lab-sticc.univ-brest.fr/~babau/

Ingénierie Dirigée par les Modèles

Transformation de modèles

(intégration dans Eclipse Modeling Tools)

Jean-Philippe Babau

Département Informatique, UFR Sciences, UBO
Laboratoire Lab-STICC

UBO

jean-philippe.babau@univ-brest.fr

Modèles et méta-modèles

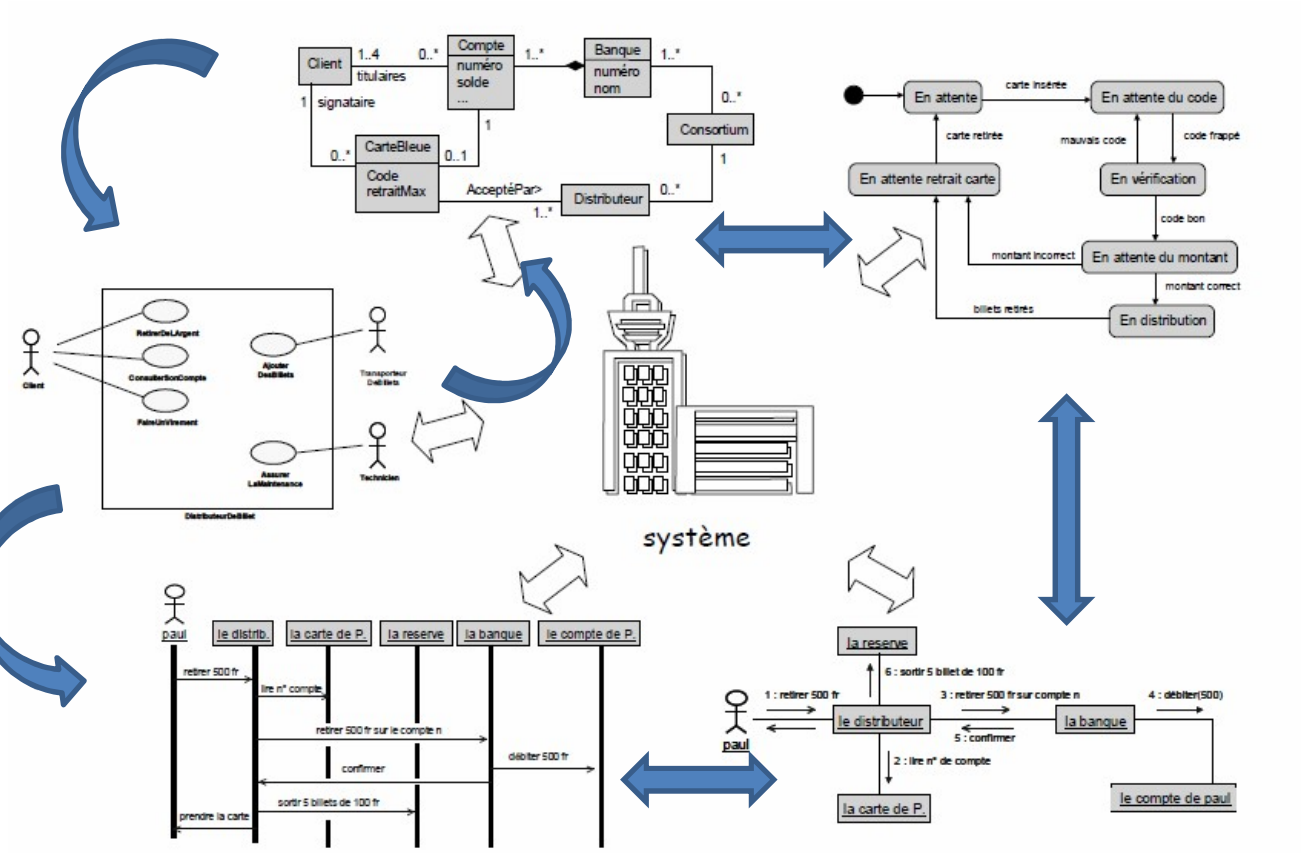
- Introduction aux modèles
  - Ca sert à quoi de faire des modèles ?
- Outils de modélisation et de méta-modélisation
  - Editeurs de méta-modèles et de modèles (EMF)
  - Vérification de modèles (OCL)
  - Editeurs de modèles graphiques et textuels (Sirius, XText)
  - Transformation de modèles M2T (Acceleo) et M2M (ATL, Modif)

Pourquoi des transformations

- Un modèle pour …
  - Editer des vues
  - Mais aussi pour …
    - Générer du code
    - Générer de la documentation
    - Analyser le modèle édité
- Beaucoup de points de vue lors du développement
  - Beaucoup de modèles …
  - Formaliser des liens explicites entre les divers modèles
  - Automatiser les transformations

Intégration de modèles

- Construire des liens et assurer la cohérence des modèles



jean-philippe.babau@univ-brest.fr

M2T

- Utilisations du Model-to-Text
  - Edition du modèle pour impression/diffusion
  - Rapport d'évaluation
    - Analyse du modèle
  - Génération de code
    - Traduction vers un langage cible
- Principes
  - Parcours du modèle
  - Génération de fichiers (.txt, .doc, .xmi, .html, .java …)
- Techniques
  - Basé sur des parseurs existants : XML/XSLT
  - Basé sur des langages de programmation
    - Approches à base de visiteurs (Java)
  - Basé sur des templates
    - Acceleo

Technique M2T : XML / XSLT

- Syntaxe des modèles d'entrée
  - XML
- Transformation de modèles
  - Langage XSLT
- Transformations simples
  - Pas d'opérations
  - Pas d'analyse sémantique
  - Difficile à maintenir

```
<xsl : template match =" test " >
   <xsl : with-param name =" xpm " >
   <xsl : with-param name =" ypm " >
   <xsl : choose>
   <xsl: when elt=" self::node()[isX='true']" >
      <xsl:call-template name =" XTemplate ">
      <xsl : with-param name =" xpm " select =" $xpm "/>
      </xsl : call-template>
   </xsl:when>
   <xsl:otherwise>
      <xsl:call-template name =" YTemplate ">
      <xsl : with-param name =" ypm " select =" $ypm "/>
      </xsl : call-template>
   </xsl : otherwise>
   </xsl : choose>
</xsl : template>
```

Technique M2T : API java

- Approche à base de visiteurs
  - Mécanismes de parcours du modèle
    - Chargement du modèle
    - Navigation
      - Attributs, relations
      - Itérateurs
  - Mécanisme de génération (write)

Opérations Java sur les modèles

- Un modèle = une structure de données Java
- Manipulation du modèle = opérations de manipulation des données Java
  - Création de packages Java sous EMF
    - Génération via codegen
      - Sélection *projectName.genModel* et *Generate Model Code*
    - *Création de projectName/src*
      - Package java *projectName et projectName.impl*
      - Package java *projectName.util*
- Dans les packages
  - Constructeur des éléments du modèle
  - Ensemble des méthodes *get/set* sur les attributs et les références
  - Accès aux méta-données (*projectNamePackage.java*)
- Services utilisés par les éditeurs EMF et, in fine, par les outils de transformation

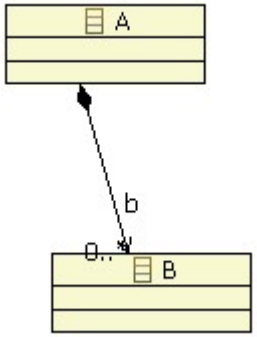
Opérations sur les modèles

- Signature des opérations Java
  - Les constructeurs d'entités
    - projectNameFactory.java
      - EClassName createEClassName()
  - Les opérations de consultation / modifications
    - EClassName.java
      - Interface Eclassname
      - attributeType getAttributeName();
      - setAttributeName(attributeType);
  - Les opérations de navigation
    - Get sur les références (non containment)
    - EClass eClass();
    - EClassPere eContainer();
    - EList<EClassName> getReferenceName();

Opérations sur les modèles

- *Opérations génériques*
  - EObject : super-type de toutes les entités manipulées
  - Toutes les méthodes de EObject …
  - La comparaison
    - boolean EcoreUtil.equals(eObject1,eObject2);
  - La copie
    - EObject EcoreUtil.copy(eOobject1);
  - Parcours de liste EList
    - Méthode iterator()

```
for (Iterator<B> iterator = a.getB().iterator(); iterator.hasNext();) {
    B b = (B) iterator.next();
    // User Code
}
```



Traitement Java des modèles

- Utilisations possibles
  - Vérification du modèle
    - Parcours et opérations de vérification
  - Génération de code
    - Parcours et génération de fichier texte
  - Transformation de modèles
    - Chargement du méta-modèle cible
    - Génération d'un modèle source

- Mais cela reste une solution orientée code
  - À quoi sert de faire de la méta-modélisation ?
  - À quoi sert de faire la formalisation (OCL) ?
  - Retour arrière
    - Modélisation Ecore (orientée modèle)
    - Transformation Java (orientée code)

Technique M2T : templates

- Approche à base de templates
  - Un template = une structure à base de textes à générer et de meta-données
  - Structure proche de la structure du code généré
    - Plus intuitif

Templates Acceleo

- Accès aux éléments du méta-modèle
  - Intègre OCL et des extensions à OCL
  - Langage basé sur JET

