

Master mention Informatique

Génie Logiciel et modélisation
M1 / 177UD01

1 Introduction à MDA

claudine.piau-toffolon@univ-lemans.fr

MDA : Model Driven Architecture

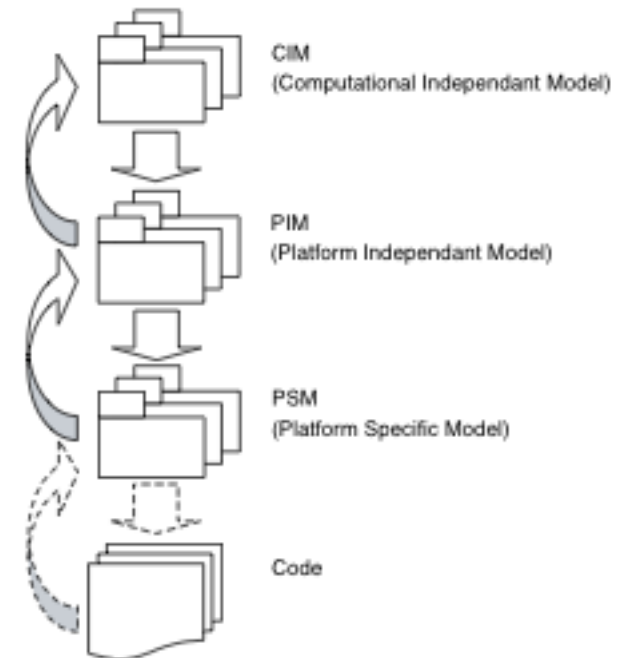
- Faire des modèles c'est bien mais on peut faire beaucoup mieux :
 - Utiliser des modèles de manière productive (\neq de manière représentative)
 - Pour générer automatiquement d'autres modèles, du code source ou de la documentation
- L'approche *MDA* (*Model Driven Architecture*) proposé par l'OMG :
 - Approche basée sur l'IDM (Ingénierie Dirigée par les Modèles)
 - IDM : placer les modèles au centre du processus de développement
- Idées principales :
 - Utilisation de modèles aux différentes phases du cycle de développement
 - Différencier chaque phase mais garder un lien de traçabilité entre modèles des différentes phases
 - Un modèle d'exigences (CIM)
 - Un modèle d'analyse et de conception (PIM) par transformation du CIM
 - Un modèle de code lié à une plateforme (PSM) par transformation du PIM
- Avantages attendus :
 - Elaboration de modèles pérennes, indépendants des plates-formes d'exécution
 - Permettre la génération automatique de la totalité du code des applications vers différentes plateformes (but à (très) long terme...)
 - Obtenir un gain significatif de productivité
- MDA préconise l'usage d'UML et de MOF (mais ce n'est pas obligatoire)

