Modélisation d’applications Openmask

Méthodologie de développement des outils de modèlisation

Développement de deux familles d'outils complémentaires

- **Des outils « *standalone* »**
  - **Générateurs de code, de documentation, de « *make files* » ..**
- **Des outils « *intégrés* » (à l'environnement de développement Eclipse)**
  - **Editeur de modèles, éditeurs d'instances**
  - **Générateurs de code, de documentation, de « *make files* » ..**
  - **Archivage de modèles (dans une base de données relationnelle)**

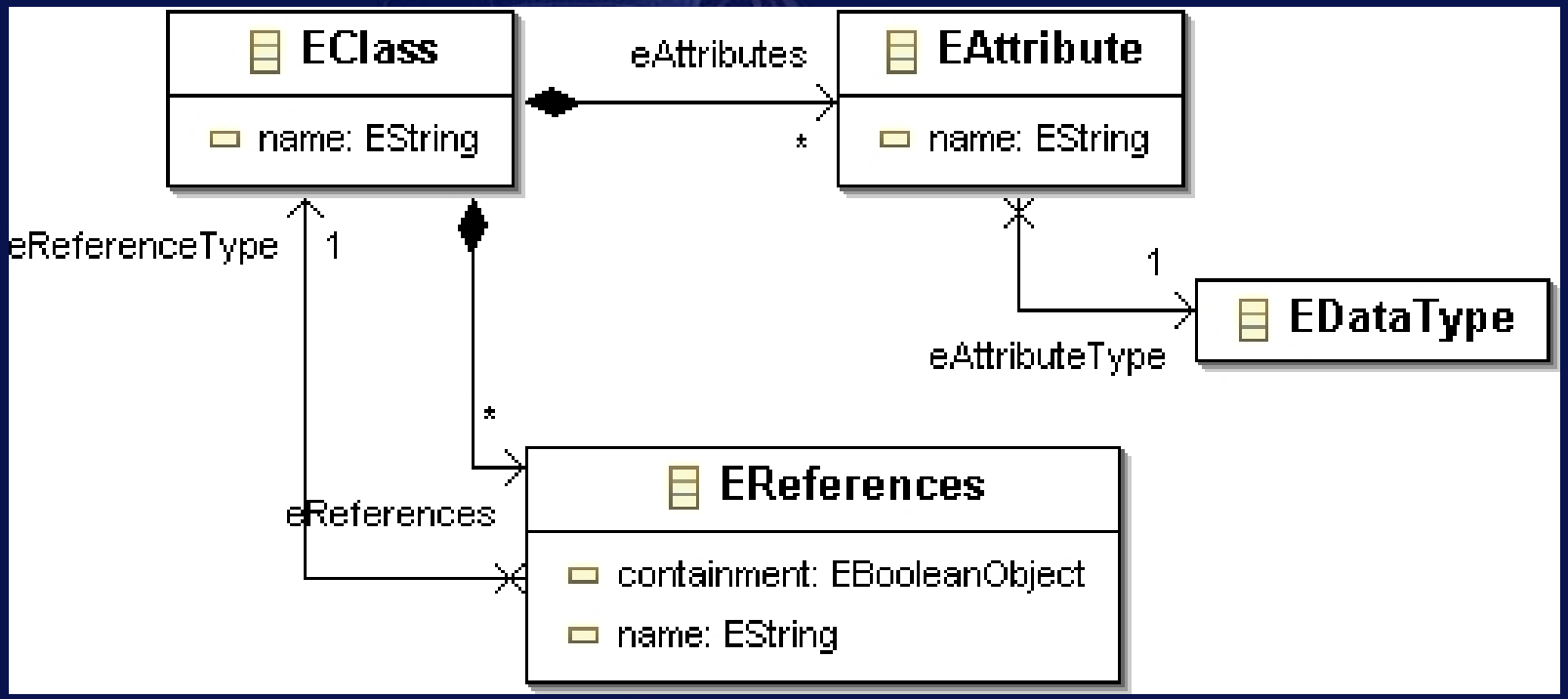
Pourquoi Eclipse et pas un/une autre ?