- Syntaxe Acceleo
  - Déclarations

```
[comment encoding = UTF-8 /]
[module modName ('http://fr.ubo.name.myMetamodel ')/]
[template public transfoName(rc : EClassRoot) { OCLvariable declarations } ]
[comment @main/]
[file ('report_'.concat(rc.name).concat('.txt'),false)]
    texte
[/file]
[/template]
```

file : fichier généré

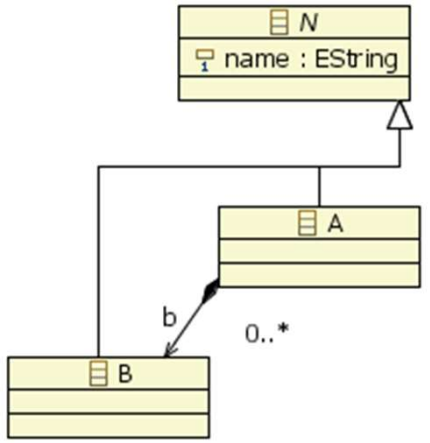
- rc possède un attribut name
- Si l'attribut name de la EClassRoot s'appelle a1 pour le modèle considéré
  => Création du fichier report_a1.txt qui contient la ligne « texte»

Rédaction des templates Acceleo

- Corps du [file] … [/file]
  - Texte constant à générer
    - A présenter tel quel
    - Attention aux retour à la ligne et aux tabulations …
  - Métadonnées et structures
    - Utilisation des séparateurs [… /]
  - Commentaires

    [comment leCommentaire/]
  - Vérification de la syntaxe à l'écriture
    - Respect du méta-modèle
      - Accès aux attributs existants
  - Complétion à l'écriture
    - Après [

Templates Acceleo



- Corps du template : parcours du modèle
  - Parcours multiple du modèle possible
  - Utilisation d'OCL

  - Accès aux attributs
    - Nom de l'attribut ou de la référence
      name , b, …
    - Accès aux éléments
      - eAllContents()

  - Affichage de l'attribut name de type String des éléments contenus par la EClassRoot

[b.name/] [comment liste des attributs name de tous les éléments contenus dans A /]

  - Affichage de l'attribut name de type String des éléments de type B

[eAllContents(B) .name/] [comment liste des attributs name de tous les éléments de type B/]

Templates Acceleo : structures

– Structure répétitive

[for (elt : typeElt | expression)]
for-statement (expression)
[/for]

- Décorations pour structure répétitive

[for (Sequence{1, 2, 3}) before (‘ sequence: ') separator (', ') after (';')] [self/][/for]

Génère

sequence: 1, 2, 3;

– Structure conditionnelle

[if (condition)]
true-statement
[else]
false-statement
[/if]

[if (condition)]
true-statement
[/if]

Template Acceleo

- Quelques opérations utiles
  - Ecriture d'une chaine toString()

```
[a.toString()/]
-- renvoie une description de a : " package_de_
-- a.type_de_a(attribut1 : valeur1, attribut2 : valeur2, ...) "
```

  - Affichage du nombre d'objets d'une collection (ici les références b de a) size()

```
[a.b->size().toString()/]
```

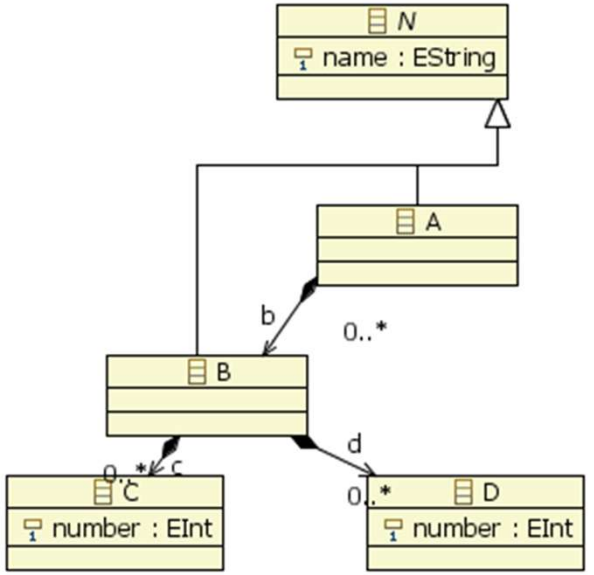
  - Numéro de l'itération i

```
[for (eAllContents(B))][if (i>1) ][name/] est le [i/]eme[else][name/] est le 1er [/if]
[/for]
```

Documentation sur Acceleo :
http://help.eclipse.org/helios/index.jsp?topic=/org.eclipse.acceleo.doc/doc/overview.html

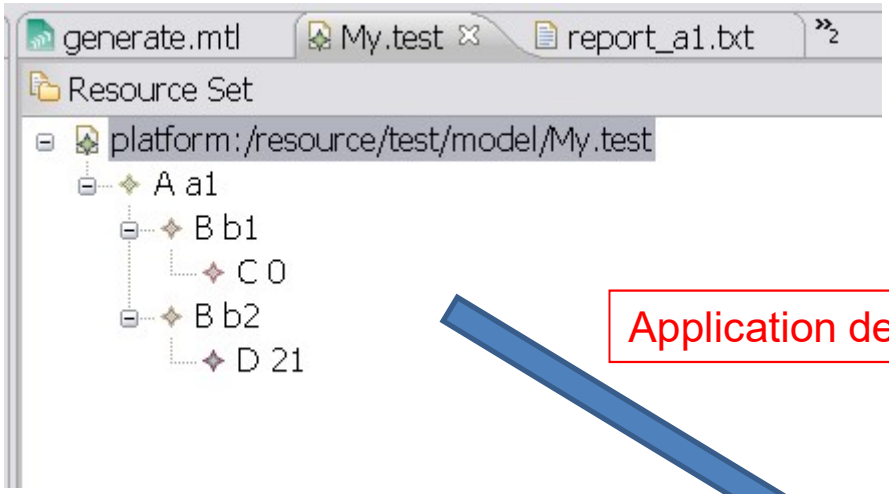
Exemple

```
[module generate ('http://fr.ubo.babau.test')/]
[template public generate(a : A)]
[comment @main /]

[file ('report_'.concat(a.name).concat('.txt'),false)]
--  debut texte généré
rapport sur [a.name/]
[comment  parcours des noms de b /]
[for (b) ]
[if (name = 'b1')]
                [name/] trouvé
[else]
                [name/] est différent de b1
[/if]
[/for]
[comment parcours des numéros de C et D /]
[for (b)]
[for (c) ]
                pour [name/], C de numéro [number/]
[/for]
[for (d) ]
                pour [name/], D de numéro [number+4 /]
[/for]
[/for]
[/file]
```



jean-philippe.babau@univ-brest.fr

Exemple



Application de la transformation generate

report_a1.txt

```
-- debut texte généré
rapport sur a1

        b1 trouvé
        b2 est différent de b1

        pour b1, C de numéro 0
        pour b2, D de numéro 25
```

Intégration de code utilisateur

- Le code généré n'est pas complet
  - Squelette de code
  - Portions de code

- Balises pour encadrer le code utilisateur

```
// [protected ('unTexteUnique')]
// user code to add
// [/protected]
```

- Comportement
  - Tout ce qui est entre les balises **n'**est ajouté **qu**'à la **première** génération
  - unTexteUnique doit être unique pour chaque zone généré
    - C'est l'identifiant de la zone à ne pas écraser pour Acceleo
    - On peut ajouter une partie variable pour bien différencier les zones générées
    - C'est un objet de type string

Lien avec Java

- On peut lancer le générateur de code depuis une application java
  - Génération d'une classe java avec le même nom que le template Acceleo
  - La classe java contient un main qui permet de lancer la génération de code
- On peut intégrer du code java dans acceleo

```
invoke (String class, String method, Sequence(OclAny) arguments ) : OclAny

invoke('java.lang.String', 'toUpperCase()', Sequence{root.name})
```

M2M

- Utilisations du Model-to-Model
  - Abstraction : extraire un point de vue
    - Modèle de la structure, modèle de QoS, …
  - Fusion de modèles : intégration de divers aspects
    - Intégration de politiques de sécurité, …
  - Refactoring : améliorer la structure du modèle
    - Renommage
    - Application de design pattern
    - Simplifications (factorisation d'attributs dans une classe ancêtre, …)
  - Raffinement : enrichir et préciser le modèle
    - Intégration d'aspects technologique (MDA)
      - Plateformes d'exécution et de communication
  - Traduction et interprétation : passage d'un monde à un autre
    - Changement de version (L1 to L2), …

- Mise en œuvre
  - Analyse syntaxique et sémantique du modèle
  - Génération d'un nouveau modèle défini lui-même par un méta-modèle