MDA : CIM, PIM et PSM

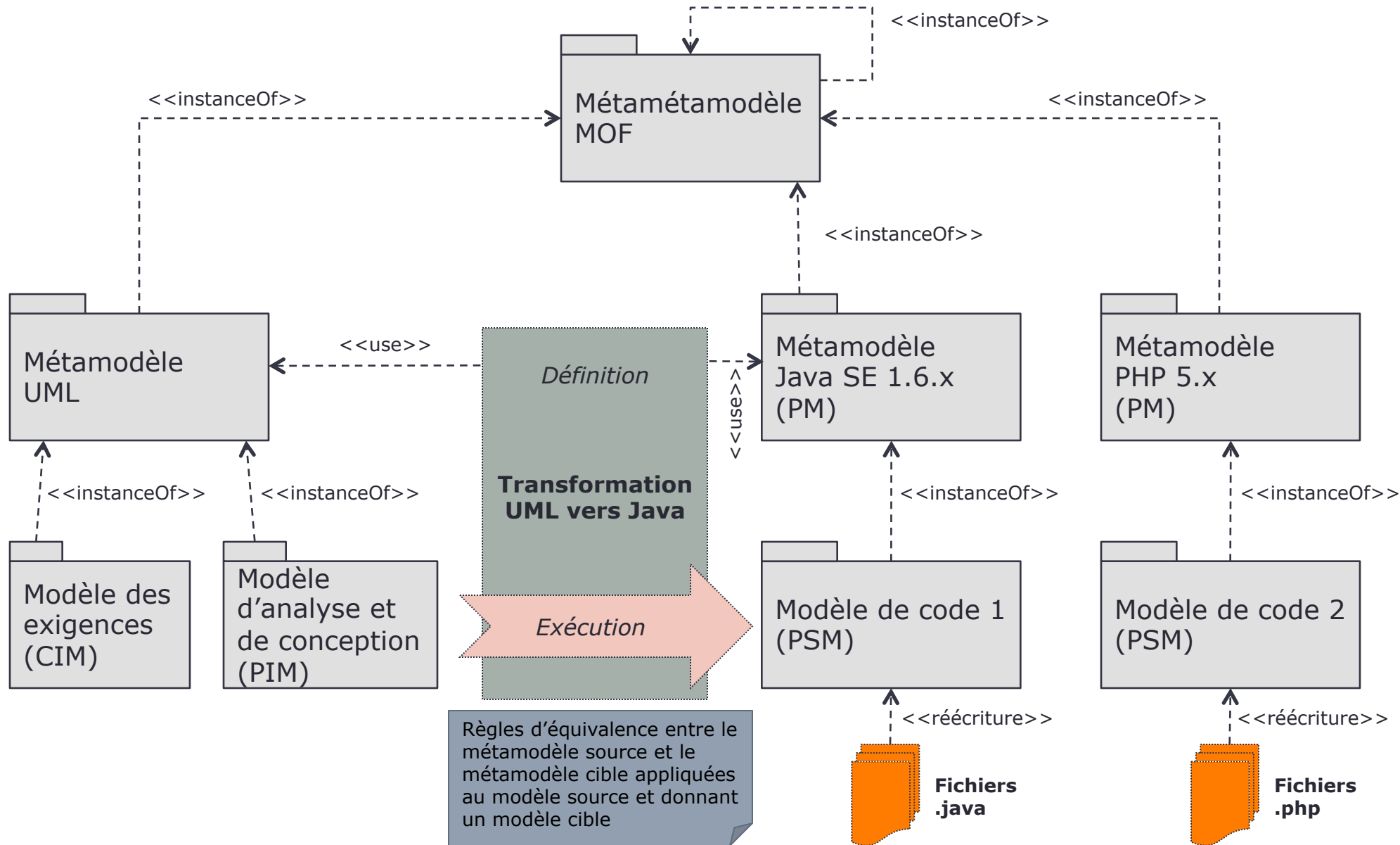
- CIM (*Computation Independent Model*) :
 - Modèle des exigences indépendant de toute technologie numérique
 - Représente l'application dans son environnement :
 - Services offerts par l'application
 - Les entités avec lesquelles elle interagit
 - Peut contenir des diagrammes de cas d'utilisation, des diagrammes de séquences, des diagrammes d'activités (processus métiers)
 - Modèle pérenne : les besoins métiers évoluent moins vite que les technologies
- PIM (*Platform Independant Model*) :
 - Modèle d'analyse (concepts du domaine) et de conception (architecture, design pattern)
 - Doit faire référence au CIM (lien de traçabilité)
 - Modèle abstrait et pérenne indépendant de toute technologie
 - Modèle productif : socle pour la génération du PSM
- PSM (*Platform Specific Model*) :
 - Modèle lié à une plateforme d'exécution (Java, .NET, PHP, etc.)
 - Obtenu par transformation à partir d'un PIM et d'un PM (*Platform Model*)
 - Permet de générer du code source
- Le passage d'un modèle à l'autre se fait par transformation
- La génération de code source est une simple réécriture textuelle du PSM

MDA : les transformations (1/2)

- Les transformations sont le cœur de MDA : elles assurent l'automatisation et les liens de traçabilité
- Il y a deux sens de transformation :
 - Les transformations descendantes
 - Les transformations montantes (ou par rétro ingénierie)
- Une transformation est un ensemble de règles entre les éléments du métamodèle source vers les éléments du métamodèle cible. Par exemple :
 - Source : modèle PIM sous forme de diagramme de classe UML
 - Métamodèle source : UML
 - Cible : modèle PSM sous forme de classes Java
 - Métamodèle cible : la plateforme d'exécution Java
 - Règles :
 - Transformer une classe UML en classe Java
 - Transformer les associations UML en attributs de classe Java
 - Transformer l'héritage multiple UML en héritage simple Java
 - Etc.



MDA : les transformations (2/2)

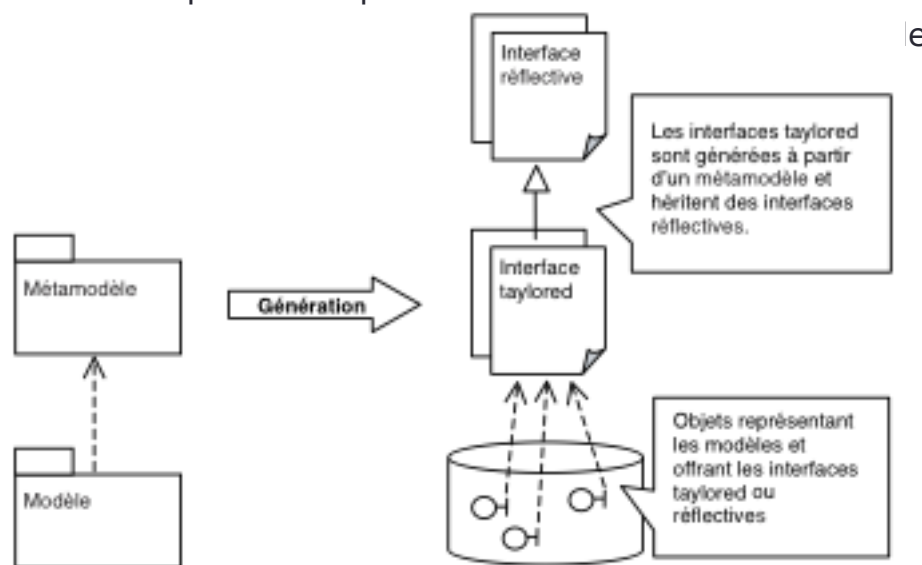


MDA : les standards et outils (1/2)

