Génération de code à partir de modèles

Vincent Aranega

vincent.aranega@genmymodel.com
Axellience

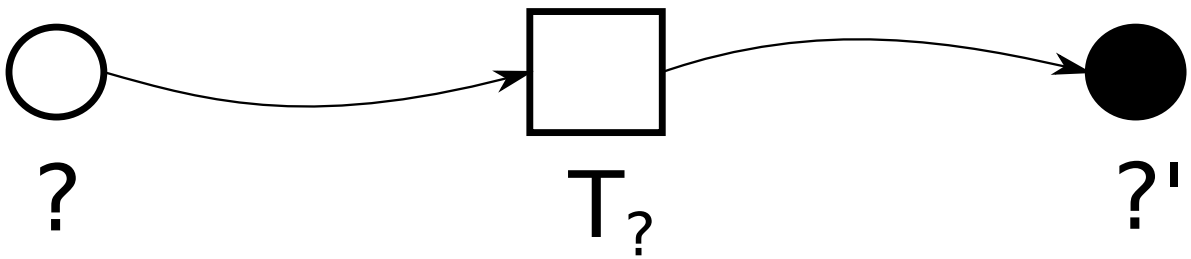
Les déjeuners technologiques

–

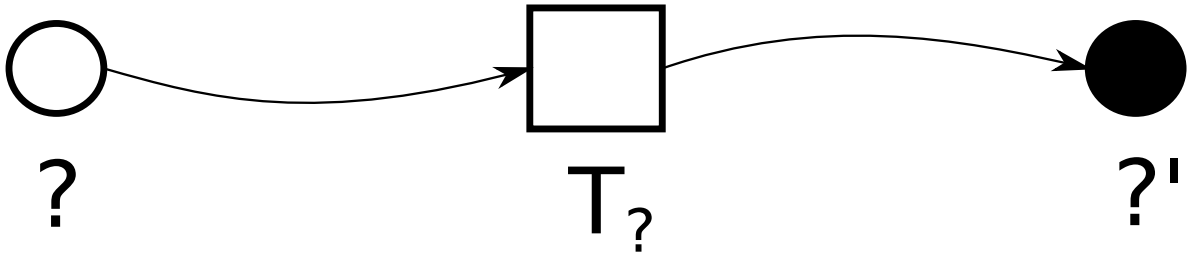
9 décembre 2014

Table of Contents

Génération d'artefact
De quoi on parle



Génération d'artefact
De quoi on parle




```
ld a,0
out (PORTB1), a
call TIMER1
ld a,0FFH
```

```
...
00110100
10011000
10010001
00111001
...
```

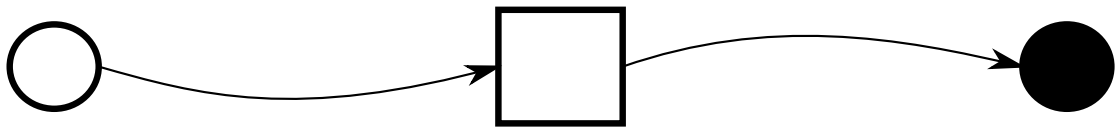
Word

PDF

```
int main(void)
{
   int *p;
      ...
}
```

```
ld a,0
out (PORTB1), a
call TIMER1
ld a,0FFH
```

Génération d'artefact



source

cible
sémantique
equivalente

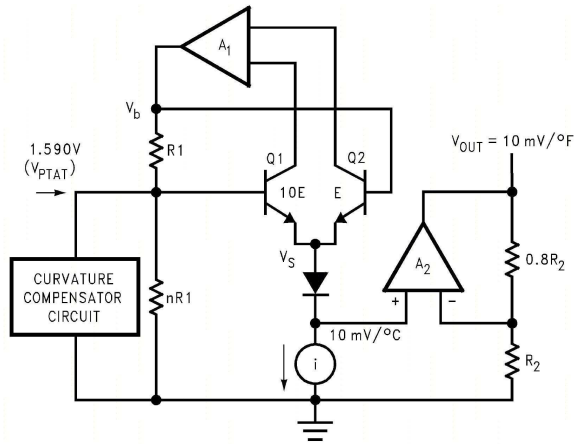
En une (longue) phrase

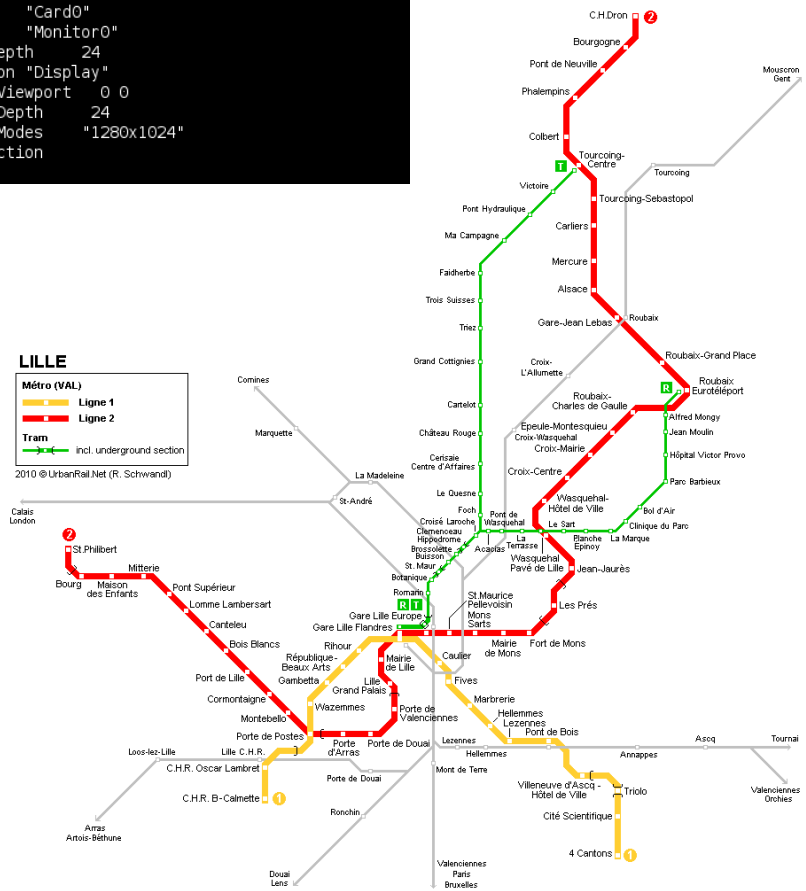
Production d'un ou plusieurs artefacts cible à partir d'un ou plusieurs artefacts source où les artefacts cibles sont sémantiquement équivalents aux artefacts sources.

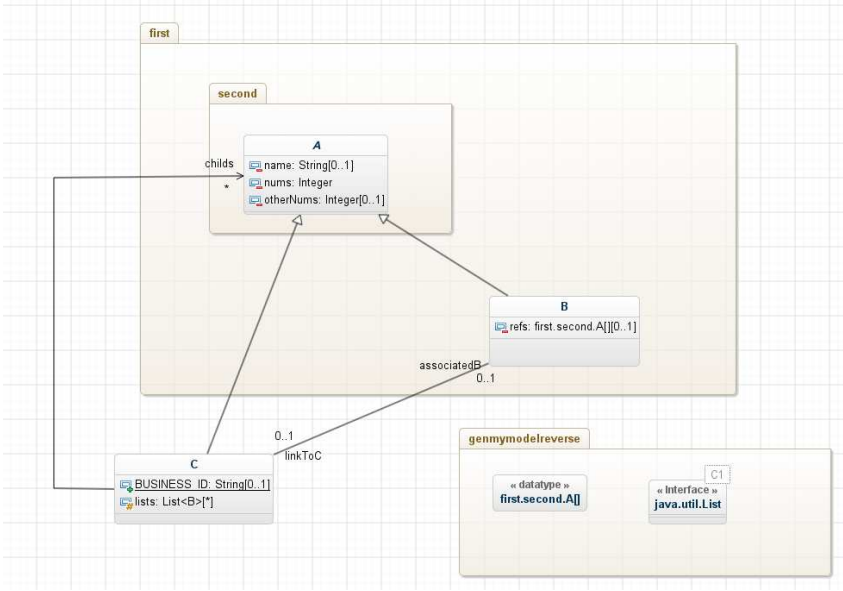
Table of Contents

Un modèle ?
À quoi ça ressemble ?

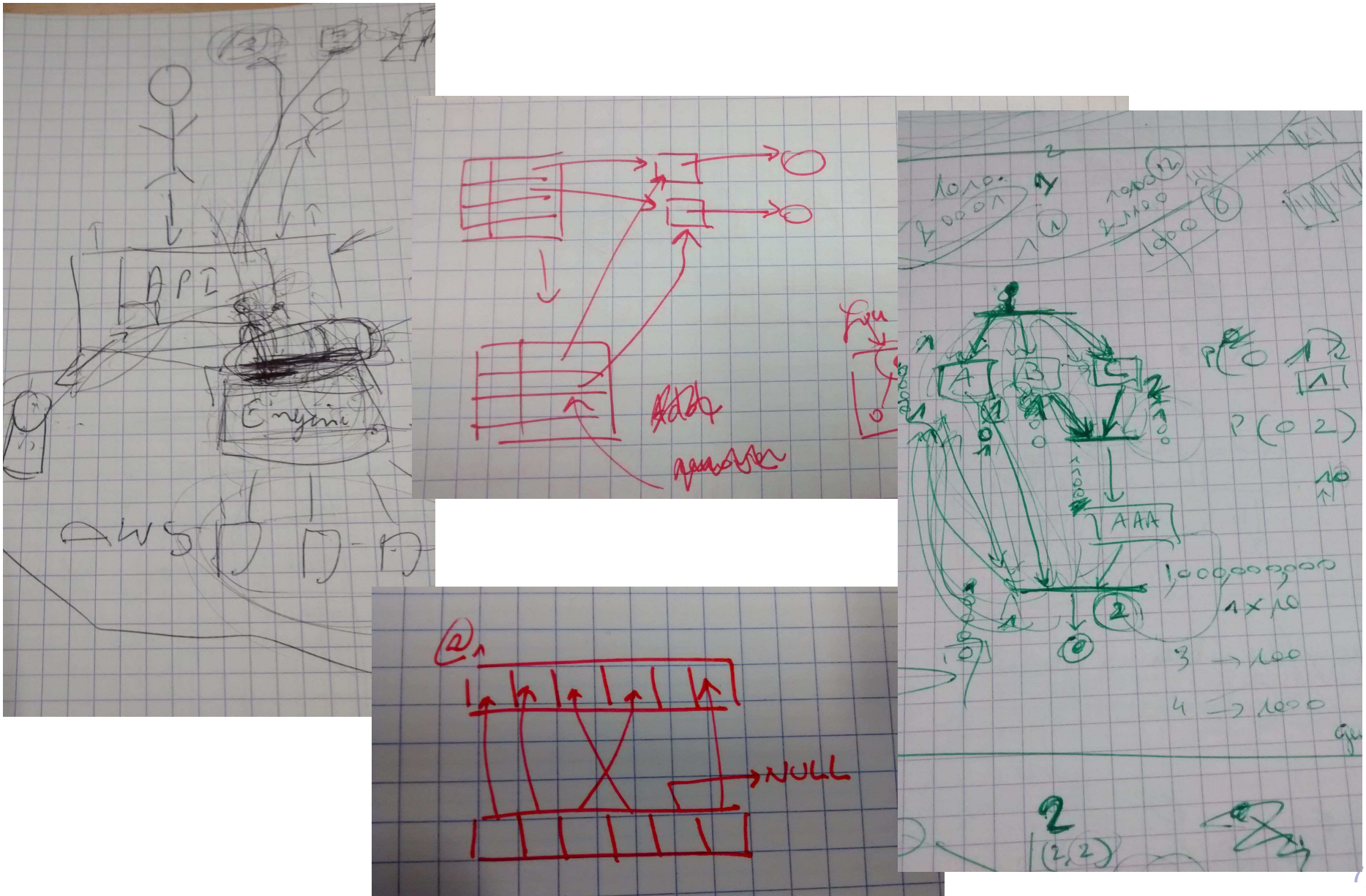
```
File  Edit  View  Terminal  Go  Help
### Comment all HorizSync and VertRefresh values to use DDC
       #HorizSync    31.5 - 57.0
       #VertRefresh  50.0 - 100.0
EndSection

Section "Device"
       Identifier  "Card0"
       Driver      "nv"
       Card        "* NVIDIA"
EndSection

Section "Screen"
       Identifier "Screen0"
       Device     "Card0"
       Monitor    "Monitor0"
       DefaultDepth     24
       SubSection "Display"
              Viewport   0 0
              Depth     24
              Modes    "1280x1024"
       EndSubSection
EndSection
```