- **Eclipse n'est pas dédié à un type de développement particulier**
  - **ouvert, extensible, tous ses sources sont disponibles et peuvent servir de modèle de code**
  - **Eclipse est d'abord un outil de developpement d'environnements de développements**
- **Le premier service offert par Eclipse**
  - **son PDE *Plug-in Developement Environment***
- **Le second service offert par Eclipse**
  - **EMF Eclipse Modeling Framework**

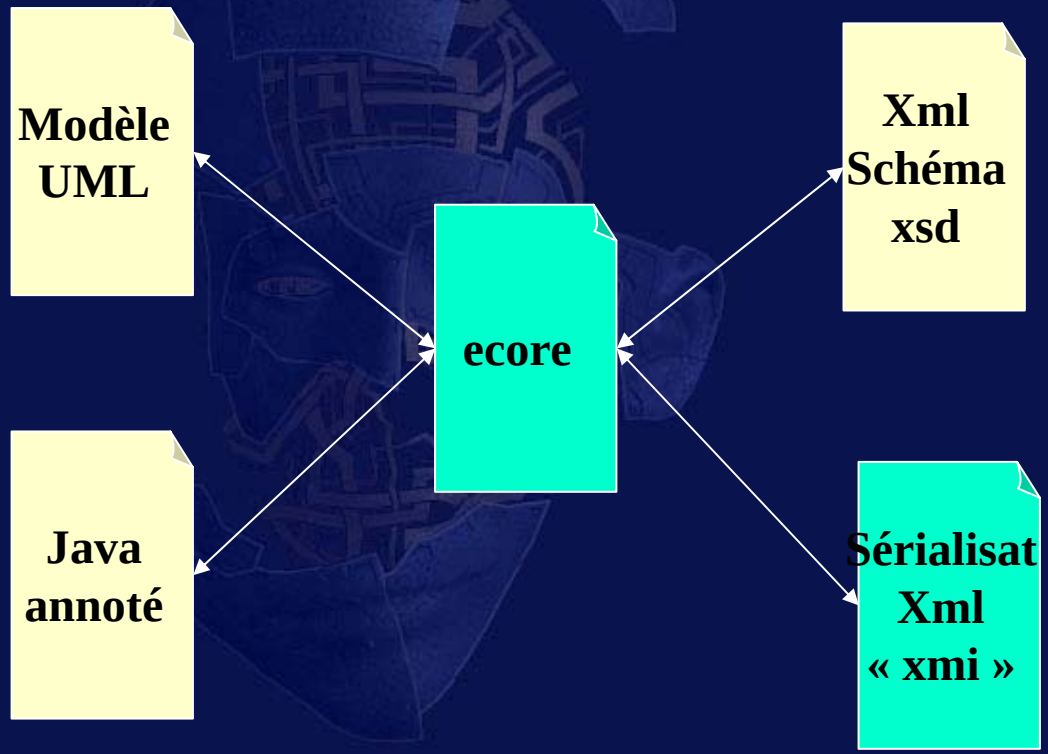
EMF *Eclipse Modeling Framework*

- **EMF un environnement de développement à base de modèles**
- **EMF permet de construire rapidement de toutes sortes d'outils ou d'applications basés sur un modèle de données structurées**
- **EMF offre un langage de description de modèles (méta-modèle) « ecore »**

