

# Ingénierie des Modèles

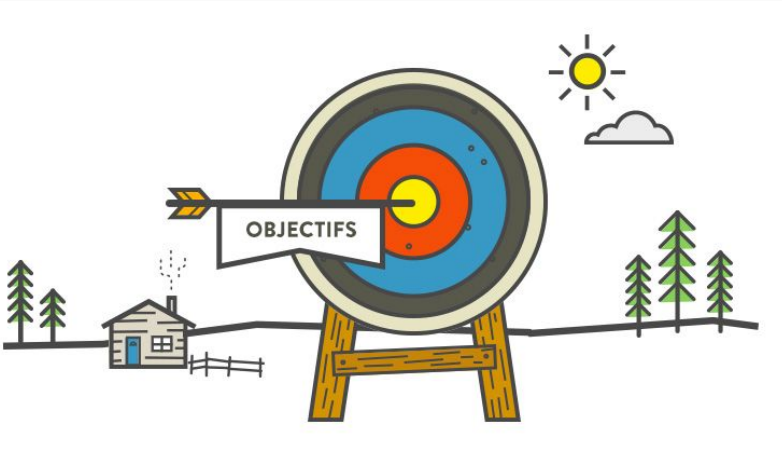
## Introduction générale

**Léa Brunschwig**  
✉ [lea.brunschwig@univ-pau.fr](mailto:lea.brunschwig@univ-pau.fr)

**M2 Technologies de l'Internet**

**Université de Pau et des Pays de l'Adour**  
Collège STEE  
Département Informatique

# Objectifs du cours



- Concevoir des **langages dédiés** par **méta-modélisation**,
- Définir une **syntaxe concrète** visuelle et/ou textuelle pour un langage dédié,
- Définir des **transformations** de modèles,
- S'initier à la **génération de code**.

# Organisation du cours

## Evaluation :

- **70 % Contrôle continu**
  - 60 % Projet
  - 10 % présentation orale
- **30 % Examen**

## Chapitre 1

### Concepts principaux

Modèle, Méta-modèle,  
Transformation.

## Chapitre 2

### Modélisation logicielle

UML et OCL.

## Chapitre 3

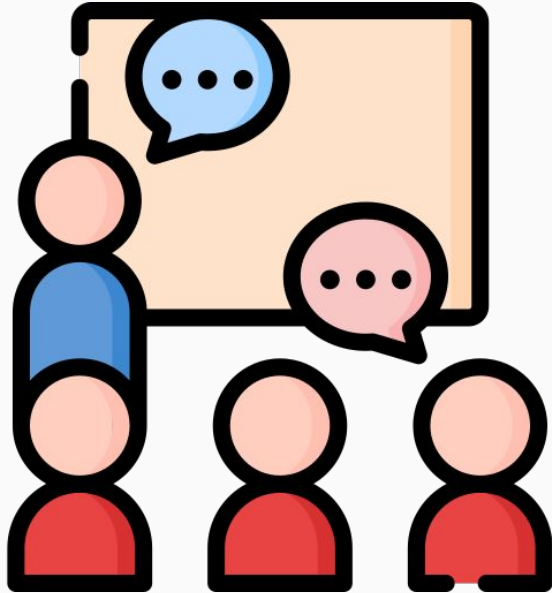
### Méta-modélisation et DSL

Syntaxe abstraite et concrète,  
Sémantique.

## Chapitre 4

### Transformation de modèles

Transformation M2M,  
M2T/M2C.



## Objectifs :

- Se familiariser avec la **recherche scientifique**,
- Approfondir sa **compréhension des concepts** liés à l'IDM,
- Rendre l'IDM **moins abstraite**.

## Durée de la présentation :

- **10 - 15 minutes** de présentation,
- **5 minutes** de questions.

## Planning :

- Entre la **semaine 45** et la **semaine 49**,
- Date de passage annoncée **semaine 40** max,
- Notes de l'orale délivrées **après la fin de toutes** les présentations.