- Une transformation est un programme qui manipule des modèles :
 - Elle a besoin de charger les modèles et les métamodèles (cf. standards MDA)
 - Une fois les « données » chargées, il existe 3 approches possibles :
 - Approche par programmation seule : programmer la transformation comme une application « normale » où les données manipulées sont les modèles
 - Approche par template (éléments cibles paramétrés) : remplacer les paramètres de la cible par des données du modèle source ; basée sur un langage de template ± riche, flexible...
 - Approche par modélisation : modéliser la transformation elle-même dans un métalangage de transformation telle que MOF 2.0 QVT (*Query View Transformation*) ; au stade expérimental
- Les standards *MDA* :
 - XMI (*XML Metadata Interchange*) : format XML pour l'enregistrement des modèles
 - JMI (*Java Metadata Interface*) : spécification Java pour manipuler des modèles MOF
 - EMF (*Eclipse Modeling Framework*) : identique à JMI + implémentation ; basé sur ECore
- Quelques outils :
 - EMF soit comme plugin Eclipse soit en dehors d'Eclipse
 - MDR (*MetaData Repository*) basé sur JMI ; notamment utilisé par NetBeans
 - Acceleo (basé sur EMF) : outil pour définir ses propres templates de générations
 - Editeurs UML avec génération de code intégrée (limitée aux langages de l'éditeur)
 - Astah, Modelio, ArgoUML, etc. en versions professionnelles ou *community*
 - Certains sont basés sur JMI, sur EMF ou sur MDR ; certains export/import en XMI
 - Des outils expérimentaux issus de la recherche !

MDA : les standards et outils (2/2)

- Avec JMI ou EMF, la manipulation des modèles se fait :
 - Soit par interfaces réflexives :
 - Classes Java indépendantes de tout modèle ou métamodèle mais liées au framework utilisé
 - Elles sont basées sur l'idée simple suivante : tout modèle est un ensemble d'éléments (instances de métaclasse) reliés entre eux avec un élément racine conteneur de tous ces éléments
 - Elles permettent d'accéder aux metapackages et, à partir de là, de tous ses éléments constitutifs et ainsi de suite
 - Soit par interfaces *taylored* (taillées sur mesure) :
 - Classes Java générées automatiquement à partir du métamodèle
 - Elles permettent de r
 - Elles héritent des int



Exercice 1

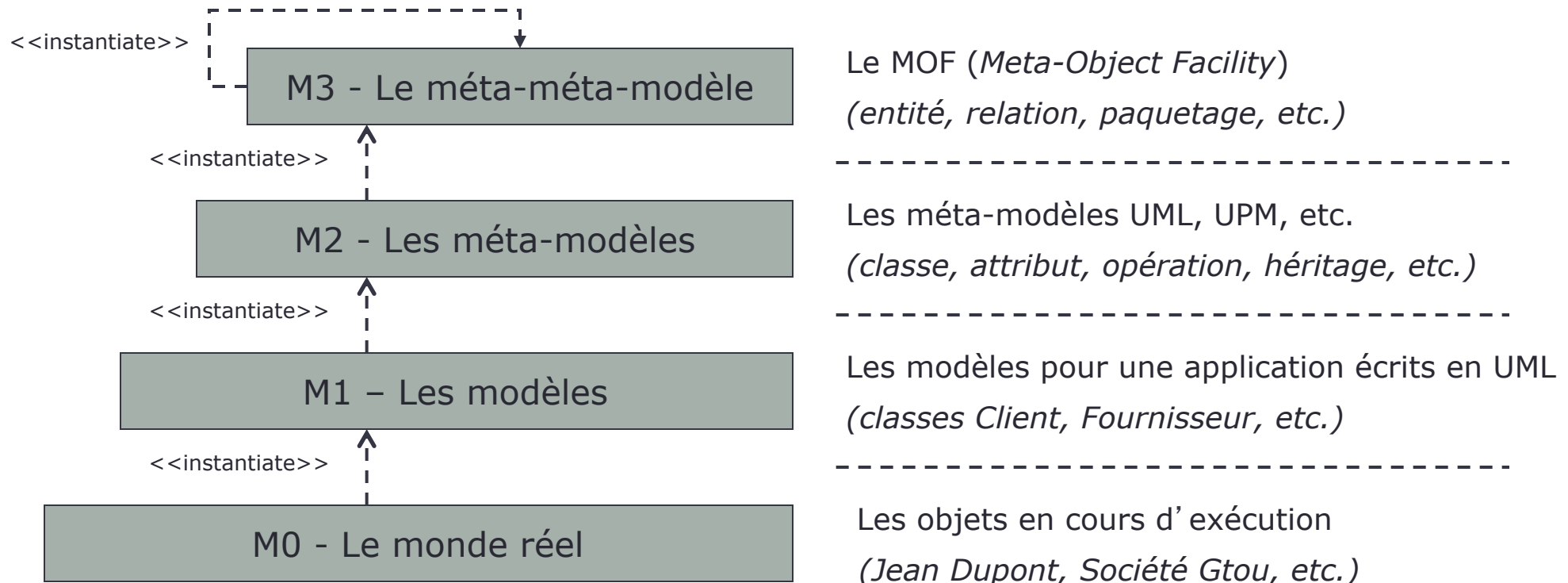
Proposer une séquence d'activité (processus) permettant de créer un schéma E/ A pour une application de prêt de livres dans une bibliothèque