Ecore un modèle de modèles



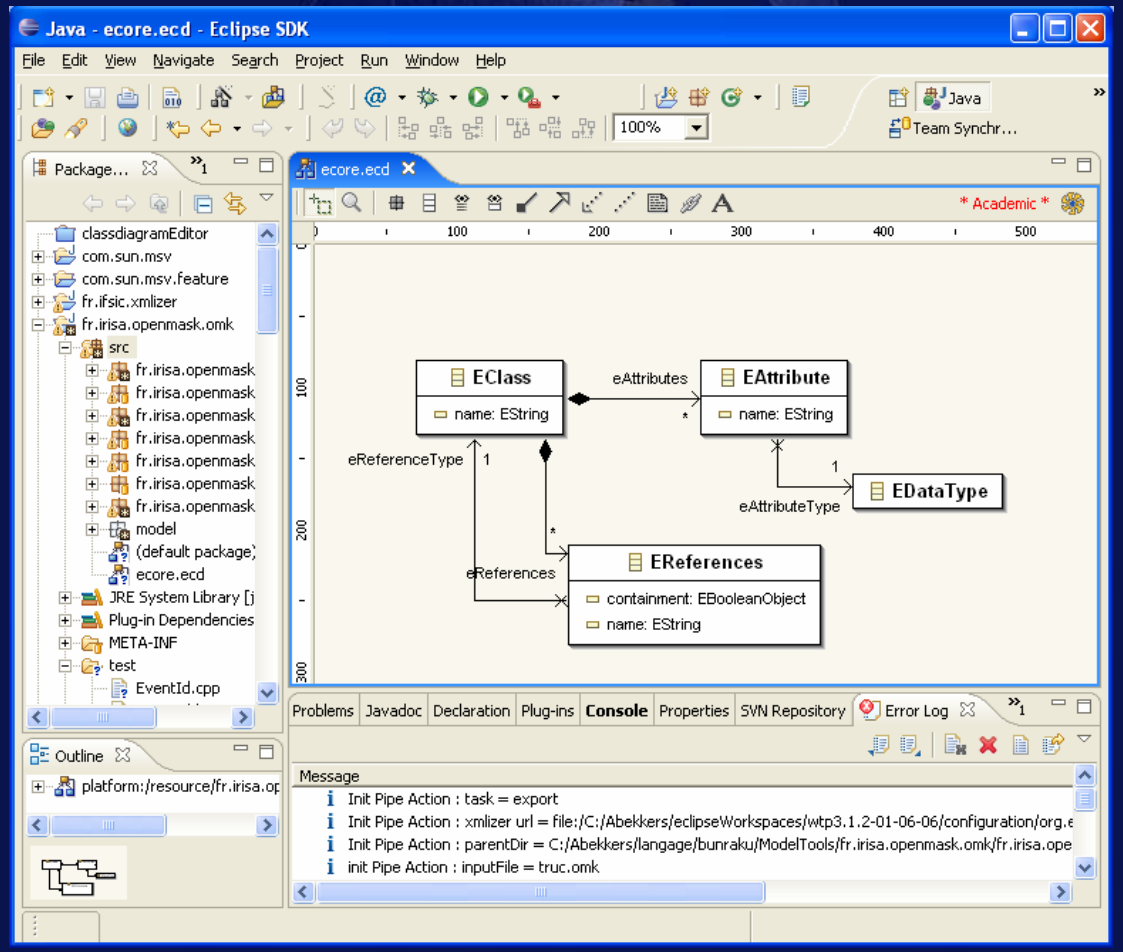
Ecore au centre d'EMF



Ecore : la forme « canonique » des modèles EMF

- **Sources de modèles EMF**
  - **Edition directe du modèle Ecore**
  - **Modéles UML « *Rational* »**
  - **Modèles XML-Schema**
  - **Interfaces ou classes java annotées**
  - **Modèles UML « *Omondo UML* »**
- **Des outils d'import et d'export**
  - **pour passer d'une représentation à une autre**
- **Prise en compte du cycle de vie du logiciel**
  - **On peut revenir aisément sur le modèle**

Edition UML de modèles Ecore



Particularités d'Ecore

- **C'est un modèle de modèles**
- **Ecore s'est autogénéré (grâce à une description de son propre modèle en lui-même)**
- **Tout modèle « *Ecore* » est sérialisable en xml**
  - **Sérialisation construite sur xmi, un dialecte xml normalisé par l'OMG « Object Management Group »**

XMI

- **Un dialecte XML pour importer, exporter, échanger des instances d'objets indépendamment des langages**
- **Tout modèle Ecore est totalement équivalent à sa sérialisation xmi**
- **D'où l'abus de langage « document *.ecore » = instance de modèle ecore en xmi**

Ce qu'offre EMF autour d'Ecore

- **A partir du modèle Ecore d'un utilisateur, EMF génére**
  - **un modèle élaboré et robuste de classes Java qui mettent en œuvre le modèle utilisateur**
  - **Le code de sérialisation/désérialisation des instances d'objets du modèle utilisateur**
  - **Un éditeur d'instance d'objets utilisateurs**

Quel modèle pour notre méta modèle ?