Informatique + dessins = ♡



Structure et représentation graphiques

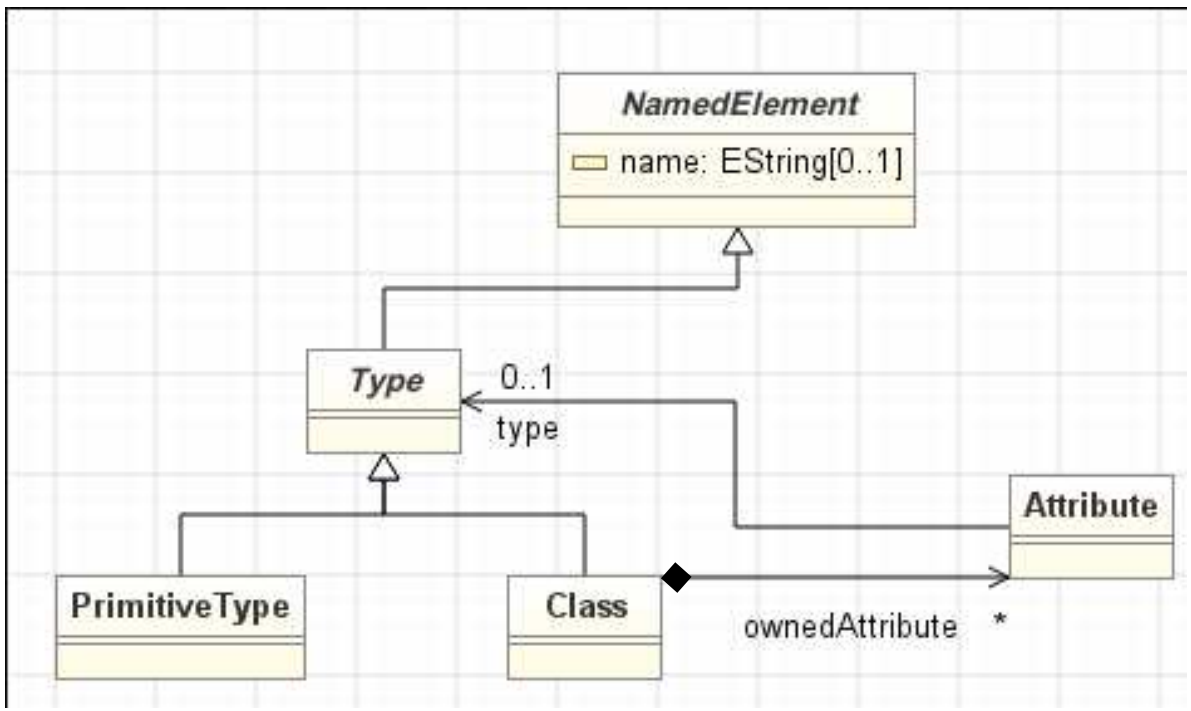
- Modèle est une représentation **structuré** et **abstraite** d'un domaine/principe/entitée...
- Modèle **conforme** à une "convention" (méta-modèle)

Méta-modèle

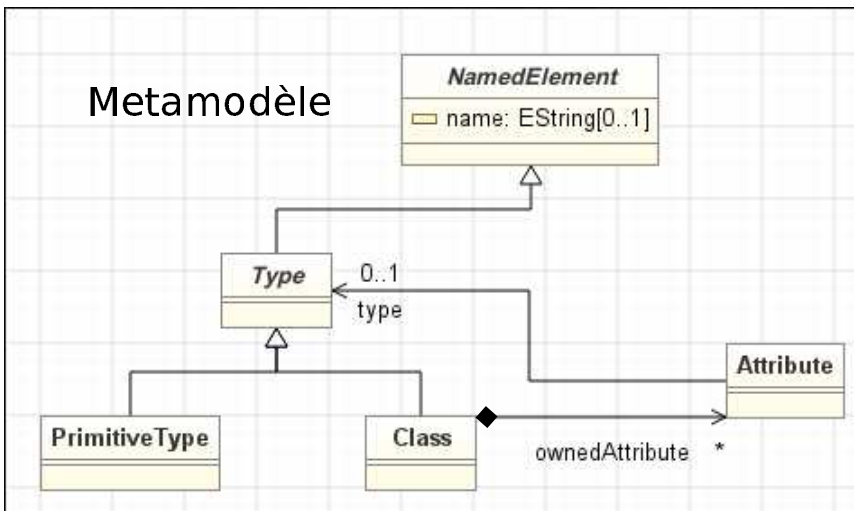
- Décrit la structure des modèles qui lui seront conformes
- Exemples
  - Sur la carte : la légende
  - Code java : grammaire java

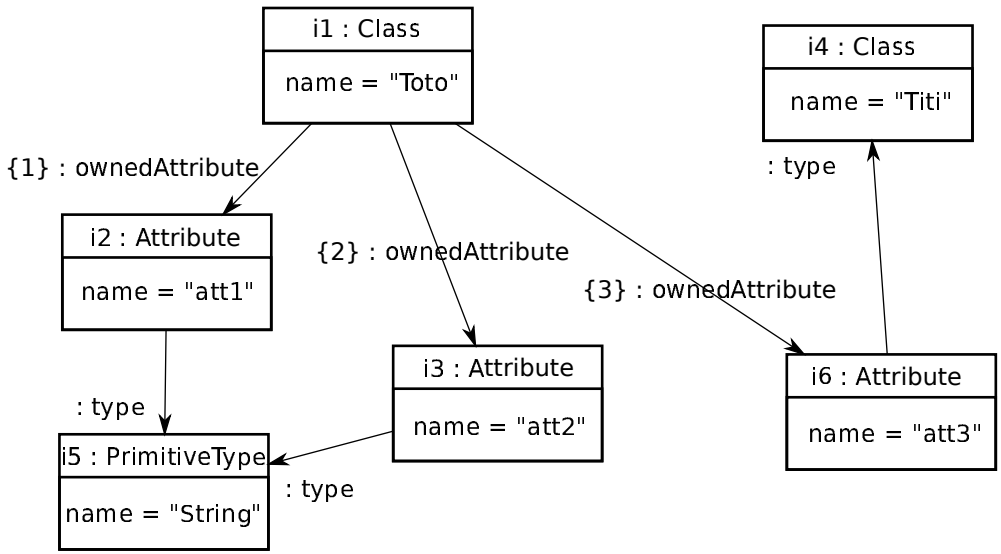
- *Modèle graphique* = modèle qui possède une représentation sous forme de dessin.

Déssine moi un méta-modèle et un de ses modèles



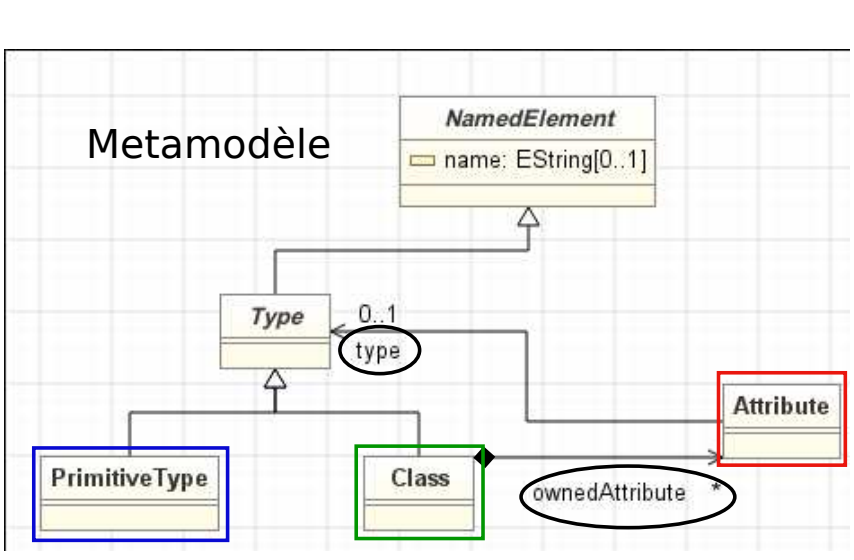
Déssine moi un méta-modèle et un de ses modèles

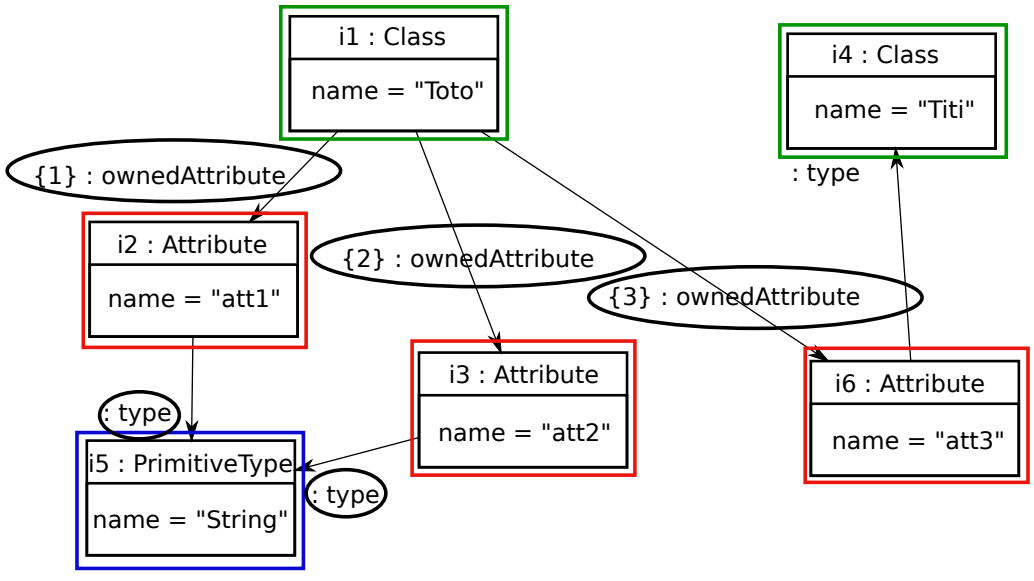




- UN modèle possible (droite) conforme au méta-modèle (gauche)