# Pourquoi l'IDM ?

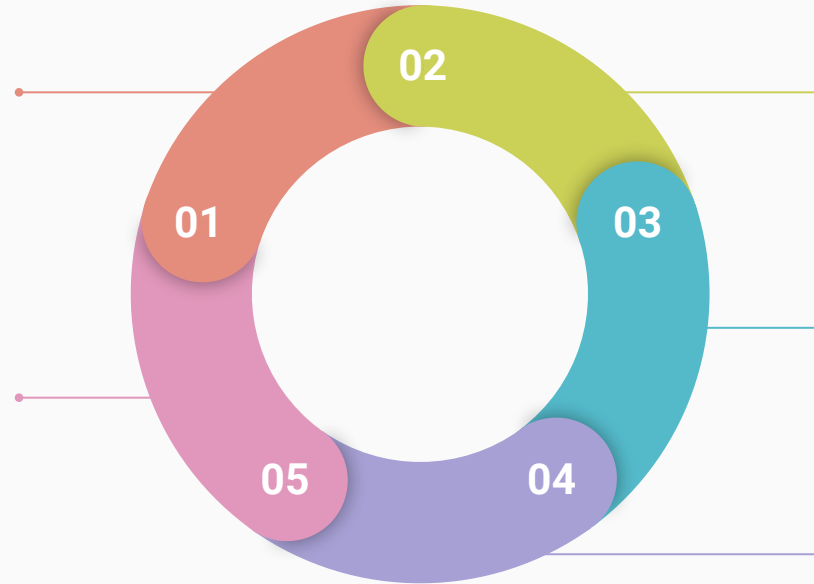
# Rappels : Développement d'un programme informatique

## Analyses des besoins

Comprendre les exigences du client et les objectifs du projet

## Déploiement et maintenance

Installer le logiciel en production et assurer sa maintenance continue.



## Conception

Planifier l'architecture et les fonctionnalités du logiciel.

## Implémentation

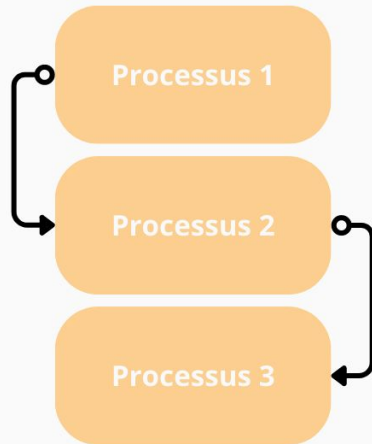
Écrire le code source basé sur la conception.

## Tests et validation

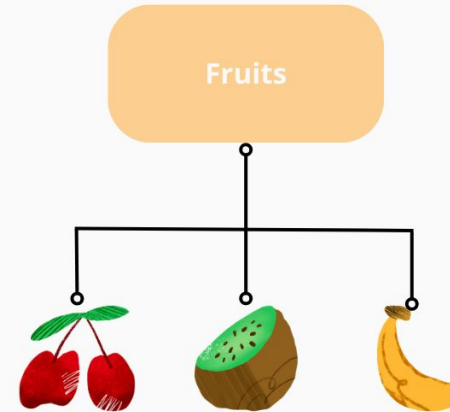
Vérifier le bon fonctionnement du logiciel par des tests.

