Université de Liège

Département d'aérospatiale et mécanique

Développer du code avec une équipe de « non-geeks » à l'ULg

Romain BOMAN

Plan


Contexte et passé


Environnement de travail

Interface python


Gestion du code source


Documentation


Batterie de tests


Gestion mémoire


Visualisation

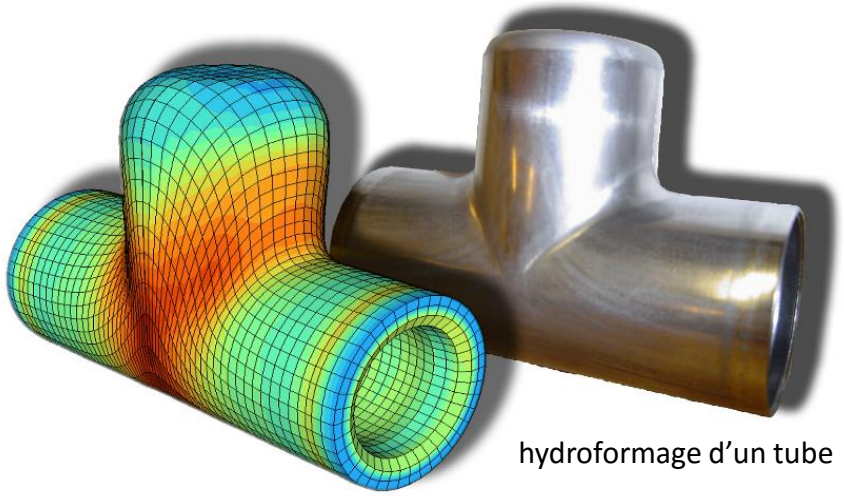

Futur

Position du labo à l'ULg

Université de Liège

UNIVERSITÉ de Liège Sciences appliquées

Département
Aérospatiale & Mécanique



hydroformage d'un tube

Mécanique Numérique Non Linéaire

- Simulation numérique.
- Mécanique du solide.
- Méthode des éléments finis.
- Développement de logiciels.

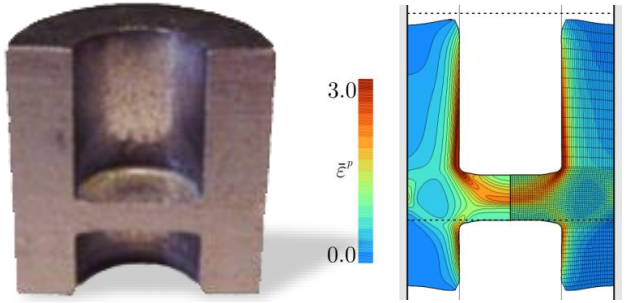
Keywords

J.-P. Ponthot

Notre logiciel : Metafor

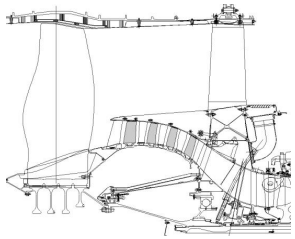
Code de calcul Element Fini implicite
pour la simulation de grandes déformations de solides

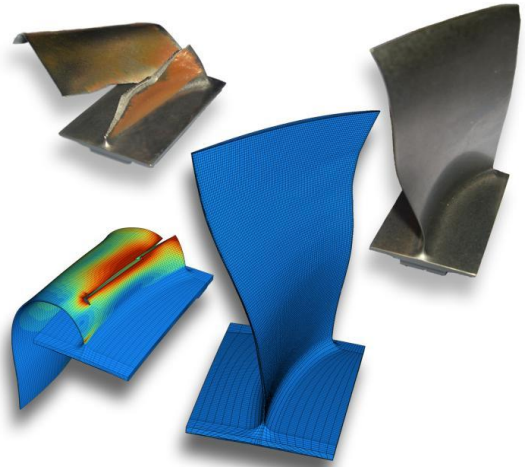
Applications « Metal Forming »



- Formalisme ALE, remaillage.
- Schémas d'intégration thermomécaniques.

Crash / Impact



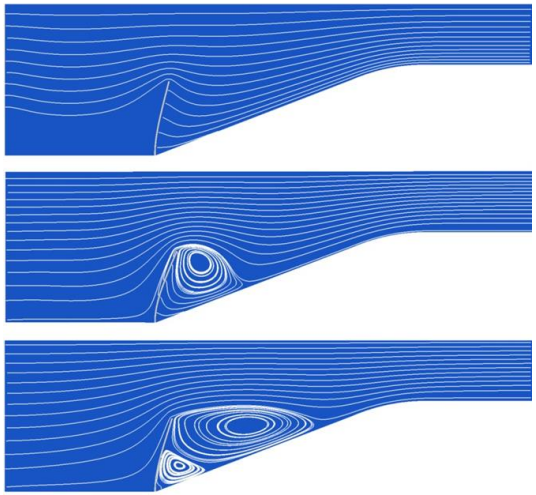


- Modélisation de fissures, rupture.
- Algorithmes de contact.

Biomécanique

- Loi constitutives non linéaires.
- Génération de maillages à partir d'images médicales.

Interaction fluide/structure



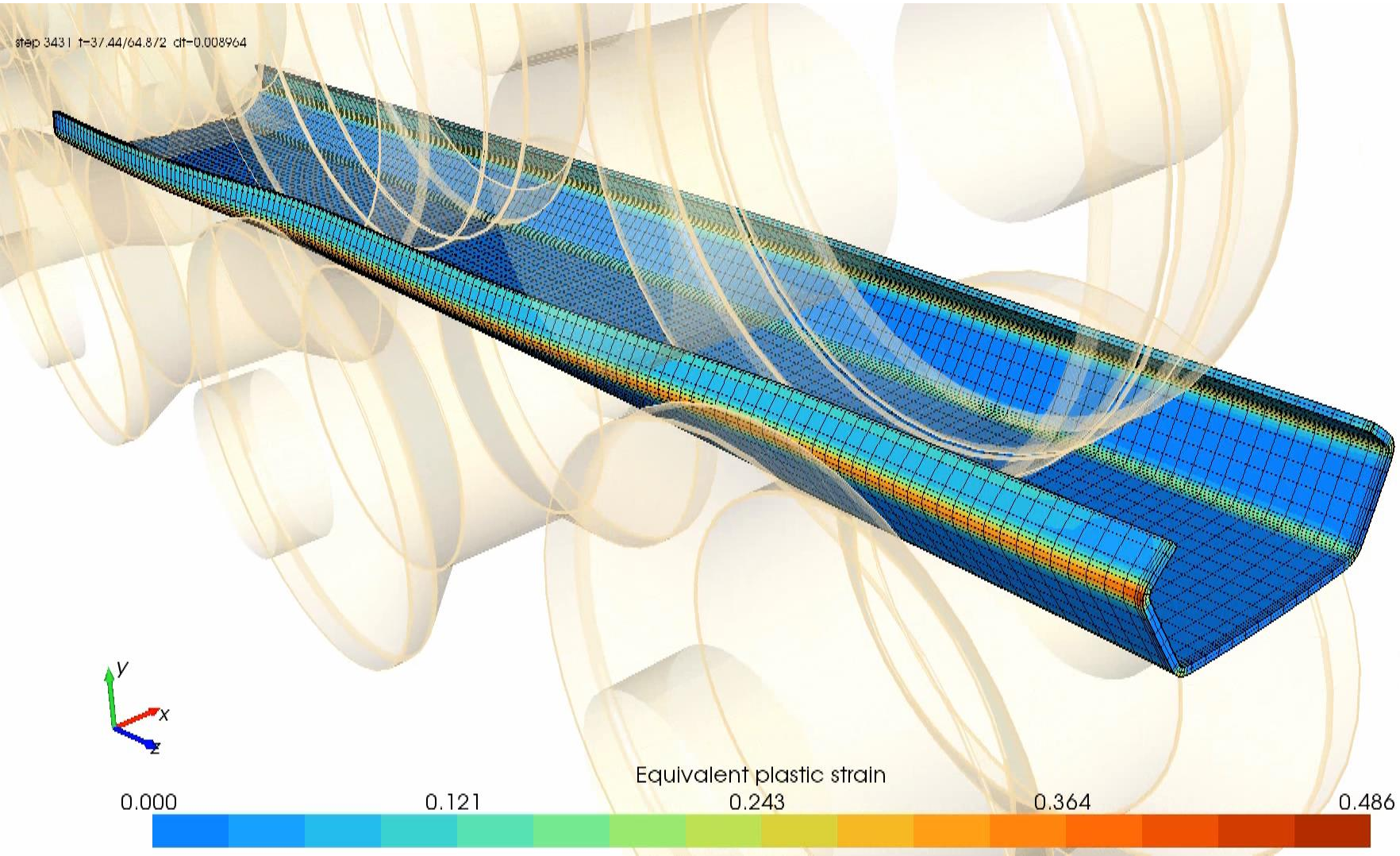
- Eléments fluides.
- Schémas monolithiques.

Simulations phares

Simulation d'une ligne de profilage

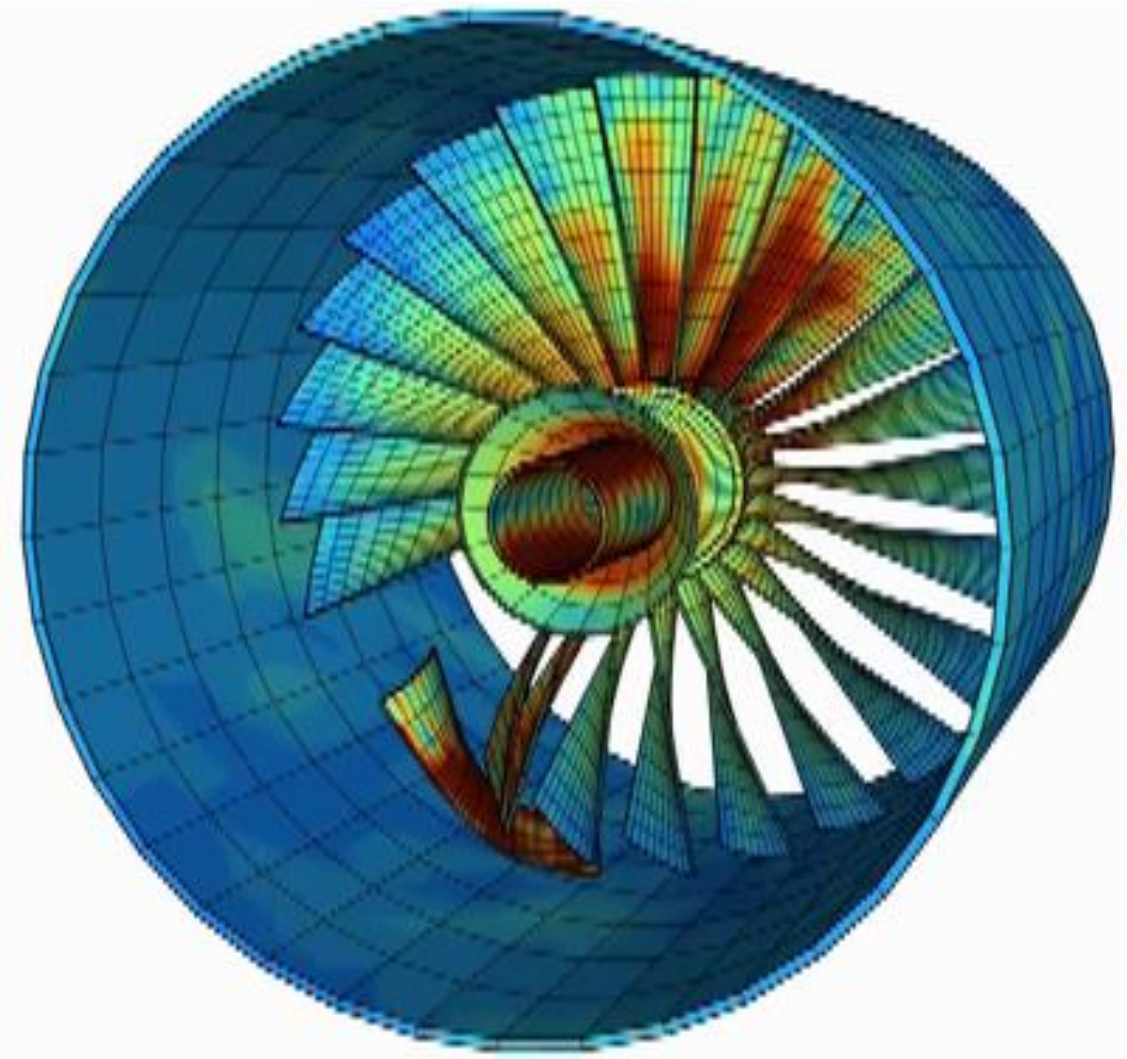
mise à forme de poutrelles à partir de tôles d'acier