1. **Saisie directe du métamodèle sous Omondo-UML**
2. **Conception d'un schéma de données XmlSchéma**

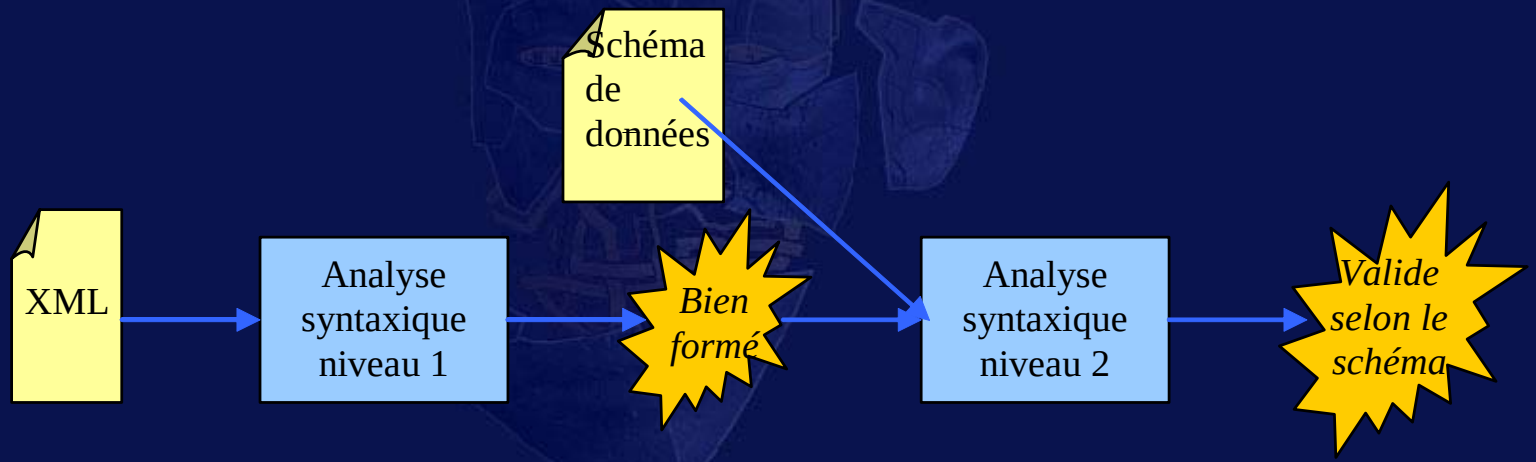
Après avoir pris la solution 1 nous avons adopté à la solution 2

Notre première tâche

- **Concevoir le langage de description de modèles**
  - **Le méta-langage**
    - **Il s'agit d'un dialect xml**
  - **Rédiger la spécification formelle de ce dialecte « omk »**
    - **Par un schéma de données XmlSchéma « xsd »**

