Tutorial : Conception de syntaxes concrètes graphiques

(TP2)

Au cours de ce tutorial on va examiner un méta modèle et on va générer une syntaxe concrète graphique en utilisant l'utile « GMF ».

Prérequis :

- Eclipse Indigo Modeling SR2
- Le plugin GMF
- JDK 1.6 min
- Le plugin SVN

Etape 1 : Compréhension du modèle sémantique fourni (méta modèle)

1. Pour commencer, lancer Eclipse et sélectionner votre workspace.
2. Changer la perspective en SVN (Window -> Open Perspective -> Other -> SVN Repository Exploring).
3. Créer un nouveau repository location (File -> new Ropository Location -> URL : http://aigle-2012.googlecode.com/svn/trunk/aigle.pfister.gmf.tpb/-> finish).
4. Enregistrer le projet en local (click droit sur le projet -> Check Out)
5. Passer en perspective Java (Window -> Open Perspective -> Java)
6. Déconnecter vous de SVN (click droit sur le projet -> team -> Disconnect)
7. Analyser les fichiers tp2.ecore qui représente la syntaxe abstraite de notre modèle et tp2.ecorediag qui représente la syntaxe concrète **(on est au niveau M2)**. Essayer de comprendre ce méta model

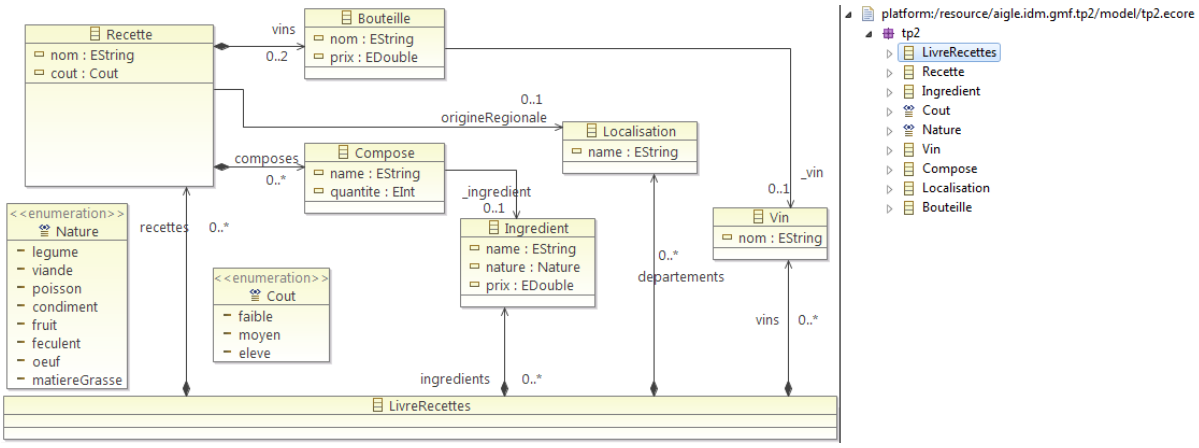

Figure 1 : tp2.ecorediag (gauche) tp2.ecore (droit)

Etape 2 : Génération d'un modèle arborescent, tester la consistance du méta modèle

1. Générer tp2.genmodel (click droit sur le projet -> new -> other -> EMF Generator Model -> ecore -> load tp2.ecore -> finish)
2. Ajouter le préfix a tp2.genmodel (Window -> Show View -> Properties pour avoir la vue des propriétés, puis sur tp2.genmodel sélectionner Tp2 -> dans la vue des propriétés sélectionner Base Package et ajouter « aigle.idm.gmf »)
3. Maintenant qu'on a le genmodel, on peut tester la consistance de notre modèle en utilisant l'éditeur arborescent offert par EMF.
4. EMF ou Eclipse Modeling Framefork se base sur deux méta modèles : Ecore et Genmodel. Ecore contient les informations des classes définies alors que Genmodel contient des informations pour la génération de code, comme par exemple le path, des informations sur les fichiers…
5. Générer l'éditeur (dans tp2.genmodel click droit sur Tp2 -> Generate All)
6. Du code a été générer dans src ainsi que trois nouveaux projets : edit, editor et tests
7. Lancer l'éditeur (click droit sur editor -> Run As -> Eclipse Aplication)
8. Nouvelle instance d'Eclipse est ouverte. Créer un projet général et click droit sur le projet new -> Other -> Tp2 Model -> --- -> Livre Recettes -> finish
9. Tester la consistance : Créer des Recettes, des ingrédients…

