Université de Liège – Département ASMA
Laboratoire Milieux Continus & Thermomécanique

Architecture de Metafor

Code EF – orienté objet

Romain Boman

LTAS-MCT – Institut de Mécanique B52/3 – 1, chemin des Chevreuils – B 4000 LIEGE
Tel: +32 (0)4 366 91 85 Fax: +32 (0)4 366.91.41 Web: http://garfield.ltas.ulg.ac.be/ E-mail: r.boman@ulg.ac.be

Sommaire

Aperçu de Metafor

Aperçu

Qu'est ce que c'est ?

Code de calcul EF « grandes déformations » initialement dédié à la simulation du metal forming.

Historique

- 1992 Metafor Fortran (thèse Jean-Philippe Ponthot).
- 1992-1998 Intégration maladroite de diverses thèses.
- 1998 Situation ingérable – ajout de « patchs » en C.
- 2000 Découverte d'Oofelie – réécriture complète en C++.
- 2000-2005 Intégration de tous les développements au sein de Metafor.

Equipe

(L. Adam), R. Boman, PP. Jeunechamps, (L. Noels), L. Papeleux.
coachés par J-P Ponthot.

Aperçu

Buts recherchés

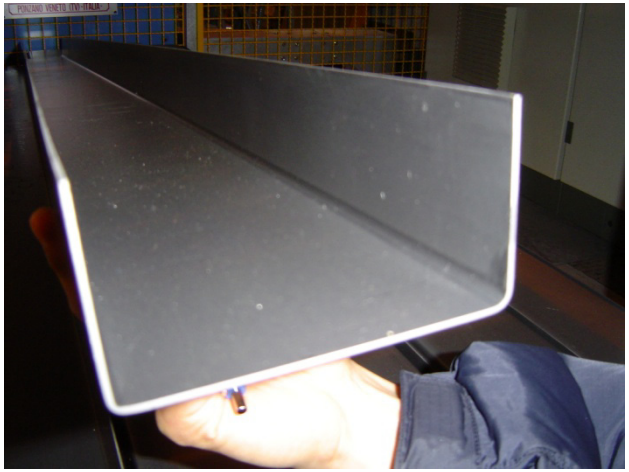
- Ne pas perdre les développements acquis au cours des années.
- Rassembler les chercheurs autour d'une plateforme unique.