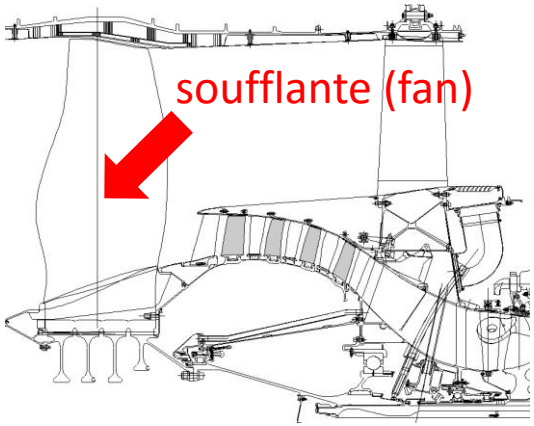
ArcelorMittal



Simulations phares

Simulation d'un test de certification de moteur d'avion (perte d'aube)



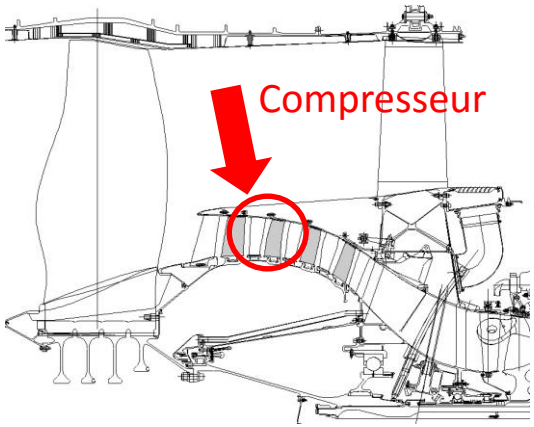


Turboréacteur double flux

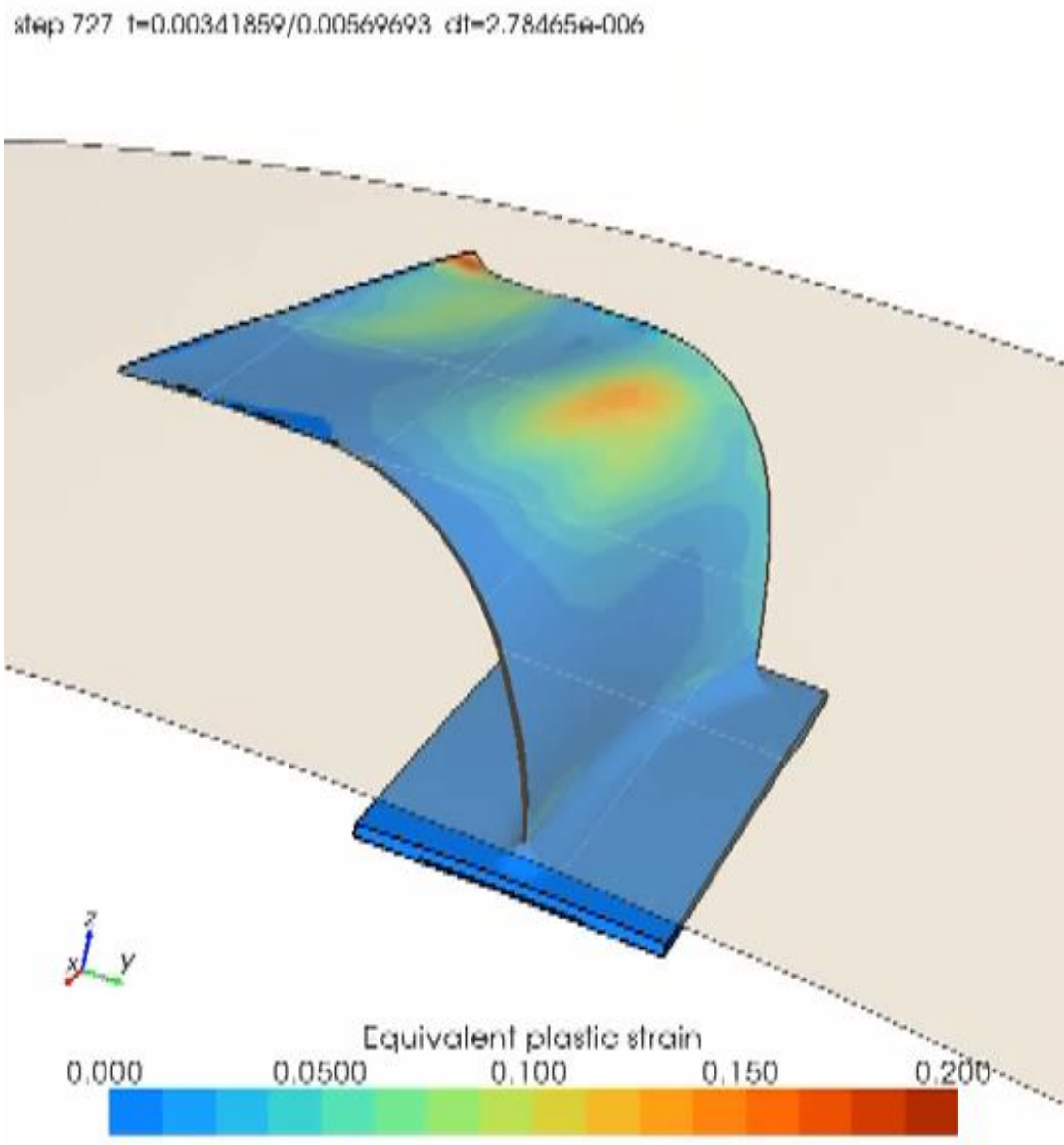
SAFRAN
Techspace Aero

- Test obligatoire très coûteux.
- On essaye ici de prédire le comportement avant le test réel.
- Une aube se détache suite à un impact d'oiseau par exemple.
- **L'aube perdue et les débris doivent rester contenus dans le carter**.

Simulations phares



- Suite à une perte d'aube l'axe du moteur vibre (balourd).
- Les aubes du compresseur basse-pression (Techspace Aero) s'écrasent sur le carter.
- Le carter ne doit pas céder.
- Néanmoins, on aimerait un carter le plus léger possible.
- Metafor permet de modéliser finement le comportement non linéaire du matériau pour prédire une **charge de flambement plus faible** que les modèles élastiques linéaires traditionnels... On espère donc un **gain de poids.**



Equipe actuelle

D. Boemer

R. Boman

C. Canales

Y. Carretta

M.L. Cerquaglia

Y. Crutzen

G. Deliége

C. Hennuyer

P. Joris

L. Papeleux

G. Wautelet

5 thésards – 4 ingénieurs de recherche – 1 scientifique permanent

Anciens développeurs


A. Stephany


E. Biotteau


C. Laurent


J. Xhardez


L. Vigneron


L. Adam

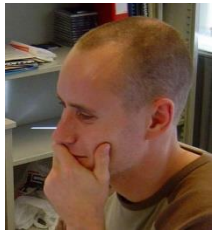

L. Noels


L. Ziane


M. Mengoni


O. Karaseva


P. Bussetta


P.-P. Jeunechamps


R. Koeune


S. Hannay


S. Trichon


V.Q. Bui


V. d'Otreppe


W. Guo

contribution au code actuel uniquement – depuis 2000

http://metafor.ltas.ulg.ac.be/dokuwiki/team

Partenaires industriels

SAFRAN Techspace Aero

S.A.B.C.A.

sonaca BELGIUM

ArcelorMittal

GDTECH engineering

geonx

open engineering

SAMTECH

SIEMENS

détient une licence de Metafor

Metafor... de 1992 à aujourd'hui

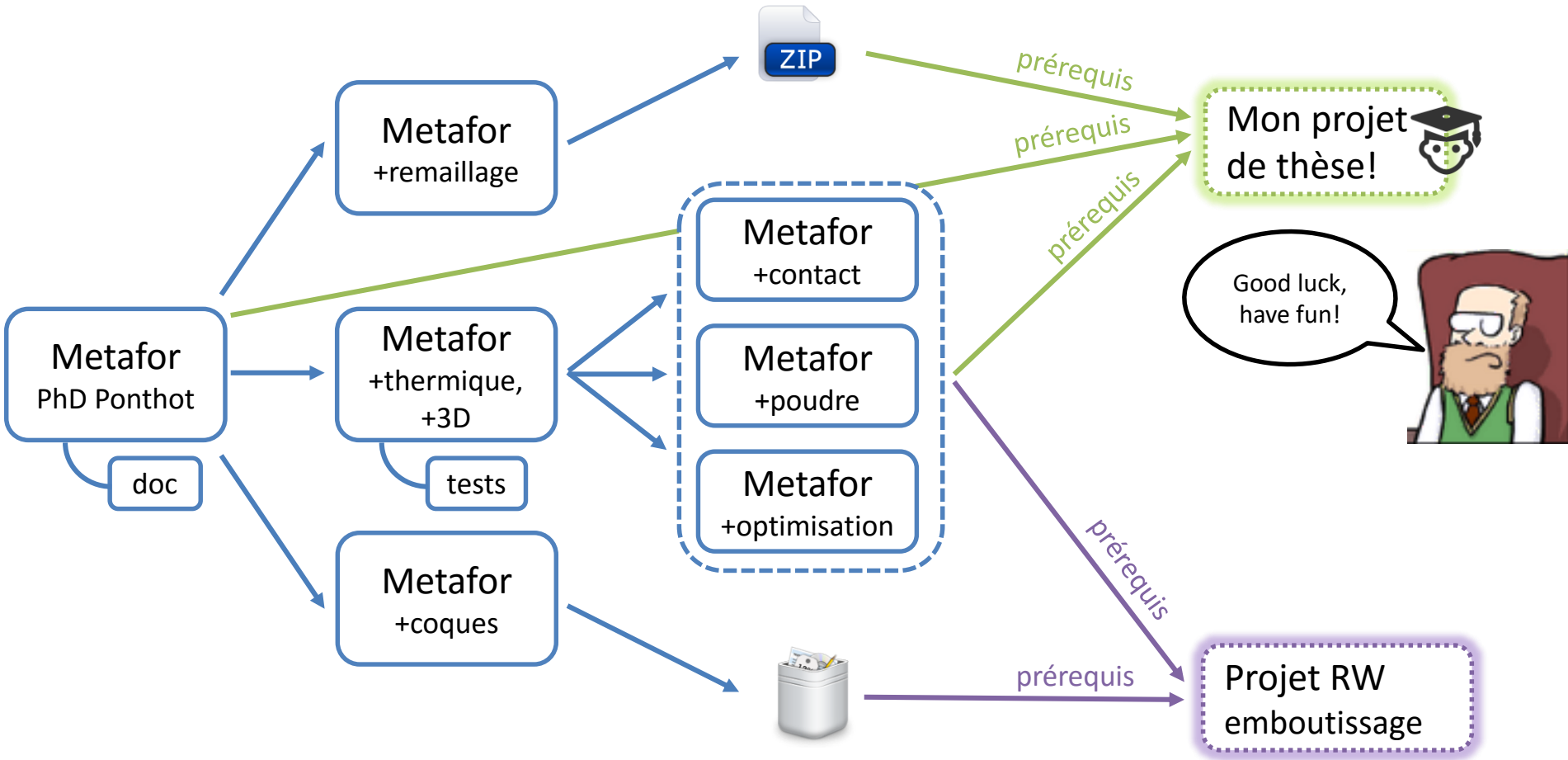
J'ai eu une super idée de projet:
On va combiner les résultats de notre projet A et de la thèse B pour développer C!

On a gardé le code source de A et B?