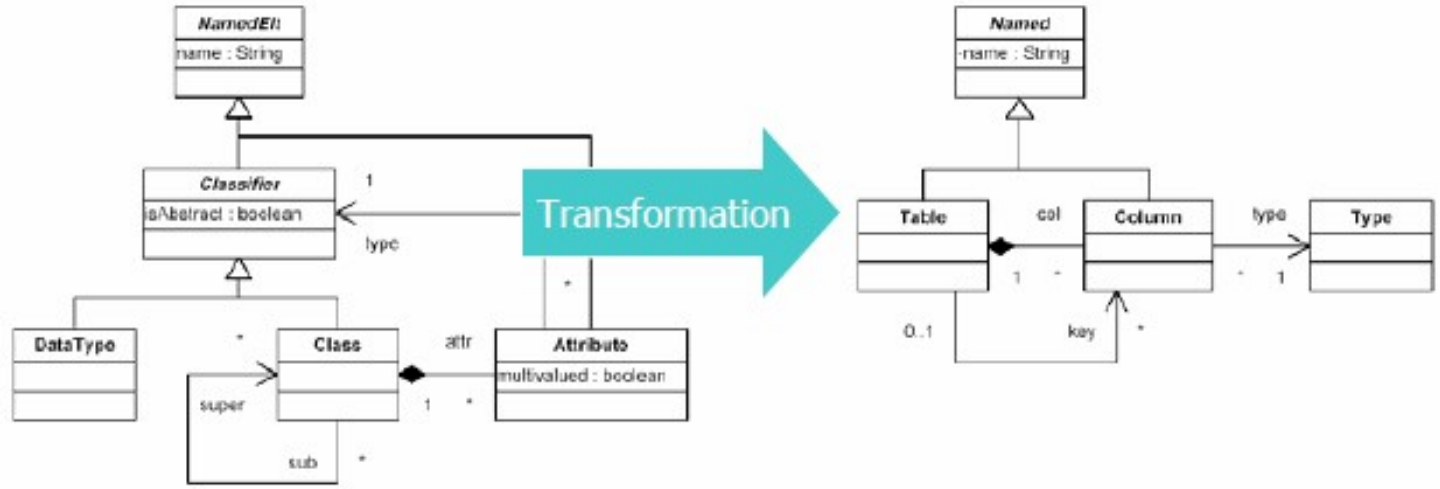
Tansformation M2M

- Principes
  - Meta-model(modelA) to meta-model(modelX)
  - Techniques de génération
    - Parcours du modèle
    - Règles de génération

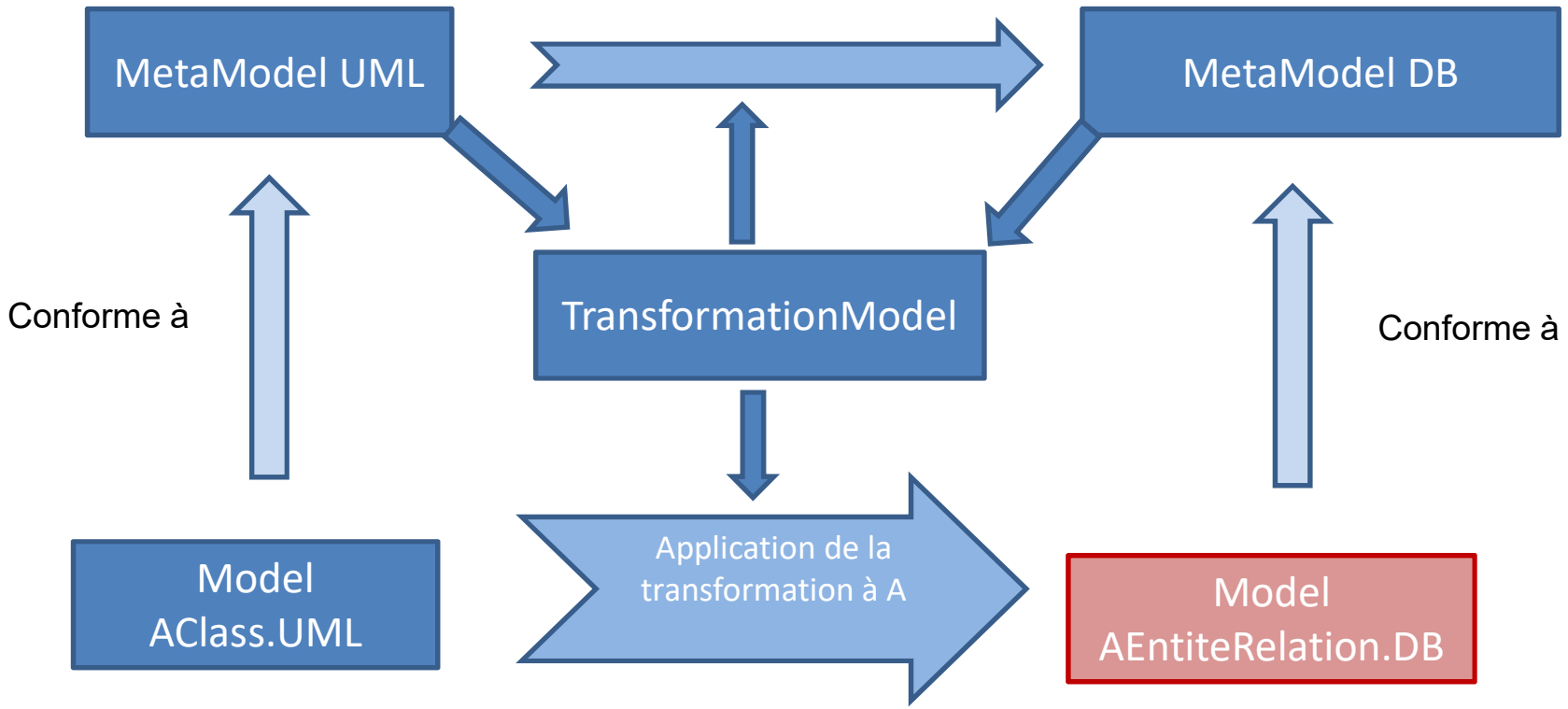
- Approches
  - Approche QVT (Query-View-Transformation)
    - Requête -> vue -> modèle généré
  - Langages de transformation
    - Déclaratif : ATL
    - Orienté objet : Kermeta
    - Dédié : Modif

Exemple

- UML 2 DB



Principes


MetaModel UML
MetaModel DB
Conforme à
TransformationModel
Conforme à
Model
AClass.UML
Application de la
transformation à A
Model
AEntiteRelation.DB

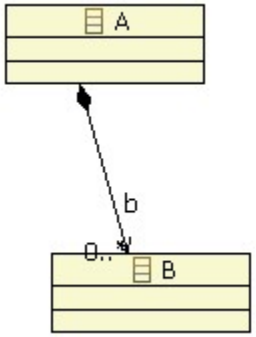
Traduction EClass -> EClass

- Principe

  Un concept du métamodèle de départ « est transformé en » un concept du métamodèle d'arrivée

- Exemple copy.atl

```
module copy;
create out : abMM from  in: abMM;
rule keepB {
        from
                b : abMM!B
        to
                copy_of_b : abMM!B()
        }
rule keepA {
        from
                a : abMM!A
        to
                copy_of_a : abMM!A(b<-a.b)
        }
```



jean-philippe.babau@univ-brest.fr

Principes d'ATL

- Langage déclaratif
  - Liste de règles (rule) de transformations
  - Une règle définit comment un concept source est transformé vers un concept cible
  - Utilisation d'opérations pour décrire la transformation
- Opérations
  - Basées sur OCL
  - Requêtes et opérations sur le modèle
- Données
  - Type OCL

ATL

- Structure d'une transformation
  - Déclaration du module
    - Header section
  - Import de librairies
    - Import section
  - Opérations
    - Helpers
  - Règles de transformation
    - Rules

ATL : header section

- Déclaration du module

```
module moduleName;
create outputModel : outMetaModel from inputModel : inMetaModel ;
```

Le nom du module est obligatoirement le même que le nom du fichier

- Import de librairies

```
uses atlLibraryName ;

uses string;
```

ATL : opérations

- Opération : helper
  - Défini pour un context donné (cf. cours OCL)
    - Opération pour une *EClass* donnée
  - Un context ne peut pas être défini par une collection

```
helper [ context contextName ] def : helperName(parameters) : returnType= atl_expression;
```

```
[ non_obligatoire ]
```

Exemples

```
helper context Integer def : double() : Integer = self * 2;
```

```
helper context MM!eclassRoot def: ConcatNames():String =
        self.oneEclass->select(e| e.name<>'')->collect(e|e.name+'_')->sum();
```