Quel est le méta-processus associé ?

Faire le méta-modèle du modèle E/ A

UML et MOF

- Le langage UML sert à écrire des modèles qui décrivent des applications informatiques : *c'est un méta-modèle*.
- UML est lui-même une instance du *méta-méta-modèle* MOF.
- MOF est réflexif (il se décrit lui-même).
- Organisation des standards de l'OMG : architecture à 4 niveaux

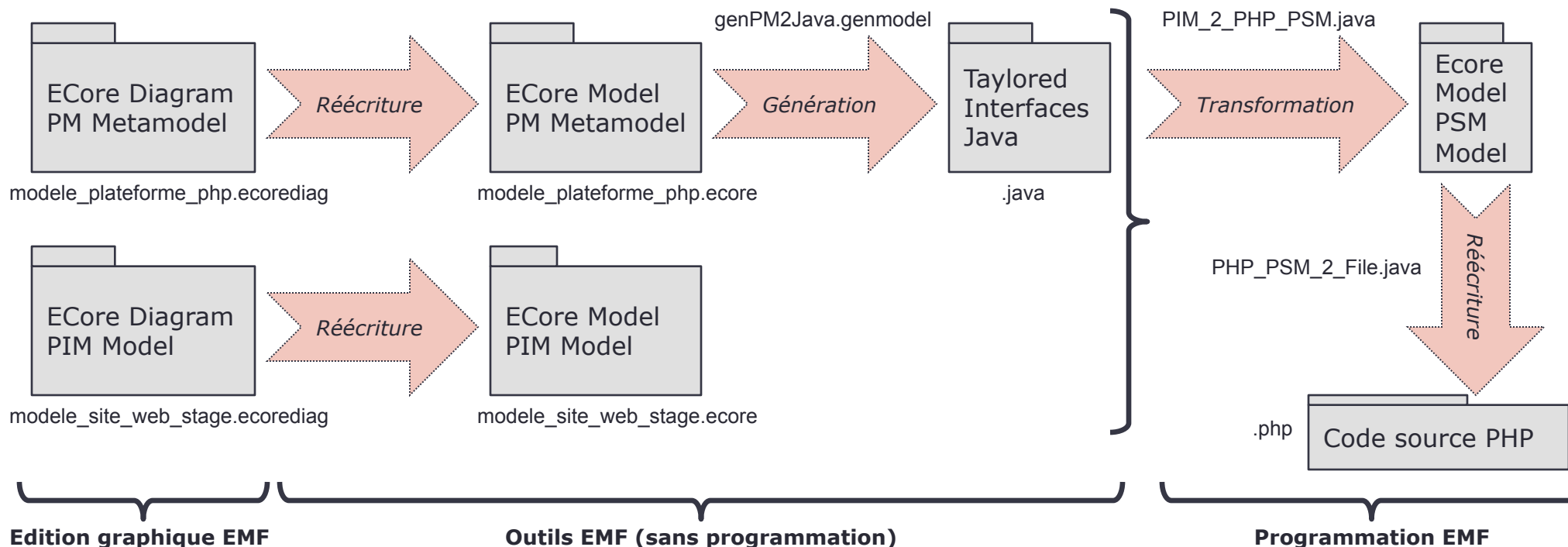


Exercice 2

- Meta-Modèle MEA
- Meta-Modèle UML?