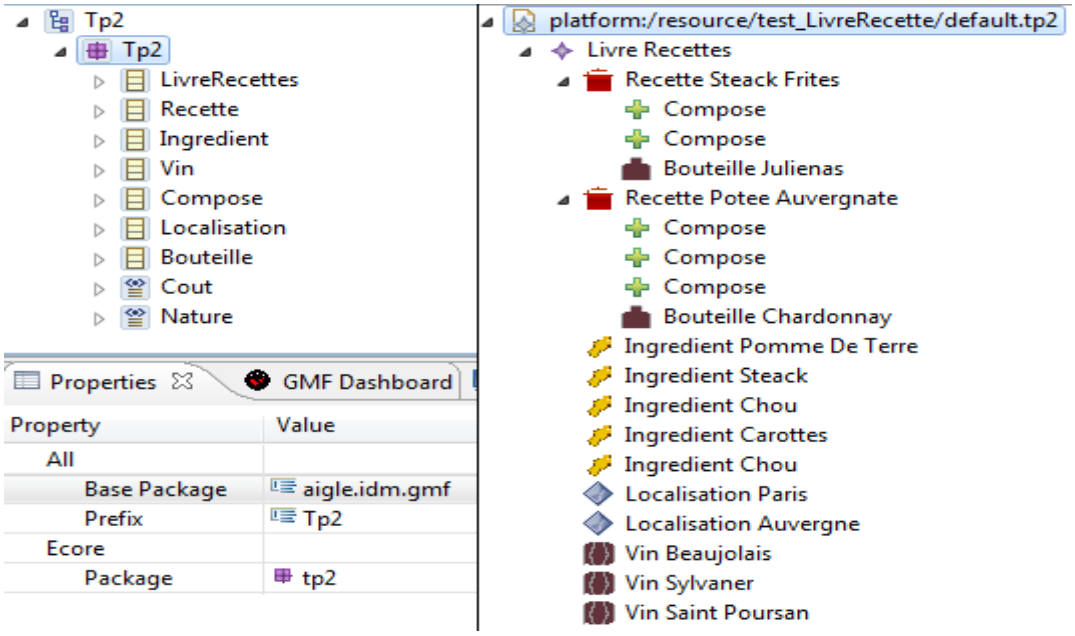

Figure 2 : tp2.genmodel (gauche), l'éditeur arborescent d'EMF (droit)

Etape 3 : Conception de la notation

Maintenant qu'on a testé l'éditeur arborescent, on veut passer à une représentation graphique dotée d'une légende en utilisant GMF. C'est beaucoup mieux de lire une représentation graphique que lire une représentation arborescente. Pour réaliser une représentation graphique, il faut d'abord imaginer à quoi ça va ressembler et faire un

premier exemple « papier-crayon ». Ensuite il faut bien commenter le méta model : des couleurs différents pour des classes représentant des choses différents, par exemple le

canvas , les nœuds du canvas, les liens avec une étiquette etc…, des commentaires pour chaque classe du méta modèle…