# Rappels : Programmation procédurale VS Programmation Orientée Objet

## Procédural



## Orienté Objet

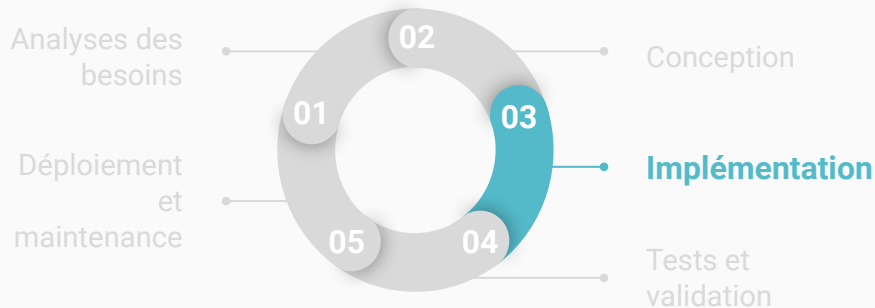


## Approche traditionnelle

VS

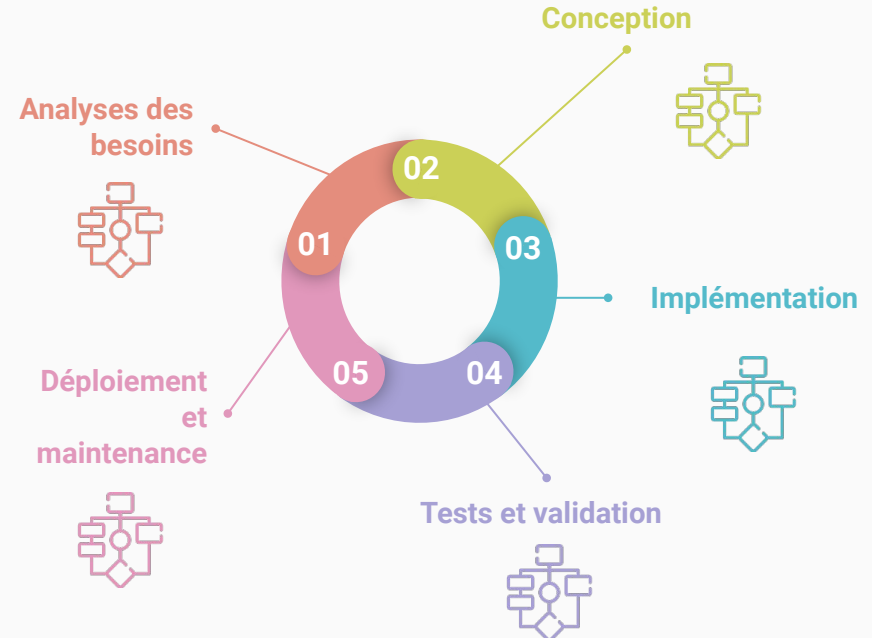
## Approche dirigée par les modèles

- **Centrée sur le Code :**  
Écriture du code source = étape principale



- **Définition dans le Code :**  
Spécifications + Décisions de Conception  
+ Fonctionnalités  
↳ Directement exprimées dans le code.

- **Centrée sur les Modèles :**  
Modèles = artefacts utile pour chaque étape



- **Séparation des Préoccupations :**  
Spécifications + Décisions de Conception  
+ Fonctionnalités = **Modèles distincts**

## Approche traditionnelle

VS

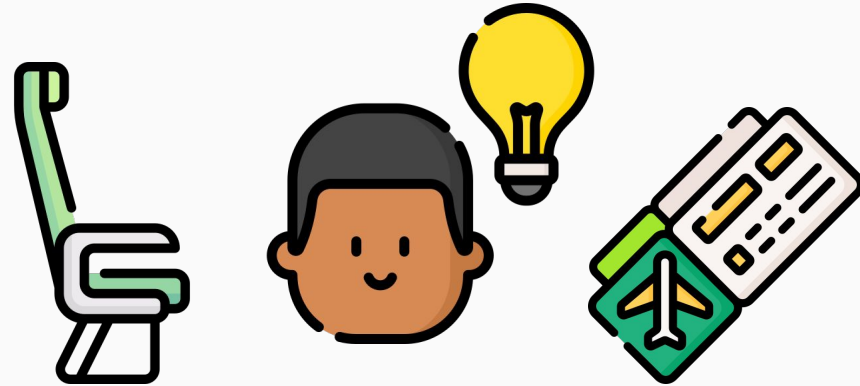
## Approche dirigée par les modèles

- **Faible Abstraction :**  
Code source = représentation concrète du système  
↪ Peut rendre difficile la compréhension des concepts abstraits et des modèles globaux.



- **Réutilisation Limitée :**  
Solutions et morceaux de code → souvent réécrits pour chaque projet.

- **Niveau d'Abstraction Élevé :**  
Modèles = abstractions du système  
↪ Permet de décrire des aspects globaux et spécifiques de manière plus claire.



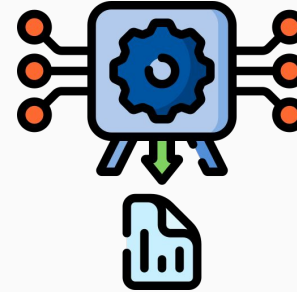
- **Réutilisation Accentuée :**  
Modèles facilement réutilisables pour ≠ parties du système.

- **Risque d'Erreur Humaine :**  
Développement repose fortement sur la compétence des développeurs → Risque d'erreurs humaines.



