Ingénierie des Modèles (IDM)

TP1 : Métamodèle et Navigation dans un modèle

L'objectif est de revisiter le TD1 avec les outils de modélisation EMF et Kermeta.

Mise en place du TP

Environnement de développement

Méthode 1 : nouvelle installation d'Eclipse

1. Télécharger et installer un Eclipse Luna ou Mars "for Java and DSL Developers" (http://www.eclipse.org/downloads/packages/eclipse-ide-java-and-dsl-developers/mars1)
2. Installer "Ecore Tools" depuis le marketplace (*Help -> Eclipse Marketplace...*)
3. Installer le plugin "K3 AL Feature" depuis l'update site http://kermeta.org/k3/update/
4. Redémarrer Eclipse

Méthode 2 : depuis une installation existante (Luna ou Mars requis)

1. Installer les plug-ins suivants depuis le marketplace (*Help -> Eclipse Marketplace…*)
    a. Xtext
    b. Ecore Tools
    c. Eclipse PDE
2. Installer le plugin "K3 AL Feature" depuis l'update site http://kermeta.org/k3/update/
3. Redémarrer Eclipse

Récupération des projets initiaux depuis Github

Méthode1 : clone depuis Eclipse

- *File -> Import -> Projects from Git*
- *Clone URI* = https://github.com/tdegueul/teaching-mde-istic/
- Importer les projets `fr.istic.idm.spreadsheet.model` et `fr.istic.idm.spreadsheet.prettyprint`

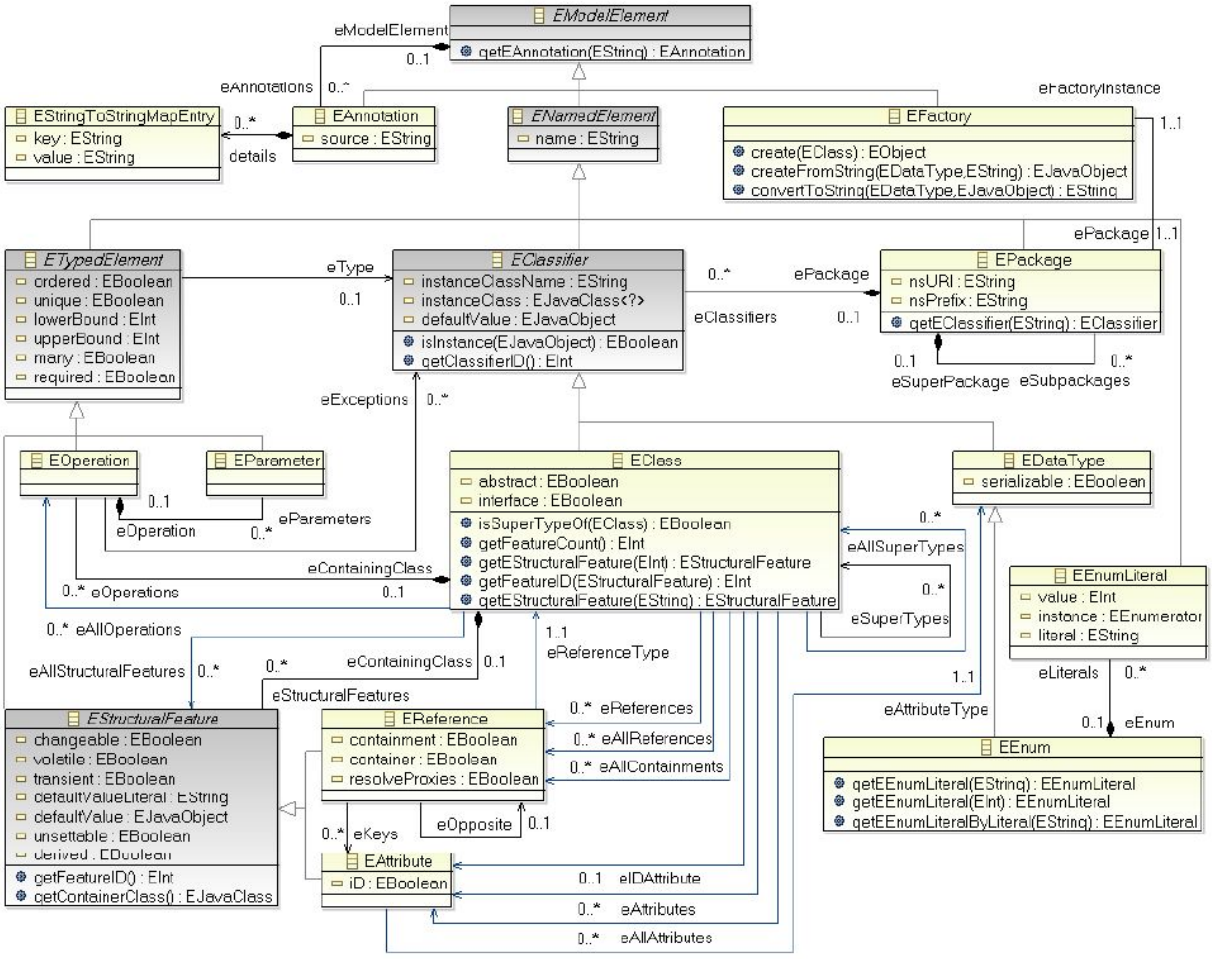
Méthode 2 : clone depuis un terminal

- `$ git clone https://github.com/tdegueul/teaching-mde-istic`
- Importer les projets `fr.istic.idm.spreadsheet.model` et `fr.istic.idm.spreadsheet.prettyprint` via *Import -> Existing projects into workspace* depuis Eclipse