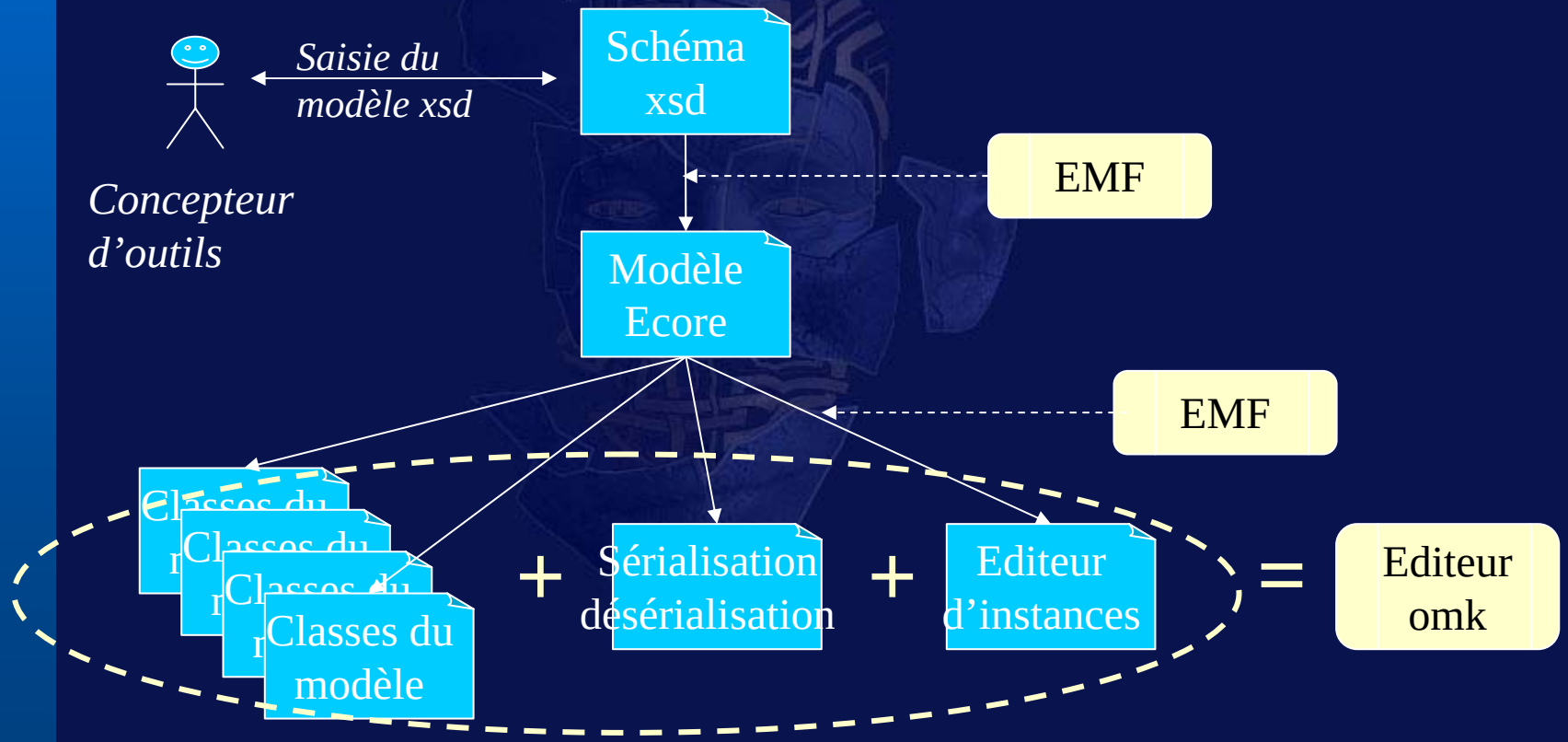
Spécification d'un dialecte xml

- **On spécifie un dialecte xml par son schéma de données**



Note : La spécification actuelle xsd du dialecte omk fait 354 lignes

Le schéma de donné xsd est le départ de toute la chaîne de traitement omk



Exemple de déclaration type de l'élément omk

```
<xs:complexType name="omk">
<xs:sequence>
<xs:element name="doc" type="xs:string" minOccurs="0" />
  <xs:choice maxOccurs="unbounded">
    <xs:element name="PsSimulatedObject"
      type="omk:PsSimulatedObjectType"
      ecore:name="PsSimulatedObject" />
    <xs:element name="PsType" type="omk:PsTypeType"
      ecore:name="PsType" />
    <xs:element name="event" type="omk:event" />
  </xs:choice>
</xs:sequence>
<xs:attribute name="name" type="xs:string" use="required" /
<xs:attribute name="prefix" type="xs:string" />
<xs:attribute name="uri" type="xs:string" />
</xs:complexType>
```