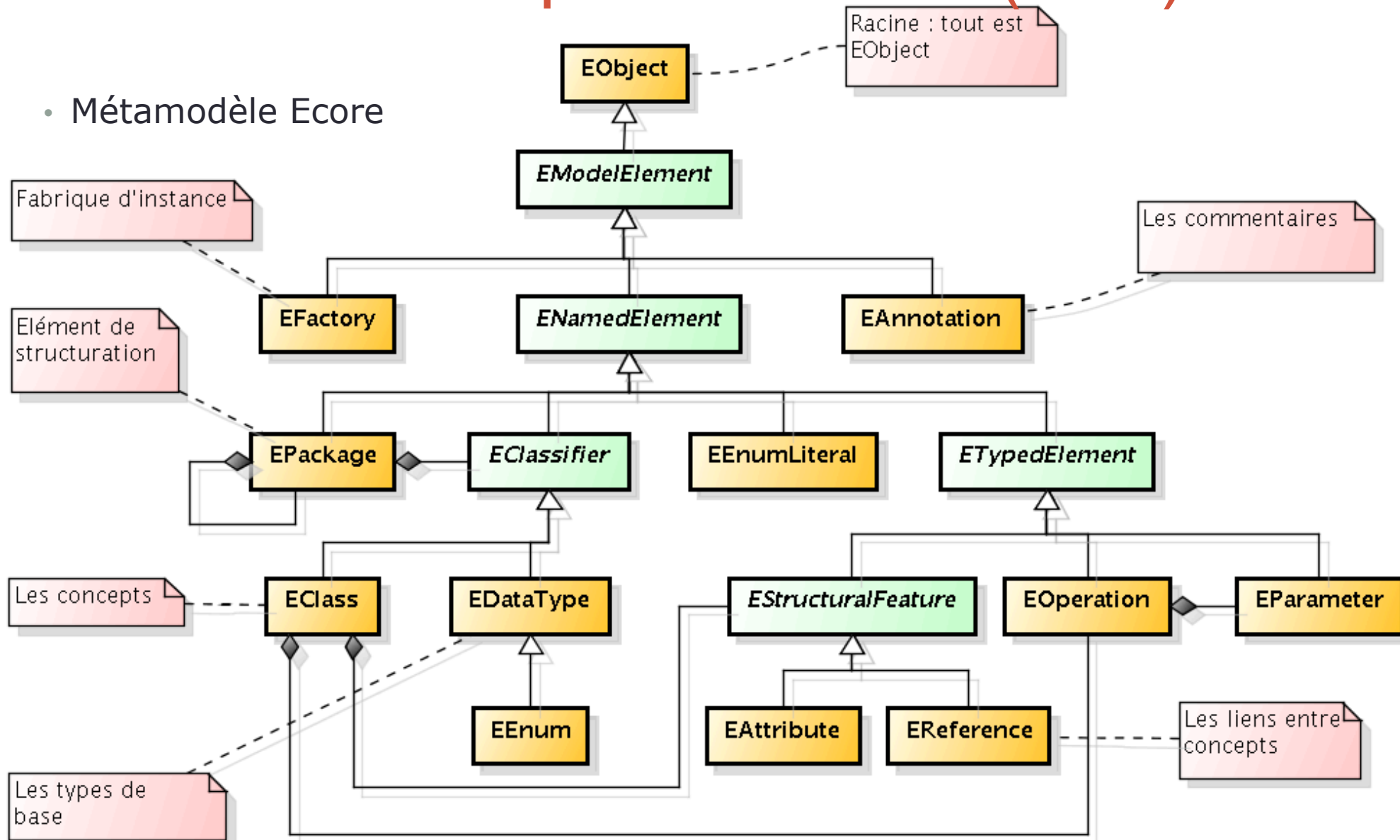
MDA : un exemple avec EMF (1/12)

- Nous allons programmer une transformation PIM vers PSM :
 - Modèle source : modèle PIM (partiel) de l'application Web de gestion des stages
 - Métamodèle source : ECore
 - Modèle cible : modèle PSM des classes entités (données stockées en base)
 - Métamodèle cible : PM PHP 5.x (partiel)
- EMF :
 - *Framework* complet Java (définition + instanciation)
 - Basé sur le métamodèle Ecore (très proche de EMOF 2.0)
 - Possibilité de transformer des modèles MOF vers des modèles ECore