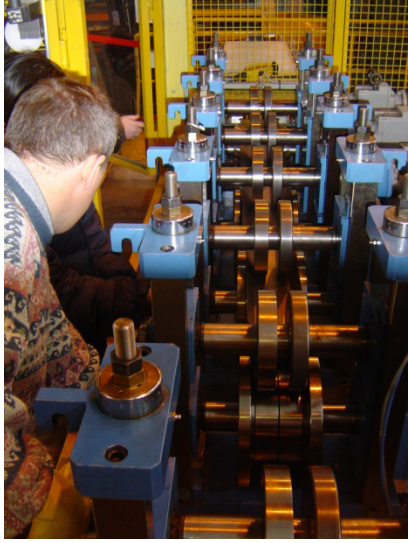
« Toolkit » de base EF

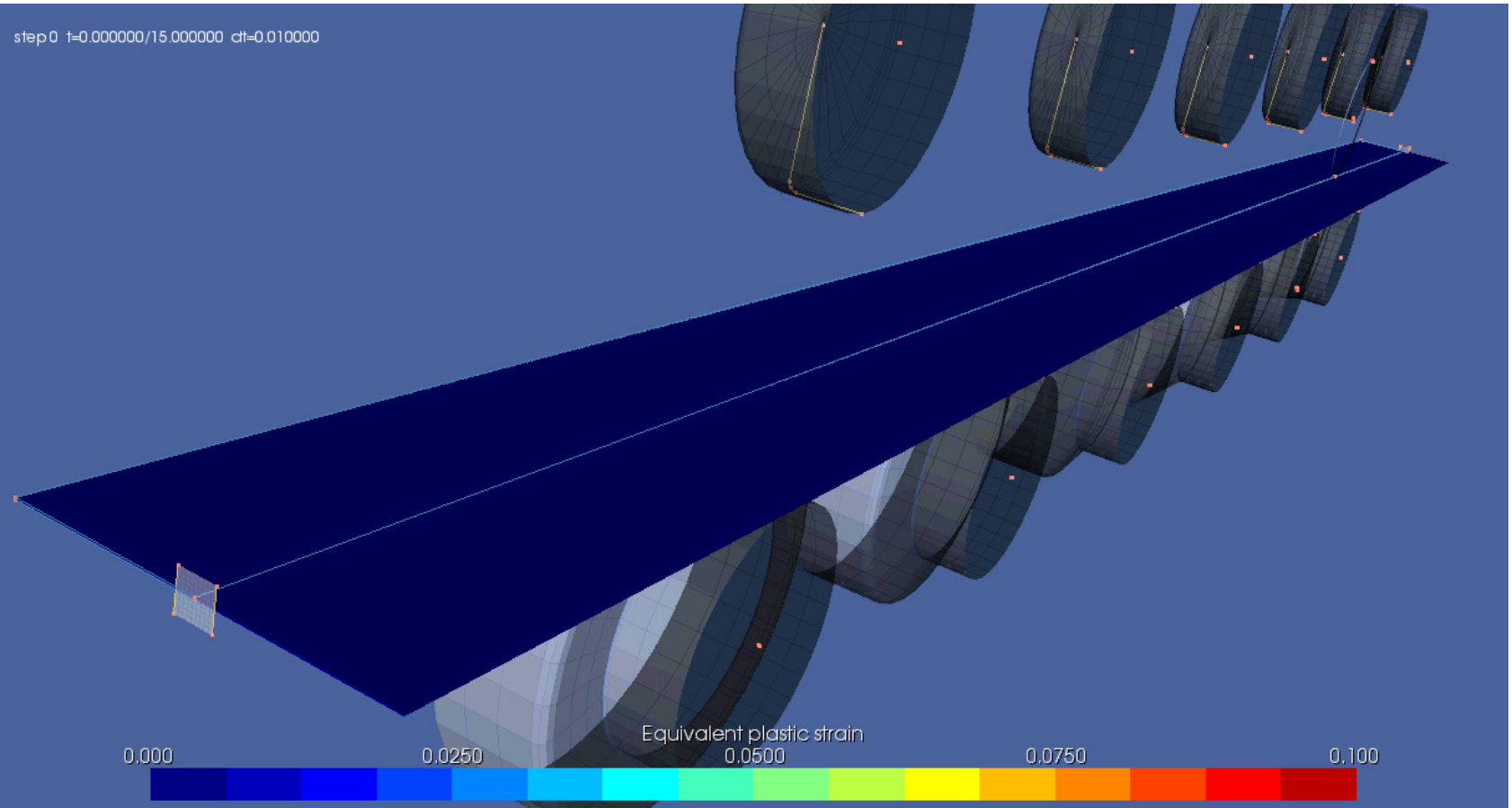
Moyens utilisés

- C++/python : langage commun
- CVS : 1-2 releases / semaine
- Batterie de tests : 632 tests au 01-01-2005
- Doxygen : documentation programmeur
- Site web : documentation utilisateur
- Règles de prog. : Ellemtel (simplifié)

Exemple : Simulation du profilage d'une tôle







Metafor et ses composants

- Composants externes
- Extensions d’Oofelie
- Composants internes

Composants externes

Oofelie ($)

- (Fragments de) code EF commercialisé par *Open Engineering* (Samtech).
- Fournit une libraire mathématique de base (matrices, solveurs).
- Fournit une interface EF (gestion des ddls, partitions, fixations, etc).
- Décembre 2004 : version réduite, figée et améliorée pour Metafor.

Nurbs++ (LGPL)

- Gestion des B-Splines pour modéliser des outils (contact).

VTK (LGPL)

- Interface graphique – visualisation des résultats.

Qt ($)

- Interface graphique – configuration de l'affichage VTK.
- Gestion des threads graphiques.