ATL : exemple

- Exemple issu de la documentation ATL

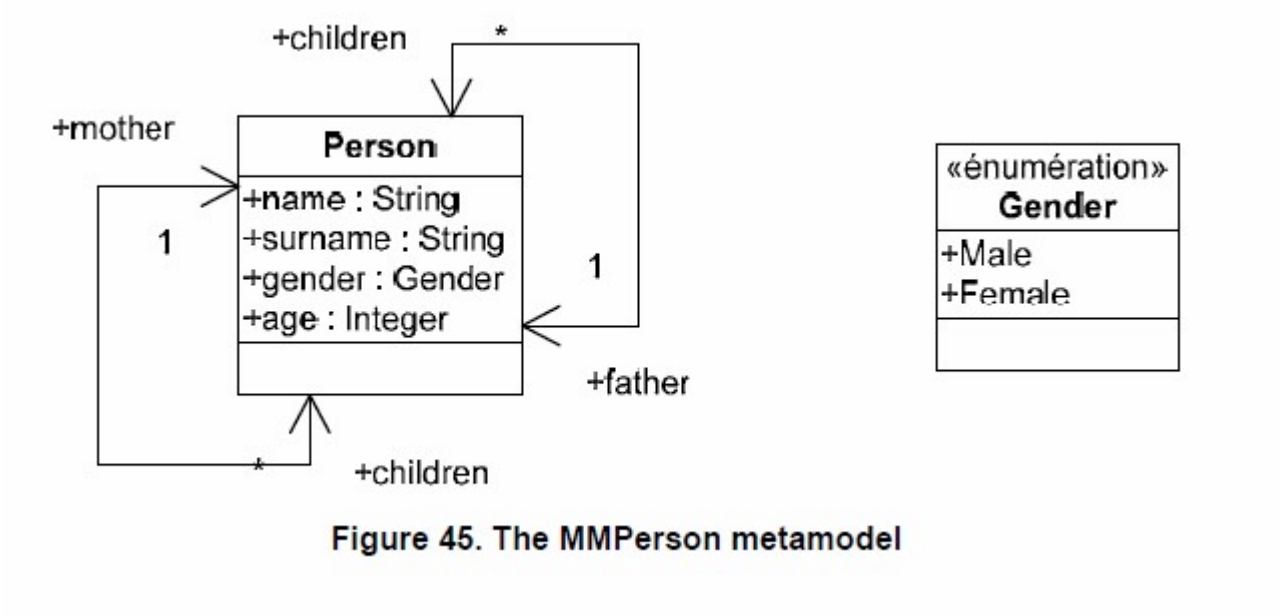


Figure 45. The MMPerson metamodel

```
helper def : getYoungPersons (s : Set(MMPerson!Person), age : Integer) :  Set(MMPerson!Person) =
        s->select(p | p.age < age);
```

ATL : données

- Données : helper
  - Pas de paramètre
  - Référence à la donnée dans les opérations
    - Si pas de contexte : thisModule.dataName

helper [context contextName] def : dataName : dataType = initialValue ;

Exemple

helper def: elementNumber : Integer = 0 ;

… thisModule.elementNumber

ATL : expressions et données

- Données
  - Types OCL
    - Boolean : true, false
      - and , or , xor, not
    - Integer et Real
      - * + - / max(), min(), div(), abs()
      - Pour exprimer –x : 0 - x
      - Opérations ATL sur Real
        - » cos() sin() tan() atan() asin() exp() log() sqrt() toDegrees() toRadians()
    - String : 'Hello'
      - size() + concat() substring(lower,upper) toInteger regexReplaceAll(regex, replace)

- Expressions
  - Expression OCL (cf. cours OCL)
    - Différent : <>
  - Opérations ATL spécifiques de retour pour String
    - chaine.writeTo(fileName) -- écriture de chaine dans le fichier fileName
    - chaine.println() -- affichage de chaine sur la console

ATL : expressions et données

- Déclaration de variable dans une expression
  - OCL déclaration

```
let var_name : var_type = var_init_expression in expression
```

- Exemple
  - SetNumber renvoie l'attribut number pour un élément de type C, si il est défini et non nul, le paramètre id sinon

```
helper context MM!C def: SetNumber( id :Integer): Integer =

Let numberDefault : Integer = id in

if (self.number=0 or self. number.oclIsUndefined()) then numberDefault else self.number endif;
```

ATL : règles de génération

- Règle de génération (1 -> 1)

précondition sur inVar pour application de la règle

```
rule rule_name {
    from
        inVar: inMetaModel!EClassNameIn [( condition )]
    to
        outVar : outMetaModel!EClassNameOut ( initialisation)
    [do
    {
        statements
    }]
```

Intialisation des attributs de outVar

- Initialisation des attributs
  - Directement à partir des attributs de inVar
  - Par appel d'un helper

```
( attribute_name <- expression  [, listeInitialisations] )
```

ATL : exemple

- Exemple issu de la documentation ATL

Figure 44. The MMAuthor metamodel

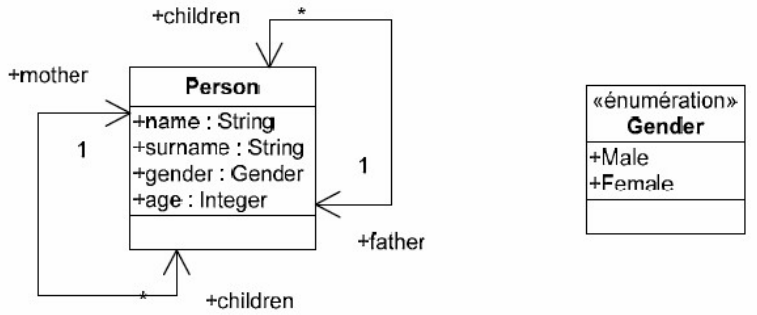


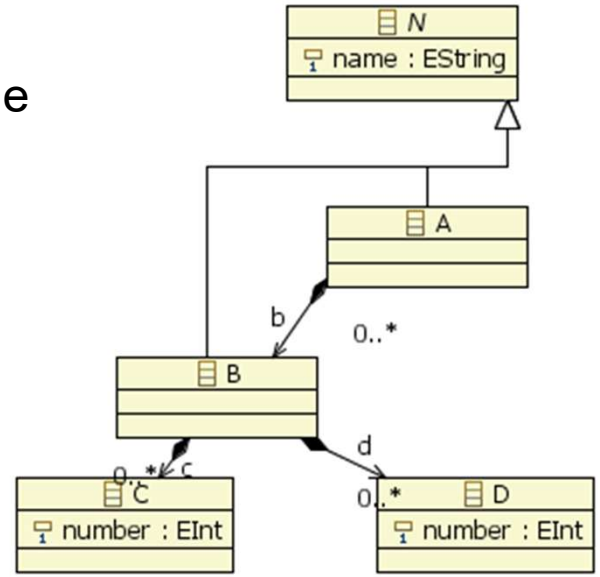
Figure 45. The MMPerson metamodel

```
rule Author2Person
{
    from
        a : MMAuthor!Author
    to
        p : MMPerson!Person (
            name <- a.name,
            surname <- a.surname )
}
```

Les attributs age et gender ne seront pas initialisés pour p

ATL : exemple

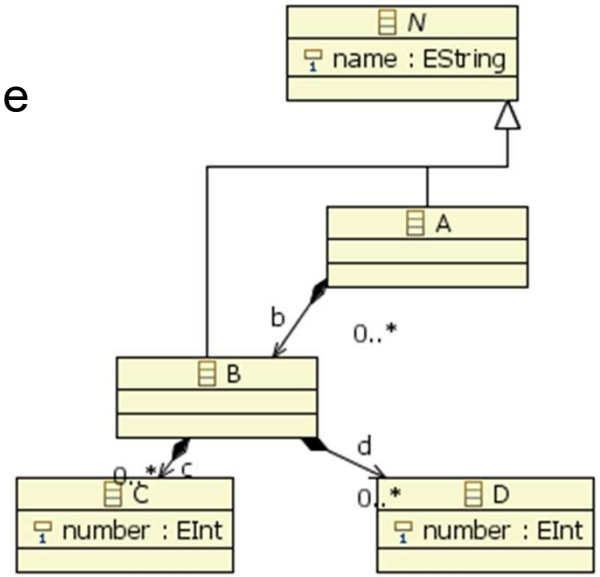
- Exemple
  - D'un métamodèle abcd vers lui même




```
…
rule copyC
{
      from
            cPre : abcd!C
      to
            cPost : abcd!C (
                  number<- cPre. SetNumber( 1 )
                  )
}
```

ATL : exemple

- Exemple
  - D'un métamodèle abcd vers lui même




```
…
rule copyC
{
        from
                cPre : abcd!C (number > 0 )
        to
                cPost : abcd!C (
                        number<- cPre.number
                )
}
```

ATL : exemple

- Exemple issu de la documentation ATL

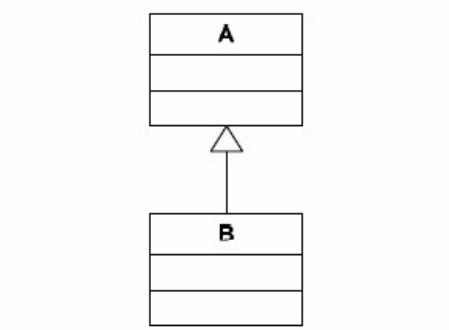


Figure 8. Simple inheritance case

```
rule Author
{
from  a : MM!A
      ( a.oclIsTypeOf(MM!A)  )
to
…}
```

Règle appliquée seulement aux A, et pas aux B

ATL : règles de génération

- Principes
  - Langage déclaratif
    - Règle appliquée à chaque élément de type EClassNameIn, vérifiant condition
  - Mise en œuvre des règles
    - Une règle pour chaque élément à considérer
    - Remplir tous les attributs de out_var à initialiser
  - Ordre de déclaration des règles
    - Libre
  - statements : partie impérative

ATL : partie impérative

- Statements
  - Module impératif
  - Opérations effectuées après chaque application de la règle
  - Affectation target <- source;
  - Conditionnelle if(*condition*) { *statements*} [else { *statements*}]
  - Répétition for(*iterator in collection*) { *statements* }

- Exemple

```
helper def: elementNumber : Integer = 1;

rule Numeroter {
    from … to …
    do
    {
        thisModule.elementNumber <- thisModule.elementNumber + 1 ;
    }
}
```