Exemple de description d'un objet de simulation dans le langage omk

```
<PsSimulatedObject name="PsTrajectory"
  exitOnInitError="true">
  <listData name="targets">
    <refType>#//@omk/@psType.1</refType>
  </listData>
  <data name="step" type="float" />
  <output name="position">
    <refType>#//@omk/@psType.0</refType>
  </output>
  <emittedEvent>#//@omk/@event.0</emittedEvent>
</PsSimulatedObject>
```

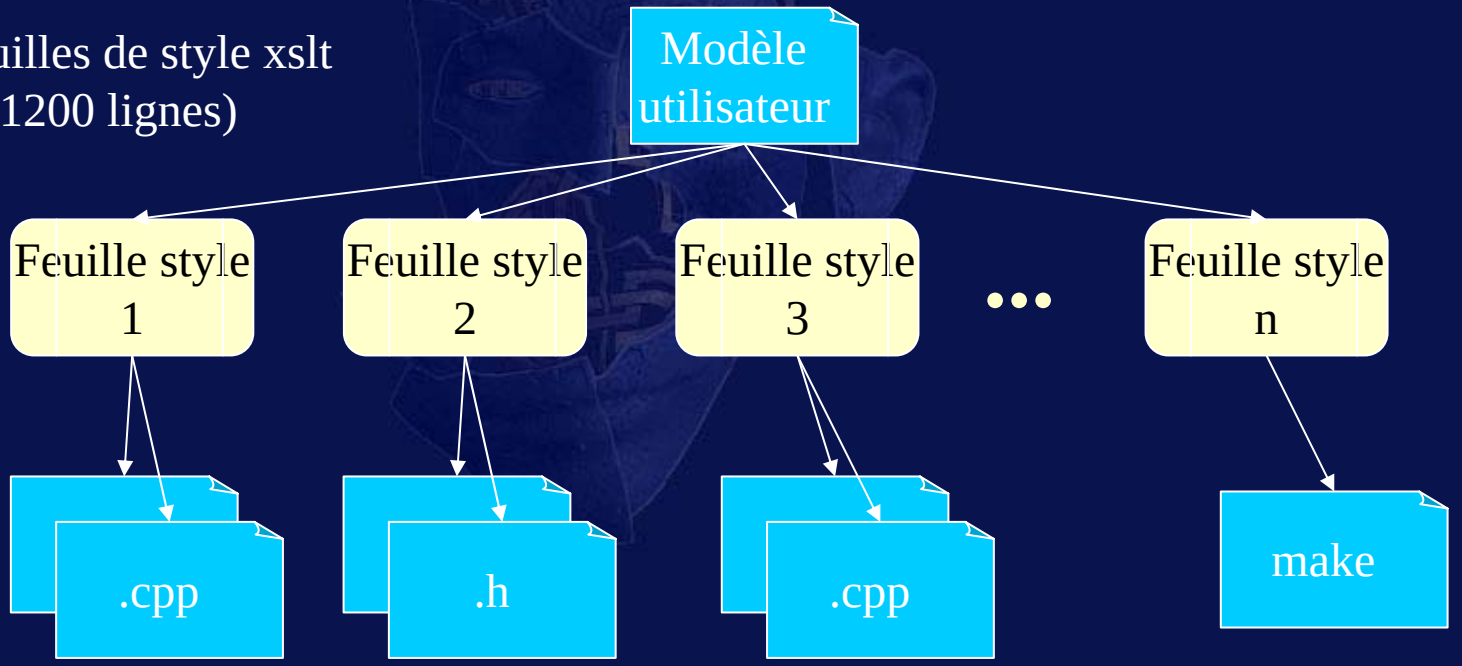
Résumé de la première étape

- **On définit un langage de modèles**
  - **Omk = 354 lignes de description formelle**
- **EMF génère le code du modèle**
  - **Environ 30 interfaces Java**
  - **Environ 30 classes d'implémentation**
  - **Quelques classes de service :**
    - **Sérialisation/désérialisation**
    - **robustes mécanismes d'introspection, de parcours de rémanence des modèles**