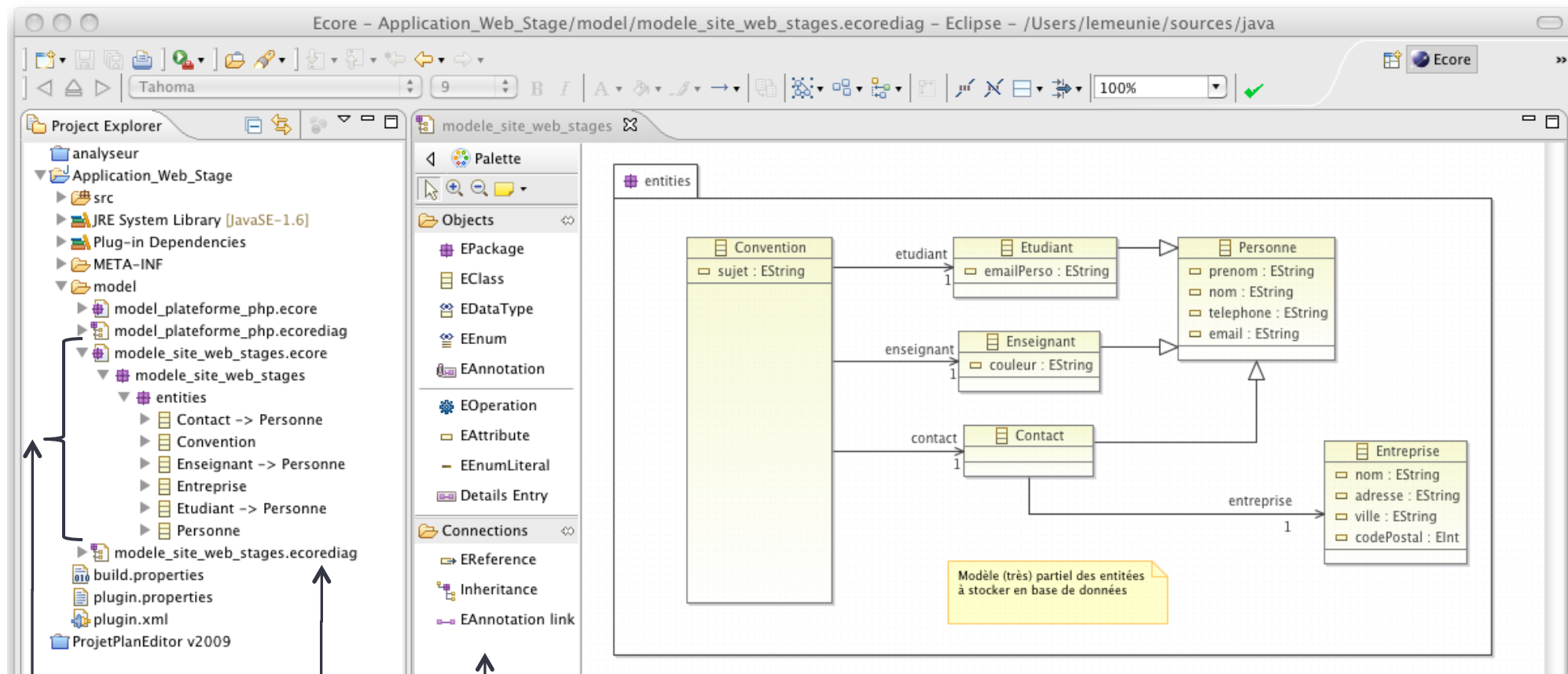
MDA : un exemple avec EMF (2/12)

- Métamodèle Ecore



MDA : un exemple avec EMF (3/12)

- Modèle PIM :
 - Edition graphique dans Eclipse (EMF Diagram Editor)
 - Sauvegarde automatique du fichier modèle .ecore