Déssine moi un méta-modèle et un de ses modèles

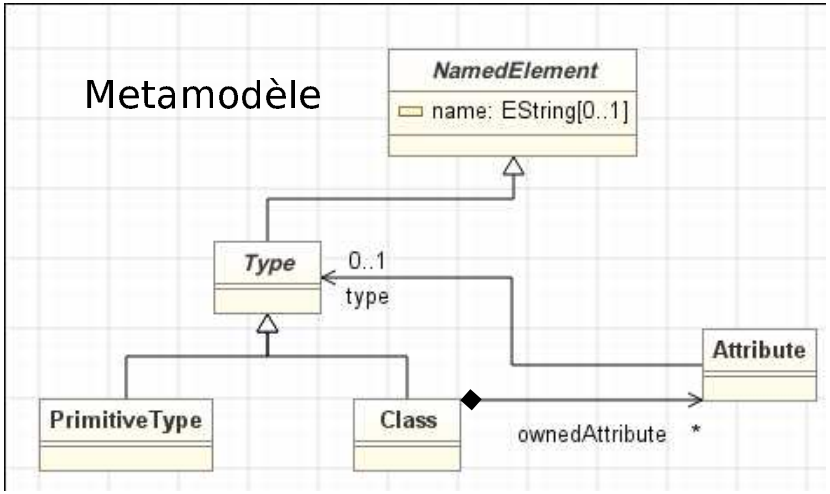


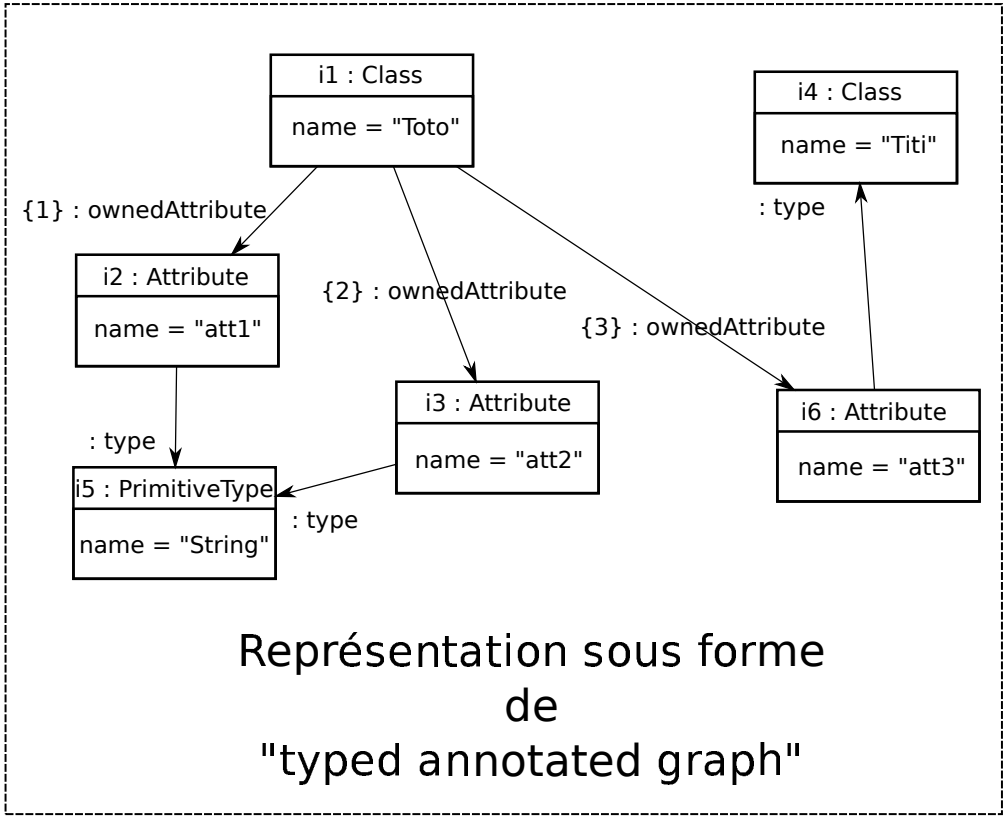


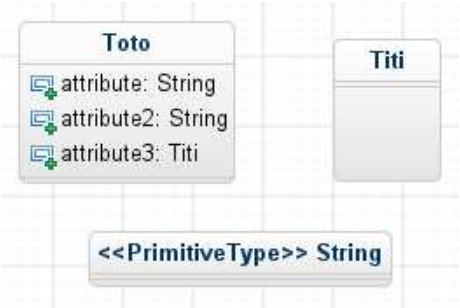
- UN modèle possible (droite) conforme au méta-modèle (gauche)

Déssine moi un méta-modèle et un de ses modèles

Une autre représentation



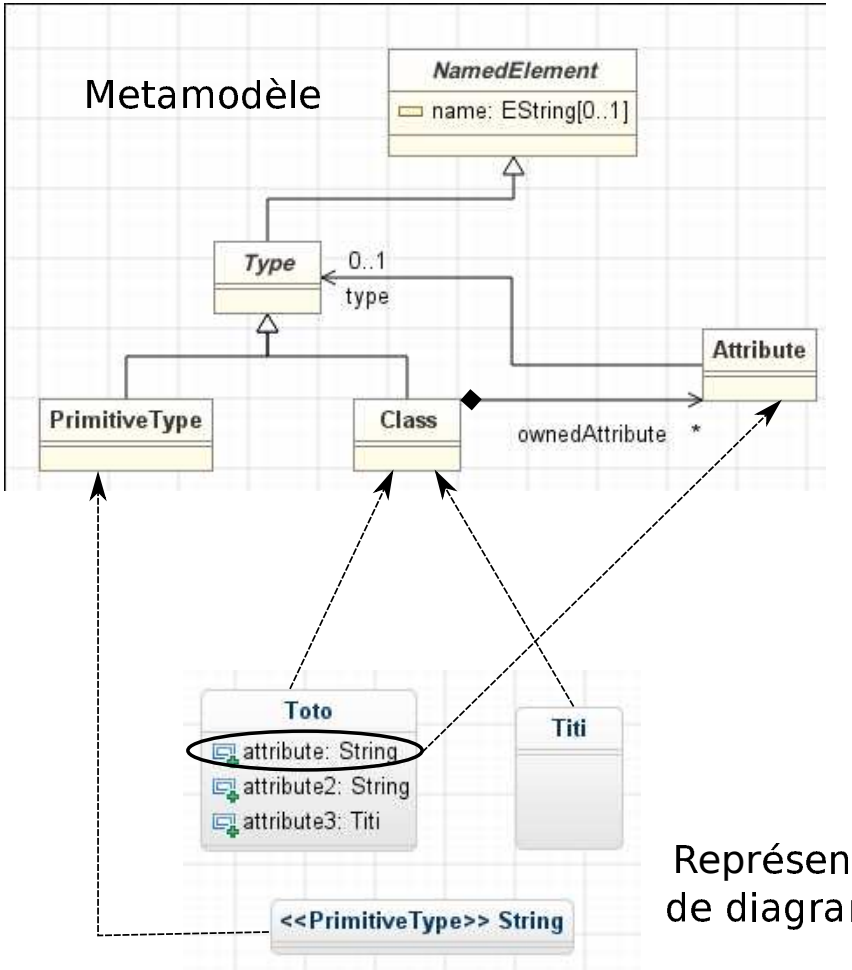




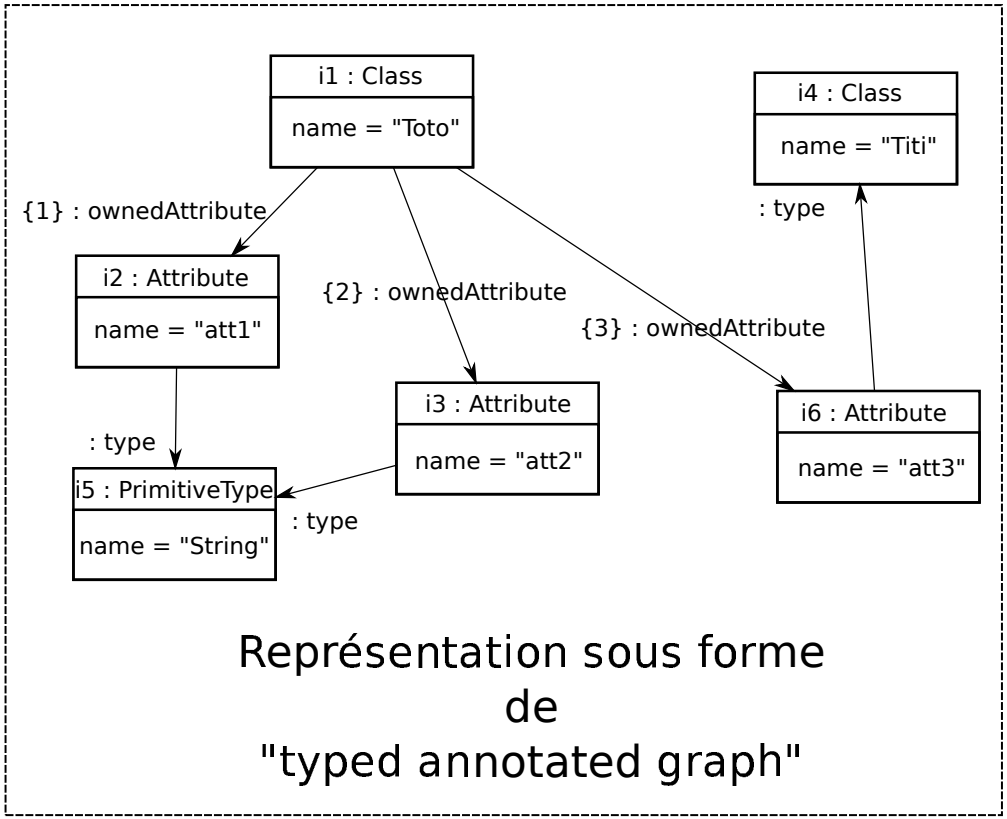
Représentation sous forme de diagramme de classe du modèle

Déssine moi un méta-modèle et un de ses modèles

Une autre représentation



Représentation sous forme de diagramme de classe du modèle



Représentation sous forme de "typed annotated graph"

Méta-modèle = langage

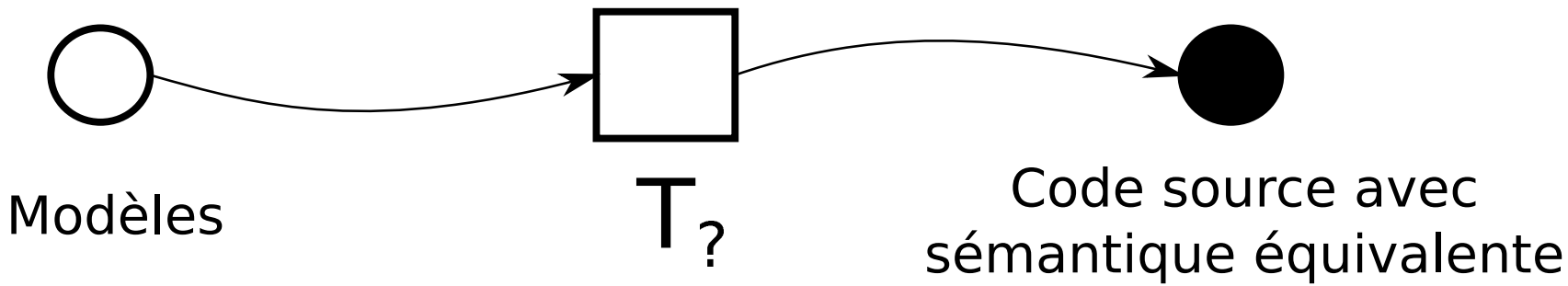
Plusieurs langages de modélisation avec syntaxe graphique existant :

- UML : utilisé pour définir différents aspects d'un logiciel
- BPMN : utilisé pour définir des processus métiers
- SYSML : utilisé pour spécifier, valider, analyser des systèmes complexes (hardware/software)
- IFML : utilisé pour décrire le contenu, les interactions utilisateur et le comportement d'applications front-end
- ...

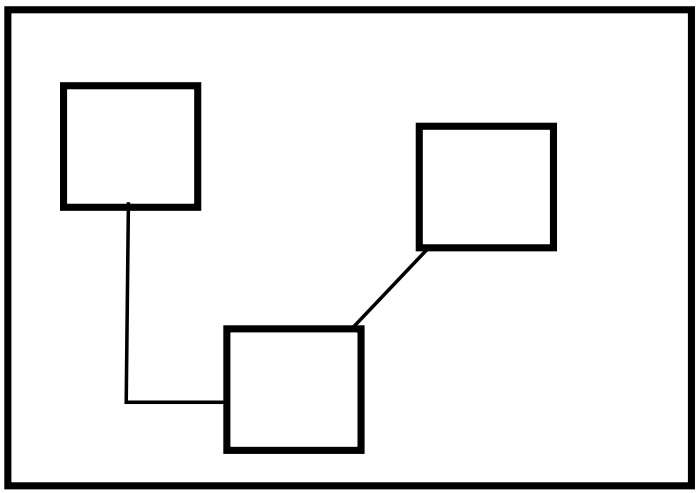
Possible de créer son propre méta-modèle, donc propre langage de modélisation.