ATL : règles de génération

- Règle de génération (0 -> 1)

```
rule name ( ParameterList) {
    to
        outVar : outMetaModel!EClassNameOut ( initialisation)

    [do
    {
        statements
    }]
```

- Principe
  - Génération par création d'élément
  - Appel via la partie impérative d'une rule (1-1) ou via une autre rule (0-1)
    - thisModule.ruleName();

Fusion de modèles

- Une transformation est issue de 2 modèles
  - 2 métamodèles

```
module moduleName;
create outputModel : outMetaModel from IN1 : in1MetaModel1, IN2 : in2Metamodel2 ;
```

- Préfixage par le nom du métamodèle

in1MetaModel1!Econcept1 et in2MetaModel2!Econcept2

- 1 modèle origine pour les transformations, l'autre modèle utilisé pour les opérations de transformation (helper défini sur le 1er métamodèle)

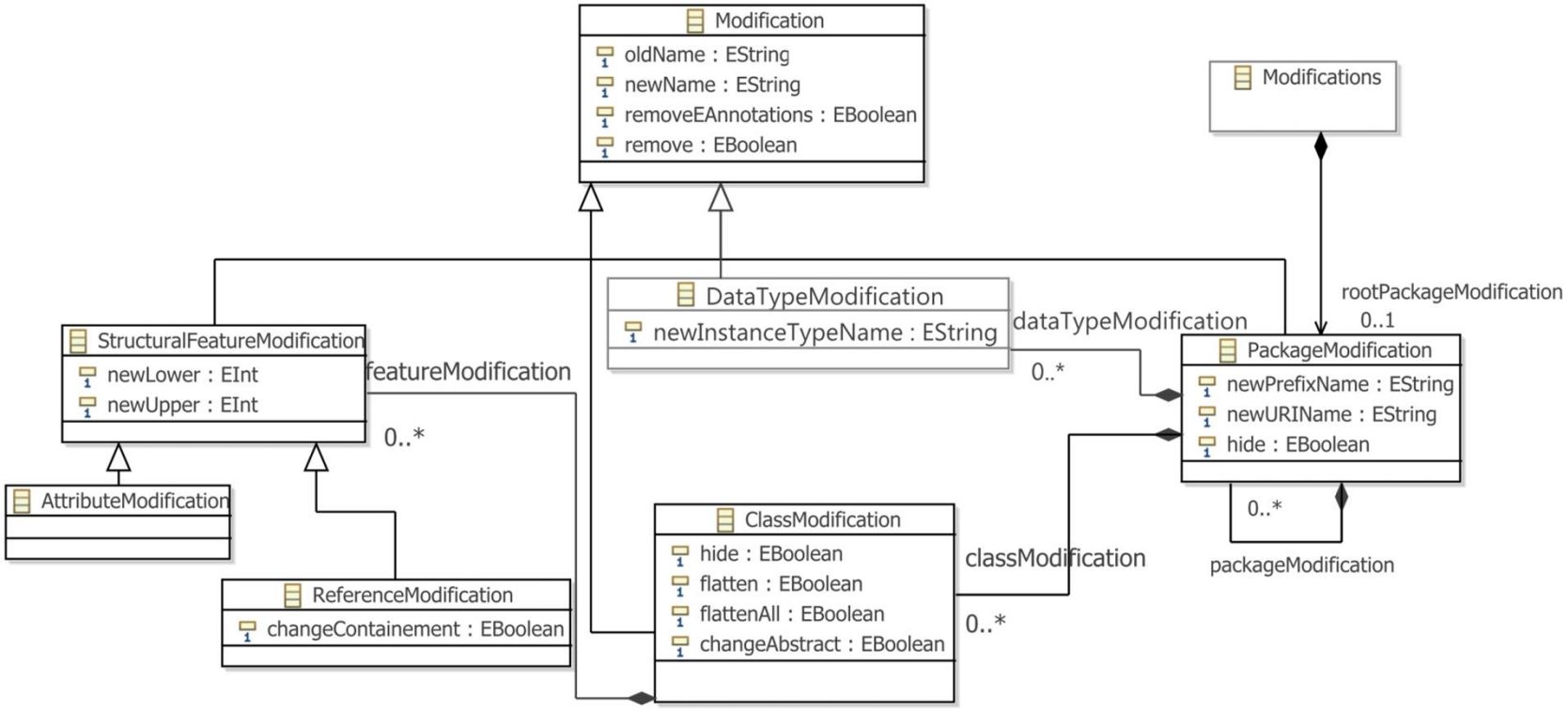
```
helper def : testA(att : in1MetaModel1!A ):Boolean = … in2MetaModel2!ClassRoot.allInstances() …

rule transfoA
{
from  a : in1MetaMode1l!A (thisModule.testA(a))
To b : in2MetaModel2!B (…)
}
```

Transformations ATL

- ATL
  - Déclaratif
  - Une seule passe sur le modèle
    - Une élément ne peut pas être utilisé 2 fois par 2 règles de générations
  - Modèle de transformation
- Fusion de modèles
  - Le métamodèle d'entrée contient plusieurs métamodèles
- Possibilité d'avoir plusieurs sorties
- Pas de génération des attributs dérivés
- Références doubles (EOpposite)
  - Génération d'une seule référence

Modif

- Liste d'opérateurs associés à des concepts ecore



jean-philippe.babau@univ-brest.fr

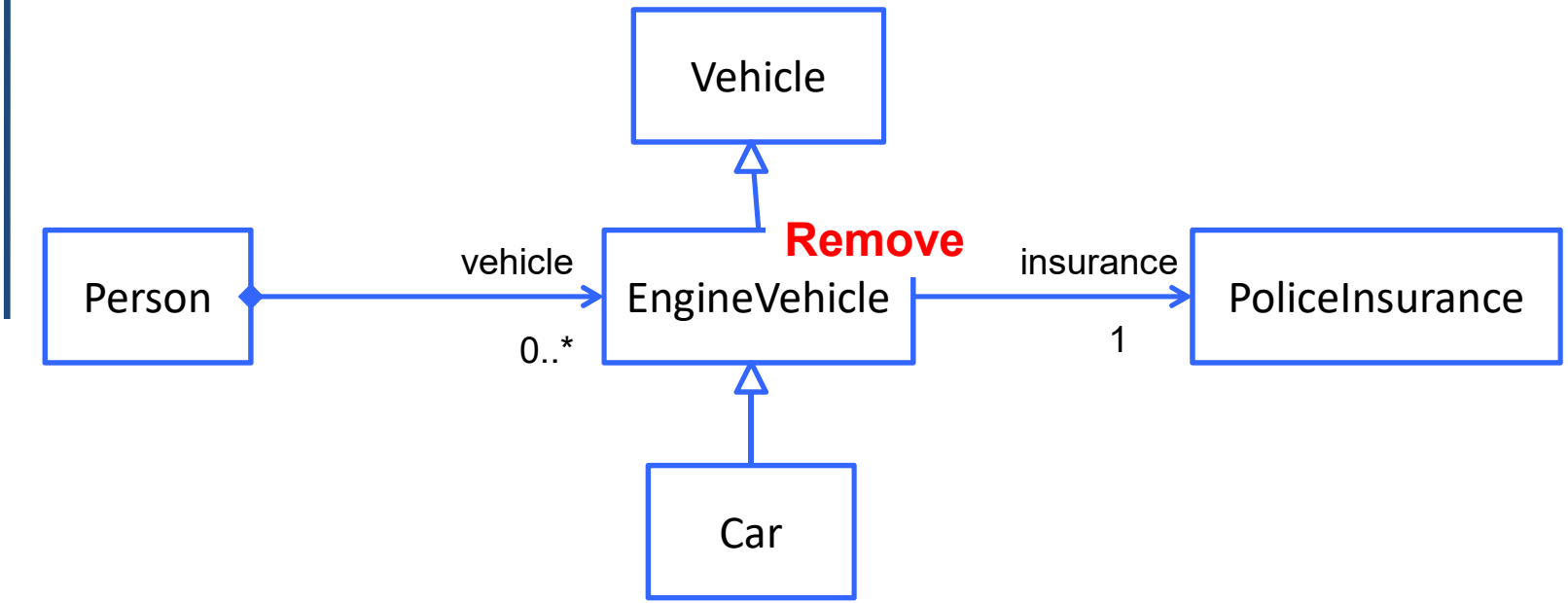
Opérateurs Modif

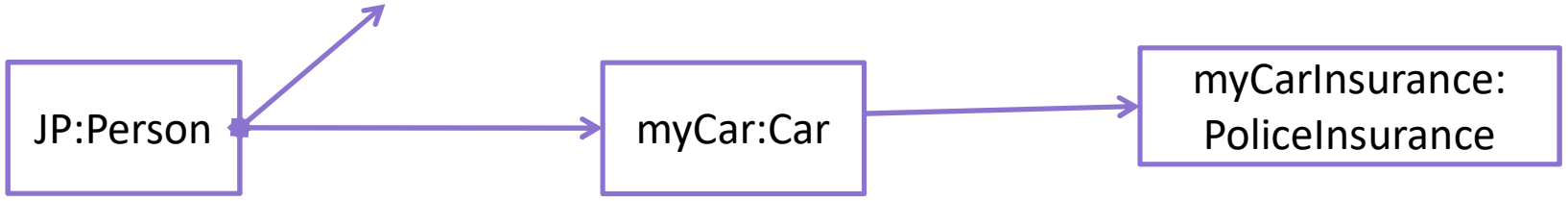
- Opérateurs “CRUD” et refactoring
- Opérateurs basiques
  - Rename, Remove
  - Flatten EClass
  - Change lower and upper bounds
  - Change abstract property of Eclass
  - Change containment property of EReference
  - Add EAttribute
  - Move EStructuralFeature
- Opérateurs complexes
  - Hide
  - FlattenAll
  - AddRootClass, …

