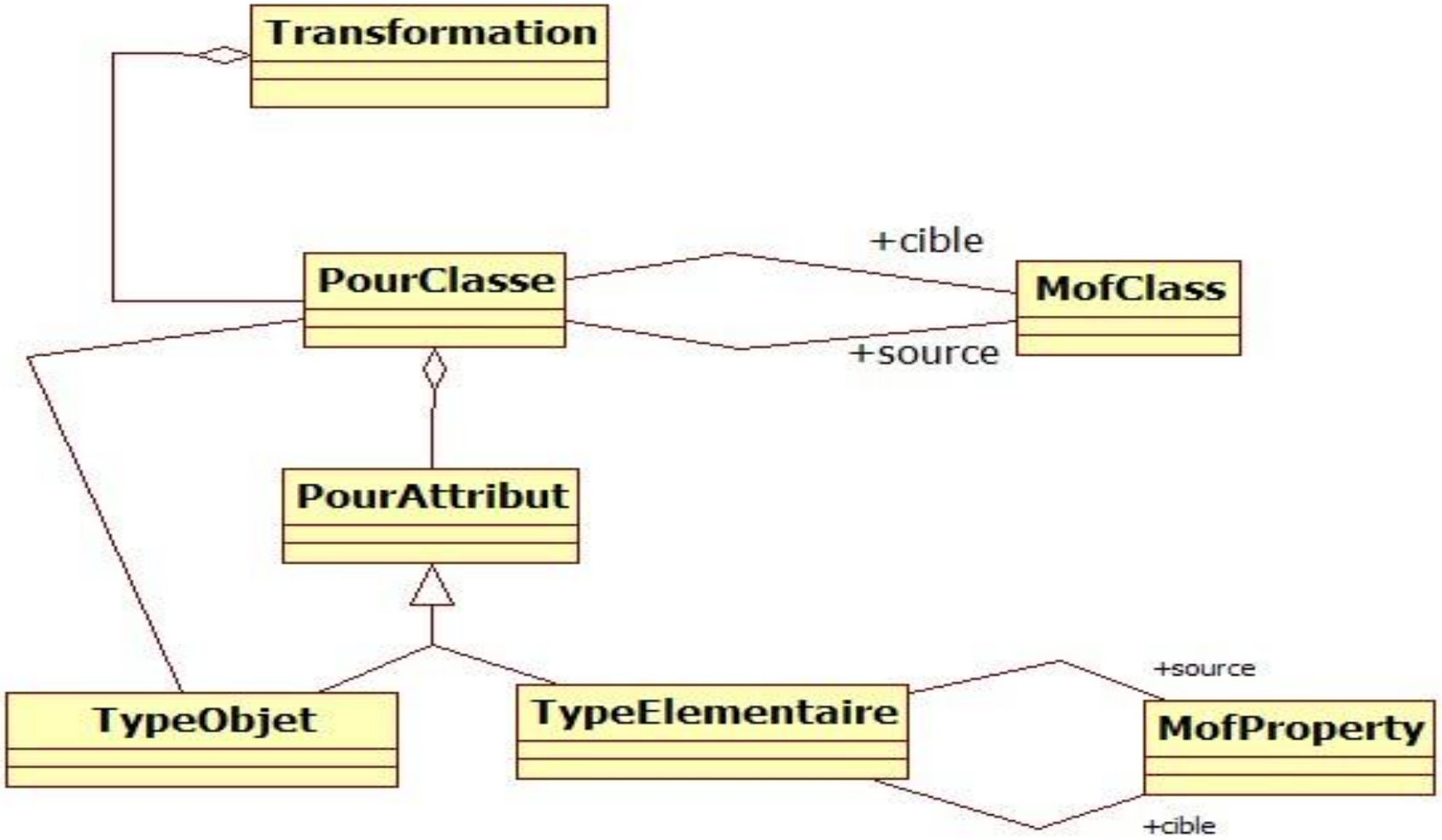