Table of Contents

Génération de code source

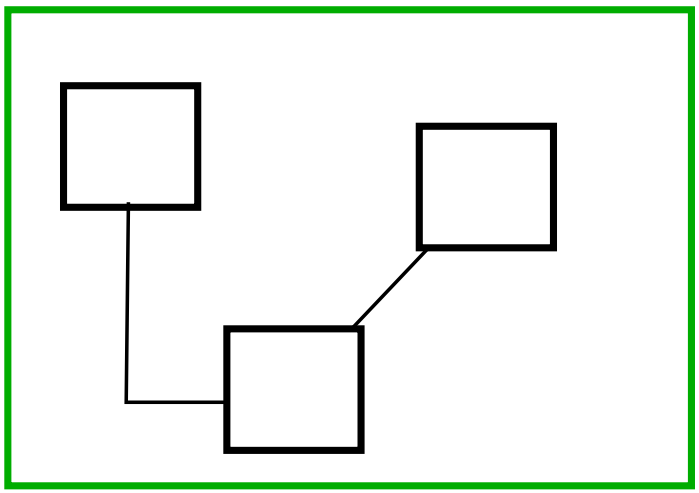


Du modèle vers le code
Overview



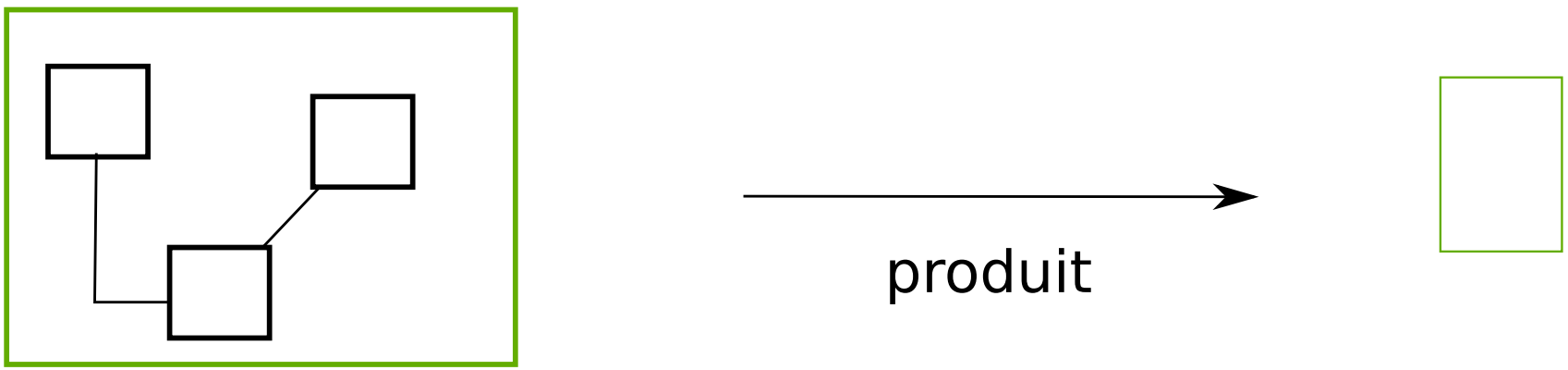
Du modèle vers le code

Overview



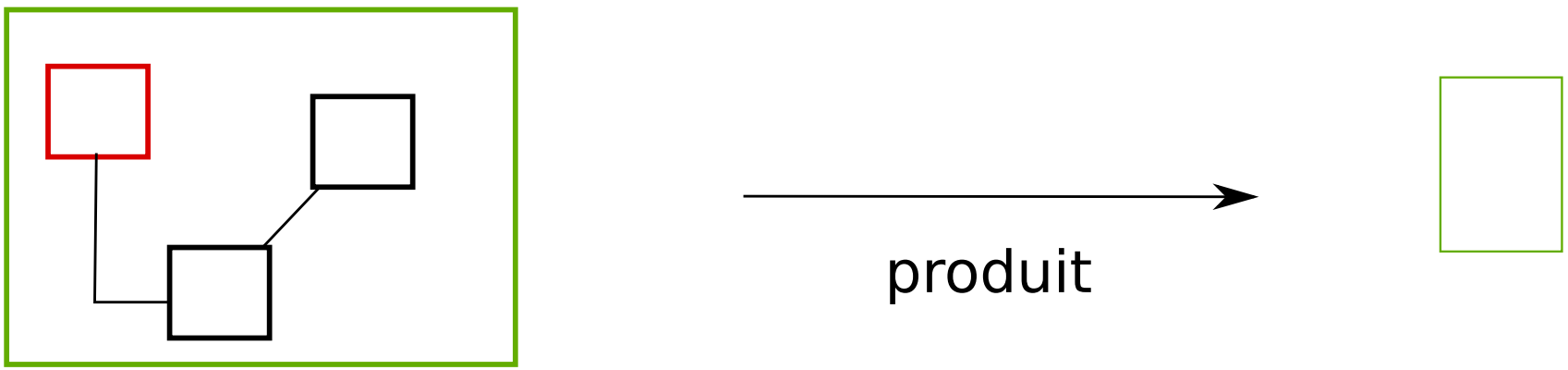
Du modèle vers le code

Overview



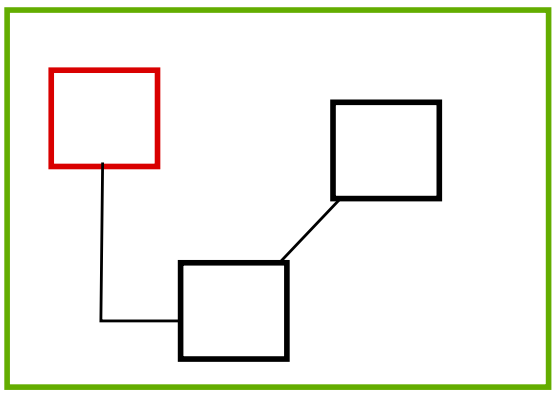
Du modèle vers le code

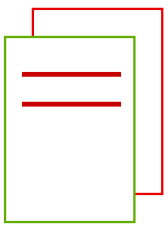
Overview



Du modèle vers le code

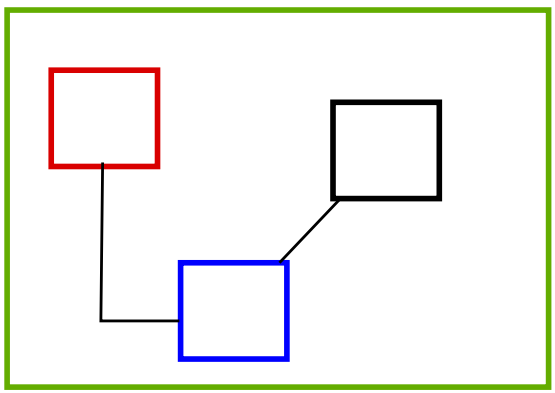
Overview

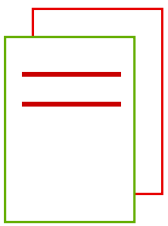




Du modèle vers le code

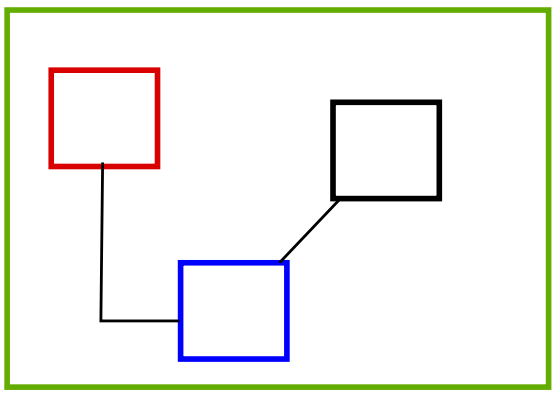
Overview

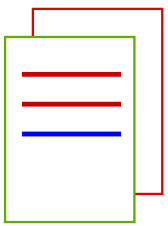




Du modèle vers le code

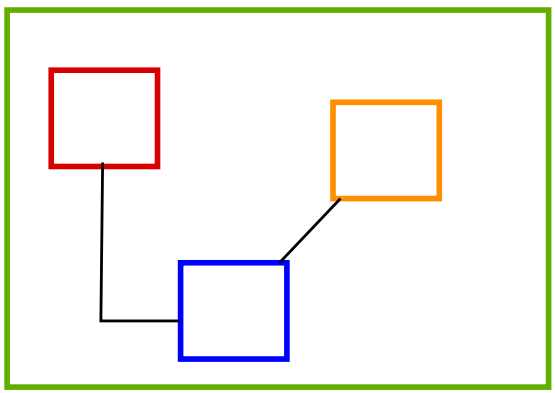
Overview

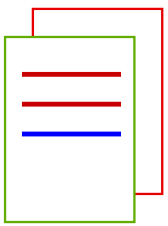




Du modèle vers le code

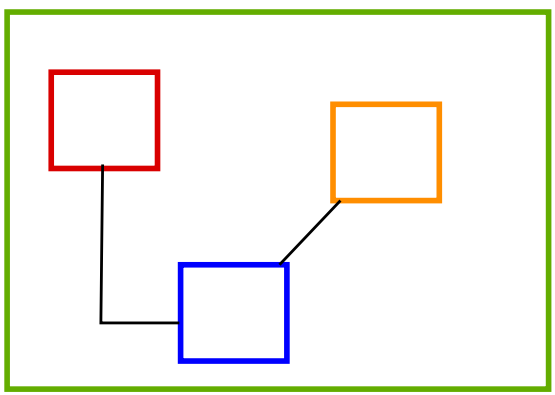
Overview

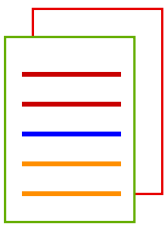




Du modèle vers le code

Overview



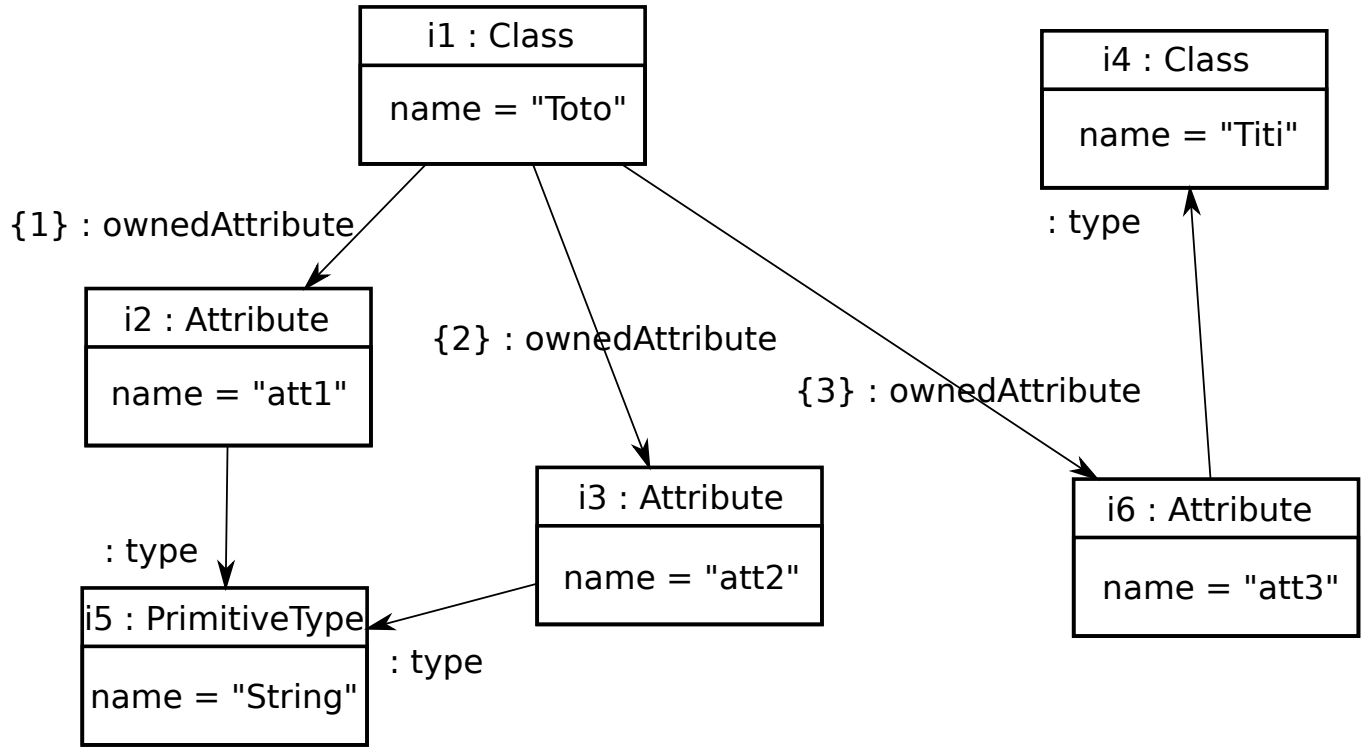