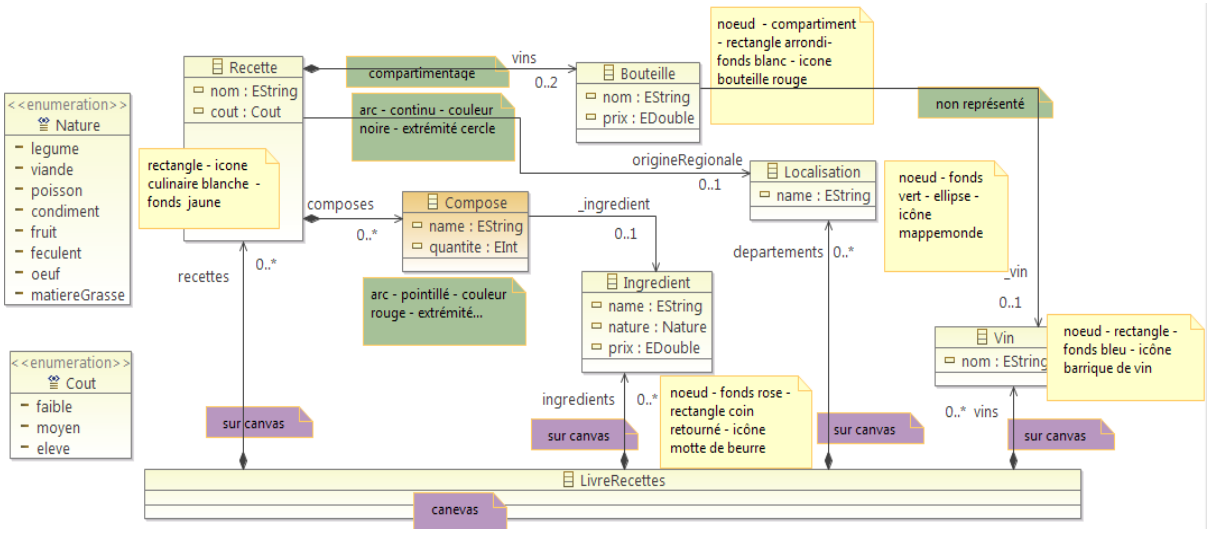

Figure 3 : Commenter le méta modèle

Etape 4 : Implémentation de la notion

Apres avoir fait l'étape 3 qui sert à nous facilite la vie, on peut attaquer à la création de la représentation graphique avec l'utile GMF. Lancer utile GMF (Window -> Show View -> Other ->GMF Dashboard).

7 étapes à faire pour générer le langage contrait (Figure 4).

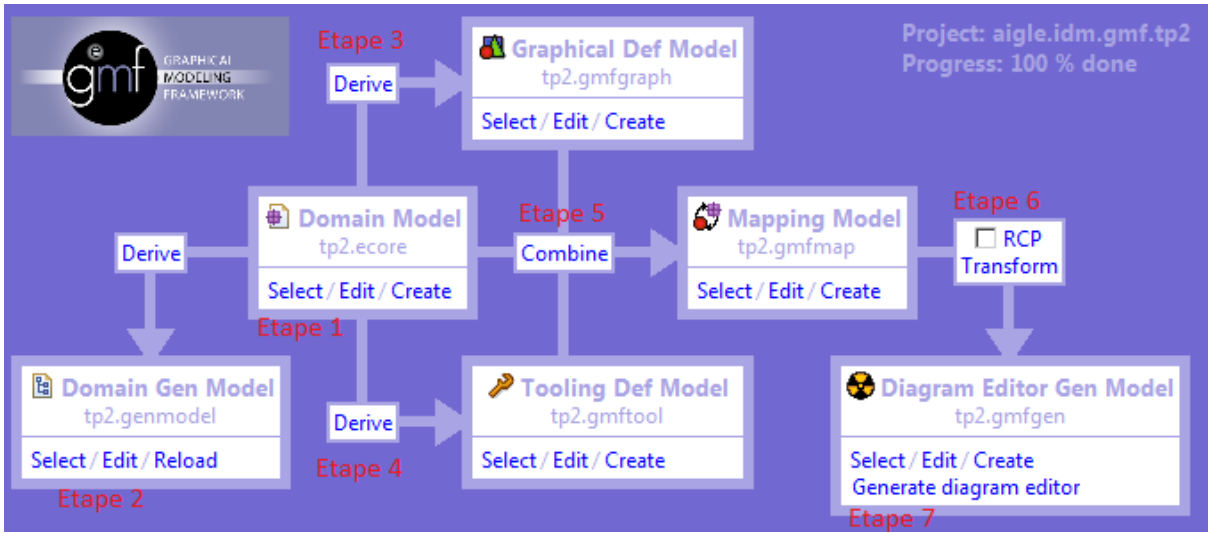

Figure 4 : Les différentes étapes qu'on doit faire pour générer le générateur du langage et le langage

1. Sélection du domaine modèle qui est notre fichier tp2.ecore
2. Sélection du domaine générateur model qui est notre tp2.genmodel
3. Dériver le modèle de définition graphique (Figure 5) qui nous permet de spécifier le design du langage (style des nœuds et arcs)