Je ne sais pas, mais le projet est accepté!
On démarre demain!

images de http://geek-and-poke.com/

Mon arrivée au labo en 1997



Etat de Metafor en 1997

Code source
- Fortran 77
- Gestion manuelle des révisions (`cp/tar` à partir un répertoire NFS partagé).

Documentation
- Une vieille note technique de 1992 imprimée.

Tests
- Procédure non systématique et incomplète.

Compilation
- Un `Makefile` mal foutu (compilation en debug par défaut).

Portabilité
- Nulle: le code dépend étroitement d'une lib propriétaire dont on n'a pas les sources.
- Compilation uniquement sous DEC Alpha / VAX.

Terminal VT220

Institut de Mécanique du Val Benoît

Etat de Metafor en 1997

Très longue tradition de **Fortran 77** en simulation par éléments finis...

En **2016**, tous les grands frères de Metafor sont toujours écrits en Fortran 77!

...et même en calcul numérique plus généralement...
(cfr BLAS: http://www.netlib.org/blas/)

Etat de Metafor en 1997

Il est bien évidemment possible d'écrire un bon code en Fortran.
La **rigueur** nécessaire et **l'effort de maintenance** sont juste colossaux!


```
CALL META
     (S,IREST,LS,INOE,NPRDOM,NTOPDO,P,Q,
      X00,X0,X1,X2,LOCSIG,LOCEL,LOCEL2,LOCNOD,
      FINT,FEXT,IDENT,ACR,NREMA,T,DEPL,NREAC,
      DEPKHI,NREKHI,FIMP,NFI,LOCS,LOCSIT,S(I2),
      S(I3),S(I4),DX2,VIT,ACC,DIAMAS,SMS,DJ,
      DETJ0V,DETJ1V,SIG0,SIG1,EPL0,EPL1,BETA1,
      ,NEPG1,S(I5),AMAT,LOCP,LOCP2,ALOCP,
      ANGR,LOCC,LOCC2,VLOCO,ICONT,CODIR,
      T0,RCONT1,LMATER,PMATER,NAEL,
      ,RAY1,DRM,MDE,MAINOE,IMPTIME,IMPDDL,
      EPS0,EPS1,NOTIN,VALTIN,NFIXT,VALFIX,
      LOCLT,LOCLT2,ALOCLT,DETFV0,DETFV1,
      F0,F1,
      SIGV0,SIGV1,SIGH0,SIGH1,
      INTMET,LISNOE,LISMAI,HGQ0,HGQ1,
      ALP0,ALP1,ICTM,TT,STPG,HHT,HHTA,
      CONMAS,LIA,LIA2,END0,END1,COEHYP,
      DCHYP,NAELCL,DXT,RHO0,RHO1,FTCAP,
      AMULA0,AMULA1,FPENAL,XFON,MATMOR,
      IDON_INV(1+IPOPT),IDON_INV(1+IPOPT+NXOPT),
      RDON_INV(1+IPOPT),
      RES_INV(2),RES_INV(2+NPOIOPT),RES_INV(1),TRAV1JP,X1SSB,
      X2SSB,SIG1SSB,DETJ1VSSB,EPL1SSB,RAY1SSB,EPS1SSB,END1SSB,
      HGQ1SSB,ALP1SSB,DETFV1SSB,F1SSB,RHO1SSB,RCONT1SSB,UEFPLUS,
      UEFMOINS,SIG0SSB,DETJ0VSSB,EPL0SSB,RAY0SSB,EPS0SSB,
      END0SSB,HGQ0SSB,ALP0SSB,DETFV0SSB,F0SSB,RHO0SSB,
      SIGV0SSB,SIGV1SSB,SIGH0SSB,SIGH1SSB,
      RCONT0SSB,X0SSB,RESPLUS,X00SSB,PMATERSSB,VLOCOSSB,
      IDIFFSSB,DPSSB,VITSSB,ACCSSB,HHTSSB,HHTASSB,STPGSSB,
      PSSB,QSSB,X000,XM00,XM0,XM1,XM00SSB,
      XM0SSB,XM1SSB,S(I12),S(I8),S(I9),S(I13),
      S(I10),S(I11),S(I14),S(I15),S(I6),
      S(I7),IROBO,NNZ,DJSSB,UPWTMP,
      S(I16),S(I17),S(I18),S(I19),
      LOCGAU,LOCELM,LOCNG,MAIADJ,LOCSITG,FCL,
      MAIADJ2,LOCGAP,LOCSITP,
      S(I20),S(I21),S(I22),S(I23),
      NBRNOETFX,LNOETFX,NBRPTSFCT,FIXT_TIME,VALFCT,ALNPSY,
      DIAMT,ALNKPSY,ACCLNM1,ACCLN,FINERLN,FINERLNM1,MQLN,
      DX2LN,ALPN0,ALPN1,EVPL0,EVPL1,SMALLR0,SMALLR1,V_FCT)
```

aperçu d'un appel d'une routine de Metafor Fortran

Le Fortran 77 dans les mains de non-geeks:

- **pas d'allocation de mémoire:**
  - modification très intrusives (mémoire allouée en C et passée en argument),
  - tentation de réutiliser des variables temporairement non utilisées pour un but différent du but initial,
  - tentation d'utiliser des variables globales.

- **nécessité d'apprendre un autre langage** pour accéder à certaines fonctionnalités (allocation mémoire, widgets GUI, autres bibliothèques, etc.)
  - bénéfice de simplicité de syntaxe perdu!

Metafor se marie à Oofelie

En 2001, j'envisage d'abandonner ma thèse...

Je suis contacté par le WSL et Igor Klapka pour créer **Open Engineering** dont le but est de commercialiser **Oofelie**, un code éléments finis écrit en C++.

Je décide de réécrire Metafor dans Oofelie!

Le succès est énorme à tel point que trop d'acteurs entrent en jeu (WSL, Interface, ULg, LTAS, Samtech, Ansys) et se partagent le gâteau...

...sans moi! Ils m'ont oublié dans le partage... Je décide de me retirer du projet.

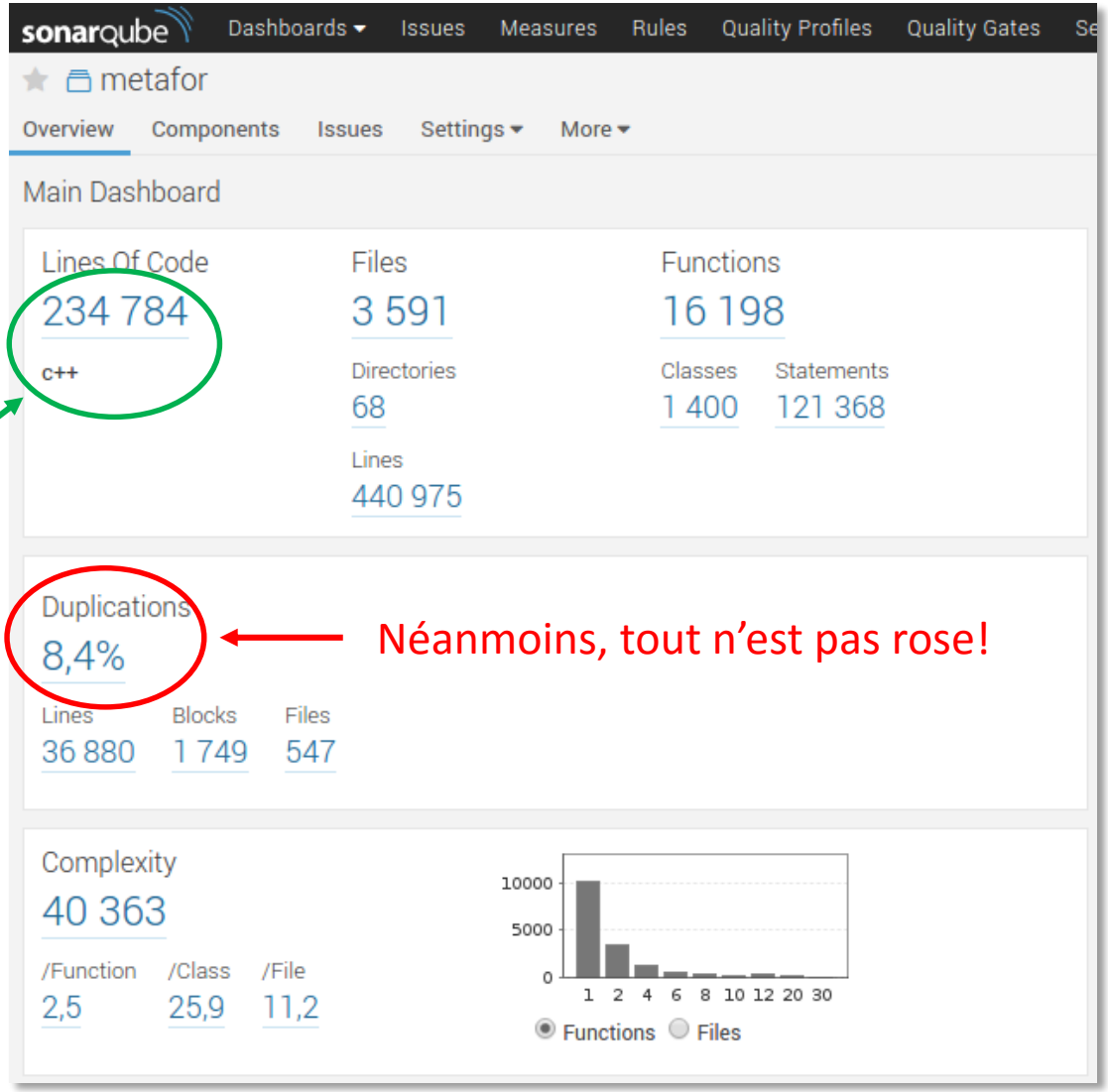
Après une négociation dure, **j'obtiens le droit d'utiliser mon propre code source** pour continuer ma thèse (mais pas de le mettre en open source).

Aujourd'hui

Une version **unique** depuis 2001.

Quasiment **aucun** développement perdu!

- Plus de 230k lignes de C++.
- 1400 classes.
- 33 modules (.dll/.so).
- ~3100 modèles EF testés.
- OS: Windows/Linux.



Néanmoins, tout n'est pas rose!

Environnement de travail

C’est génial! j’ai trouvé une super lib qui va me permettre de calculer le volume d’un cube.

Tu peux me la compiler rapidement sur toutes les machines? Je t’envoie le lien...

T’as essayé sur ton PC?

Essayé quoi?

images de http://geek-and-poke.com/

Environnement de travail

Cursus ULg des **ingénieurs aéro-méca** (et physiciens)

- Pas de cours de programmation/informatique.
- De moins en moins de projets où il faut programmer.

De plus:

- Connaissance OS limitée (y compris Windows!)
- « La programmation, c'est pas de la science! »

Au mieux:

- Connaissance (très sommaire) de Matlab.
- Parfois: connaissance d'un peu de python, C.

PROBLEME:
Ils veulent faire du « High-Performance Computing! »

Environnement de travail

Windows

Utilisateurs:
- Chercheur lambda.
- Etudiants.
- Industriels.

Avantages:
- IDE plus convivial (Visual Studio).
- Visualisation des résultats plus fluide (drivers graphiques optimisés).
- Pas besoin d'un deuxième OS pour MS Office, SolidEdge, etc.
- Compilation très rapide.

Inconvénient:
- Compilation des bibliothèques.

Linux

Utilisation:
- Calculs intensifs (clusters du CECI).