OCL : Object Constraint Language

Manipulation de modèles conformes à un méta-modèle

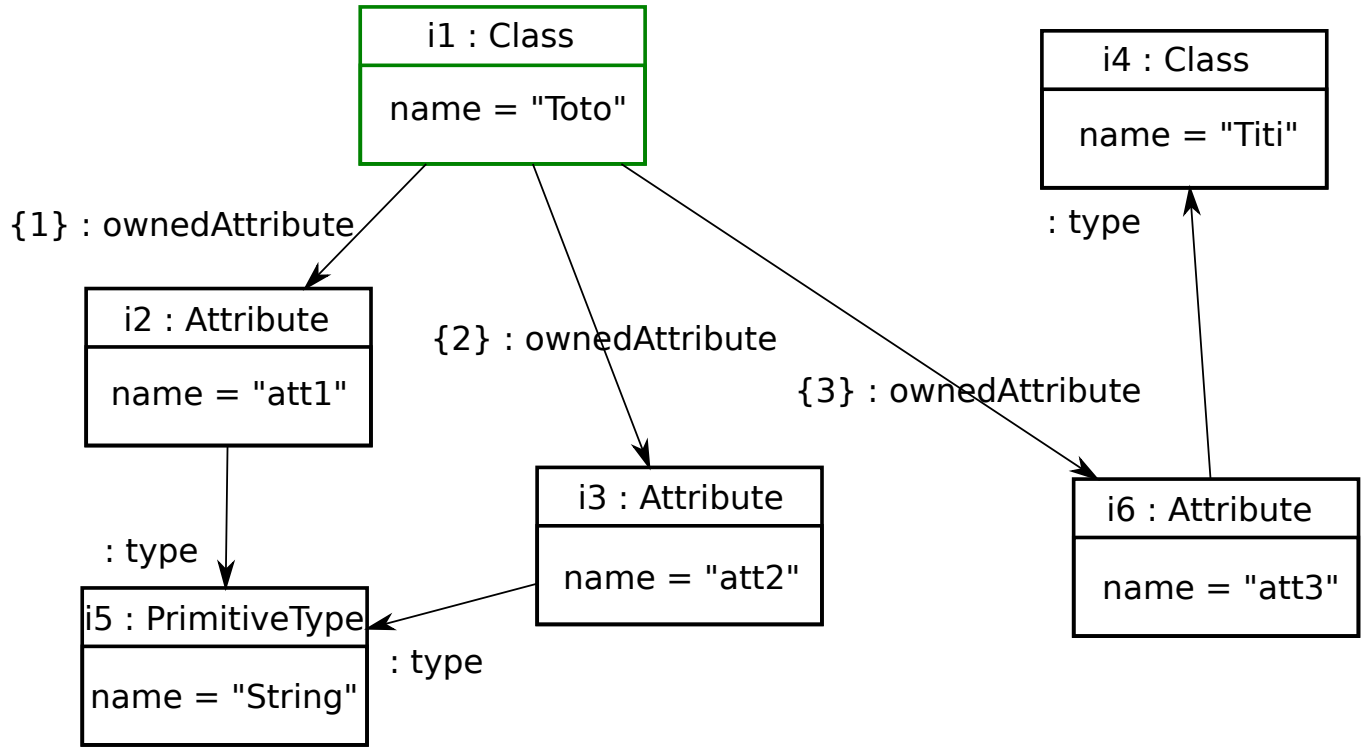
- **Language développé en** 1995 **par IBM**
- **specification :** `http://www.omg.org/spec/OCL`
- **Objectif :** "OCL est avant tout un **langage de requête** qui permet de calculer une expression sur un modèle en s'appuyant sur son méta-modèle." *(B. Combemale)*
- OCL est un **langage typé**
- OCL est un langage **sans effet de bord**
- Permet de **naviguer**, **filtrer** des éléments de modèles

OCL : Object Constraint Language



Context i1 :
self.name
self.ownedAttribute
self.ownedAttribute->at(1).type.name
self.ownedAttribute->select(e | e.type.oclIsKindOf(Class))

OCL : Object Constraint Language



Context i1 :

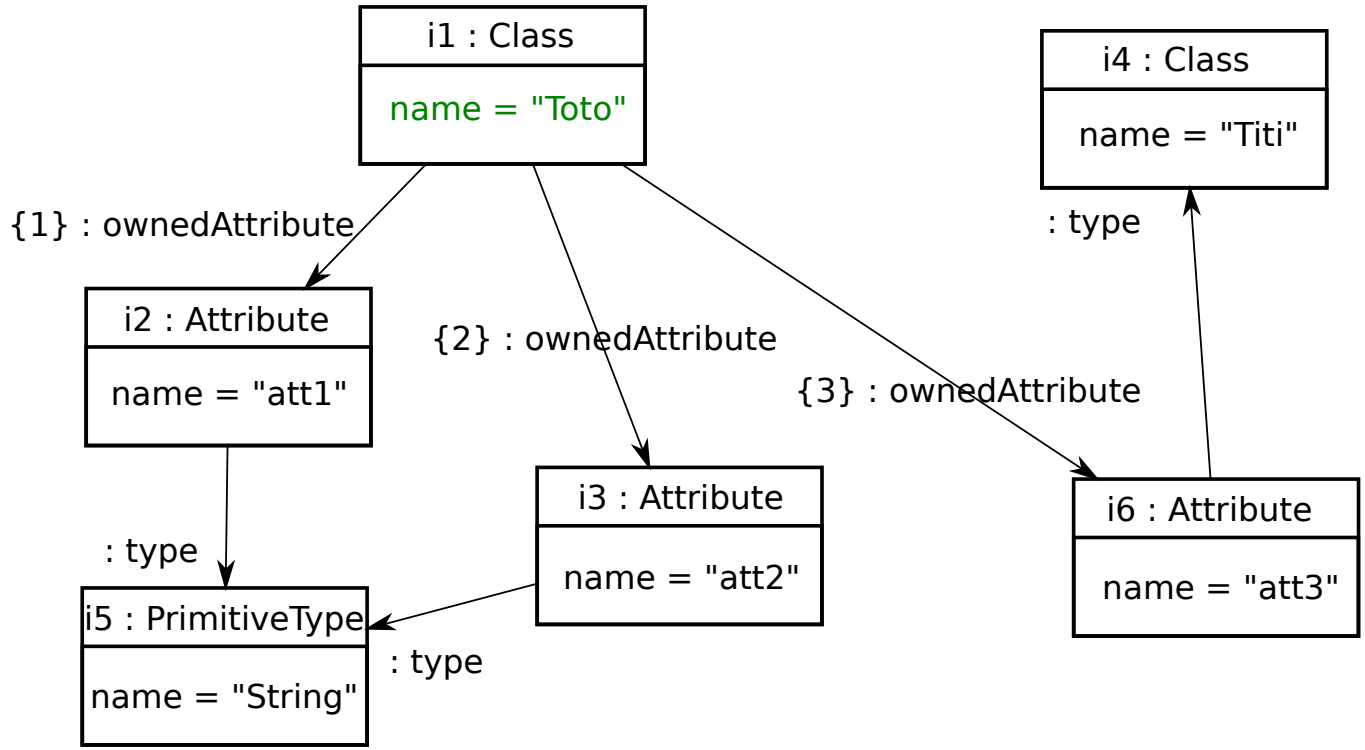
self.name

self.ownedAttribute

self.ownedAttribute->at(1).type.name

self.ownedAttribute->select(e | e.type.oclIsKindOf(Class))

OCL : Object Constraint Language



Context i1 :