Seconde étape

- **Ecriture de feuilles de styles xslt**
  - **Elles génèrent le code à partir du modèle**

21 feuilles de style xslt
(40 à 1200 lignes)



Troisième étape

- **Extension de la plateform Eclipse**
- **On choisit les points de la plateforme qui devront être étendus, exemples :**
  - **Ajouter un menu contextuel sur les documents omk pour lancer la génération de code**
  - **Ajouter une vue sur la base de données de modèles omk**
  - **Ajouter un éditeur de modèles omk (automatique)**

Quatrième étape

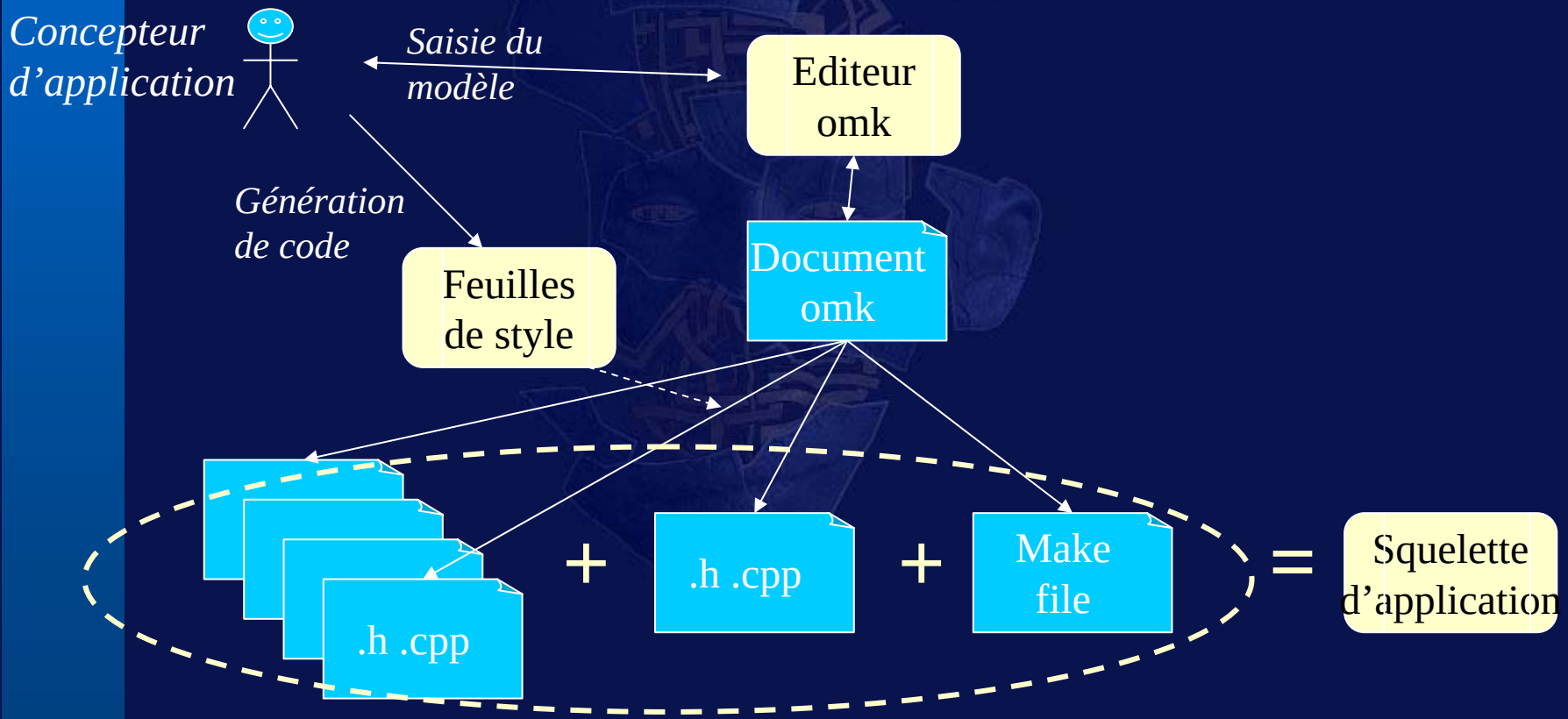
- **Mise en place d'un outil générique de séquencement de traitements xml**
- **On définit un langage de traitements xml**
  - **Lire un document xml**
  - **Valider un documen xml selon un schéma**
  - **Transformer un document xml**
  - **Sérialiser un document xml**
- **Spécification en xsd du langage**
- **Utilisation d'EMF ...**