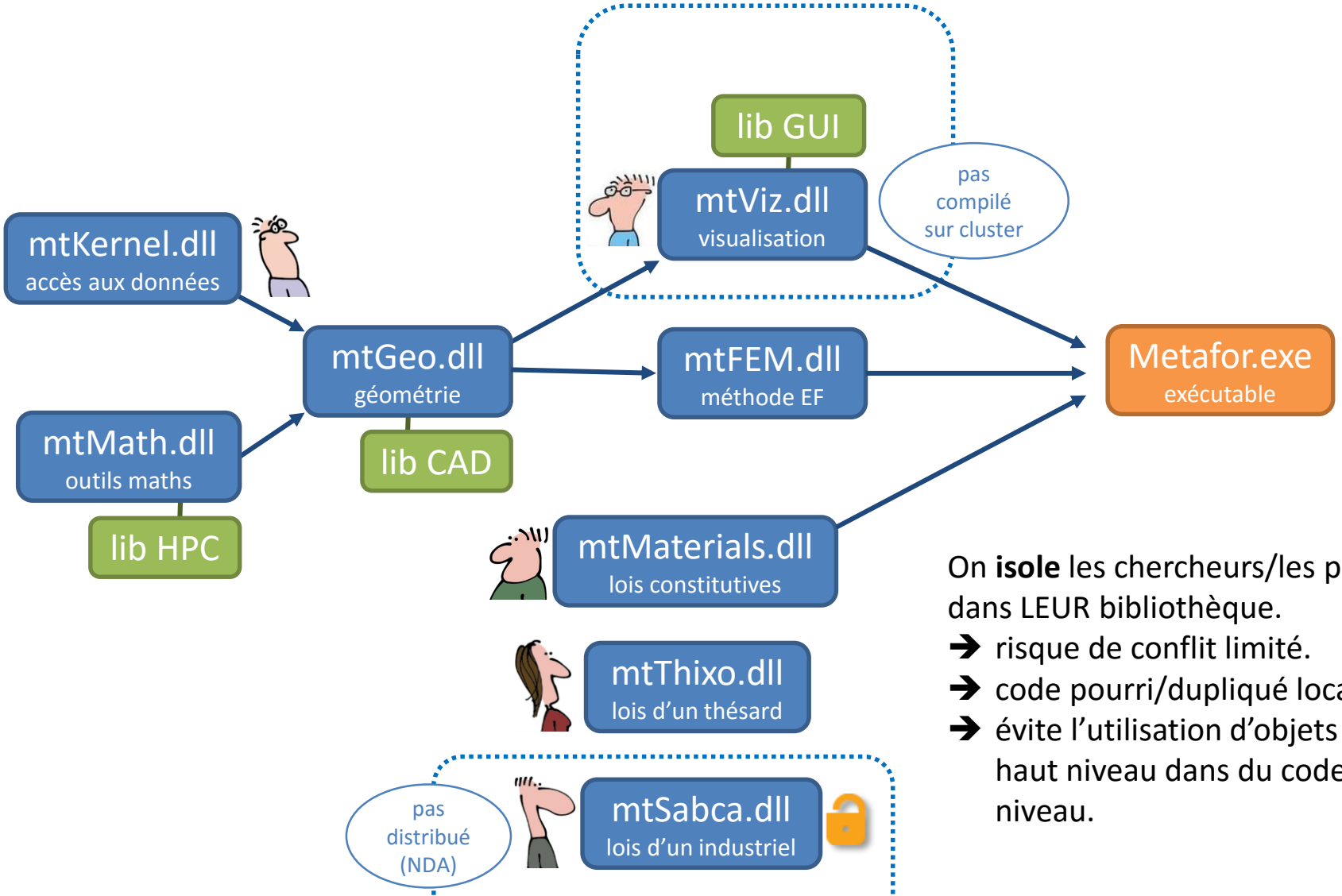
Avantages:
- Exécution ~10% plus rapide.
- Rien à installer en plus du système.
- Prototypes/essais simplifiés:

```
apt-get install new-lib
```

Inconvénient:
- Convivialité – IDE?

Environnement de travail

Le code source est scindé en **33 bibliothèques dynamiques** (.dll)



On **isole** les chercheurs/les projets dans LEUR bibliothèque.

- ➔ risque de conflit limité.
- ➔ code pourri/dupliqué localisé.
- ➔ évite l'utilisation d'objets de haut niveau dans du code de bas niveau.

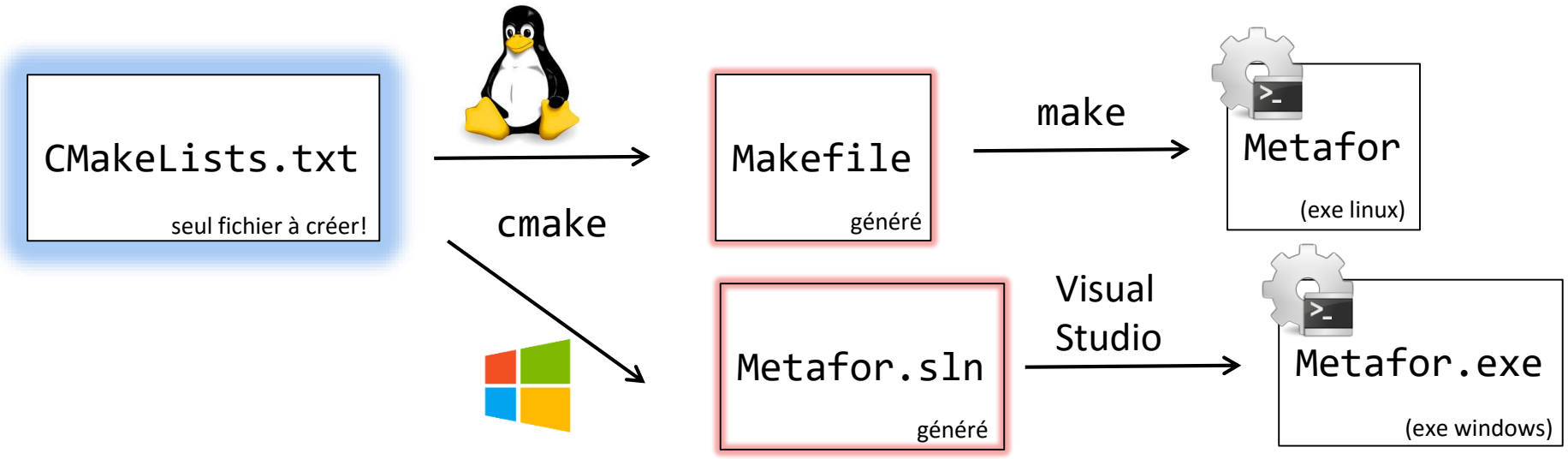
Environnement de travail

Compilation multiplateforme

CMake permet de générer un fichier `Makefile` qui sera utilisé par le programme `make` pour effectuer la compilation.

CMake est multiplateforme. Il est par exemple capable de générer de projets "Visual studio" sous Windows ou des projets “Eclipse" sous Linux.

CMake utilise un fichier `CMakeLists.txt` dont la syntaxe est très simple.



http://www.cmake.org/

Environnement de travail

Exemple de fichier CMakeLists.txt et compilation

CMake

CMakeLists.txt

```
cmake_minimum_required(VERSION 2.6)

project(Metafor CXX)

set(SRCS main.cpp
         class1.cpp
         class2.cpp)

add_executable(Metafor ${SRCS})
```

On définit la version minimale de CMake à utiliser

On définit un projet nommé `Metafor`. il est codé en C++ (CXX). Un projet peut contenir plusieurs programmes, des bibliothèques, etc.

On crée une variable SRCS qui contient la liste des 3 fichiers (`*.cpp`) à compiler.

On déclare un programme exécutable nommé « Metafor » et composé des sources définies par la variable SRCS.

```
mkdir build
cd build
cmake ..
make
./Metafor
```

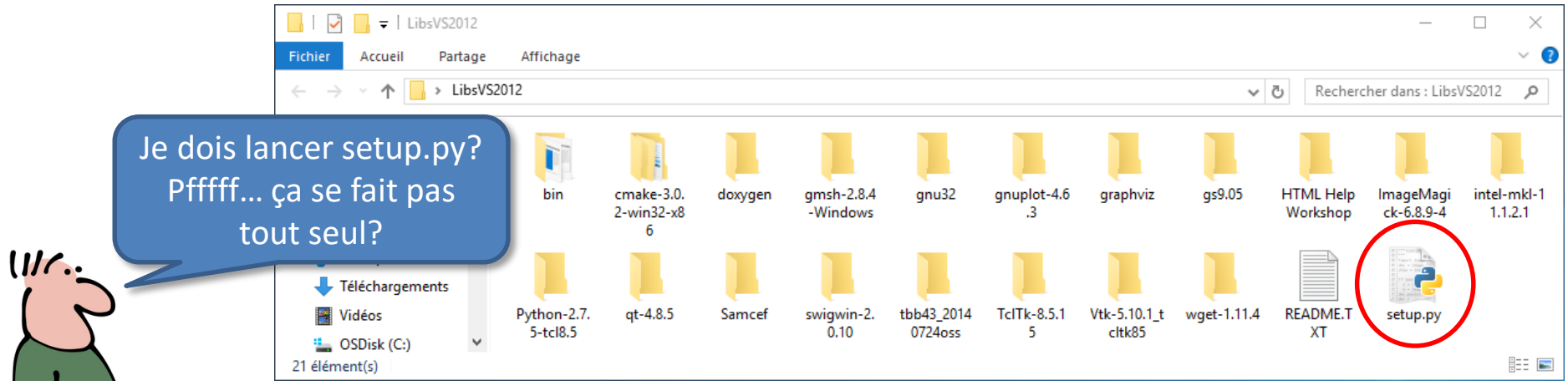
Environnement de travail

Choix des bibliothèques externes

Quelles versions utilise-t-on?

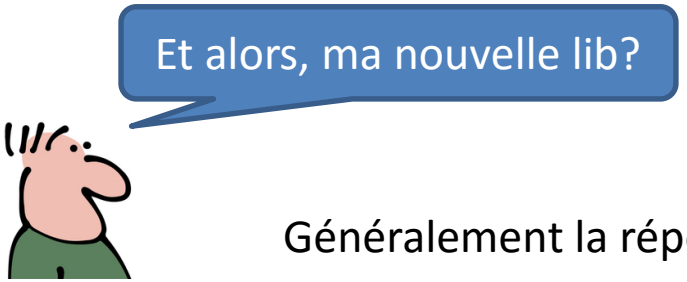
Linux:
on utilise les versions disponibles sous Ubuntu LTS
➔ compiler metafor = `sudo apt-get ...` + `svn co ...` + `cmake` + `make`
➔ python 2.7 et pas python 3.x

Windows:
on fournit des versions binaires release/debug de ces mêmes versions compilées avec le visual studio 2012, ainsi qu'un script d'installation.



Environnement de travail

Choix des bibliothèques externes



Généralement la réponse est non.

Questions qu'on se pose avant de dire oui:

- Possibilité de compilation Windows MSVC (release/debug)?
- La fonctionnalité voulue est-elle disponible dans une lib déjà utilisée?
- Quel risque et quel coût de maintenance sur le long terme?