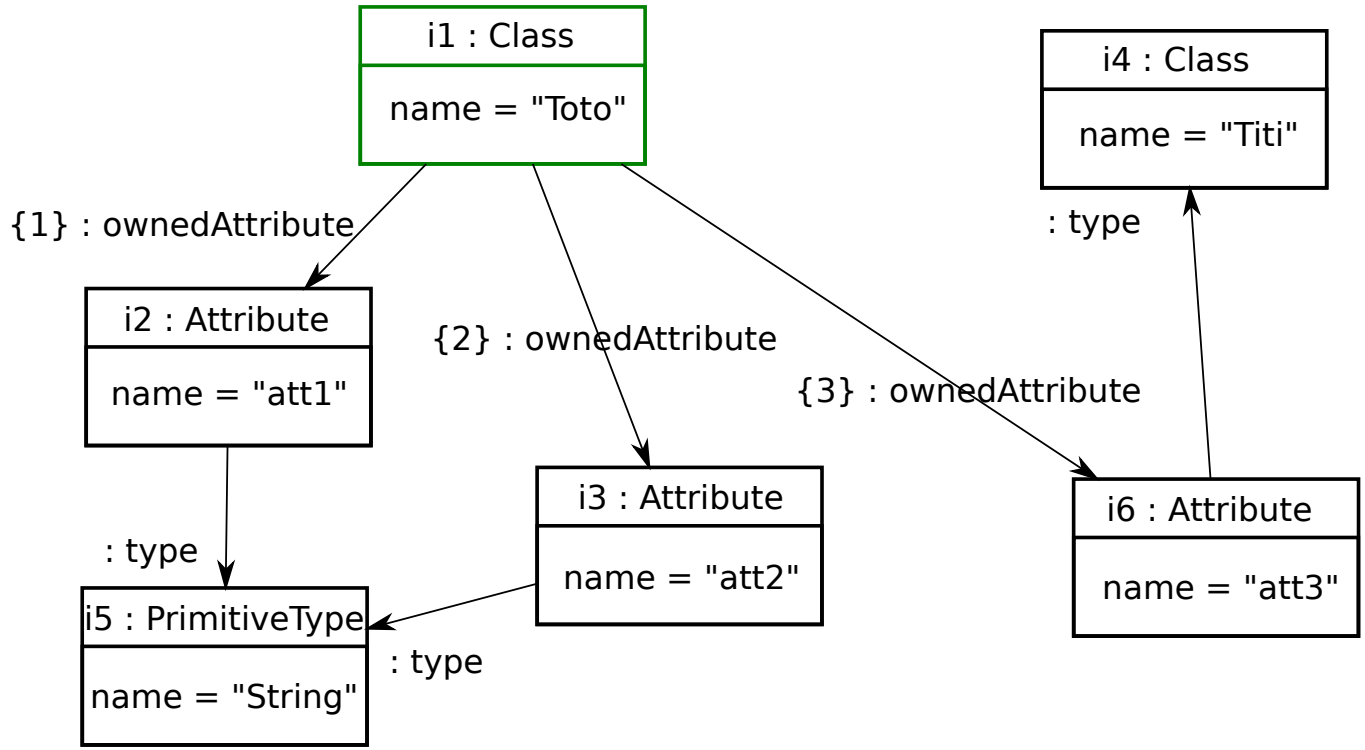
self.name = Toto

self.ownedAttribute

self.ownedAttribute->at(1).type.name

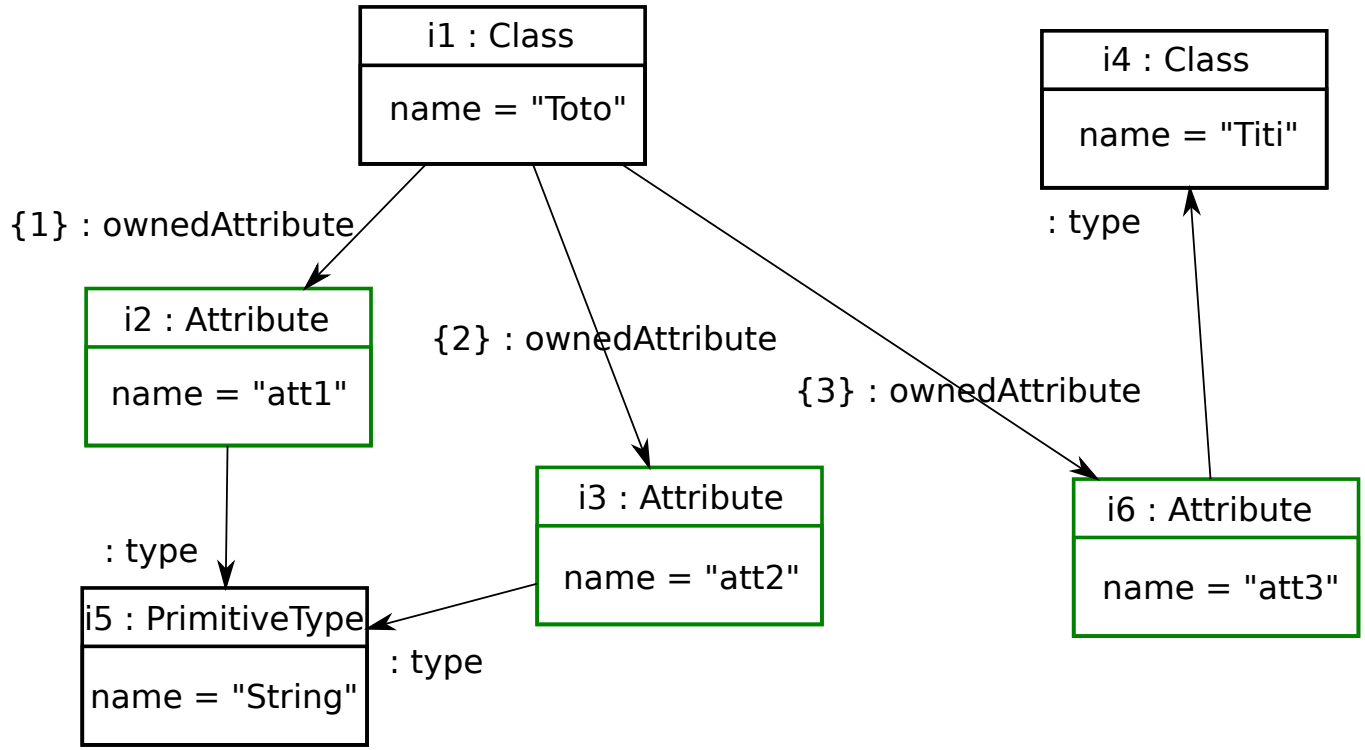
self.ownedAttribute->select(e | e.type.oclIsKindOf(Class))

OCL : Object Constraint Language



Context i1 :
self.name = Toto
self.ownedAttribute
self.ownedAttribute->at(1).type.name
self.ownedAttribute->select(e | e.type.oclIsKindOf(Class))

OCL : Object Constraint Language



Context i1 :

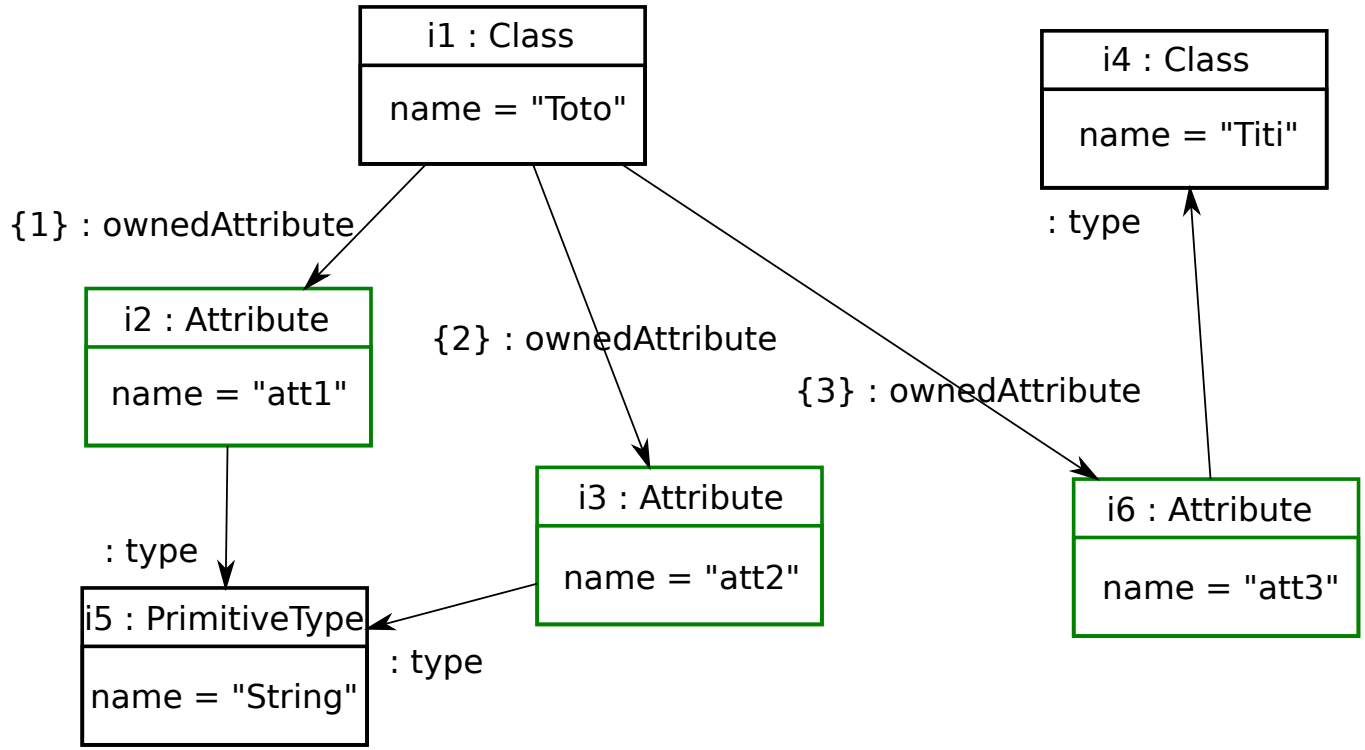
self.name = Toto

self.ownedAttribute = $\{i2, i3, i6\}$

self.ownedAttribute->at(1).type.name

self.ownedAttribute->select(e | e.type.oclIsKindOf(Class))

OCL : Object Constraint Language



Context i1 :

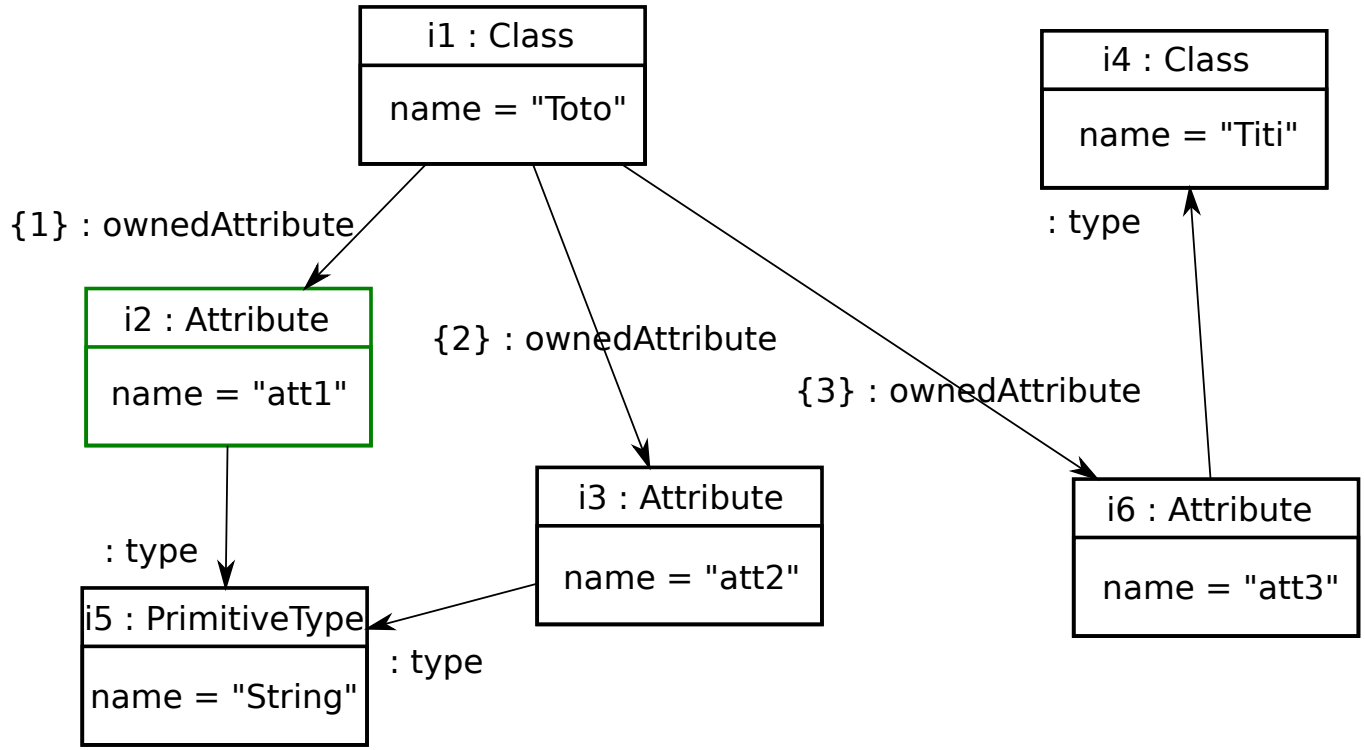
self.name = Toto

self.ownedAttribute = $\{i2, i3, i6\}$

self.ownedAttribute->at(1).type.name

self.ownedAttribute->select(e | e.type.oclIsKindOf(Class))

OCL : Object Constraint Language



Context i1 :

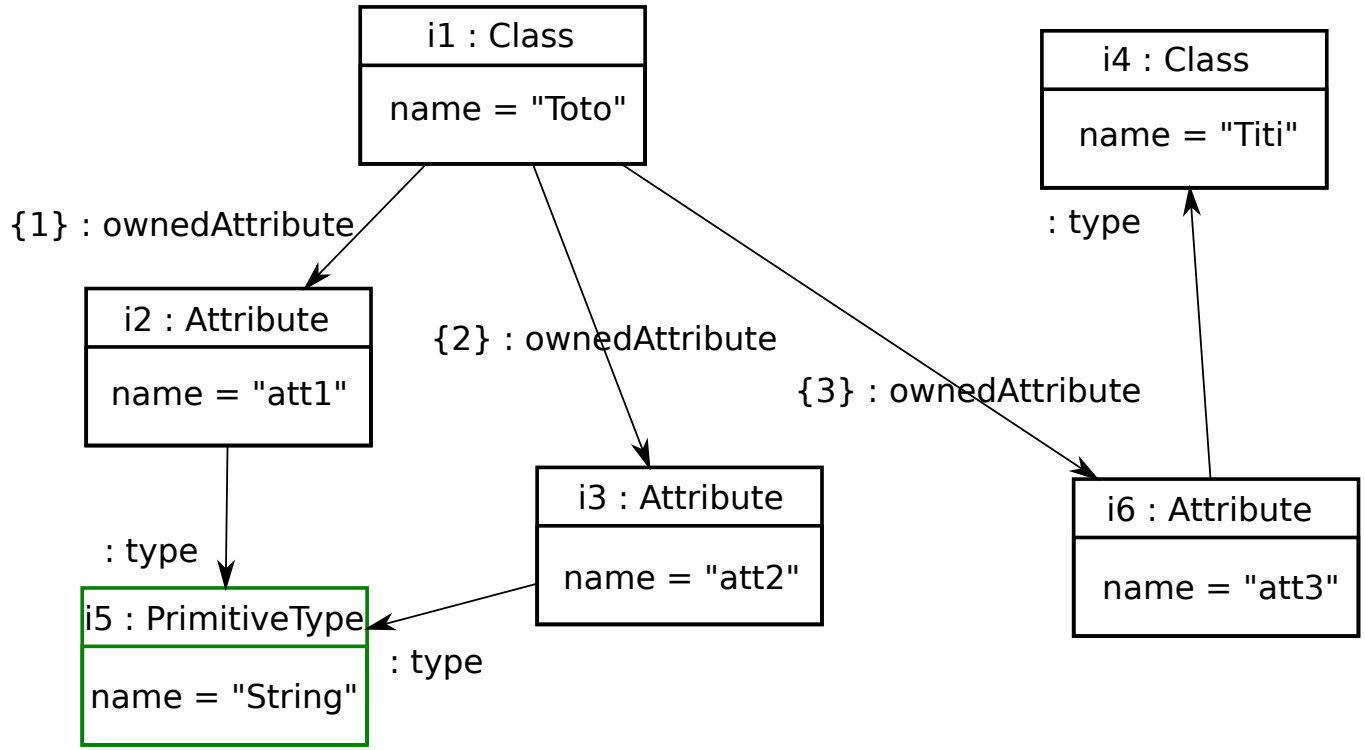
self.name = Toto

self.ownedAttribute = $\{i2, i3, i6\}$

self.ownedAttribute->at(1).type.name

self.ownedAttribute->select(e | e.type.oclIsKindOf(Class))