Comportement des operators

- Pas de modification
  - Au niveau métamodèle : copie des concepts
  - Au niveau du modèle : copie des instances
- Rename
  - Au niveau métamodèle : renommage des concepts
  - Au niveau du modèle : copie des instances
- Remove
  - Au niveau métamodèle : suppression des concepts et de **tous les concepts associés**
    - EPackage : toutes les Eclass contenues
    - EClass : tous les Eatttribute, les EReference
  - Au niveau du modèle : suppression des éléments et des éléments associés
  - Différent de l'opération EMF *Delete*
    - Obtention d'un Ecore valide

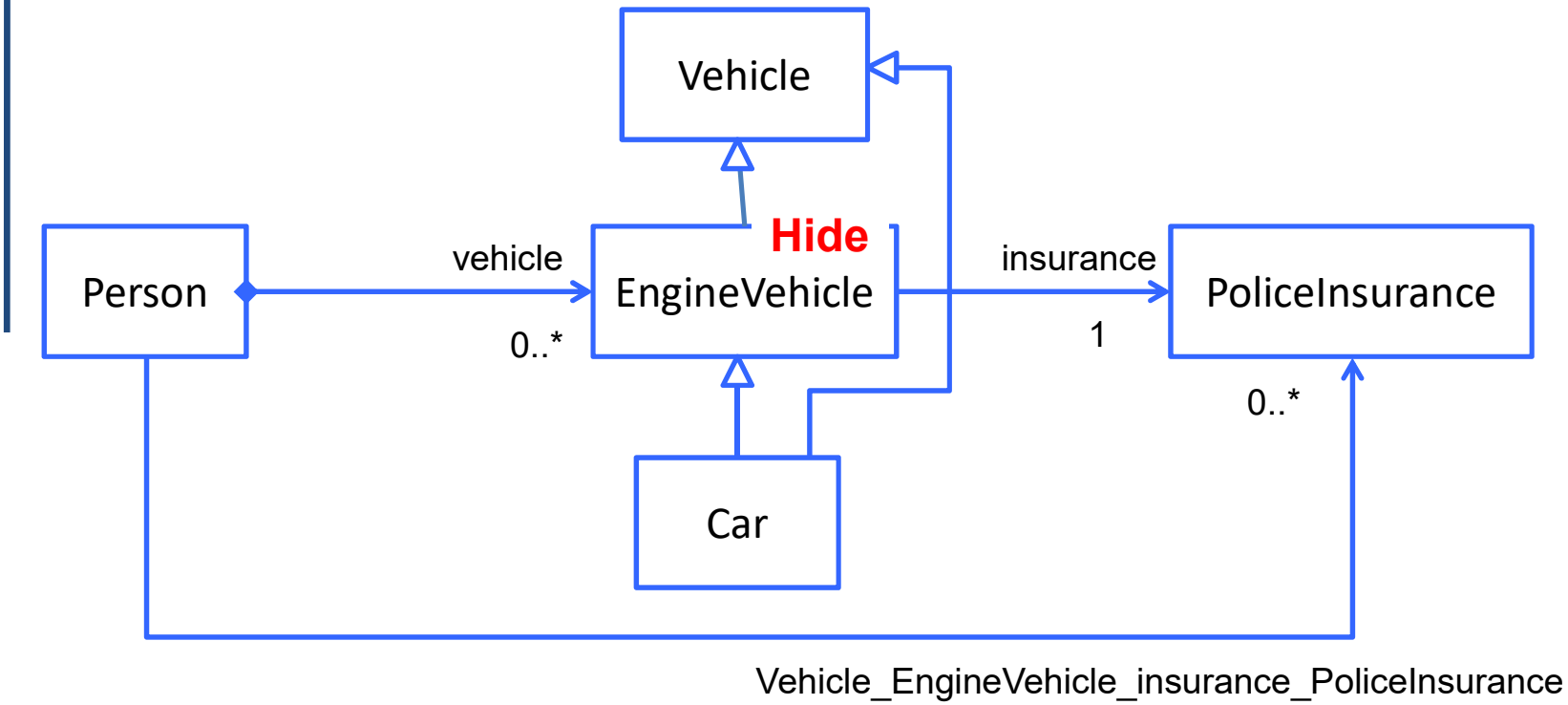
Illustration de l'opérateur remove



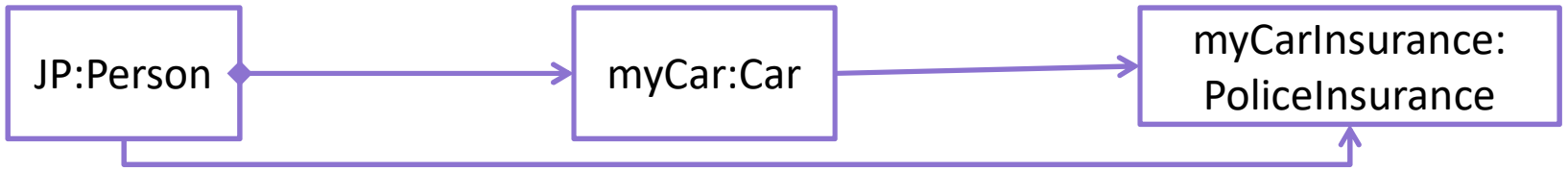


jean-philippe.babau@univ-brest.fr

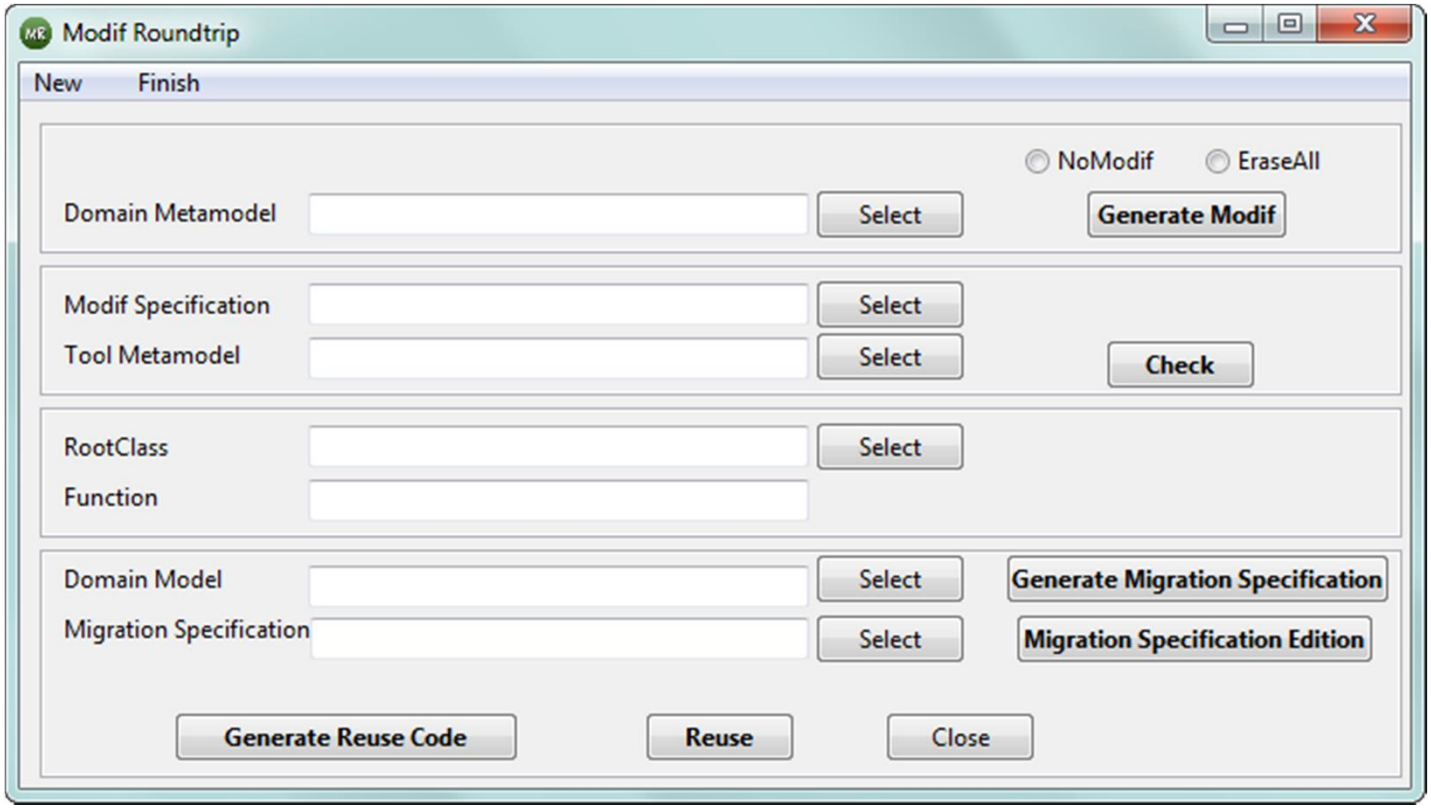
Illustration de l'opérateur hide



Vehicle_EngineVehicle_insurance_PoliceInsurance



Modif IHM



jean-philippe.babau@univ-brest.fr

Epsilon

- Famille de langages de transformation
  - EOL : manipulation de modèles EMF
  - EGL : model-to-text
    - langage à balise
  - ETL : model-to-model
    - À base de règles
    - Déclaratif
  - EVL: model validation
    - Proche d'OCL
    - Programmation des retours utilisateurs en EOL
  - Flock : model migration
  - EML : model merge
  - EWL : model refactoring
    - interactif