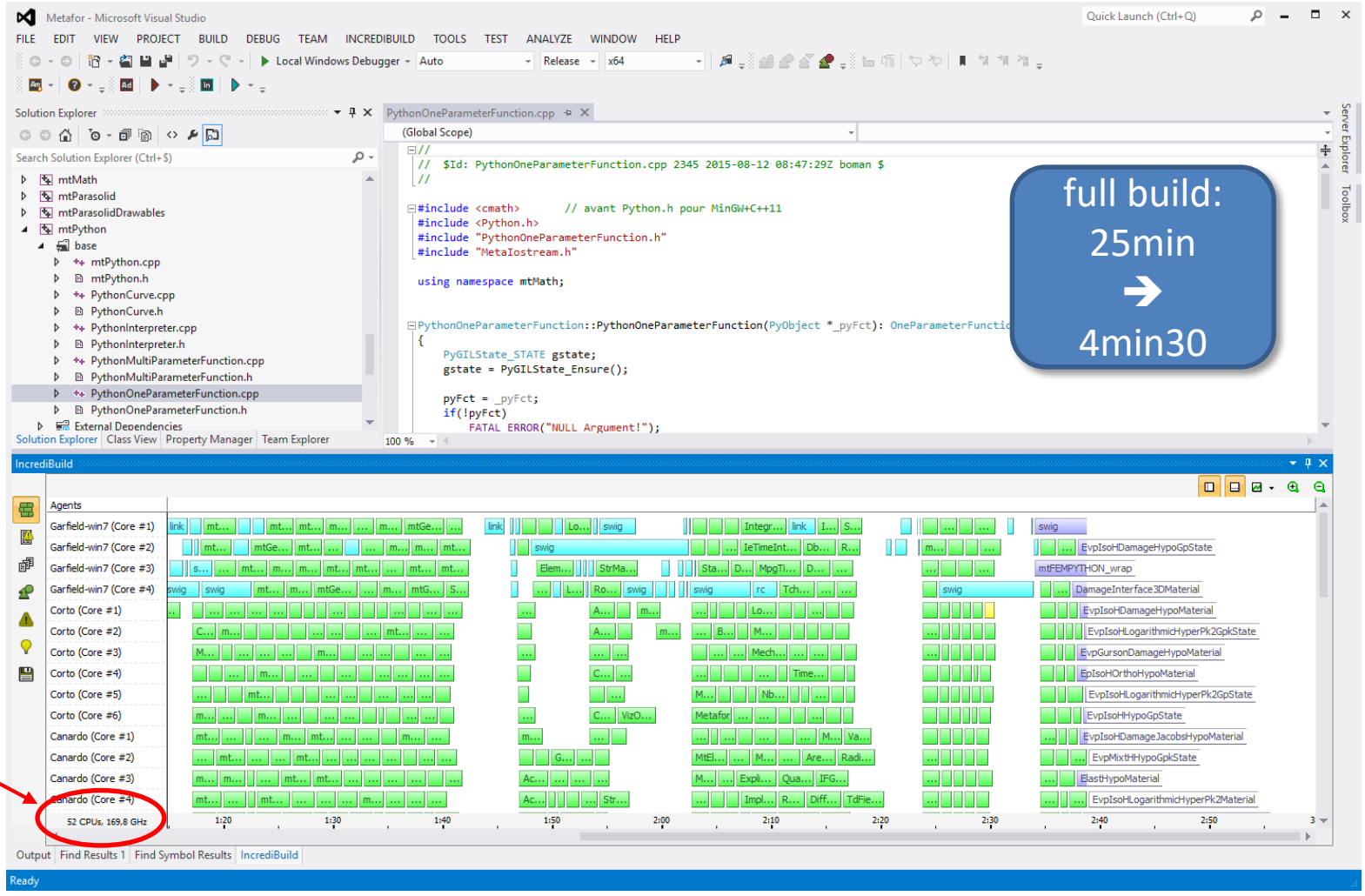
Environnement de travail

Compilation Windows distribuée

IncrediBuild
BEYOND ACCELERATION

https://www.incredibuild.com/

300$+24$/an
(prix univ par agent 8 cœurs)



Interface python

images de http://geek-and-poke.com/

Interface python

Les objets C++ de Metafor sont accessibles à travers une interface python dans 2 buts précis:

- **Ecriture des "jeux de données" des simulations**
  - **Moins de code**: pas besoin de parseur maison (toujours bugué).
  - **Langage complet**: utilisation de boucles, branchements conditionnels, objets dans le jeu de données.
  - **Extensibilité**: appel à des libs externes (Qt, wxWidgets, numpy, …)
  - **"Glue language"**: appel de codes externes (gmsh, SAMCEF, Abaqus, Matlab, etc.)
  - **Sécurité**: erreurs correctement traitées (y compris les exceptions C++!)
- **Extension du code ("user subroutines")**
  - Beaucoup de classes C++ (conditions aux limites, commandes de postprocessing, éntités géométriques, matériaux, mailleurs, etc.) peuvent être dérivées en python et utilisées dans le jeu de données.

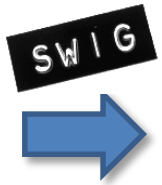
Interface python

Scripts Python comme « jeux de données »

Une classe Python est créée **automatiquement** par SWIG pour chaque classe C++

materials.i

```
%module materials
%{
#include "ElasticMat.h"
%}

%include "ElasticMat.h"
```



_materials.pyd (Module python compilé)

materials.py (Shadow classes)

JEU DE DONNEES

```
from materials import *

mat = ElasticMat(E, nu)

model.setMaterial(mat)
model.run()
```

Ajouter un nouveau matériau revient à ajouter seulement 2 lignes dans le fichier d'entrée de SWIG (`materials.i`) pour le rendre accessible en Python!

Interface python

Héritage de classes C++ en python : "user subroutines"

SWIG peut générer le code (énorme et complexe) requis pour dériver un classe C++ en Python!

MATERIAU C++ ELASTIQUE

```
class ElasticMat
{
public:
   virtual
   T computeStress(T &strain);
};
```

MATERIAU PYTHON ELASTO-PLASTIQUE

```
from materials import *

class ElasticPlasticMat(ElasticMat):

   def computeStress(self, strain):
      # compute stresses
      # from strains
      # using python cmds here…
      return stress
```

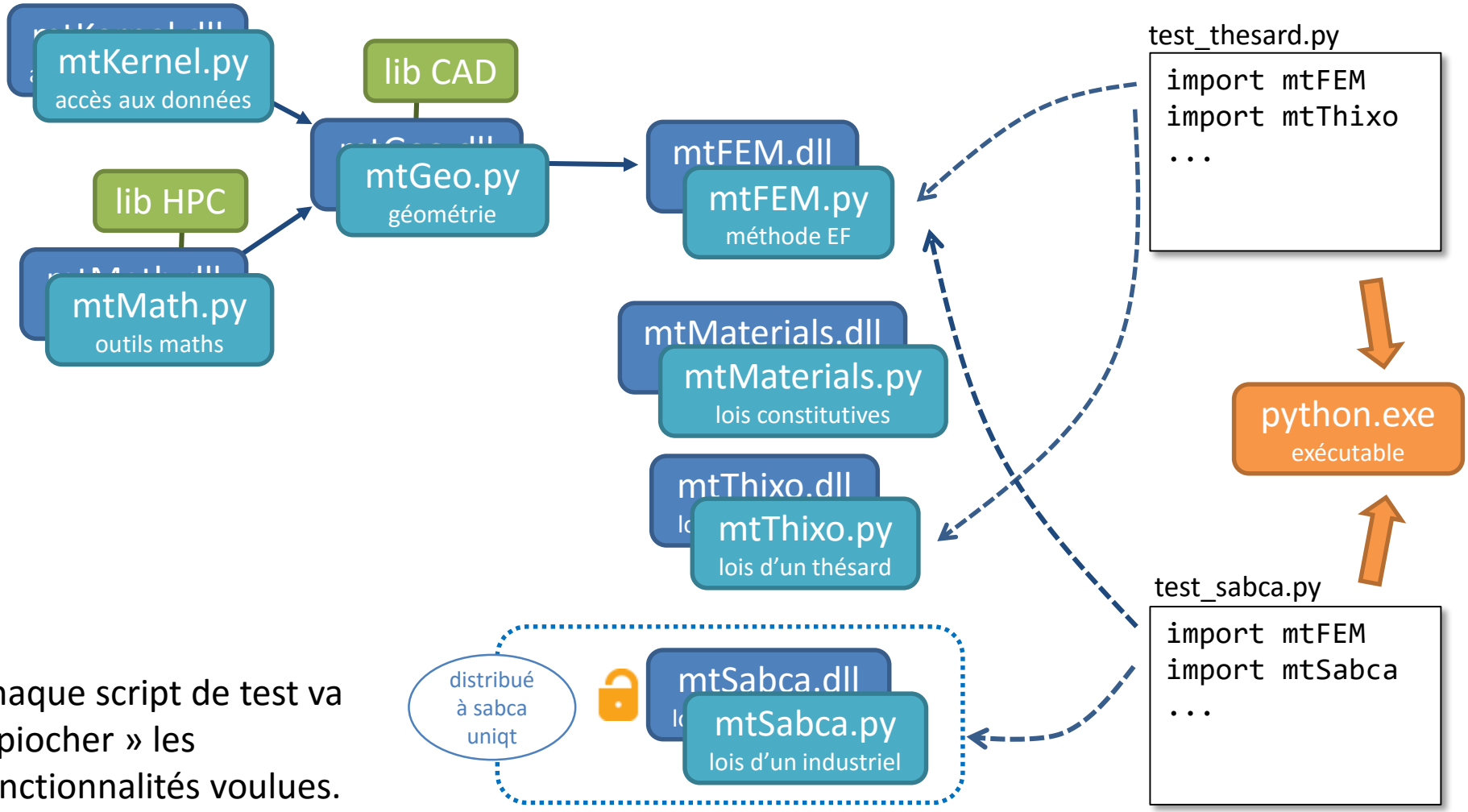
Ce code python sera appelé à partir du code C++!

Une nouvelle loi de comportement est disponible sans compilateur!

Très utile pour les étudiants (TFE)

Interface python

Le code source scindé en 33 bibliothèques dynamiques (.dll)
... et 33 bibliothèques d'interface python



Chaque script de test va « piocher » les fonctionnalités voulues.

Gestion du code source

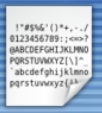
C’est quand que tu fais ton commit? J’aurais besoin de tes nouveaux trucs pour ma thèse.

Demain, ce sera fait. Et toi? ton commit?

Ah? Heu… Oui, oui… mais pas maintenant; j’ai un papier à réviser là…

images de http://geek-and-poke.com/

Gestion du code source



A relire: svn/git aux Geeks Anonymes

- Sébastien Jodogne : « *Programming practices and project management for professional software development* », mai 2013.
- Cyril Soldani : « *Git your life for fun and profit* », juin 2013.

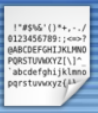
Buts

- Gestion de l'historique des modifications du code.
- Une seule version de référence pour tout le monde (~10 personnes).
- Conserver un maximum de développements du passé.

Historique de Metafor

- CVS jusqu'avril 2007
- SVN depuis lors
- Quelques sous-projets sous git depuis 2015

Gestion du code source



L'utilisation quotidienne doit être **très simple** car le non-geek **ne lira pas le manuel**...

...et menacera implicitement de ne pas commiter son travail si on insiste.

COMPROMIS TROUVE

1. On n'utilise pas de branches.
2. La version commitée est **toujours stable**.
3. Procédure de développement très rigide.

```
svn checkout
[modifs locales]
svn update
…
batterie de tests [1 nuit] (voir plus loin)
écriture d'une « page web de commit » (voir plus loin)
svn commit
```

Gestion du code source

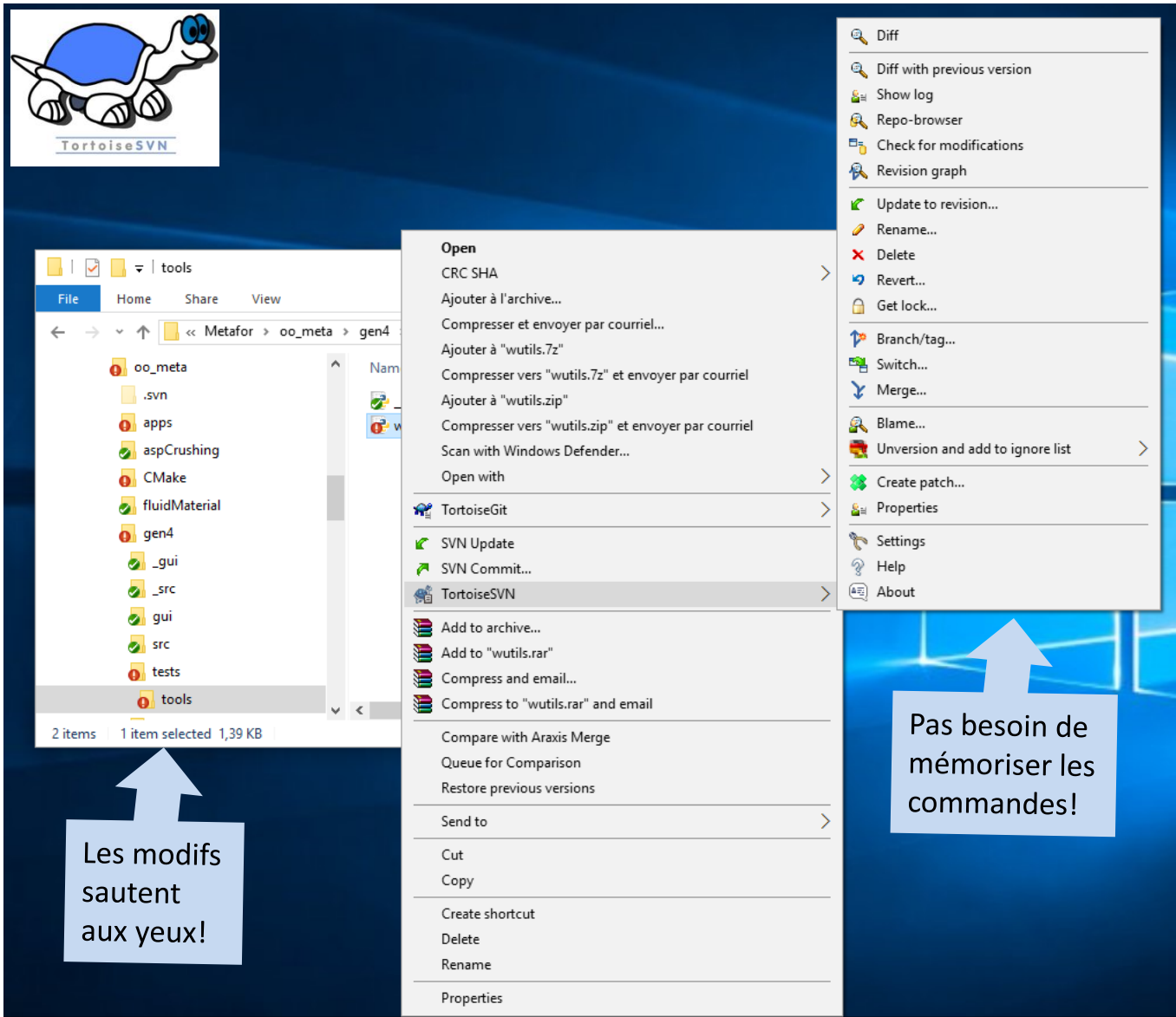
Tortoise SVN

- Utilisation instinctive.
- Bien utile pour résoudre d'un coup d'œil ceci:

mon code marche pas... J'ai pourtant rien modifié...