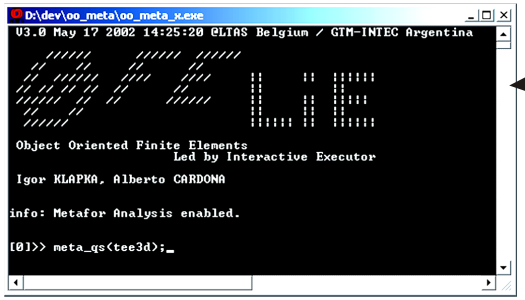
Extensions d'Oofelie

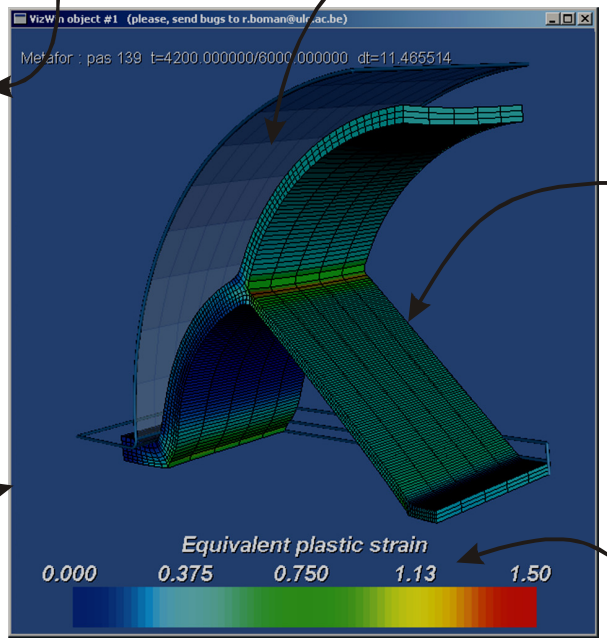
Algorithms' library

- Quasi static
- Implicit dynamic (HHT/Newmark/Chung Hulbert/SMG)
- Explicit dynamic (Centered differences/Chung Hulbert)
- Thermomechanical staggered / fully coupled schemes
- Combined Implicit/Explicit
- ALE formalism (using finite volumes for convection)

Geometry library

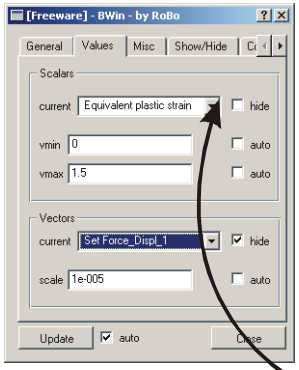
- the problem is entirely defined on the geometry (boundary conditions, materials,...)
- complex curves/surfaces (Nurbs)
- highly optimised for contact
- transfinite 2D/3D meshers





Elements' Library

- volumic:
  - 2D/3D Large strain thermomechanical
  - Non linear springs
- boundary:
  - Pressure, convection, thermal flux
  - Rigid/Flexible Thermo Contact (penalty)