- http://www.eclipse.org/epsilon/

Intégration dans EMF

- Java
- Acceleo
- ATL
- Modif

Intégration de code Java

- Création d'un plugin Java
  - New Other -> Plug-In Development -> Plug-In Project
  - Project projetJavaName
  - Type de plugin
    - Hello world Command : ajout d'un menu et d'un bouton
    - Plug-In with a popup menu : ajout d'un menu contextuel par sélection du fichier

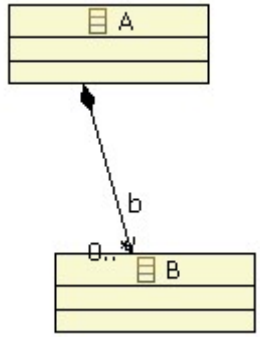
- Ajout de dépendances
  - Sélection de Manifest.MF, onglet Dependencies
    - org.eclipse.emf.ecore.xmi
    - projetEcoreName (plug-in désormais intégré)

Actions liées au popup menu

- Sélection de MANIFEST.MF
  - onglet Extensions
  - Modification du menu
    - Via le contenu de org.eclipse.ui.popupMenus
    - Ajout d'actions, modification des labels, …
  - Modification du type de fichier traité
    - Champ nameFilter
      - type de fichier sur lequel le menu est activé : *.extension
  - Lien avec les classes Java
    - Champ class liée à l'action
    - Par défaut : dans src/ package projetJavaName.popup.actions / class NewAction.java
      - Méthode selectionChanged : chargement du fichier
      - Méthode run : exécution suite à sélection du menu

Actions liées au popup menu

- Fichier NewAction.Java




```
…
import java.util.Iterator;
import org.eclipse.core.resources.IFile;
import org.eclipse.jface.viewers.StructuredSelection;
import org.eclipse.emf.ecore.resource.Resource;
import org.eclipse.emf.ecore.resource.ResourceSet;
import org.eclipse.emf.ecore.resource.impl.ResourceSetImpl;
import org.eclipse.emf.common.util.*;
import org.eclipse.emf.ecore.util.Diagnostician;

import metaModelName.metaModelNamePackage;          // nom du package généré pour le métamodèle
import metaModelName.A;                             // nom du package généré pour la EClass Root

public class NewAction implements IObjectActionDelegate {

private IFile  theSelectedFile;
private A objectRoot;                               // déclaration de la racine

…
```

Création et chargement du modèle

```
public void selectionChanged(IAction action, ISelection selection) {
URI uri;
Resource res;
ResourceSet resourceSet = new ResourceSetImpl();

if (selection instanceof StructuredSelection) {
      StructuredSelection currentSelection = (StructuredSelection)selection;
      Iterator it = currentSelection.iterator();
      while(it.hasNext()) {
            theSelectedFile = (IFile)it.next();
      }}
uri= URI.createURI(theSelectedFile.getFullPath().toString(),true);

resourceSet.getPackageRegistry().put(metaModelPackage.eNS_URI,metaModelPackage.eINSTANCE);
res = resourceSet.getResource(uri, true);

objectRoot = (A) res.getContents().get(0);
}
```

Charge le fichier sélectionné

Récupère l'URI du fichier

Charge le méta-modèle

Charge le modèle à partir de la racine

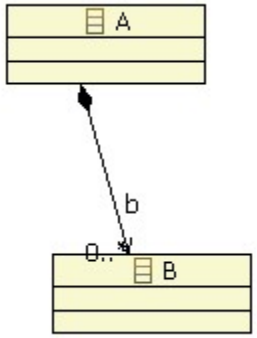
jean-philippe.babau@univ-brest.fr

Traitement du modèle

```
public void run(IAction action) {

Diagnostic diagnostic = Diagnostician.INSTANCE.validate(objectRoot);

if (diagnostic.getSeverity() == Diagnostic.OK)
{ // user code
      InfoA getterA= new InfoA();
      MessageDialog.openInformation(shell, "TestJavaEMF Plug-in" , getterA.numberOfB(objectRoot) );}
else {
      MessageDialog.openInformation(shell,"TestJavaEMF Plug-in","model error");}
}
```

Vérifie le modèle
(contraintes OCL incluses)

```
package toolsA;

import metaModelName.A;

public class InfoA {
public String numberOfB(A a) {
      return (String.valueOf(a.getB().size()));  }
}
```



jean-philippe.babau@univ-brest.fr

Acceleo et ATL

- Installation du logiciel dans Eclipse (si non présent)
  - Menu Help -> Install Modeling Components
    - Cocher Acceleo
    - Cocher ATL
  - Relancer Eclipse
- Export du plugin où se trouve le méta-modèle

Créer un projet de template Acceleo

- Basculer en perspective Acceleo
  - Window / OpenPerspective / Other / Acceleo
- Ouvrir un projet Acceleo qui crée un fichier de transformation
  - File/ New/ Acceleo Project
    - Project Name repTransfo
    - Cliquez sur Next
  - création de modName.mtl dans le répertoire repTransfo/src/repTransfo.main et
    - Parent Folder repTransfo/src/repTransfo/main
    - Module name modName
    - Metamodel URIs : http://fr.ubo.mde.Name.projectName (URI du projet)
    - Template name transfoName
    - Type EClassRoot
    - Cochez Generate File et Main Template
- Editer la transfo
  - Fichier modName.mtl

Exécuter un template Acceleo

- Lancer la transformation depuis Eclipse
  - Sélection du fichier contenant la transformation modName.mtl
    - Runs As/ Launch Acceleo Application
    - Configuration de l'exécution
      - » Project : repTransfo
      - » Main Class: transfoName.main.ModName
      - » Model: fichier source ou se trouve le modèle
      - » Target : répertoire ou va se trouver le fichier généré
- Re-lancer la transfo
  - Runs / Run History / modName

Exécuter un template Acceleo

- Lancer la transfo dans une application Java

- Code Acceleo

```
[template public generateLocalWPS(aWPS : LocalWPS)]
[comment @main/] … [/template]
```

- Code Java

```
LocalWPS aWPS; File folder;
List<String> arguments = new ArrayList<String>();
GenerateLocalWPS generator;
try {
        generator = new GenerateLocalWPS(aWPS, folder, arguments);
        generator.doGenerate(new BasicMonitor());
} catch (IOException e) {
        e.printStackTrace();
}
```

Intallation Xtext / Acceleo

- Intégration du bridge « Xtext with Sirius 4.0 » et « Acceleo »
  - Help/Install New Software…
  - http://www.obeo.fr/download/release/designer/9.0/community/latest/repository
  - **Feature**: Obeo Designer Community Edition – Extensions/ Acceleo Core SDK
  - **Feature**: Obeo Designer Community Edition – Extensions/ Sirius Integration with Xtext
  - Next/Next/I accept/Finish/Yes (restart)