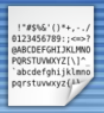
- Seul bémol:

j'ai fait un `svn blame`! ça plante dans ton code...



https://tortoisesvn.net/

Gestion du code source



Avantages SVN

- SVN peut être utilisé sans lire un manuel.
- Chaque commit possède un seul numéro de version qui s'incrémente automatiquement.

Inconvénients SVN

- Pas de « *pull request* » : on observe la qualité du code après commit.
- La méthode utilisée ne pousse pas le non-geek à commiter.

En étude actuellement:

- Utilisation de branches de développement git avec commits quotidiens.

Nécessité de définir une méthode d'utilisation qui se résume en **quelques commandes simples et figées.**

TO BE CONTINUED...

Documentation

Tu sais que la doc sur la classe `Machin` n'est pas écrite?
Pfff, quel guignol cet ancien thésard...

C'est bien possible...
Tiens, à propos, t'as écrit la doc de ton dernier commit?

Ah ben non, j'ai pas eu le temps, tu sais, je pars en conf la semaine prochaine...

images de http://geek-and-poke.com/

Documentation

documentation des commits

Qu'est ce qui a été modifié? Quand? Par qui? Pourquoi?

svn log!

documentation « utilisateur »

Utilisation de l'exécutable

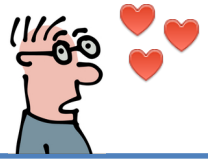
LaTeX! (+svn)

3 types de documentation

documentation du code source

Doxygen!

Documentation



Compromis trouvés

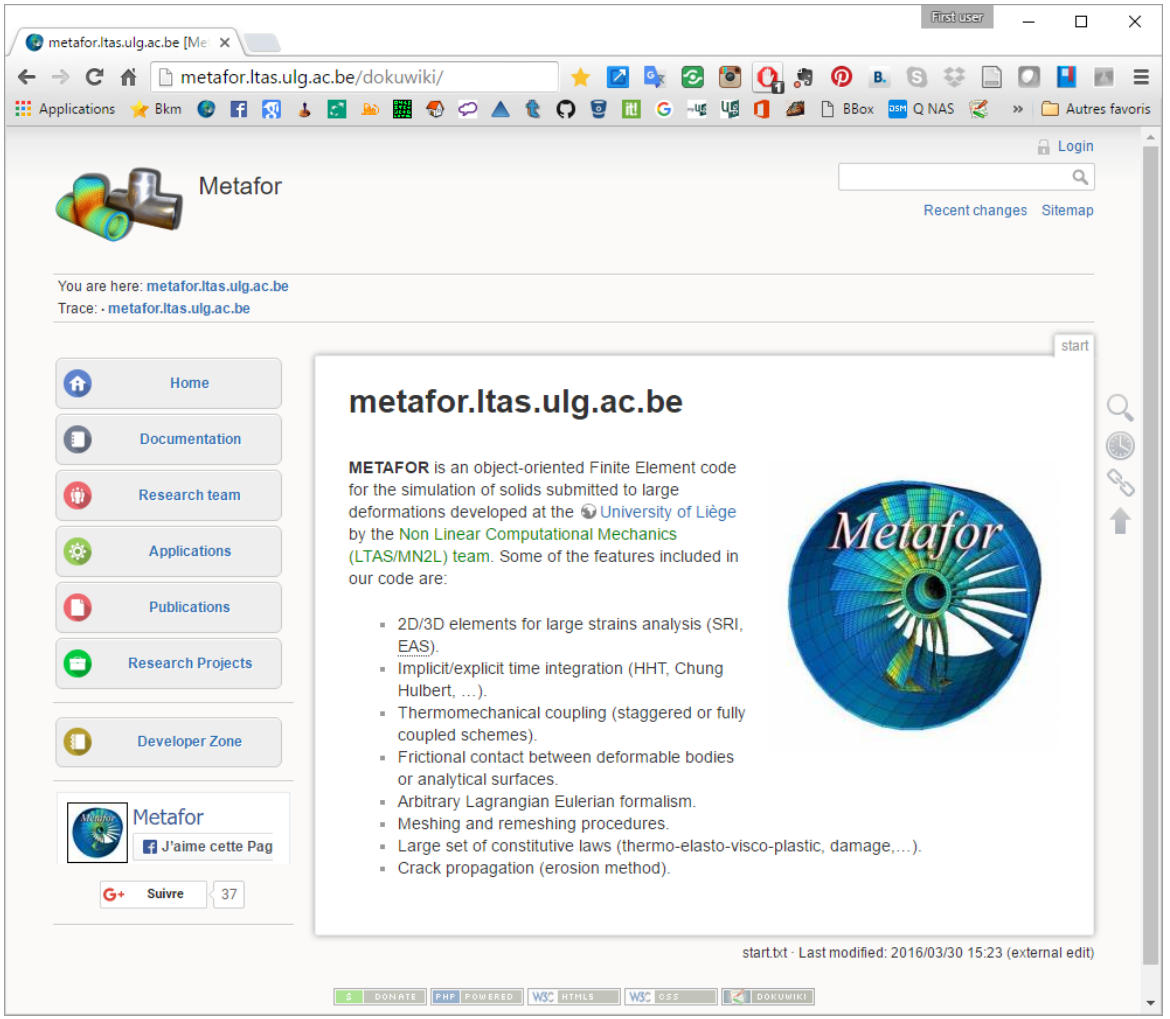
Code source complètement documenté (Doxygen).	Nom de classes, fonctions, variables d'autant plus explicites qu'elles sont visibles.
Doc « utilisateur » complète.	Fonctions de base documentées (pour le reste, utilise « `grep` » dans la batterie de tests ou « `svn blame` » sur le source).
Langue: en anglais et français.	Règle: Utilise une langue que tu maitrises (mieux vaut une bonne doc en français que de l'anglais incompréhensible).
Version papier et en ligne.	Version en ligne sur wiki.
Rappels théoriques avec équations LaTeX.	Mathjax sur le wiki.
Tutoriels, vidéos, etc.	1 powerpoint mis à jour chaque année.

Documentation

https://www.dokuwiki.org/

Système de wiki choisi depuis 10 ans: ***Dokuwiki***

http://metafor.ltas.ulg.ac.be/



Points forts:

- Installation très simple (DB format texte).
- Compatible LDAP.
- Nombreux plugins.
- Mises à jour régulières et automatiques.
- Dispo sur NAS Synology.

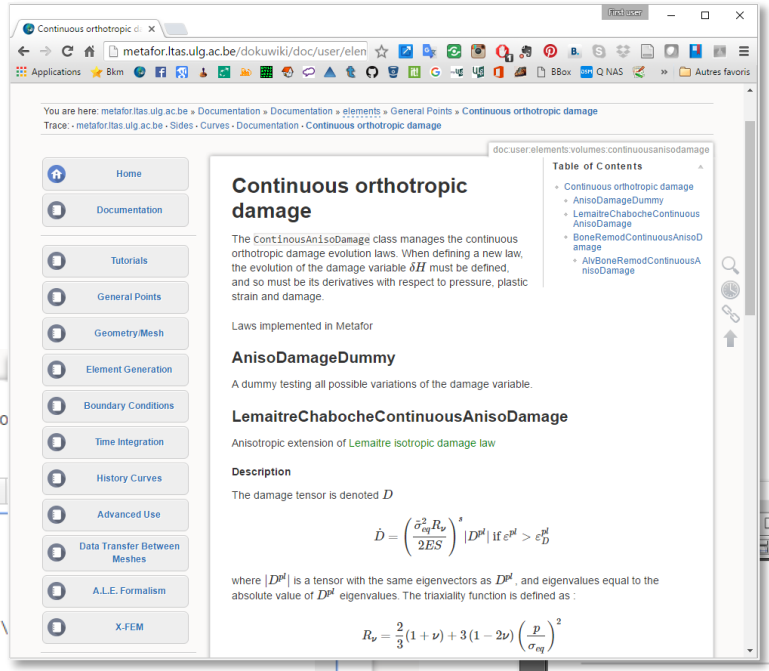
- ... et pas besoin de lire un long manuel pour éditer une page

Documentation

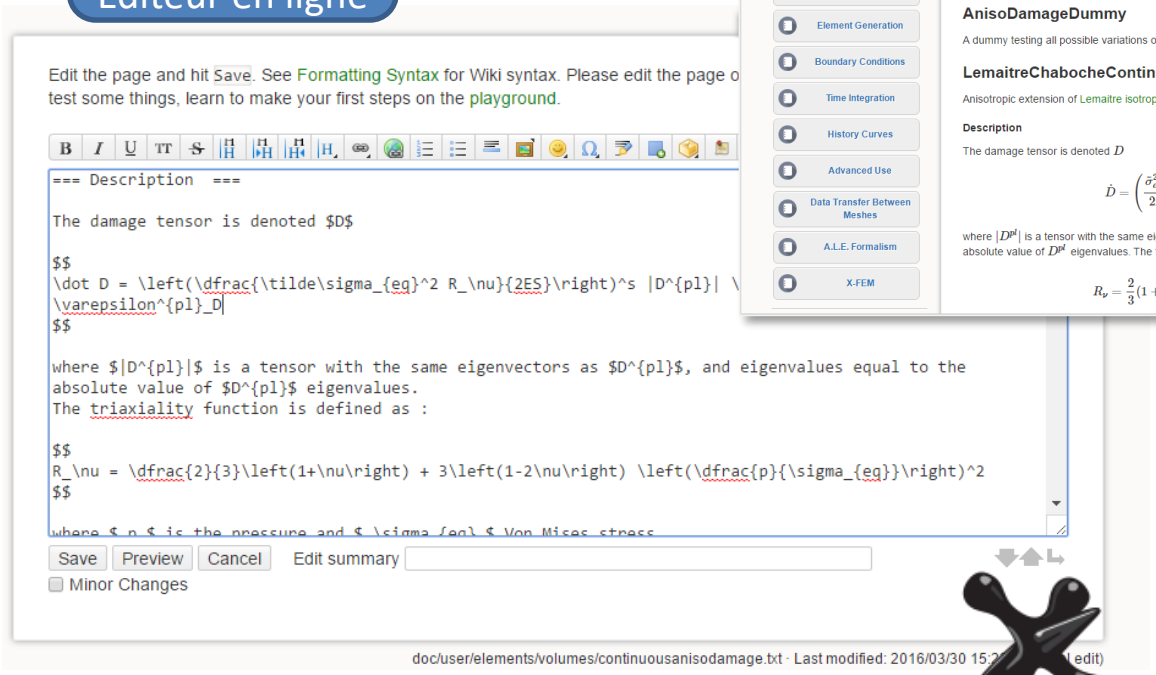
Comme tout wiki:

- Edition en ligne.
- Conservation de l'historique.
- Ajout d'images, vidéos, etc.