- **Changements Complexes :**  
Modifications dans le code → Possible répercussions sur l'ensemble du système.  
↪ Maintenance complexe.

- **Moins d'Erreurs Humaines :**  
Erreurs potentielles réduites → Transformations de modèle automatisées.

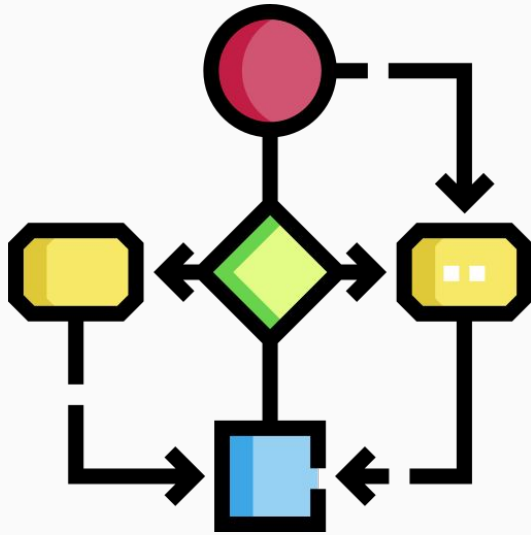


- **Maintenance Simplifiée :**  
Modifications sur modèles → Génération automatique du code.  
↪ Code mis à jour → Maintenance facilitée.



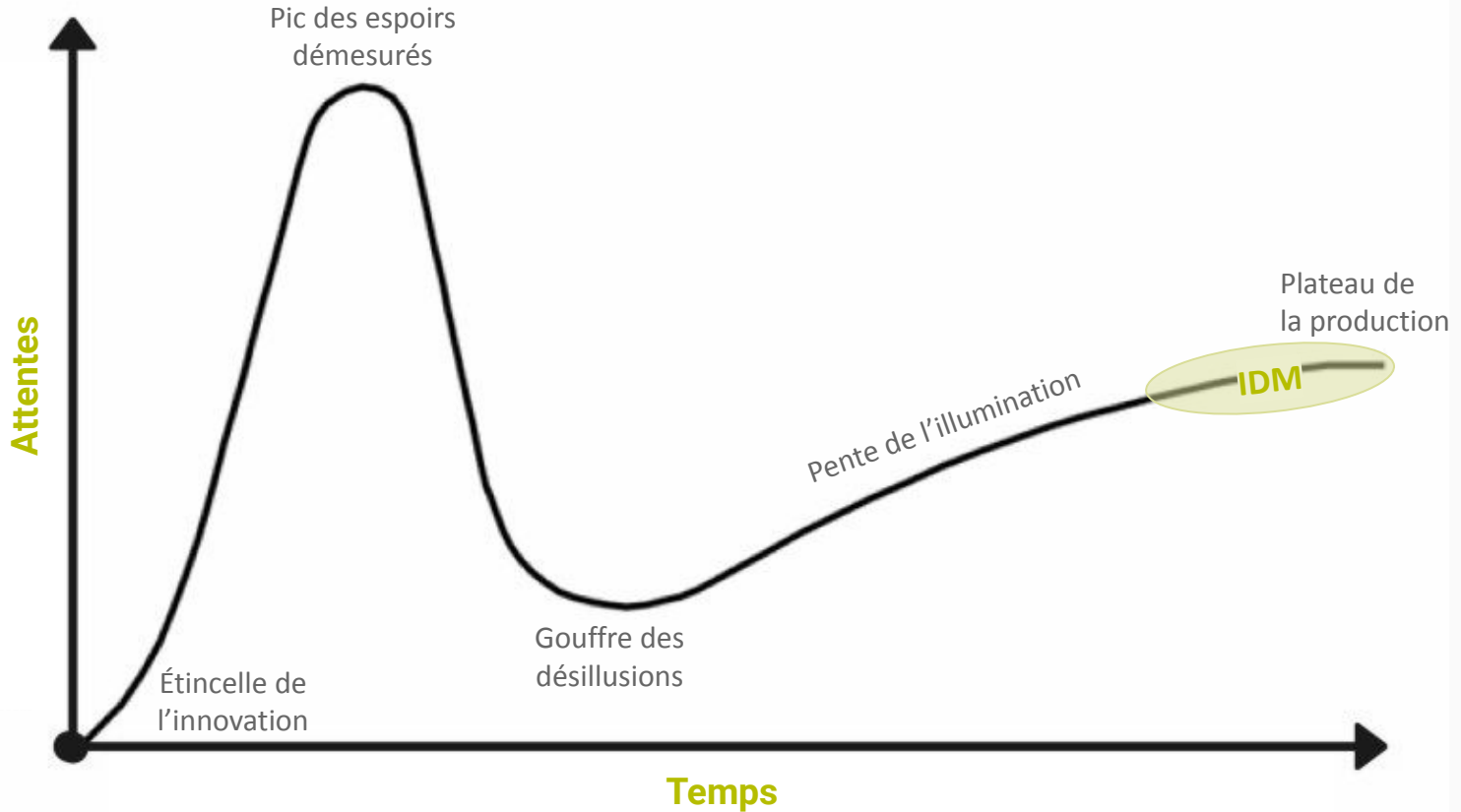
# Qu'est-ce que l'IDM ?