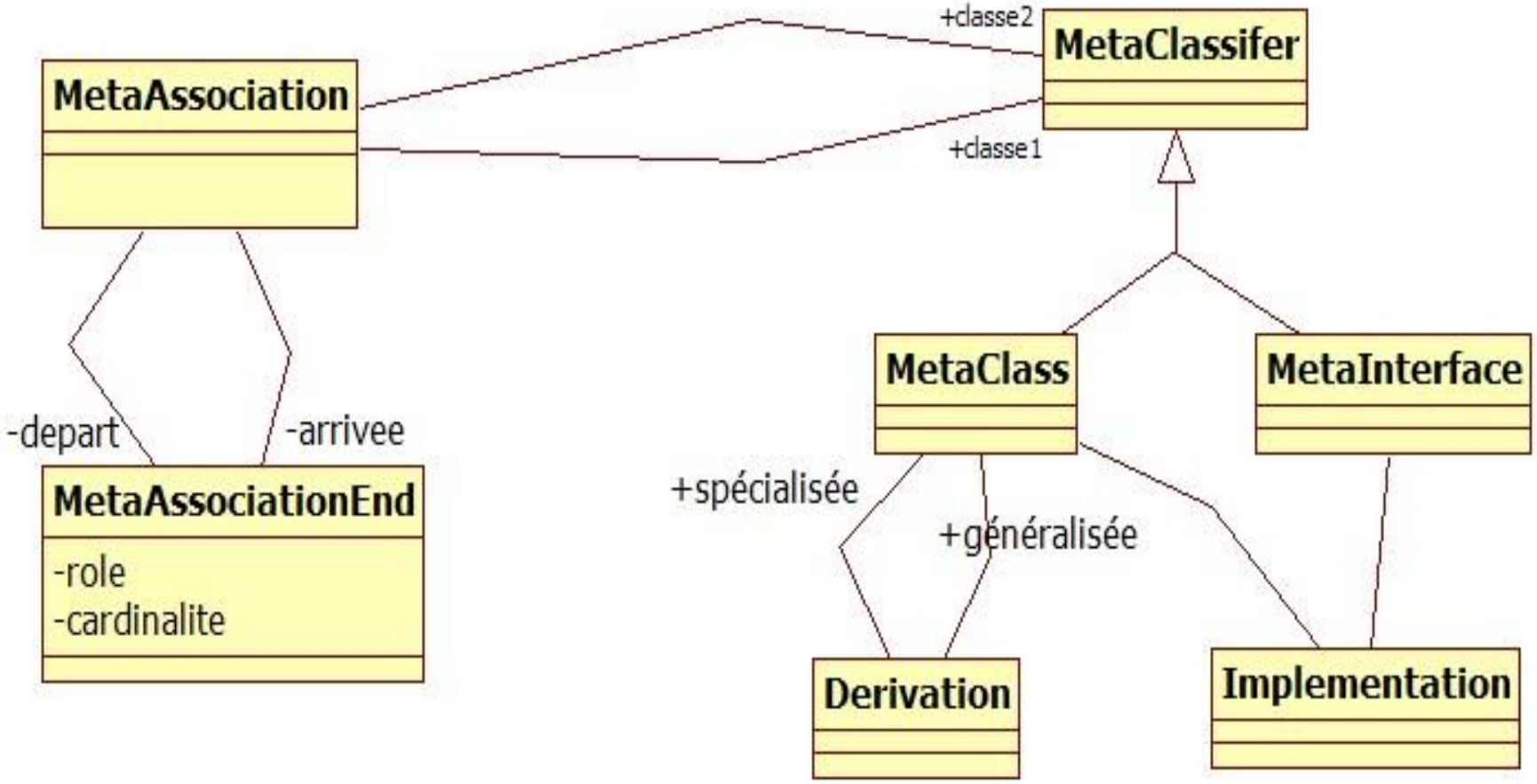
qvt (one to one)