OCL : Object Constraint Language



Context i1 :

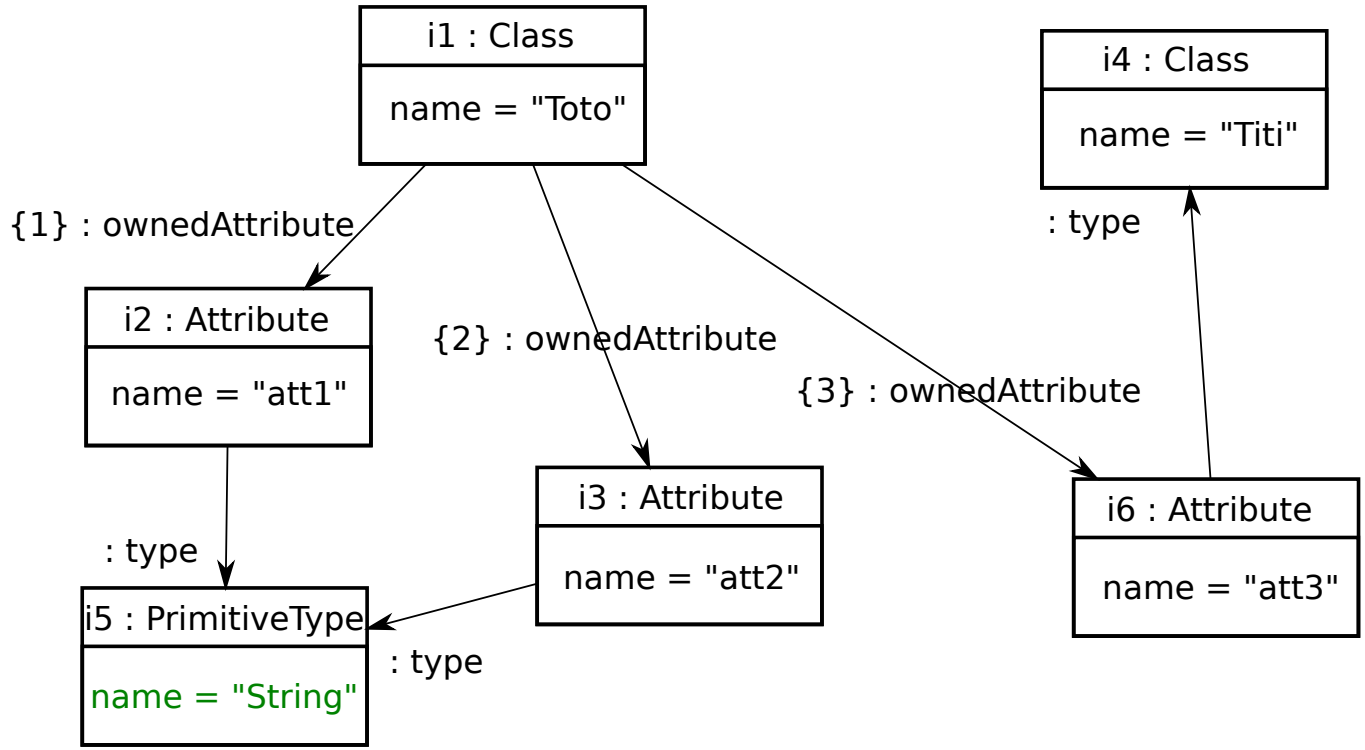
self.name = Toto

self.ownedAttribute = $\{i2, i3, i6\}$

self.ownedAttribute->at(1).type.name

self.ownedAttribute->select(e | e.type.oclIsKindOf(Class))

OCL : Object Constraint Language



Context i1 :

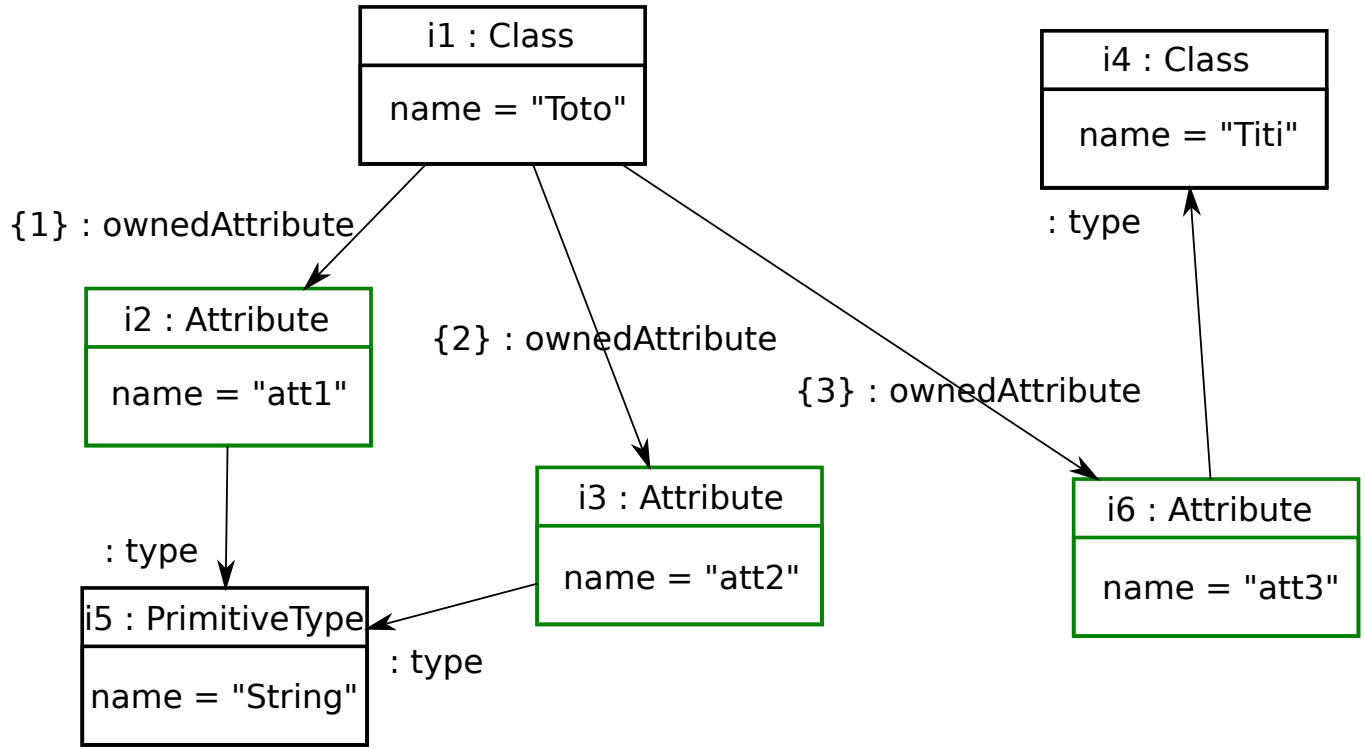
self.name = Toto

self.ownedAttribute = $\{i2, i3, i6\}$

self.ownedAttribute->at(1).type.name = String

self.ownedAttribute->select(e | e.type.oclIsKindOf(Class))

OCL : Object Constraint Language



Context i1 :

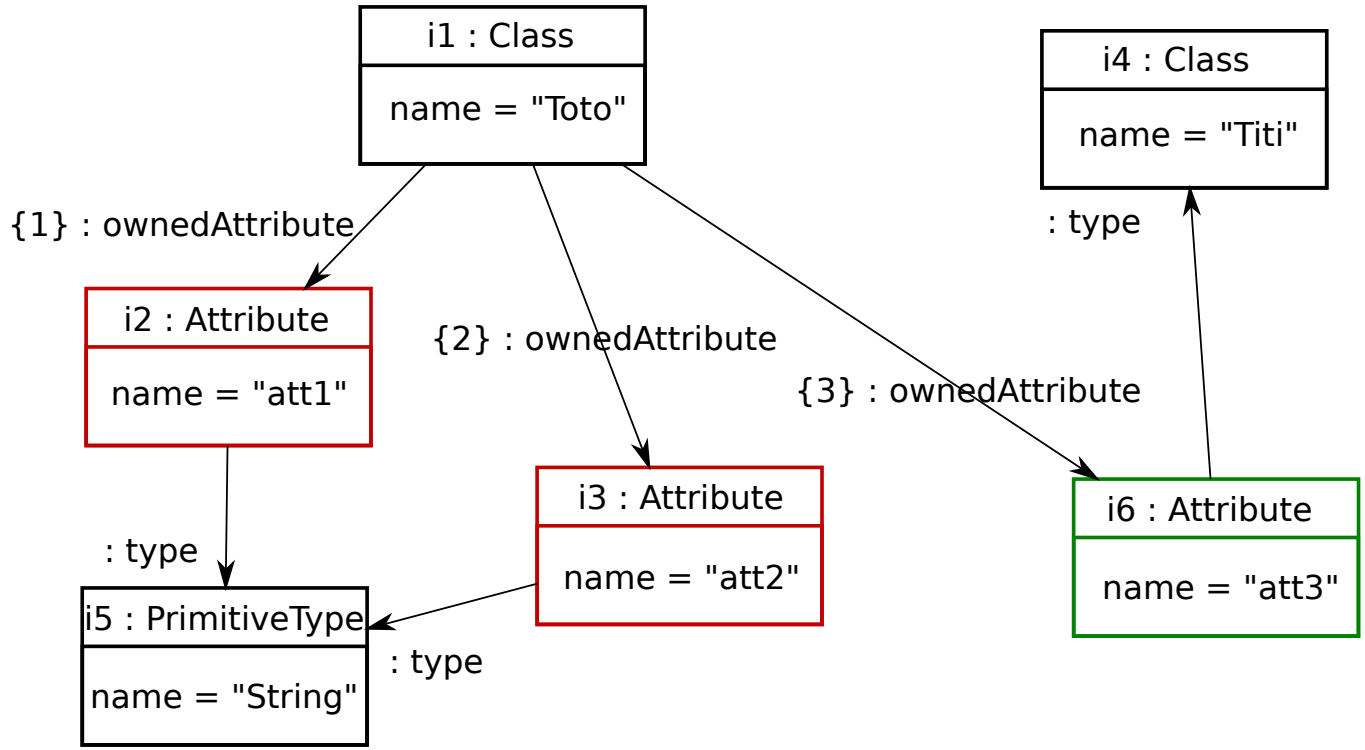
self.name = Toto

self.ownedAttribute = $\{i2, i3, i6\}$

self.ownedAttribute->at(1).type.name = String

self.ownedAttribute->select(e | e.type.oclIsKindOf(Class))