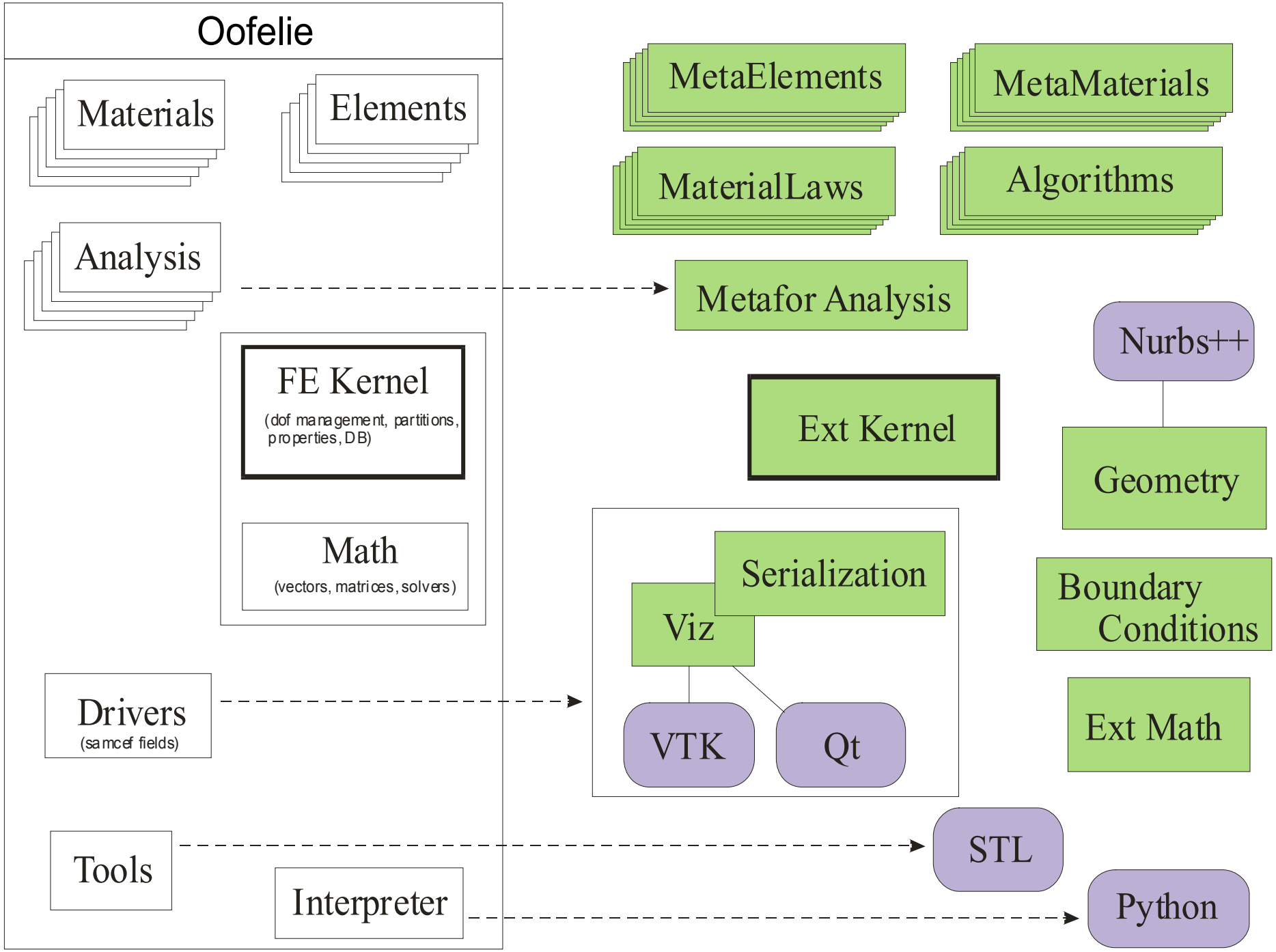


Materials' library

- ThermoElastoViscoPlastic behaviour
- Damage & microstructural evolution
- Hypo & Hyperelastic formulation

Visualisation library

- Real time visualisation using VTK/Qt libraries
- Automatic generation of movies
- Bitmap/Vector postscript export


Oofelie
Materials
Elements
Analysis
FE Kernel
(dof management, partitions, properties, DB)
Math
(vectors, matrices, solvers)
Drivers
(samcef fields)
Tools
Interpreter
MetaElements
MetaMaterials
MaterialLaws
Algorithms
Metafor Analysis
Ext Kernel
Nurbs++
Geometry
Boundary Conditions
Ext Math
Serialization
Viz
VTK
Qt
STL
Python

Utilisation du langage C++

- Avantages du C++
- Inconvénients du C++

Avantages du C++

- Permet de programmer plus facilement en « orienté objet » que C, Fortran.
  - Design patterns (Gamma & al).
  - Refactoring (Fawler) : http://www.refactoring.com.
  - Interaction minimale entre 2 développeurs (classes).
  - Réutilisation de composants sans connaître les détails d'implémentation (encapsulation des données).
  - Algorithmes génériques (abstraction et polymorphisme).
  - STL : conteneurs et algorithmes standards.
- Tout en gardant une rapidité d'exécution correcte (vs Java, Python, etc).
- Interfaçage possible avec tous les langages.
- Nombreuses librairies (calcul, graphisme) disponibles.

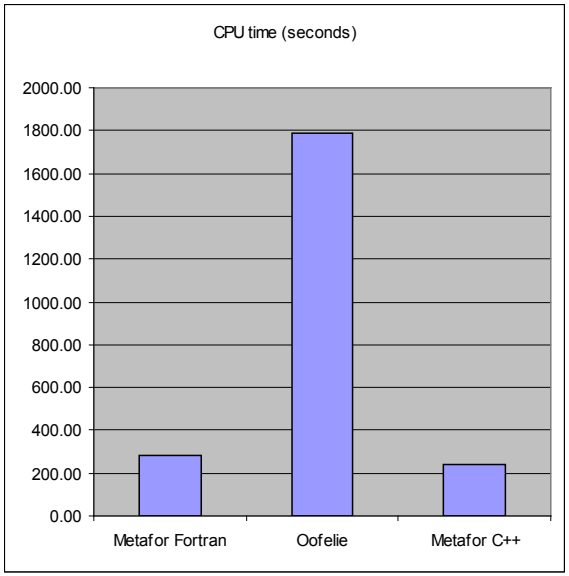
➢ Informatique intelligente.

Inconvénients du C++

- Difficulté de programmer « orienté objet »
  - Connaître la syntaxe du langage n'est pas suffisant.
  - Nécessité de lire énormément avant d'être opérationnel (>1mois)
  - Etudiants non formés (à l'informatique en général)
  - Directeurs de labo difficiles à convaincre (idée: « un bon informaticien != bon scientifique »)

C++ in Action (Milewski) : http://www.relisoft.com/book/index.htm

- Lenteur du code
  - Mal coder en C++ => coût CPU élevé
  - Nécessité d'utiliser un bon outil de profiling et vérifier le CPU à chaque release.