Pseudo UML - Pseudo Java

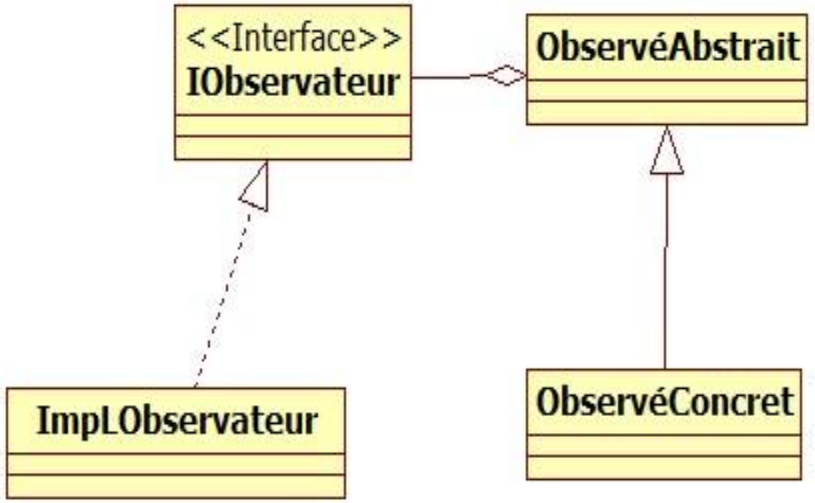


Pattern Observateur/Observé

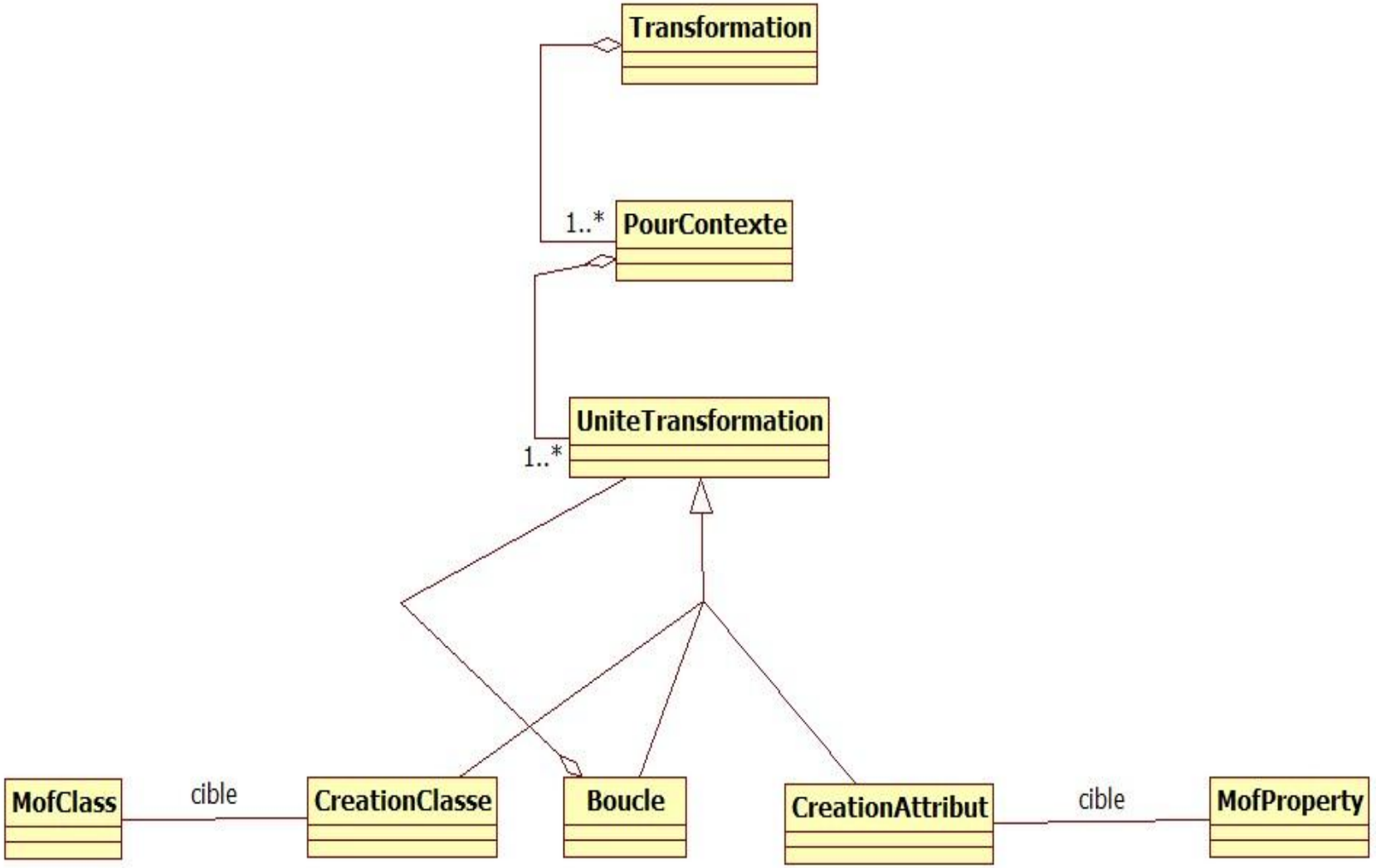
Pattern synthétisé

Nb : l'observateur peut observer plusieurs sortes d'observés.
==> une boucle serait utile

Pattern analysé



Pour traiter les patterns, le traitement 1 <----> 1 est insuffisant ==> on va utiliser OCL comme déclencheur/testeur



~ Langage QVT

- créerClasse nom type
- créerAttribut nom
- évaluerAttribut valeur
- expressionOCL { suite d'instructions }
- relierClasses nomDépart nomArrivée typeLiaison
- à compléter suivant besoin

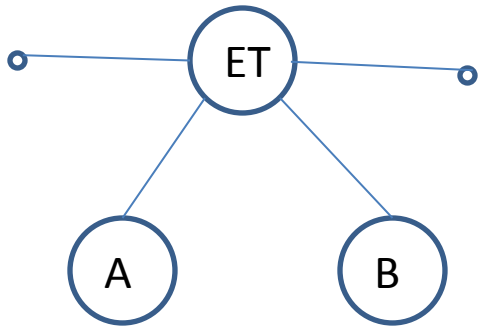
concrète, abstraite ou interface

constante ou expression OCL

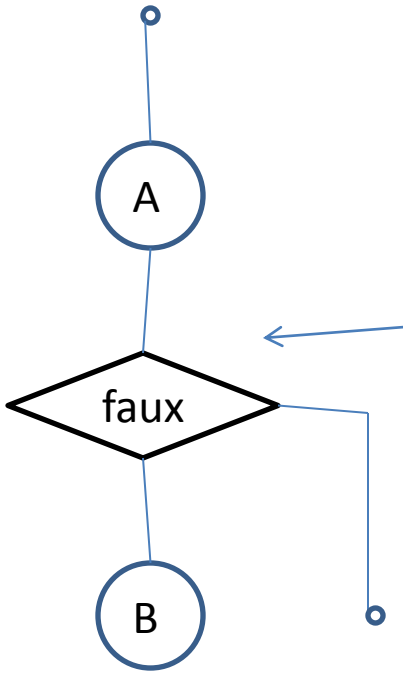
utilisée en lien de "if" ou "while"

association,dérivation, implémentation,…

Optimisation de test (refactoring)



évaluer A
évaluer B
faire ET
renvoyer résultat



Pour "OU" on teste le vrai

évaluer A
si faux renvoyer faux
évaluer B
renvoyer résultat

TP à rendre

Dans ce travail pratique, on fait une première approche de spécification de l'OMG pour la modélisation de transformation de modèles avec le langage QVT.

Il faut définir dans un premier temps :

- Modèle détaillé des transformation 1 --> 1
- Modèle détaillé des transformation n --> p
- Modèle détaillé du mini Java/UML employé

Puis dans un deuxième temps, l'une des transformations suivantes rédigées en texte tel qu'indiqué dans le support de cours

- Observateur/Observé
- MVC
- Ou tout autre pattern
- Evaluation court-circuit des expressions booléennes