# Ingénierie des Modèles : Qu'est-ce que c'est ?



- Remplacer le **code** par des **modèles** haut niveau **pour développer** des logiciels,
- **Conception** de **langages** pour des tâches ou **domaines spécifiques**,
  - Langages dédiés *graphiques* ou *textuels*.
- **Augmentation** de la **compréhensibilité** pour les développeurs grâce aux modèles,
  - Développeur *pas* nécessairement *informaticiens*,
  - Réduit les *complexité accidentelle*.
- Activité principale : **transformations** de modèles,
  - Génération de code.

# Ingénierie des Modèles : Hype Cycle



## Présence dans l'industrie

Spécialisé dans l'IDM :

**SPARX**  
SYSTEMS

Capgemini

 **modelio**  
the open source modeling environment

**MES**  
MODEL ENGINEERING SOLUTIONS

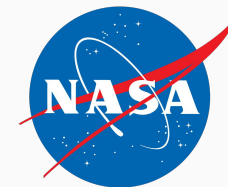
 **mendix**

 **MetaCase**

Utilise l'IDM :

**IBM**

**AIRBUS**



**ABB**

**THALES**

 **RENAULT**

**SIEMENS**



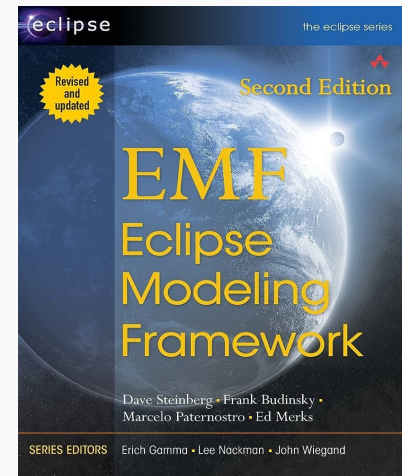
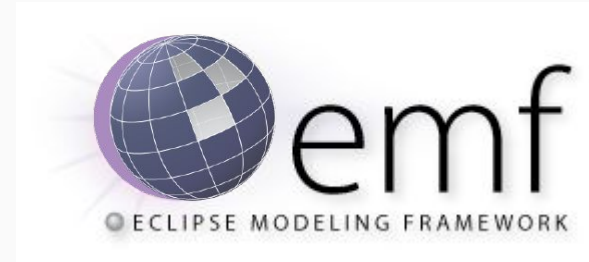
# EMF



- *Plateforme de développement intégrée (IDE)* open-source pour la **création**, le **développement** et la **gestion** de logiciels :
  - Java, C/C++, Python, etc ...
  - Editeur de code, outils de débogage avancés, gestion de projets, ...
- **Extensible** : système de plugins pour ajouter de nouvelles fonctionnalités personnalisées en fonction de besoin spécifiques,
  - communauté active de développeurs = mise à jour régulières et nouvelles fonctionnalités.

# Eclipse Modelling Framework (EMF)

- Composant d'Eclipse qui se concentre sur la **modélisation** des données et la **création de métamodèles**,
- Permet de **générer du code Java** à partir des méta-modèles créés,
  - Le code généré *doit être complété*.
- **Éditeur de modèles** qui peut-être agrémenté de nombreux plugins pour ajouter de nouvelles fonctionnalités,
  - Transformation de modèles,
  - Syntaxe concrète,
  - Langage de contrainte (OCL),
  - ...