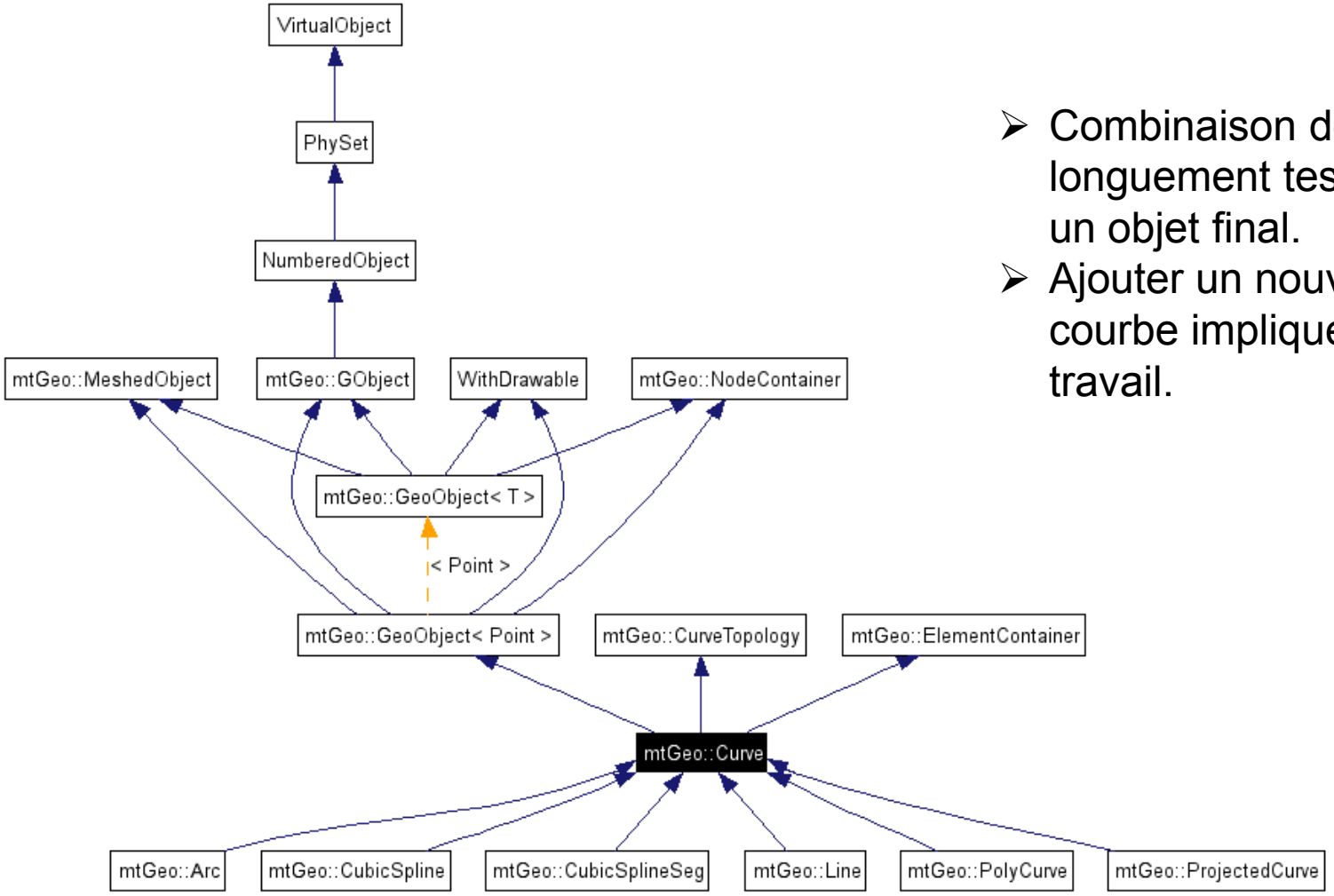


Zoom sur Metafor

- Enrichir Metafor
- Utilisation de schémas de conception
- Réutilisation de code existant