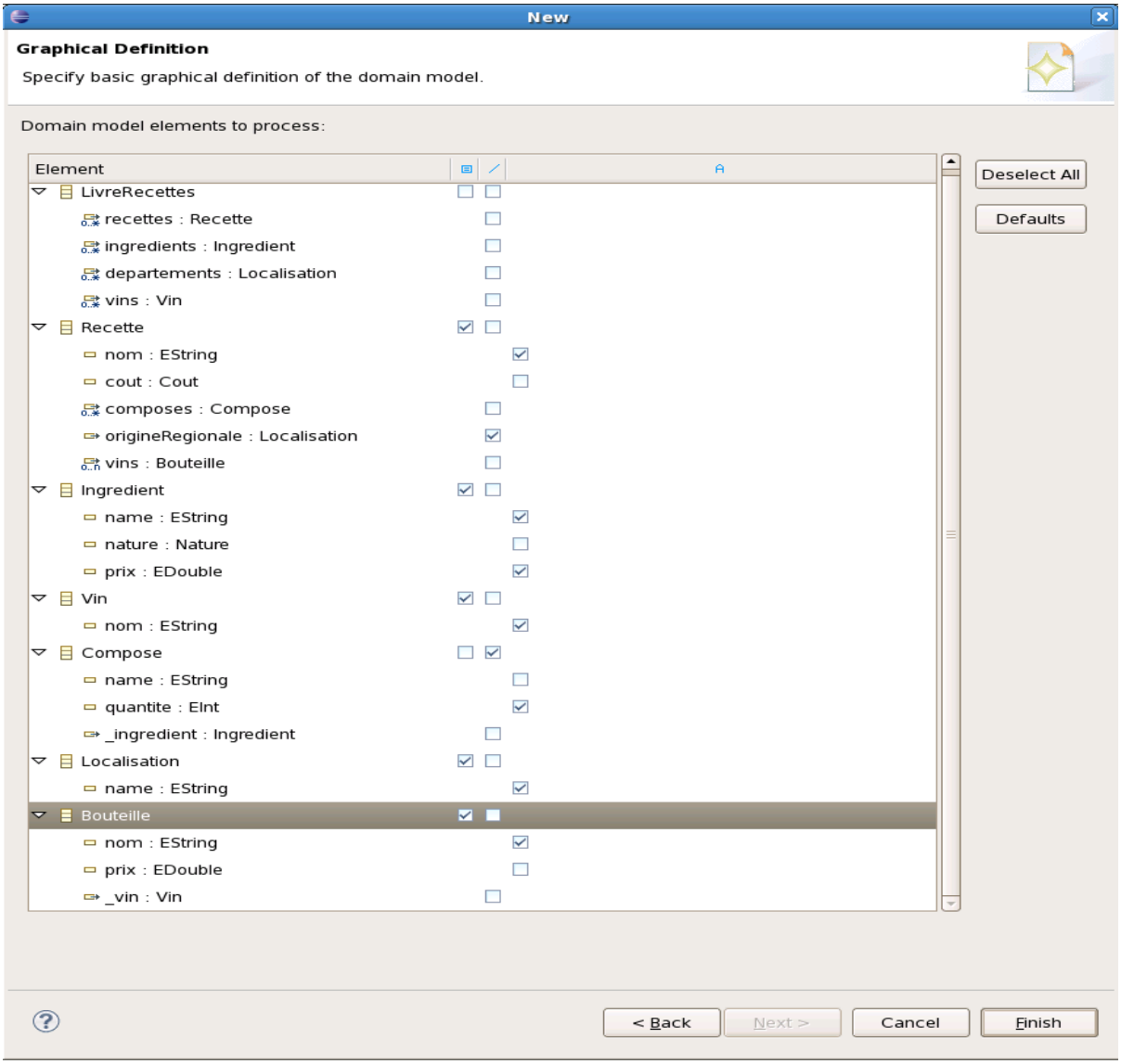

Figure 5 : lors de la création du modèle graphique

Pour chaque classe deux choix : nœud (la première case) ou arc (la deuxième case) pour chaque attribut une case qui signifie attribut visible

4. Dériver le modèle de définition d'utiles (Figure 6) qui nous permet de spécifier la palette qu'on va utiliser pour créer des syntaxes concrètes, et aussi nous permet des spécifier les icones qu'on va visualiser durant la création des syntaxes concrètes.
5. Combiner le modèle d'utile et le modèle graphique, faire le « mapping » pour spécifier quel nœud corresponde à quel utile dans la palette. Il faut sélectionner la class LivreRecettes comme containeur ou « Point of view »
6. Transformer pour obtenir le générateur des modèles
7. Générer le langage qu'on vient de définir