OCL : Object Constraint Language



Context i1 :

self.name = Toto

self.ownedAttribute = $\{i2, i3, i6\}$

self.ownedAttribute->at(1).type.name = String

self.ownedAttribute->select(e | e.type.oclIsKindOf(Class)) = $\{i6\}$

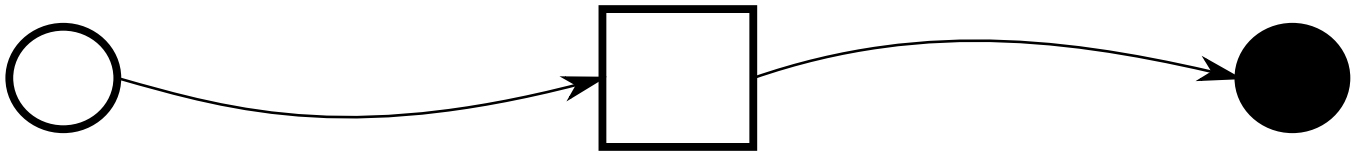
Du modèle vers le code
Les technologies

- Laguages dédiés (DSL) : **MOFM2T**, Epsilon Generation Language, XPand, Jet, ATL (en trichant), Kermeta, QVT (en trichant)...
- MOFM2T = OCL + Templates
- **Specification :** `http://www.omg.org/spec/MOFM2T/1.0/`

Et un exemple de plus

```
[template public class2Java(c:Class)]
class [c.name/] {
    // Constructor
    [c.name/]() {
    }
}
[/template]
```

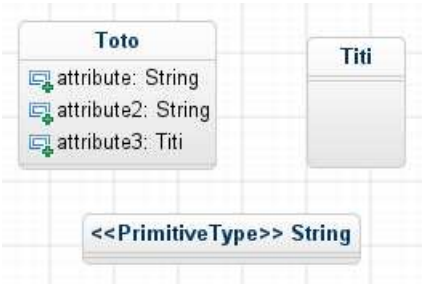
On met tout ensemble



Modèles

$T_?$

Code source avec sémantique équivalente



Artefact source (modèle)

```
3  [template public generate(c : Class)]
4  [comment @main/]
5  [file (c.name + '.java', false, 'UTF-8')]
6  public class [c.name/] {
7      [c.ownedAttribute.genAttribute()/]
8
9      // Constructor
10     [c.name/]() {
11
12     }
13 }
14 [/file]
15 [/template]
16
17 [template public genAttribute(a : Property)]
18 [a.type.name/] [a.name/];
19
20 [/template]
21
```

Transformation M2T vers Java

Code Java sémantiquement équivalent

Hypothèse

Comme le modèle est une abstraction, il n'est lié à aucun langage de programmation. Il serait donc possible de générer du code à partir de notre modèle vers un autre langage.

En bref, mapping appliqué pour Java :

E. modèle		**Java**
Class	$\rightarrow$	Class Java
Attribute	$\rightarrow$	Attribut Java

Mapping pour C : ?????

Distance sémantique
Solutions

1. Trouver une traduction dans le langage cible pour complètement couvrir la sémantique du modèle d'entrée

Distance sémantique
Solutions

1. Trouver une traduction dans le langage cible pour complètement couvrir la sémantique du modèle d'entrée
2. Restreindre/spécialiser la sémantique du modèle

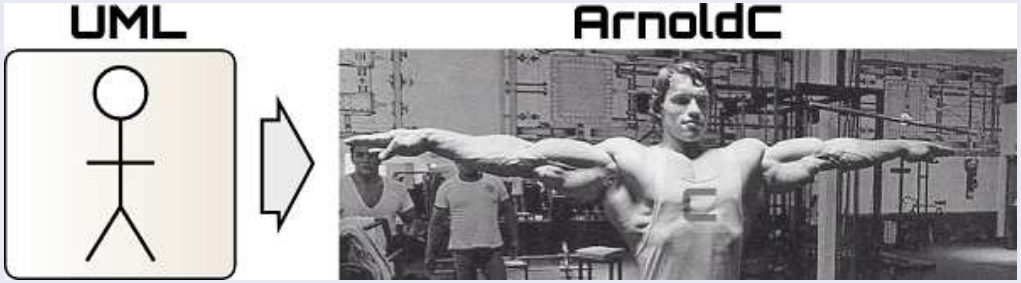
Solution $2 \rightarrow 1$

Distance sémantique
Solutions

1. Trouver une traduction dans le langage cible pour complètement couvrir la sémantique du modèle d'entrée
2. Restreindre/spécialiser la sémantique du modèle

Solution $2 \rightarrow 1$

UML vers ArnoldC



- En collaboration avec Ali Gourch
- Passage d'un diagrame d'activité UML à un code ArnoldC (esolang)

Abstraction vers Concret
Problèmes et solutions

Problème

- Trop forte abstraction dans le modèle et cible complexe $\rightarrow$ transformation M2T complexe à écrire

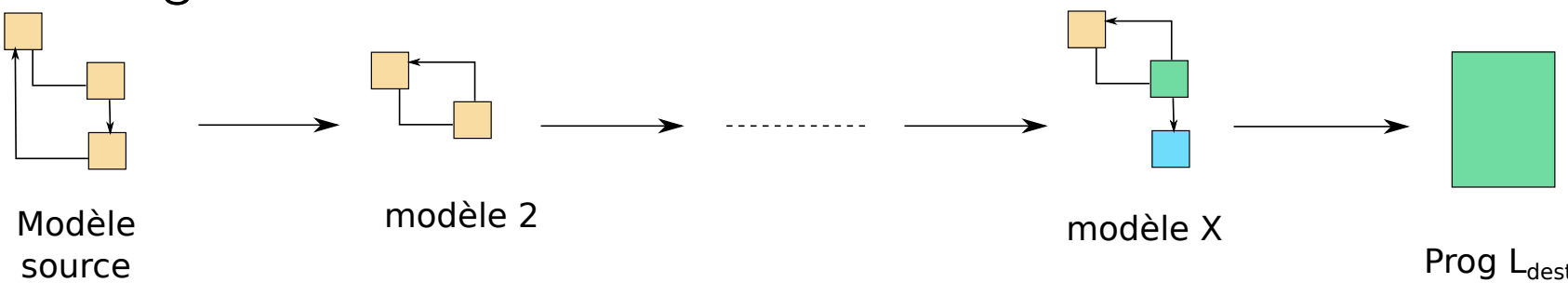
Solution

Raffiner le modèle d'entrée en le spécialisant et en le ramenenant à une structure proche de celle attendu pour le language cible

Abstraction vers Concret
Nouvelles technologies

- Génération de texte (code) à partir de modèle → M2T (model to text)
- Raffinage d'un modèle ou changement d'espace technologique = Transformation de modèles → M2M (model to model)

Chainage de M2M et de M2T → chaines de transformations.



Abstraction vers Concret
Exemple de chaines

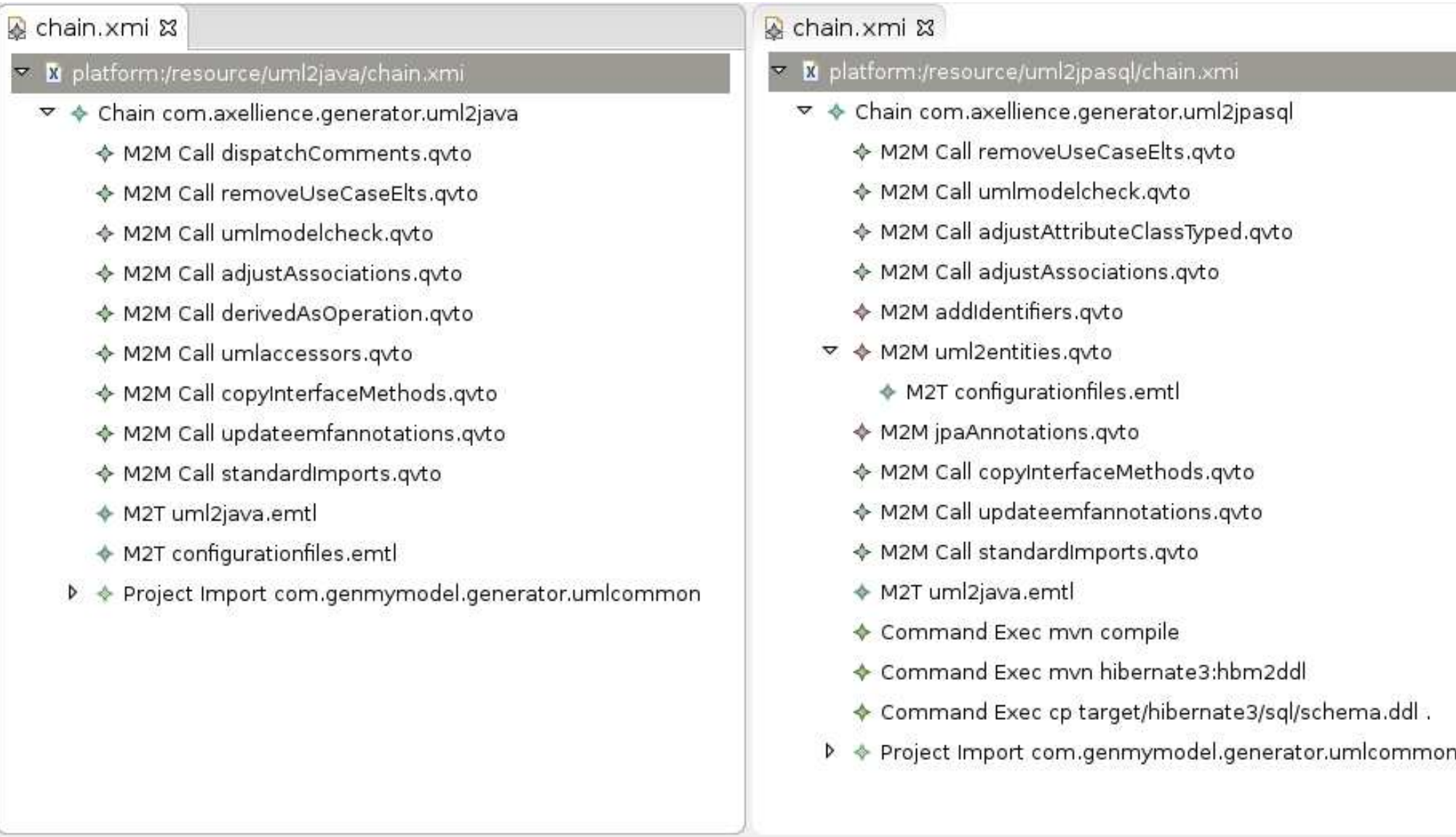


Table of Contents

Dans le vrai monde
Pour faire des trucs avec

Beaucoups d'outils disponible pour Eclipse:

- EMF (Eclipse Modeling Framework) pour écrire des méta-modèles/modèles conformes
- Modeleurs UML (Papyrus ...)
- QVT(o-r), ATL, Acceleo

Des outils disponible en ligne:

- Modeleurs (dessin): LucidChart, Cacoo...etc, MAIS pas de conformité ni de lien avec un métamodèle
- Modeleurs: GenMyModel
  - UML/FlowChart/Ecore
  - chaines de génération de code (QVT + Acceleo)
  - ...

Conclusion

- Modèle $\rightarrow$ reflexion niveau abstrait
- Plusieurs langages de modélisation
- Génération de code = outils
- Gain de temps
- Manipulation de modèles: OCL
- Génération de code : plusieurs langages
- MOF2MTL (Acceleo) = OCL + templates
- Permet un shift rapide de technos
- Anticiper les erreurs

Dans le vrai monde

Les idées reçues - 1

Dans le vrai monde
Les idées reçues - 2

Dans le vrai monde
Les idées reçues - 3

Nouvelle couche - encore de la complexité

ModèleS $\rightleftharpoons$ codeS

Pour aller plus loin

Transformation d'un langage textuel vers un autre, un flôt possible :