Enrichir Metafor

Exemple : hiérarchie de classes pour une courbe géométrique



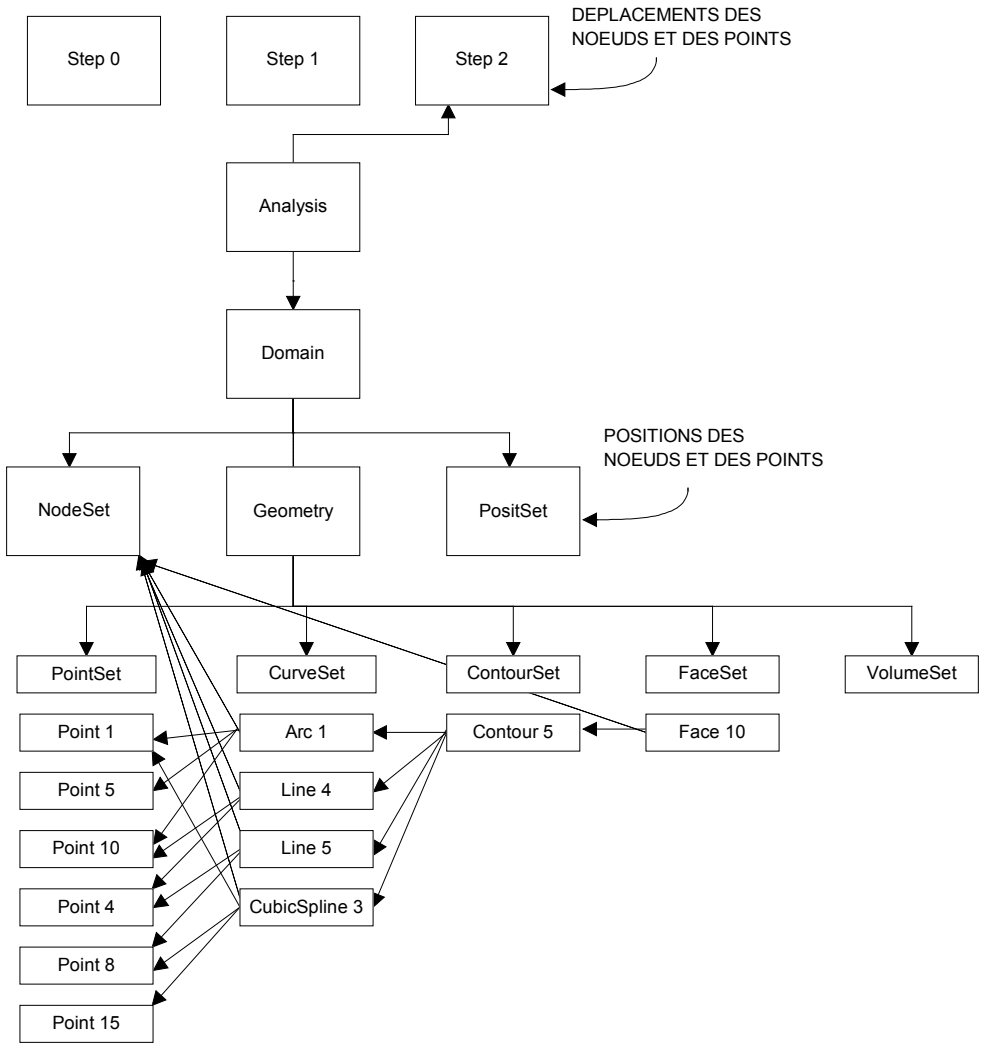
- Combinaison de fonctionnalités longuement testées pour former un objet final.
- Ajouter un nouveau type de courbe implique très peu de travail.

Design patterns

Utilisation de schémas de conception de la littérature (Gamma & al) comme des recettes de cuisine

Exemple: Communication entre objets par « *Chaîne de responsabilité* »

- Arrangement des objets en arbre et transmission des requêtes d'un objet à son « père » si échec.
- Permet d'accéder à n'importe quoi de n'importe où.
- Classes « *Proxy* » pour accélérer les requêtes.

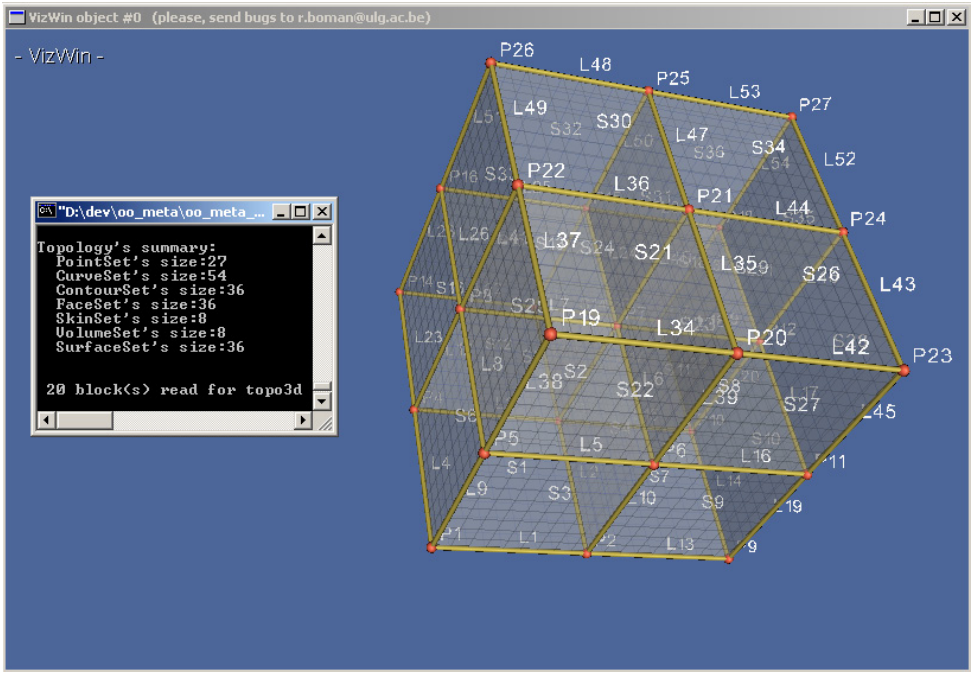


Réutilisation du code existant

Exemple : Géométrie du maillage EF

- Chaque élément fini possède une géométrie complète (sommets, arêtes, facettes, surfaces, etc) : la maille associée
- Réutilisation des outils géométriques au niveau de l'élément.