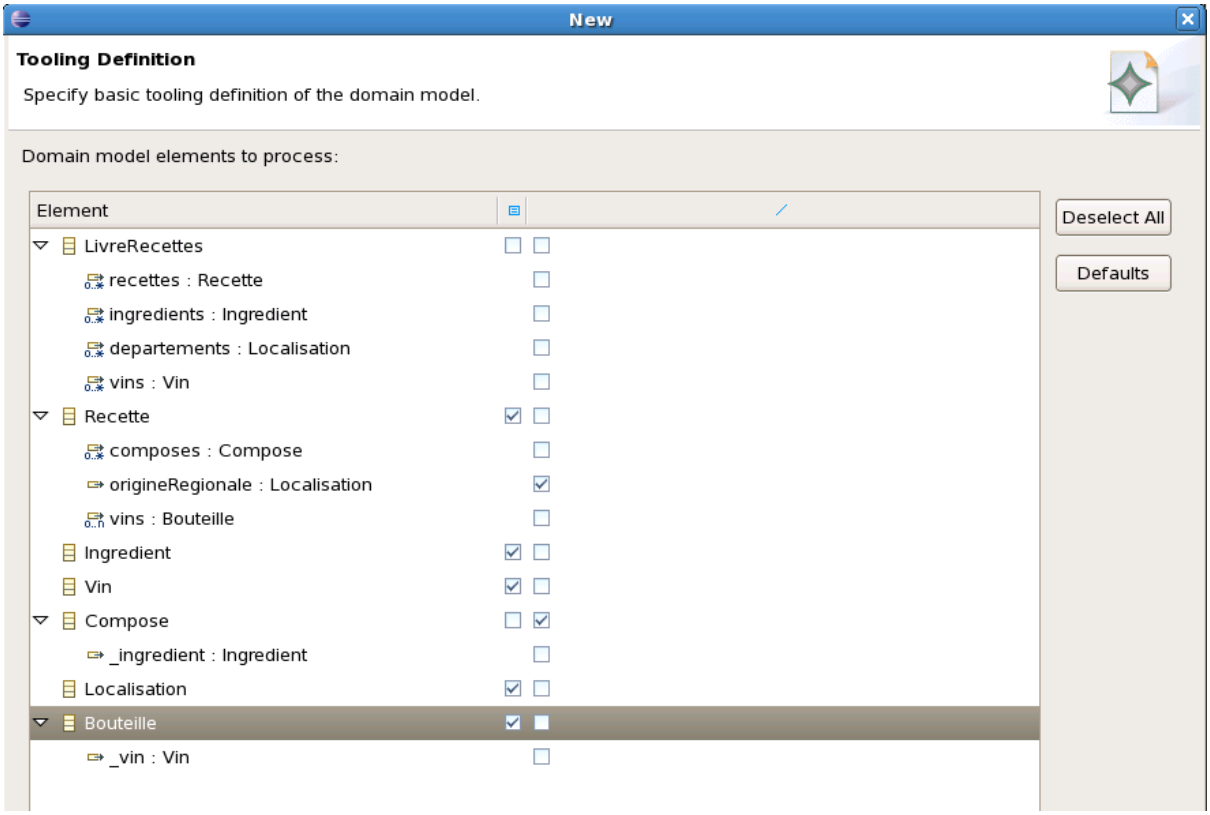

Figure 6 : lors de la création de la palette d'utile, on spécifie les nœuds et les arcs

Apres avoirs générer l'éditeur, il peut ajouter des icônes pour la palette. Par défaut on ajoute les icones dans le fichier « icones/full/obj16 » en format .gif de largeur de 16pixeles, sinon on peut spécifier un chemin pour chaque icone de la palette dans le fichier .gmftool qu'on vient de générer.

Pour lancer l'éditeur, click doit sur le projet .editor -> Run As - >Eclipese Application. Une deuxième instance d'Eclipse se lance ou on va créer un projet général et on va créer un .tp2_diagram dans le projet qui est en fait l'éditeur qu'on vient de générer.

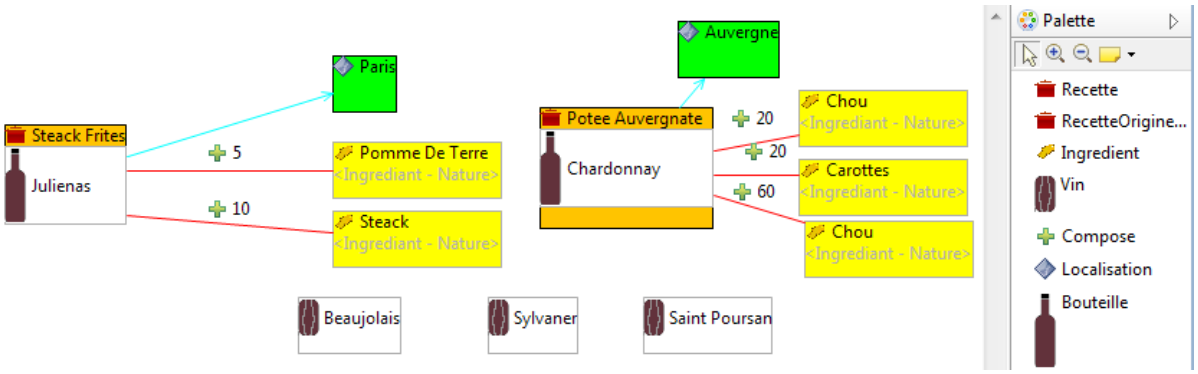

Figure 7 : des recettes définies avec l'éditeur