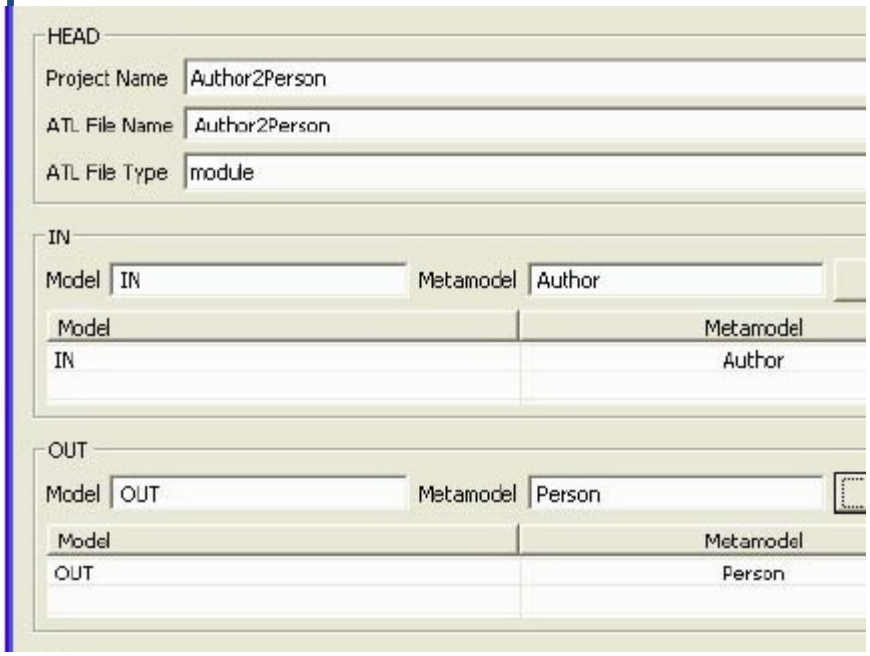
Intégration Acceleo / Xtext

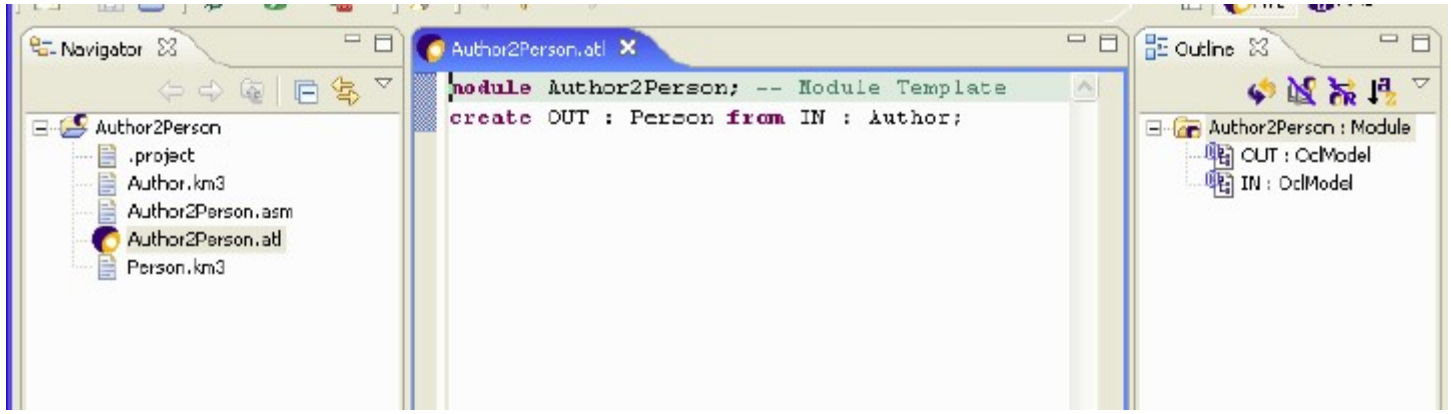
- **Enlever** * @generated **dans les commentaires de la méthode**
  public void registerPackages(ResourceSet resourceSet) { …
- **Ajouter la ligne suivante à la fin de la méthode**
  resourceSet.getPackageRegistry().put(
  ModelNamePackage.eNS_URI, ModelNamePackage.eINSTANCE);
- **Enlever** * @generated **dans les commentaires de la méthode**
  public void registerResourceFactories(ResourceSet resourceSet) { …
- **Ajouter la ligne suivante à la fin de la méthode**
  ModelNameStandaloneSetup.doSetup();
- UI launcher : sélectionner le projet Acceleo
  - Pas d'export du UI launcher

Configuration ATL

- Basculer en perspective ATL
  - Window / OpenPerspective / Other / ATL
- Créer un projet ATL New -> ATL Project
  - Créer un fichier de transformations
    - Sélectionner le projet, puis menu New -> ATL File
    - Container ProjectName
    - ATL Module Name transformationName
    - IN et OUT model modelInName Metamodel metaModelName ADD
  - Génération du fichier Transfo1.atl
    - Editions des transformations

ATL perspective



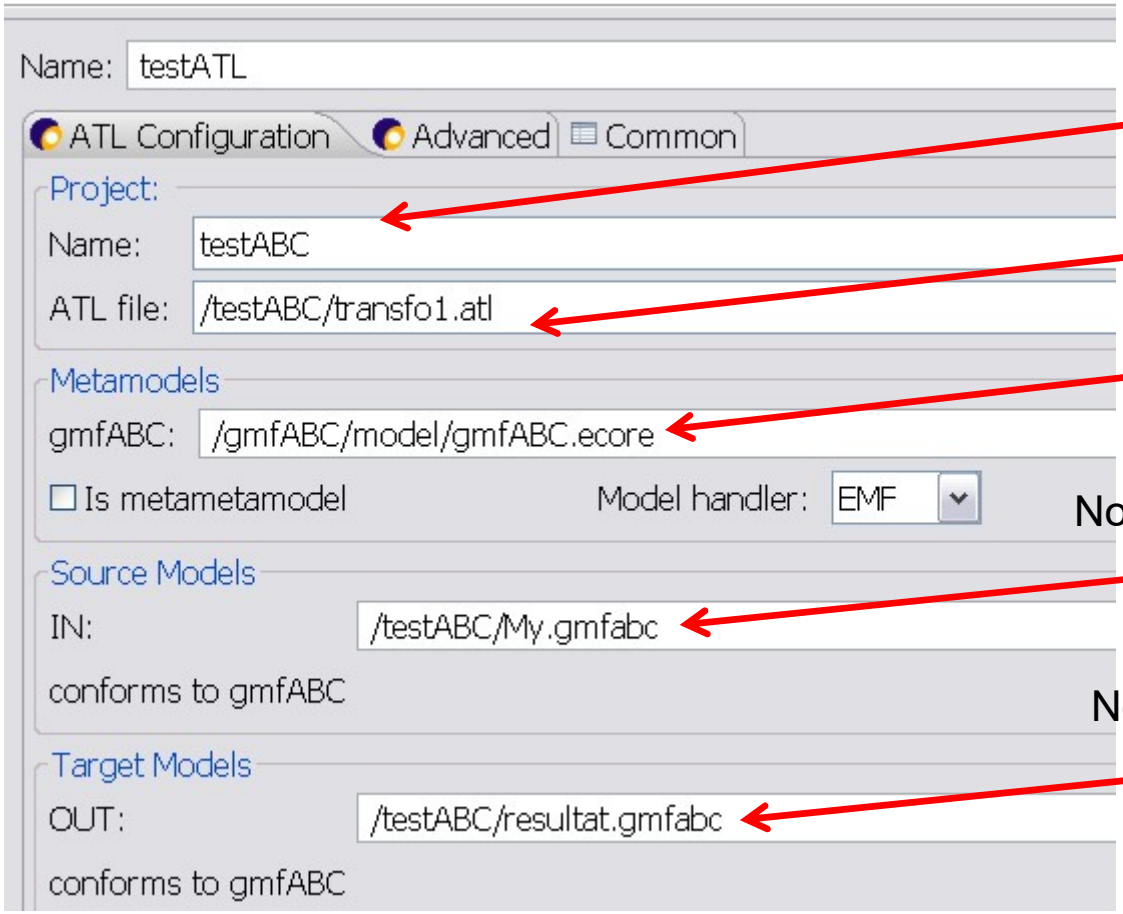


jean-philippe.babau@univ-brest.fr

Exécution ATL

- Exécution Run-> Run configurations …
  - Sélectionner ATL Transformation et New
  - Name aName
  - Project Name projectName
  - Project ATL file fileName.atl
  - Metamodels
    - Sélectionner via le workspace les métamodèles concernés par la transformation
  - Source Models
    - IN fichier contenant le modèle à considérer
  - Target Models
    - OUT nom du fichier résultat
  - Onglet Advanced
    - cocher la case Allow inter-model references
  - Apply, puis Run

ATL perspective



Nom du projet

Nom de la transformation

Nom du ou des métamodèles

Nom du modèle à traiter en entrée

Nom du modèle produit en sortie

jean-philippe.babau@univ-brest.fr

Modif

- Tout ce que vous avez voulu savoir sur Modif est sur

http://lab-sticc.univ-brest.fr/~babau/modif/modif.htm

https://github.com/vallejoco/ModifRoundtrip-project

Références utilisées pour faire ce cours

- Eclipse
  - http://www.eclipse.org/
- Cours de Jean-Marc Jézéquel
  - http://www.irisa.fr/prive/jezequel/enseignement/
- Documentation Acceleo
  - http://www.acceleo.org/pages/accueil/fr
- ATL User Manual
  - http://www.eclipse.org/m2m/atl/doc/
  - /home2/commun_depinfo/EMT/langages

- ATL
  - http://wiki.eclipse.org/ATL/Tutorials_-_Create_a_simple_ATL_transformation
  - http://wiki.eclipse.org/ATL/User_Guide_-_The_ATL_Language#Iterating_over_collections
  - Explication des règles ATL de manière détaillée avec des exemples :
  - http://wiki.eclipse.org/ATL/User_Guide_-_The_ATL_Language#ATL_Rules
  - Une liste des erreurs les plus fréquente commises dans l'utilisation d'ATL et leur solution :
  - http://wiki.eclipse.org/ATL_Language_Troubleshooter

- Acceleo
  - http://www.acceleo.org/pages/un-premier-generateur/fr
  - http://www.lifl.fr/~dumoulin/enseign/2010-2011/pje/cours/2.generationCode/2.acceleo-mtl.pdf
  - http://www.acceleo.org/pages/accueil/fr
  - http://www.lifl.fr/~dumoulin/enseign/2014-2015/pje/cours/4.generationDeCode/2.acceleo-mtl.pdf