➢ Gestion du contact « rigide - défo » et « défo – défo » identique.



Autre exemple : (templates)

➢ Code des éléments 2D et 3D identique.

Interpréteur python

- Python
- Metafor et python
- Exemple

Qu'est ce que c'est ?

- Langage interprété orienté objet (Guido van Rossum)
- S'interface avec TOUS les langages : souvent appelé « *the glue language* »

Pourquoi Python et pas Perl, Tcl, Ruby, etc ?

- Interfaçage automatique (2 lignes par classe) via SWIG
- Python possède toutes les caractéristiques du C++ (héritage multiple, exceptions)
- Syntaxe similaire au C++
- Génération automatique des « wrapper classes »

oofelie.i

```
%module oofelie
%{
#include "oeDomain.h"
%}

%include "oeDomain.h"
```

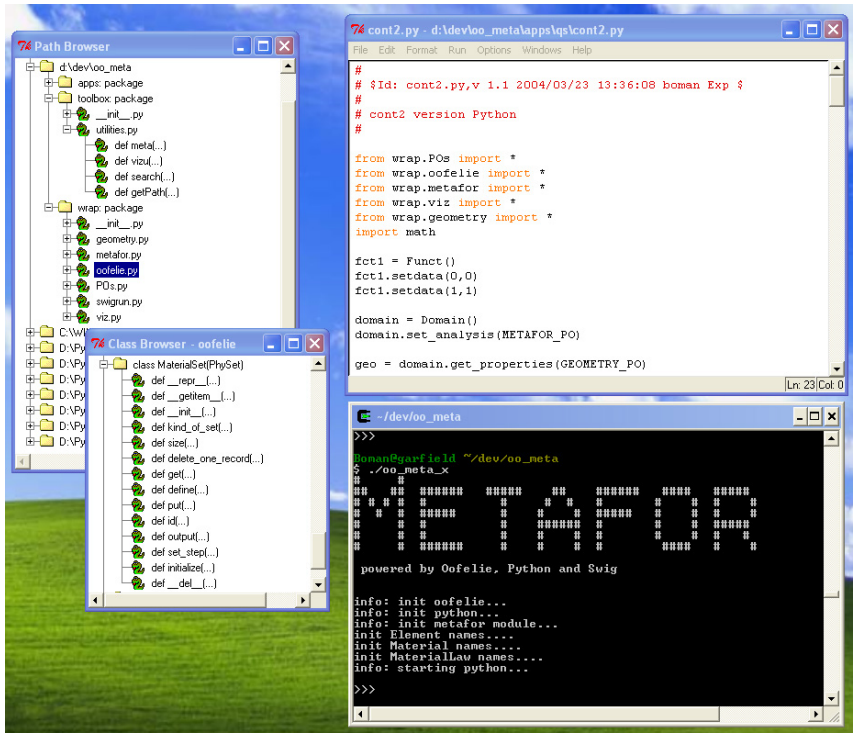
SWIG →

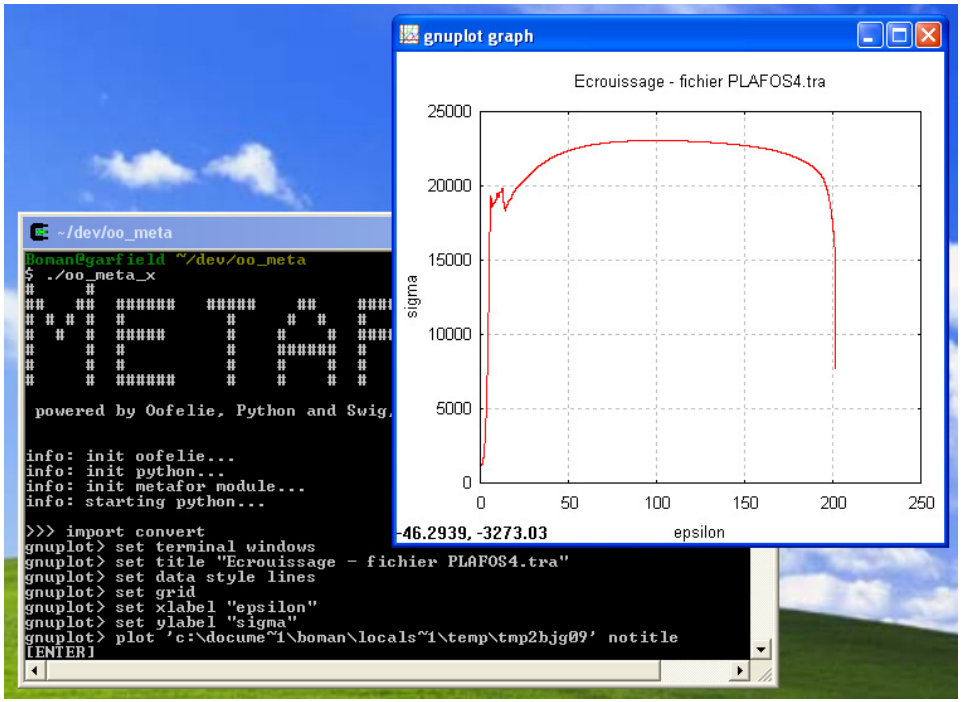
oofelie.py

```
from oofelie import Domain

domain = Domain()
...
```