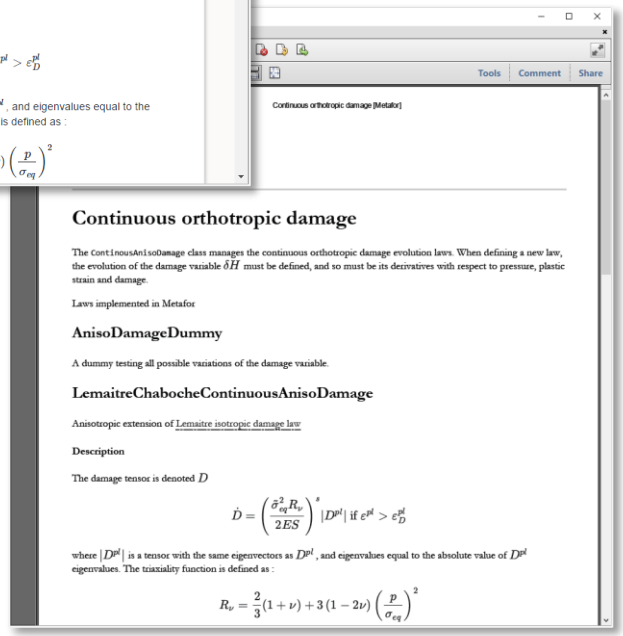
Vue en ligne



Editeur en ligne



Impression

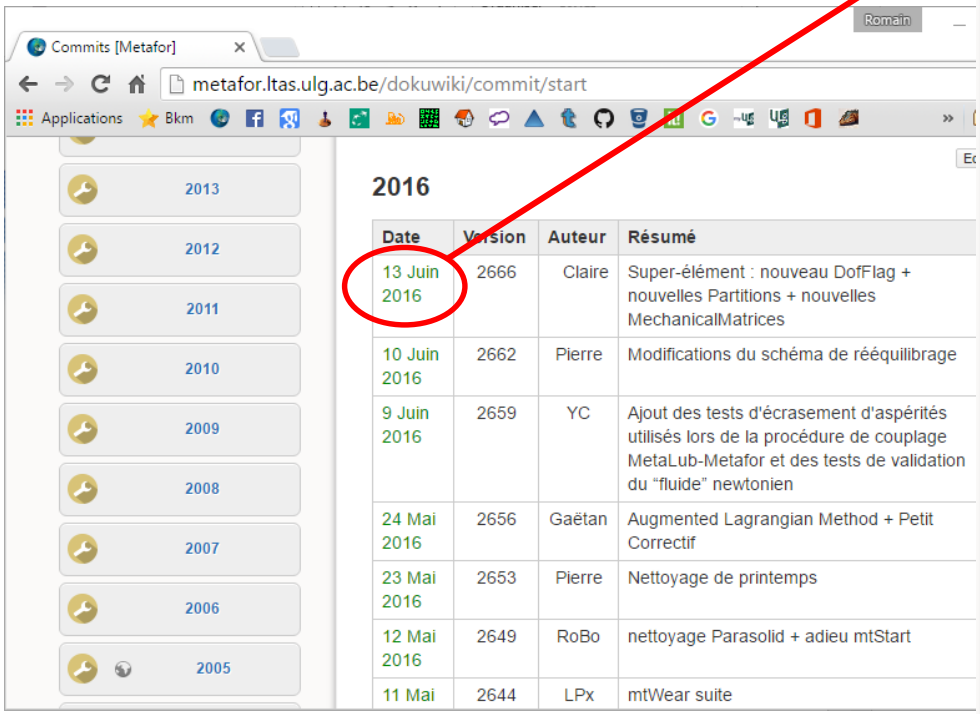


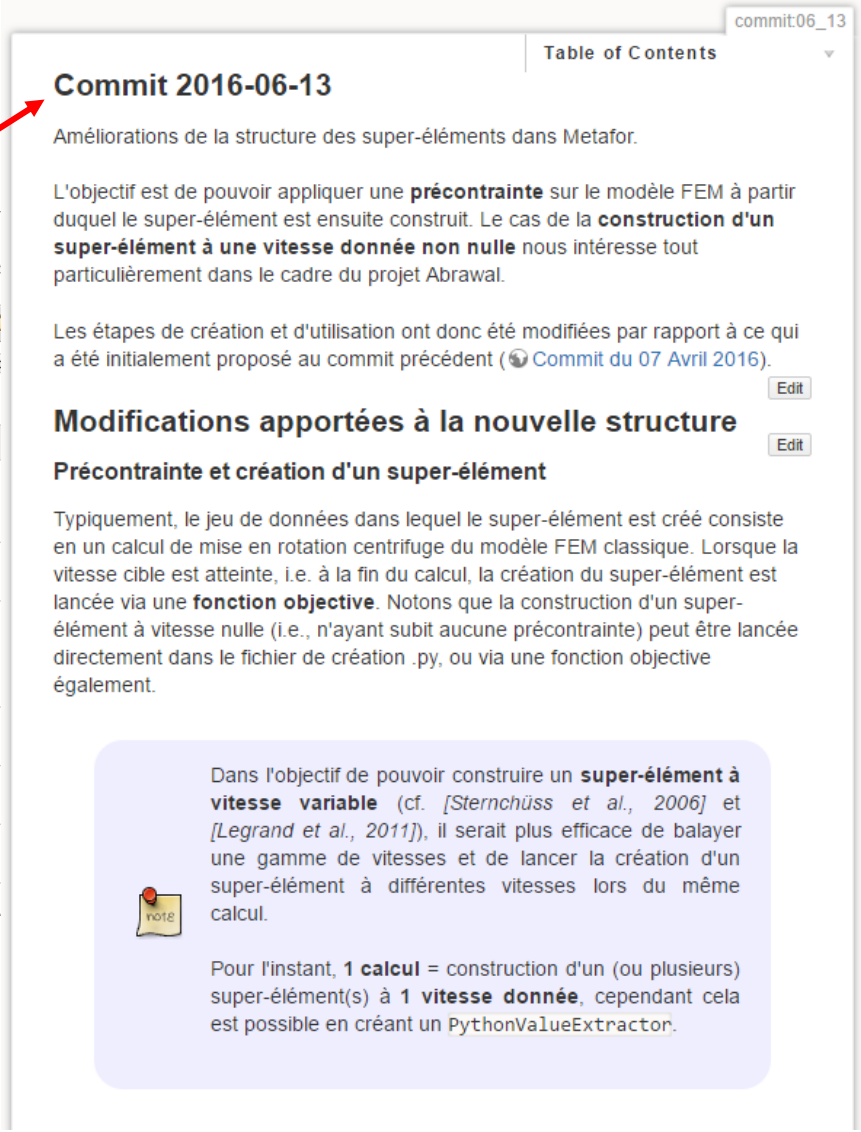
moteur javascript de rendu d'équations MathJax

Documentation

Documentation des commits

`svn log` est peu utile vu la procédure de commit utilisée.





Chaque commit est longuement documenté par son auteur sur le wiki.

Batterie de tests

J'ai mis à jour ma version après le commit de mon abruti de collègue et mes développements ne marchent plus du tout!

Mmm?.. Bizarre ça… T'es sûr que tes tests sont suffisants?

Ah oui, c'est vrai… faudra que je commite un jour des tests, j'ai pas encore eu le temps…
Tu sais... la conf, le papier, etc.

images de http://geek-and-poke.com/

Batterie de tests

Buts:

- Assurer la non-régression du code et des modèles numériques.
- Eviter les conflits entre personnes.

Principe:

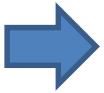
- Gérée par un script python maison.
- Série d'environ 3100 modèles EF, lancés en parallèle si la machine le permet.
- Tourne en moins d'une nuit sur un PC classique (~75000s CPU = ~5h30 sur un quad core).
- 3 configurations:
  - Windows x64, Visual Studio 2012 (généralement le PC de travail)
  - Linux Debian x64, gcc
  - Linux Debian x64, intel
- Chaque test extrait des valeurs de résultats « bien choisis » qui seront vérifiées avec les résultats de la version officielle du code (résultats commités dans le dépôt avec les sources).

Batterie de tests

2 règles simples

- Interdiction de commiter sans vérifier intégralement la batterie de tests.
- Tout développement non testé peut être détruit (inutile de se plaindre).

Risque:



Solution actuellement mise en place:

- La procédure documentée et scriptée.

Batterie de tests

Procédure scriptée

En 5min:

- Zipper ses sources modifiées (dans l'explorer Windows).
- Transfert sur les machines de test (Filezilla).
- ssh sur les machines (putty).
- Lancer « comp.py ».

Ensuite:

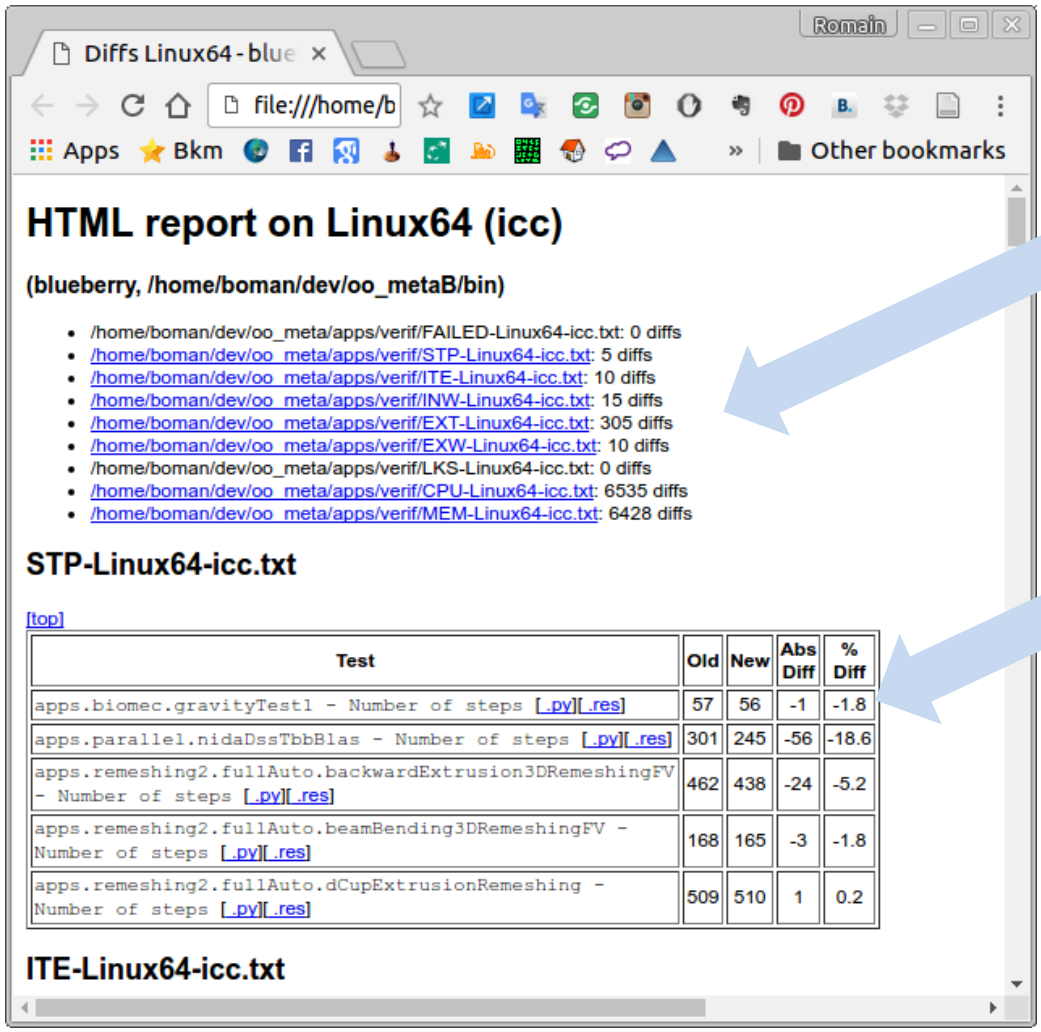
- rentrer chez soi…

durée = 1 nuit

```
boman@blueberry: ~/dev
Actions:
 a/ e-mail address (reports)            : 'boman'
 b/ archive name                        : '~/dev.zip'
 f/ build options                       : 'blueberry.cmake'
 g/ debug mode                          : False
 h/ nice value                          : '0'
 j/ nb of task launched in parallel     : '4'
 k/ nb of threads by task               : '1'
 m/ Run Method                          : 'batch'
 q/ is bacon present?                   : True

 1/ source                              : 'zip'
 2/ compile                             : True
 3/ battery                             : True
 4/ installer                           : False

 G/ GO
 S/ SAVE
 Q/ QUIT


Your choice?
```

Batterie de tests

- Le lendemain, les résultats sont reçus par e-mail.