Exemple de tâche de traitement

```
<xmlizer xmlns="http://www.irisa.fr/xmlizer">
  <pipe id="test002">
    <xmlReader in="src/abc.xml" />
    <xslTransformer xslt="xslt/xml2html.xslt" />
    <xmlWriter out="result/test002.html" />
  </pipe>
</xmlizer>
```

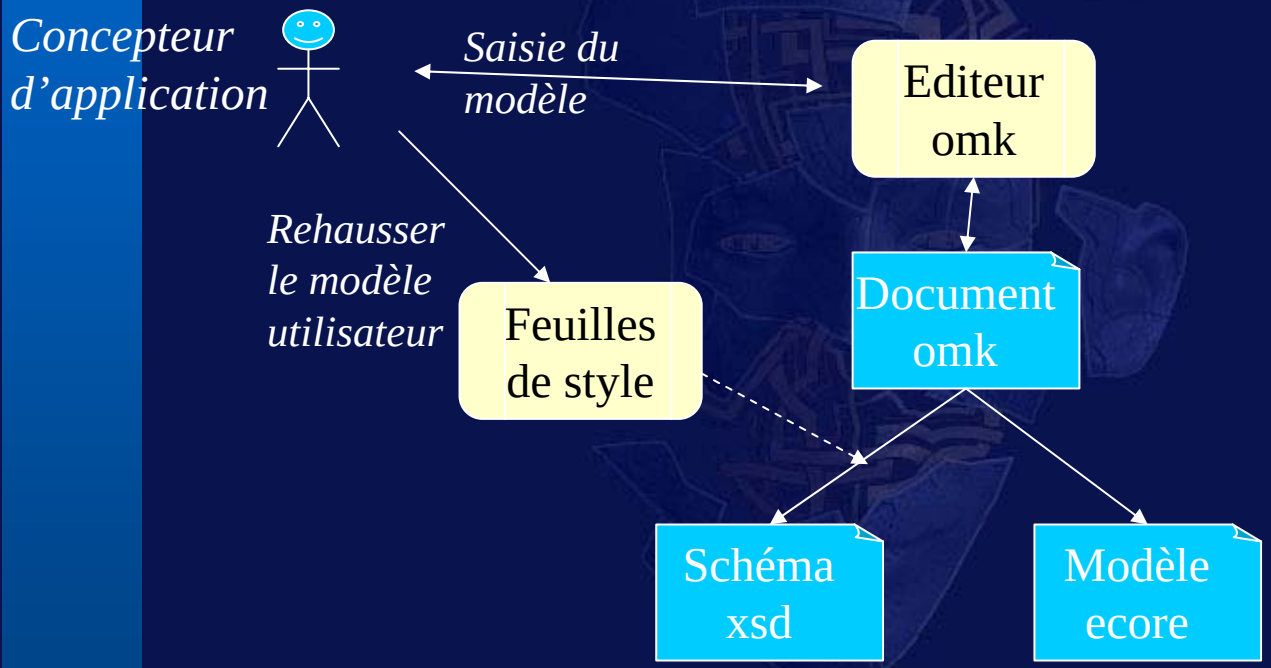
- La même tâche sert
  - pour interpréter l'action « *générer le code* » dans le menu contextuel d'un document omk
  - Pour le traitement en standalone hors eclipse

Les outils décrits jusqu'à présent



Comment aller plus loin !

« Rehausser » le modèle utilisateur comment, pourquoi ?



Génération de l'éditeur de fichiers d'instance