<String, Object> m = registry.getExtensionToFactoryMap();
    m.put("xmi", new XMIResourceFactoryImpl());
    m.put("ecore", new EcoreResourceFactoryImpl());
    rs.getPackageRegistry().put(GraphPackage.eNS_URI,
                               GraphPackage.eINSTANCE);

    Resource packageResource =
        rs.createResource(URI.createFileURI("model/Graph.ecore"));
    packageResource.getContents().add(GraphPackage.eINSTANCE);
    try { packageResource.load(null); }
    catch (IOException e1) { e1.printStackTrace(); }
    URI uri = URI.createFileURI(f.getAbsolutePath());
    Resource resource = rs.createResource(uri);
    resource.getContents().add(g);
    try {
        HashMap<String, Boolean> options =
            new HashMap<String, Boolean>();
        options.put(XMIResource.OPTION_SCHEMA_LOCATION,
                   Boolean.TRUE);
        resource.save(options);
    } catch (IOException e) { e.printStackTrace(); }
}
```

# Génération de modèle : une approche *factory*

```
public Graph generateCompleteGraph(String name, int n) {
    Graph g = GraphFactory.eINSTANCE.createGraph();
    g.setName(name);
    for(int i = 0; i < n; i++) {
        Node node = GraphFactory.eINSTANCE.createNode();
        node.setContent("node_" + i); g.getNodes().add(node);
    }
    for(Node source: g.getNodes()) {
        for(Node target: g.getNodes()) {
            Edge e = GraphFactory.eINSTANCE.createEdge();
            e.setLabel(source.getContent() + " → "
                + target.getContent());
            e.setSource(source); e.setTarget(target);
            g.getEdges().add(e);
        }
    }
    return g;
}
```

# Implémentation des EOperation :

## Localisation des méthodes dans le code généré

- 1 Dans le sous-package `graph.impl`
- 2 Dans la classe `GraphImpl`
- 3 Noyée dans le code automatique généré par EMF ...

```
/**  
 * @generated NOT  
 */  
public int order() {  
    return this.getEdges().size();  
}
```

Ne pas oublier de *marquer* pour empêcher l'écrasement !

# Un exemple d'avantage du canevas : Observer/Observable

## ■ Dans le code Java :

```
Graph g2 = GraphFactory.eINSTANCE.createGraph();
g2.eAdapters().add(new EContentAdapter() {
    public void notifyChanged(Notification notification) {
        super.notifyChanged(notification);
        System.out.println(notification);
    }
});
g2.setName("New Name!");
```

## ■ Dans le terminal :

```
org.eclipse.emf.ecore.impl.ENotificationImpl@2297d7 (
  eventType: SET, position: -1,
  notifier: graph.impl.GraphImpl@1ecfeb (
    name: New Name !),
  feature: ...
```

# Un canevas adaptable : spécialisation des entités

```
// The Usual XMI persistence factory
m.put("xmi", new XMIResourceFactoryImpl());