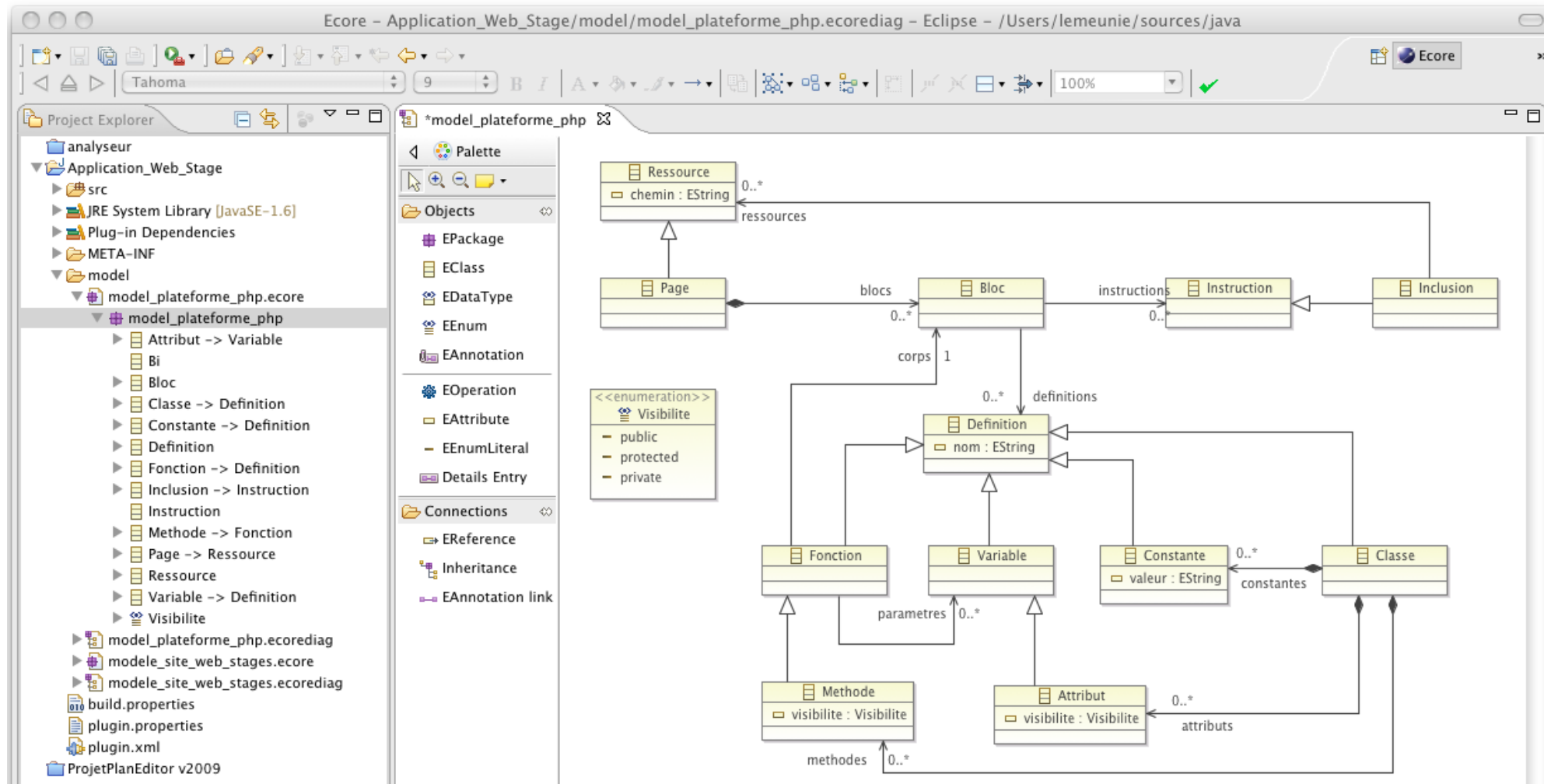
Le fichier du
modèle PIM

Le fichier du
diagramme

Éléments ECore

Zone édition graphique

MDA : un exemple avec EMF (4/12)



MDA : un exemple avec EMF (5/12)

- Modèle PM :
 - Génération des interfaces tailored Java
 - Création d'un générateur en indiquant le modèle *model_plateforme_php.ecore*

4
Classes interfaces + implémentations générées

1
Le générateur

2
Visualisation du générateur et sélection

3
Génération

Project Explorer: analyseur, Application_Web_Stage, src, generation, model_plateforme_php, impl, util, Attribut.java, Bloc.java, Classe.java, Constante.java, Definition.java, Fonction.java, Inclusion.java, Instruction.java, Methode.java, Model_plateforme_phpFactory.java, Model_plateforme_phpPackage.java, Page.java, Ressource.java, Variable.java, Visibilite.java, php, transformation, genPM2Java.genmodel, model_plateforme_php.ecore, modele_site_web_stages.ecore, JRE System Library [JavaSE-1.6], Plug-in Dependencies, META-INF, model, build.properties, plugin.properties.

Package Explorer: GenPM2Java, Model_plateforme_php, Page -> Ressource, Bloc, Instruction, Definition, Fonction -> Definition, Variable -> Definition, Constante -> Definition, Classe -> Definition, Methode -> Fonction, Attribut -> Variable, Ressource, Inclusion -> Instruction, Visibilite.

Context Menu: Generate Model Code, Generate Edit Code, Generate Editor Code, Generate Test Code, Generate All, Open Ecore, Open GenModel, Undo, Redo, Cut, Copy, Paste, Delete, Reload..., Export Model..., Run As, Debug As, Profile As, Validate, Team, Compare With, Replace With, Refresh, Show Properties View, Remove from Context.

MDA : un exemple avec EMF (6/12)

- Exemple de transformation : les EPackage en liste de Ressource (Page)

```
/**
 * Transformation d'un PIM en PSM PHP
 * @author T. Lemeunier
 * @version 0.1
 */
public class PIM_2_PHP_PSM {

    private List<EObject> lRessource = new ArrayList<EObject>(); // Stocke les ressources PHP

    /**
     * Transforme un EPackage source en un ensemble de Ressource : chaque EClass source
     * du EPackage source devient une Page et chaque EPackage source est transformé
     * par appel récursif
     * @param epackage Le package source à transformer
     */
    public void packageToPages(EPackage epackage) {
        // Parcours des sous-packages
        for (EPackage ep : epackage.getESubpackages()) {
            this.packageToPages(ep);
        }
        // Parcours des classifieurs
        for (EClassifier ec : epackage.getEClassifiers()) {
            if (ec instanceof EClass) {
                EClass classe = (EClass) ec;
                if (!classe.isAbstract()) lRessource.add(classToPage(classe));
            }
        }
    }
}
```


MDA : un exemple avec EMF (7/12)

- Exemple de transformation : les EClass en Page

```
/**
 * Transforme une EClass source en une Page PHP
 * @param ec La classe source
 * @return La page PHP
 */
private Page classToPage(EClass ec) {
    Page page = Model_plateforme_phpFactoryImpl.eINSTANCE.createPage() ;
    page.setChemin("src/php/" + ec.getName() + ".php");

    // La page est constituée d'un seul bloc d'instruction : la classe PHP Objet
    Bloc bloc = Model_plateforme_phpFactoryImpl.eINSTANCE.createBloc();
    page.getBlocs().add(bloc);

    // La classe PHP unique élément du bloc
    Classe classe = Model_plateforme_phpFactoryImpl.eINSTANCE.createClasse();
    classe.setNom(ec.getName());
    bloc.getDefinitions().add(classe);

    // Transformations des attributs de la classe source en attributs de classe PHP
    attributeToAttribut(ec, classe);

    // Transformations des références de la classe source en attributs de classe PHP
    referenceToAttribut(ec, classe);

    // Créer les accesseurs
    creerAccesseur(classe);

    return page;
}
```