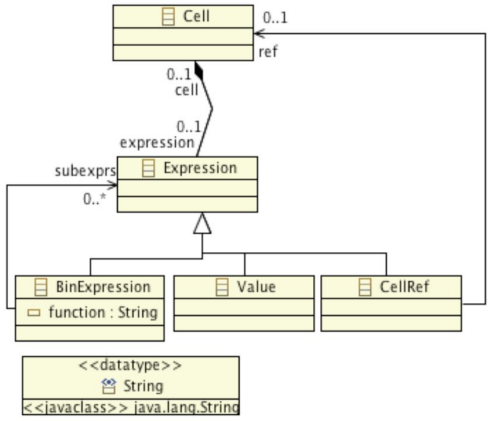
Resources

- Une introduction à EMF : http://www.vogella.com/tutorials/EclipseEMF/article.html
- Une introduction à Xtend: http://www.eclipse.org/xtend/documentation/index.html
- Une introduction à K3 : http://tinyurl.com/K3Tutorial

Ecore, un langage de méta-modélisation



Q1.1 : L'objectif est d'écrire le méta-modèle du tableur (cf. TD1) en utilisant le méta-langage Ecore. Pour cela, passer en vue Modeling : *Window -> Open Perspective -> Modeling*. Une première version contenant uniquement la méta-classe `Cell` est disponible dans `fr.istic.idm.spreadsheet.model/model/spreadsheet.ecore` . Complétez cette version soit en utilisant l'éditeur arborescent (double-clic sur le fichier Ecore) soit en utilisant l'éditeur graphique de diagramme de classe (en déroulant le fichier `spreadsheet.aird` : *Design -> Entities -> spreadsheet)*. Dans les deux cas, vous pouvez utiliser la vue *Properties* (*Window -> Show View*) pour accéder aux attributs des éléments de votre métamodèle.



Q1.2 : L'éditeur de fichier Ecore est dit réflexif. Que permet cette propriété pendant la création d'un méta-modèle ?
Q1.3 : L'éditeur arborescent permet de créer simplement des modèles conformes à notre méta-modèle. Par exemple, clic droit sur `Cell` dans le méta-modèle puis *Create Dynamic Instance*... Ouvrir le fichier .xmi créé avec Sample Reflective Ecore Model Editor (*Clic-droit -> Open With*)

Q2.1 : Créer un modèle d'un tableur comportant quelques cellules. Quel problème rencontre-t-on avec l'éditeur ?
Q2.2 : Proposer une solution pour remédier à ce problème.

Navigation dans un modèle

L'objectif est de visualiser les éléments du méta-modèle du tableur à l'aide de 3 opérations :

- **flat** : affiche la hiérarchie d'héritage des classes
- **short** : décrit les attributs et les opérations d'une classe
- **flatShort** : short en incluant les membres hérités des superclasses

L'affichage peut prendre la forme suivante, par exemple pour flatShort :

```
UneClasse :
        attr nomAttr : NomType
        op nomOp (nomArg : NomType)
        SuperClasse :
                ref nomRole : AutreClasse[0..*]
```

Q3.1 : Implémenter ces 3 opérations dans le fichier `PrettyPrintEcore` du projet `fr.istic.idm.spreadsheet.prettyprint` . Vous pouvez tester votre solution en lançant la classe `Main`.
~~**Q3.2** : Implémenter ces 3 opérations dans un aspect `EClassAspect` en utilisant l'annotation @Aspect de Kermeta 3.~~ (reporté au TP2)
Q3.3 : Proposer une solution utilisant le patron de conception Visiteur.

~~**Q3.3** : Quels sont les avantages et les inconvénients d'utiliser @Aspect par rapport à du Xtend pur ?~~ (reporté au TP2)

On souhaite ajouter un identifiant unique à chaque classe. Cet identifiant sera calculé via une méthode `setIdentifier` qui concaténera le hash code du nom de la classe avec le temps actuel.

```
id = maClasse.name.hashCode + "_" + System.currentTimeMillis
```

On appellera `setIdentifier` sur toutes les classes du méta-modèle juste après le chargement de celui-ci afin de ne définir les identifiants qu'une seule fois. Cela correspond à une transformation model-to-model endogène (un “refactoring”).
Q4.1 : Implémenter `setIdentifier` dans `PrettyPrintEcore` et modifier les précédentes opérations pour supporter l'affichage de cet identifiant.
~~**Q4.2** : Idem dans `EClassAspect`.~~
~~**Q4.3** : Quels sont les avantages et les inconvénients d'utiliser @Aspect par rapport à du Xtend pur?~~ (reporté au TP2)