**Référence** : Steinberg, D.; Budinsky, F.; Paternostro, M. & Merks, E. (2009), *EMF: Eclipse Modeling Framework* , Addison-Wesley , Upper Saddle River, NJ .

**Lien BU** : [https://uppa.primo.exlibrisgroup.com/permalink/33BUPAU\\_INST/14etfq6/alma990003045490206166](https://uppa.primo.exlibrisgroup.com/permalink/33BUPAU_INST/14etfq6/alma990003045490206166)

## Exemple simple : AchatCommande



Diagramme de classe UML

```
public interface AchatCommande {
    String getAdresseLivraison();
    void setAdresseLivraison(String value);
    String getAdresseFacturation();
    void setAdresseFacturation(String value);
    List getItems();
}
...
```

Code Java

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www..."
            xmlns:PO="...">
  <xsd:complexType name="AchatCommande">
    <xsd:sequence>
      <xsd:element name="AdresseLivraison"
                    type="xsd:string" />
      <xsd:element name="AdresseFacturation"
                    type="xsd:string" />
      <xsd:element name="Items"
                    type="PO:Item"
                    minOccurs="0"
                    maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
  ...
</xsd:schema>
```

Schéma XML

# EMF : AchatCommande

CoursTest - AchatCommande/model/achatCommande.ecore - Eclipse IDE

File Edit Navigate Search Project Sample Ecore Editor Run Window Help

Model Explorer

type filter text

- AchatCommande
  - Project Dependencies
  - src-gen
    - achatCommande
      - AchatCommande.java
      - AchatCommandeFactory.java
      - AchatCommandePackage.java
      - Item.java
    - achatCommande.impl
      - AchatCommandeFactoryImpl.java
      - AchatCommandeImpl.java
      - AchatCommandePackageImpl.java
      - ItemImpl.java
    - achatCommande.util
      - AchatCommandeAdapterFactory.java
      - AchatCommandeSwitch.java
    - achatCommande.validation
      - AchatCommandeValidator.java
      - ItemValidator.java
  - src
    - build.properties
    - plugin.properties
    - plugin.xml

Tree Editor

Syntaxe graphique

Code Java Généré

Format XML

Syntaxe textuelle

Selected Object: achatCommande

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:xmi version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="achatCommande" nsURI="http://www.example.org/achatCommande">
  <eClassifiers xsi:type="ecore:EClass" name="AchatCommande">
    <EStructuralFeatures xsi:type="ecore:EAttribute" name="adresseLivraison" lowerBound="1"
      eType="ecore:EDatatype http://www.eclipse.org/emf/2002/Ecore#EString"/>
    <EStructuralFeatures xsi:type="ecore:EAttribute" name="adresseFacturation" lowerBound="1"
      eType="ecore:EDatatype http://www.eclipse.org/emf/2002/Ecore#EString"/>
    <EStructuralFeatures xsi:type="ecore:EReference" name="items" upperBound="-1"
      eType="#//Item" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Item">
    <EStructuralFeatures xsi:type="ecore:EAttribute" name="nomProduit" lowerBound="1"
      eType="ecore:EDatatype http://www.eclipse.org/emf/2002/Ecore#EString"/>
    <EStructuralFeatures xsi:type="ecore:EAttribute" name="quantity" upperBound="-1"
      eType="ecore:EDatatype http://www.eclipse.org/emf/2002/Ecore#EInt"/>
    <EStructuralFeatures xsi:type="ecore:EAttribute" name="price" lowerBound="1" eType="ecore:EDatatype http://www.eclipse.org/emf/2002/Ecore#EDouble"/>
  </eClassifiers>
</ecore:EPackage>
```

```
import ecore : 'http://www.eclipse.org/emf/2002/Ecore';

package achatCommande : achatCommande = 'http://www.example.org/achatCommande';

class AchatCommande
{
  attribute adresseLivraison : String[1];
  attribute adresseFacturation : String[1];
  property items : Item[*][1] { ordered composes };
}

class Item
{
  attribute nomProduit : String[1];
  attribute quantity : ecore::EInt[*][1] { ordered };
  attribute price : ecore::EDouble[1];
}
```