MDA : un exemple avec EMF (8/12)

- Exemple de transformation : chargement du PIM et transformation

```

/**
 * Transforme un modèle PIM en modèle PSM
 * @param cheminModeleEcore Le nom du fichier .ecore du modèle PIM
 */
public void transforme(String cheminModeleEcore) {
    // Charger le modèle PIM
    Resource.Factory.Registry reg = Resource.Factory.Registry.INSTANCE;
    Map<String, Object> m = reg.getExtensionToFactoryMap();
    m.put("ecore", new XMIResourceFactoryImpl());

    ResourceSet resourceSet = new ResourceSetImpl();
    URI fileURI = URI.createFileURI(cheminModeleEcore);
    Resource resource = resourceSet.createResource(fileURI);
    try {
        resource.load(null);
    } catch (IOException e) {
        System.err.println("Impossible de charger le modele : " + cheminModeleEcore);
        e.printStackTrace();
    }

    // Transformer le modèle PIM en modèle PSM
    this.packageToPages((EPackage)resource.getContents().get(0));
}

public static void main(String[] args) {
    PIM_2_PHP_PSM trans = new PIM_2_PHP_PSM();
    trans.transforme("model/modele_site_web_stages.ecore");
}

```

MDA : un exemple avec EMF (9/12)

- Exemple de génération du code source : génération des fichiers PHP

```
/**
 * Génération de code source PHP à partir d'un modèle PSM PHP
 * @author T. Lemeunier
 * @version 0.1
 */
public class PHP_PSM_2_File {

    public void generation(List<Ressource> ressources) {
        for (Ressource res : ressources) { // Parcours des ressources
            if (res instanceof Page) {
                FileWriter file = null;
                try {
                    file = new FileWriter(res.getChemin()); // Création du fichier
                    file.write("<?php\n");
                } catch (IOException e) {...}

                // Parcours des blocs des pages puis des définitions de chaque bloc
                for (Bloc bloc : ((Page) res).getBlocs())
                    for (Definition def : bloc.getDefinitions())
                        if (def instanceof Classe) generationClasse((Classe)def, file);

                try {
                    file.write("\n?>\n"); // Fin du fichier
                    file.close();
                } catch (IOException e) { ... }
            }
        }
    }
}
```

MDA : un exemple avec EMF (10/12)

- Exemple de génération du code source : génération des classes PHP

```
/**
 * Génération du code source d'une classe PHP
 * @param classe La classe dont le code source sera généré
 * @param file Le fichier où sera généré la classe
 */
public void generationClasse(Classe classe, FileWriter file) {
    try {
        // Début de la classe
        file.write("\nclass " + classe.getNom() + " {\n");

        // Génération des attributs
        generationAttributs(classe, file);

        // Génération des méthodes
        generationMethodes(classe, file);

        // Fin de la classe
        file.write("\n}\n");
    } catch (IOException e) {...}
}
```

MDA : un exemple avec EMF (11/12)

- Exemple de génération du code source : génération des attributs PHP

```
/**
 * Génération des attributs d'une classe PHP. On génère un attribut idNomDeLaClasse.
 * @param classe La classe à générer
 * @param file Le fichier où les attributs sont générés
 */
public void generationAttributs(Classe classe, FileWriter file) {
    try {
        // Ajout d'un attribut id
        file.write("\tprivate $id" + classe.getNom() + ";\n");

        // Parcours des attributs
        for (Attribut attr : classe.getAttributs()) {
            file.write("\tprivate $" + attr.getNom() + ";\n");
        }
    } catch (IOException e) {...}
}
```

```
public static void main(String args[]) {
    // Chargement et transformation du modèle PIM
    PIM_2_PHP_PSM trans = new PIM_2_PHP_PSM();
    trans.transforme("model/modele_site_web_stages.ecore");

    // Génération du code source
    PHP_PSM_2_File gene = new PHP_PSM_2_File();
    gene.generation(trans.getRessources());
}
```

MDA : un exemple avec EMF (12/12)

- Exemple de génération du code source : exemple de fichier généré

```
<?php

class Contact {
    private $idContact;
    private $nom;
    private $prenom;
    private $telephone;
    private $email;
    private $entreprise;

    public function Contact($nom,$prenom,$telephone,$email,$entreprise) {
        $this->nom = $nom;
        $this->prenom = $prenom;
        $this->telephone = $telephone;
        $this->email = $email;
        $this->entreprise = $entreprise;
    }

    public function getIdContact() { return $this->idContact; }

    public function getNom() { return $nom; }

    public function setNom($nom) { $this->nom = $nom; }

    ...
}

?>
```