Metafor et python

- Fournit une ligne de commande (cfr Matlab) à Metafor
- Supprime la nécessité d'un logiciel de préprocessing / parser, etc
- Permet l'écriture d'algorithmes interprétés
- Interface Metafor avec n'importe quel autre programme (GnuPlot, Matlab, etc)
- Permet de lancer Metafor à travers un navigateur web (CGI)
- …

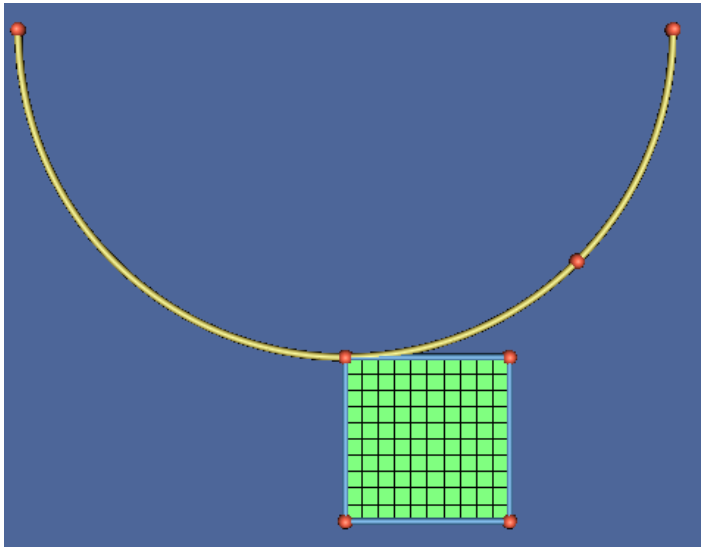




Exemple Python

```
domain = MetaDomain()
domain.setAnalysis(METAFOR_PO)

geo = domain.findObject(GEOMETRY_PO)
geo.setDimPlaneStrain()

# Parameters --

Lx = 1.0
Ly = 1.0

# Geometrie --

poiset = geo.findObject(POINTSET_PO)

poiset.define(1,   0,    0, 0)
poiset.define(2,   Lx,   0, 0)
poiset.define(3,   Lx,  Ly, 0)
poiset.define(4,   0,   Ly, 0)

poiset.define(5,   2*Lx,            Ly+2*Lx, 0)
poiset.define(6,   2*Lx*math.cos(math.pi/4),
                   Ly+2*Lx-2*Lx*math.cos(math.pi/4), 0)
poiset.define(7,  -2*Lx,            Ly+2*Lx, 0)

curset = geo.findObject(CURVESET_PO)
l1 = Line(1, 1, 2)
l2 = Line(2, 2, 3)
l3 = Line(3, 3, 4)
l4 = Line(4, 4, 1)
l5 = Arc(5, 5,6,7)
curset.copy(l1,l2,l3,l4,l5)
del l1;del l2;del l3;del l4

wireset = geo.findObject(WIRESET_PO)
w1 = Wire(1)
w1.push(1)
w1.push(2)
w1.push(3)
w1.push(4)
wireset.copy(w1)
```

```
sideset = geo.findObject(SIDESET_PO)
s1 = Side(1)
s1.push(1)
sideset.copy(s1)

#geo.output()

# Maillage

n1=10
n2=10

def ellength(s):
    return Lx/float(n1)
ellength = PythonOneParameterFunction(ellength)

mesher = DensityMesher1D(curset(1), ellength)
mesher.execute()
mesher = DensityMesher1D(curset(3), ellength)
mesher.execute()

curset(2).mesh(n2)
curset(4).mesh(n2)

sideset(1).mit(1)
```



Perspectives

Perspectives

Architecture

- Décomposition du code en DLLs.
- Chargement dynamique de matériaux, éléments, propriétés.
- Parallélisme.
- Interface web CGI.
- Interface « produit métier » dédiée au profilage (1 fichier python).
- Interface graphique plus conviviale (sans « console »).