// Can be Specialized to encrypt models
m.put("xmi", new XMIResourceFactoryImpl() {
    public Resource createResource(URI uri) {
        XMIResourceFactoryImpl resFactory =
            new XMIResourceFactoryImpl();
        XMIResource resource =
            (XMIResource) resFactory.createResource(uri);
        try {
            resource.getDefaultLoadOptions().put(
                Resource.OPTION_CIPHER, new AESCipherImpl("foo"));
            resource.getDefaultSaveOptions().put(
                Resource.OPTION_CIPHER, new AESCipherImpl("foo"));
        } catch (Exception e) { e.printStackTrace(); }
        return resource;
    }
});
```

# Agenda

- 1 Modèles, Métamodèles ...
- 2 Exemple : *Graph – Management*
- 3 Eclipse Modeling Framework
- 4 Outil de Modélisation & Spécificités EMF
- 5 EMF : Un canevas logiciel
- 6 Graphical Modeling Framework**
- 7 Prenons un peu de recul ...
- 8 Conclusions

# Générateur d'éditeur graphique dirigé par les modèles

Définir un éditeur graphique revient à définir :

- 1 Un modèle de ce que l'on veut représenter

GMF : *"Graphical Modeling Framework"*

- Métier : `Graph.ecore`

# Générateur d'éditeur graphique dirigé par les modèles

Définir un éditeur graphique revient à définir :

- 1 Un **modèle** de ce que l'on veut représenter
- 2 Un **modèle** de représentation à l'écran

GMF : *"Graphical Modeling Framework"*

- Métier : `Graph.ecore`, `Graph.gmfgraph`

# Générateur d'éditeur graphique dirigé par les modèles

Définir un éditeur graphique revient à définir :

- 1 Un **modèle** de ce que l'on veut représenter
- 2 Un **modèle** de représentation à l'écran
- 3 Un **modèle** d'action accessibles à l'utilisateur

GMF : *"Graphical Modeling Framework"*

- Métier : `Graph.ecore`, `Graph.gmfgraph`, `Graph.gmftool`

# Générateur d'éditeur graphique dirigé par les modèles

Définir un éditeur graphique revient à définir :

- 1 Un **modèle** de ce que l'on veut représenter
- 2 Un **modèle** de représentation à l'écran
- 3 Un **modèle** d'action accessibles à l'utilisateur
- 4 Un **modèle** de liaison, et un modèle de création !

GMF : *"Graphical Modeling Framework"*

- Métier : `Graph.ecore`, `Graph.gmfgraph`, `Graph.gmftool`
- Technique : `Graph.gmfmap` & `Graph.gmfgen`

# Générateur d'éditeur graphique dirigé par les modèles

Définir un éditeur graphique revient à définir :

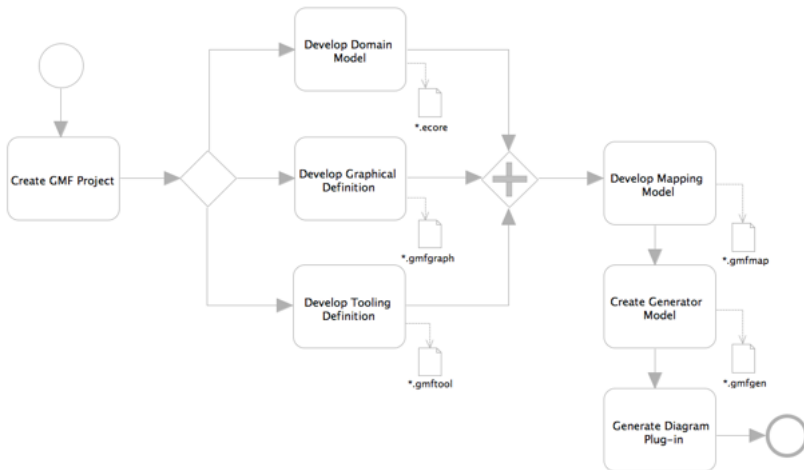
- 1 Un **modèle** de ce que l'on veut représenter
- 2 Un **modèle** de représentation à l'écran
- 3 Un **modèle** d'action accessibles à l'utilisateur
- 4 Un **modèle** de liaison, et un modèle de création !

GMF : *"Graphical Modeling Framework"*

- Métier : `Graph.core`, `Graph.gmfgraph`, `Graph.gmftool`
- Technique : `Graph.gmfmap` & `Graph.gmfgen`

⇒ Génération d'un plugin Eclipse

# Workflow GMF de A à Z



# Agenda

- 1 Modèles, Métamodèles ...
- 2 Exemple : *Graph – Management*
- 3 Eclipse Modeling Framework
- 4 Outil de Modélisation & Spécificités EMF
- 5 EMF : Un canevas logiciel
- 6 Graphical Modeling Framework
- 7 Prenons un peu de recul ...
- 8 Conclusions

# Questions ouverte : Multiplicité technologique ?



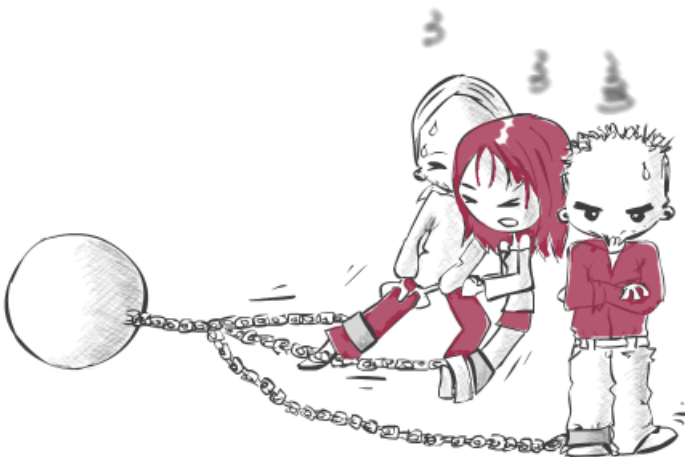
# Questions ouverte : Interopérabilité des outils ?



# Questions ouverte : Pragmatisme de la démarche ?



# Questions ouverte : Coût d'un développement IDM ?



# Agenda

- 1 Modèles, Métamodèles ...
- 2 Exemple : *Graph – Management*
- 3 Eclipse Modeling Framework
- 4 Outil de Modélisation & Spécificités EMF
- 5 EMF : Un canevas logiciel
- 6 Graphical Modeling Framework
- 7 Prenons un peu de recul ...
- 8 Conclusions

# Conclusions

## L'approche métamodèle ...

- ... est finalement naturelle
- ... permet de se focaliser sur son métier

## EMF est un canevas logiciel

- Basé sur eMOF (standard ISO), Open Source, intégré à Eclipse
- De nombreux outils l'utilisent, pas si complexe qu'il en à l'air !
- Fournit de nombreuses fonctionnalités "out of the box".

## ... mais ...

- Un manque de maturité peut rester visible
- L'approche "tout modèle" laisse parfois perplexe

# Références

- Le Web officiel d'EMF :
  - <http://www.eclipse.org/modeling/emf>
  - <http://wiki.eclipse.org/EMF>
  - <http://www.eclipse.org/modeling/emf/docs>
- Livres (très) intéressants
  - *Eclipse Modeling Framework*, Budinsky *et al*, Addison Wesley
  - *MDA en action*, Xavier Blanc, Eyrolles
- Tutoriels intéressants disponibles sur le Web :
  - <http://chie.uniandes.edu.co/~isis4712/wiki/>
  - <http://www.eclipse.org/articles>
  - [http://wiki.eclipse.org/index.php/GMF\\_Tutorial](http://wiki.eclipse.org/index.php/GMF_Tutorial)
  - <http://www.kermeta.org/documents/emfTutorial/>

# Remerciements

- Pour leurs compétences sur EMF :
  - Alban Gaignard (Modalis, Sophia)
  - Franck Chauvel (HCST, Pékin)
- Pour leurs différents points de vues sur l'IDM
  - Mireille Blay–Fornarino (Modalis, Sophia)
  - Xavier Blanc (MoVe, Paris 6)
  - Pierre–Alain Muller (Triskell, Mulhouse)
  - Robert France (CSU, Fort Collins)
  - Geri Georg (CSU, Fort Collins)
- et tous les membres de la communautés MODELS ... :)

*“Aucun modèle n’a été maltraité durant la préparation de cet exposé.”*