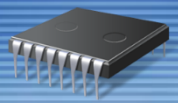


Les résultats sont triés en catégories
On voit le nombre de diffs avec la dernière version commitée

détail pour chaque diff

- Etape finale:
  vérifier que les diffs observées ne sont pas significatives avant commit.

Gestion mémoire



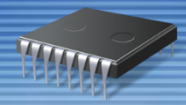
Faut que je me rachète un PC avec plus de RAM; mes calculs sont tellement gros que j'arrive plus à travailler

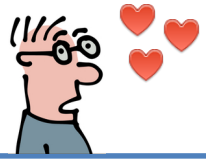
A mon avis, les destructeurs de tes classes ne sont pas correctement appelés...

Heuu, c'est quoi ça?

images de http://geek-and-poke.com/

Gestion mémoire



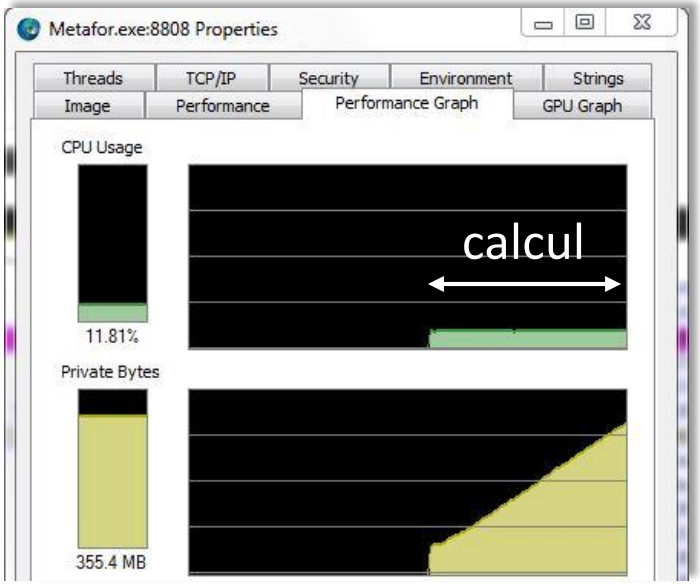


Compromis trouvé

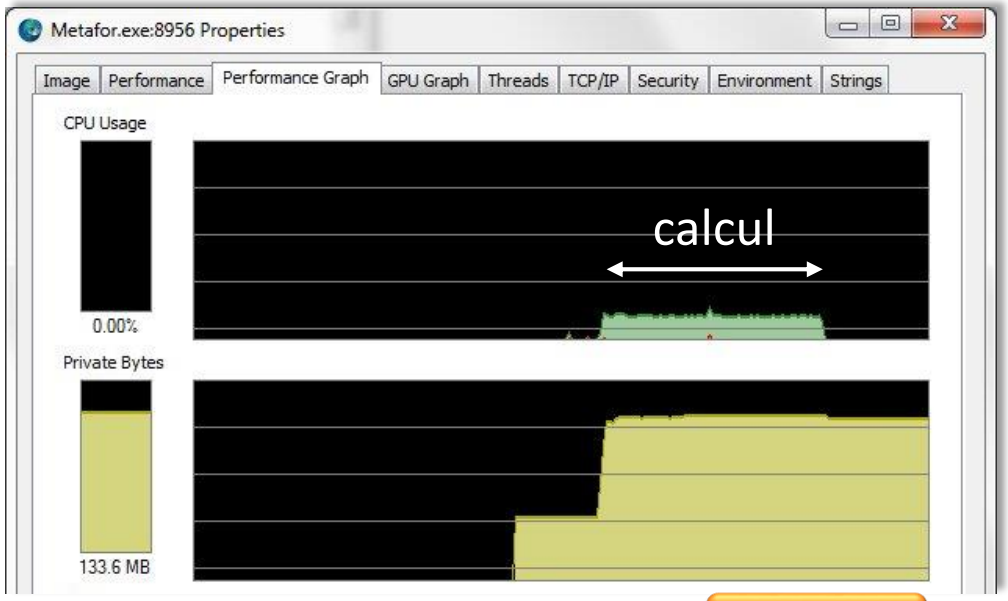
Toute mémoire allouée doit être libérée.

Pas de memory leak en cours de calcul.

Exemple:

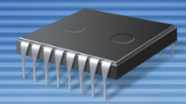


Inacceptable!



Toléré...

Gestion mémoire



RAM limitée sur machines de test.

Certains objets sont comptés
(et le compteur est un résultat de batterie).

Garde-fous

GUI:
Détection de fuites via `vtkDebugLeaks`.

De temps en temps:
Intel® Inspector.

Visualisation

Aha! J'ai trouvé un bug dans tes routines de visu!
Quand je visualise mon nouveau type de maillage, il y a tout qui plante!

Ca ne m'étonne pas, j'ai écrit ça bien avant tes développements.
Mais je peux t'expliquer comment ça marche...

Tu crois que j'ai le temps de m'amuser avec du graphisme?

images de http://geek-and-poke.com/

Visualisation

Visualisation continue des résultats en cours de calcul

Initialement 2 buts:

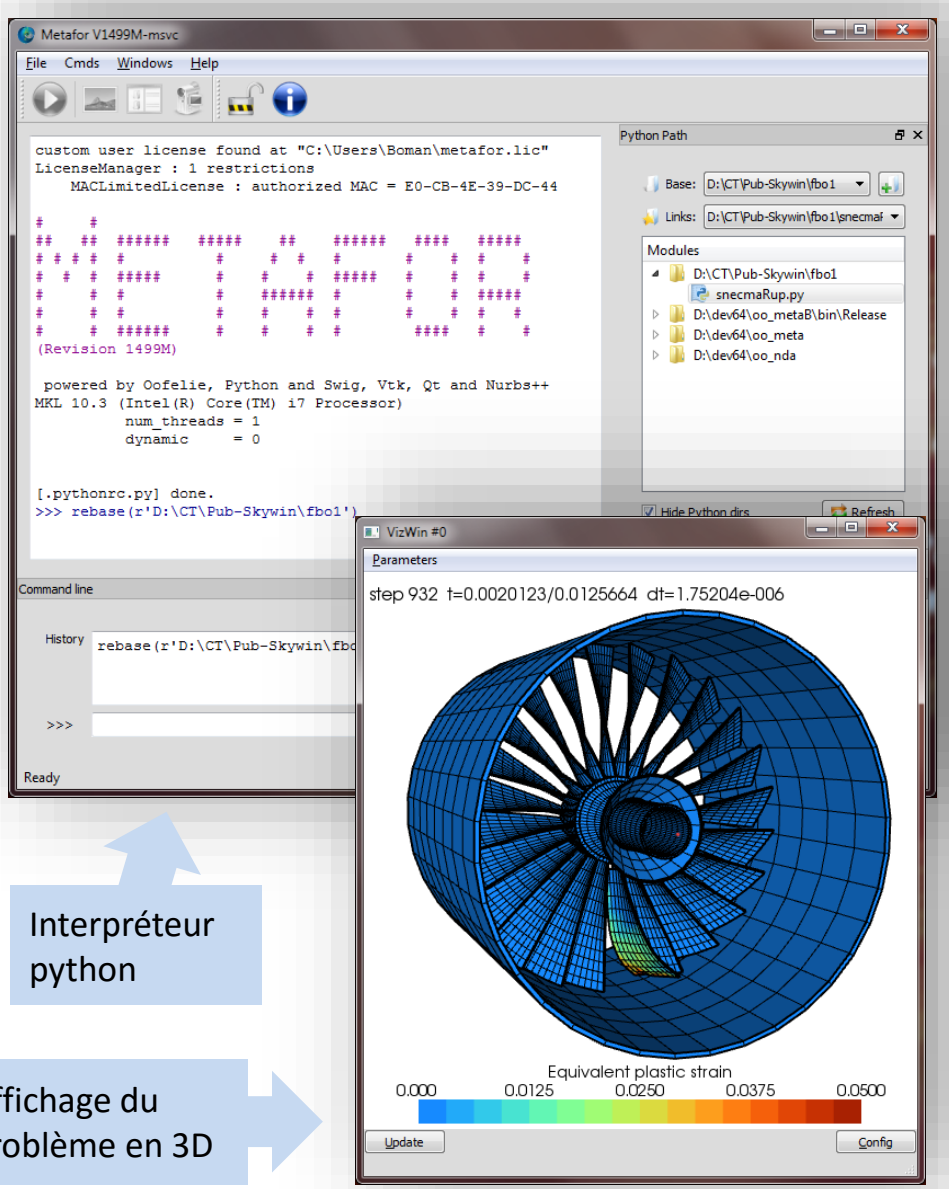
- Voir à tout moment ce qu'on fait.
- Faciliter l'utilisation pour les étudiants et les industriels.

Particularités:

- 1 thread graphique à côté du/des thread(s) de calcul.
- On affiche simplement en 3D l'état de la mémoire.
- Pas d'accès disque.

Conséquence:

- Plus besoin d'outil externe de post-processing.



Visualisation

Gestion des widgets avec Qt

Points forts de Qt:

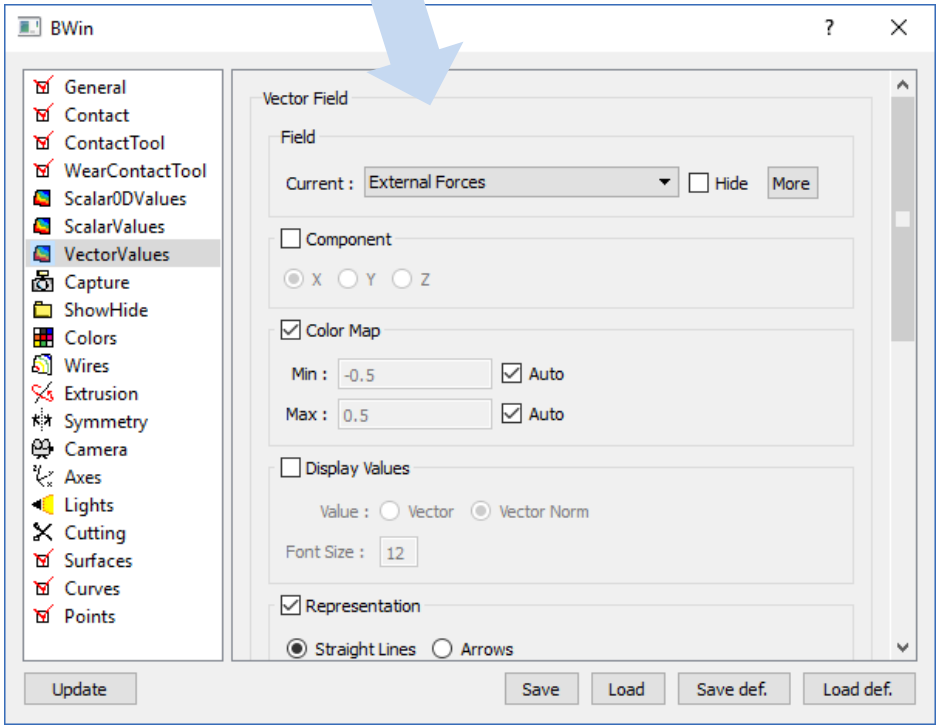
- Portable (Windows, Linux, etc.) sans être trop laid.
- Nombreux utilisateurs (base de KDE).
- Open source (LGPL).
- Interface python (PySide – PyQt).
- Moderne : Orienté objet (C++), mécanisme signal/slot.
- Bien documenté.
- Framework complet (p. expl. `QThread`, `QDataStream`, etc.).

Essayés et abandonnés:

- wxWidgets, GTK+, Tcl/Tk

The Qt company
http://www.qt.io/

Boutons, listes déroulantes, cases à cocher, etc.



Visualisation

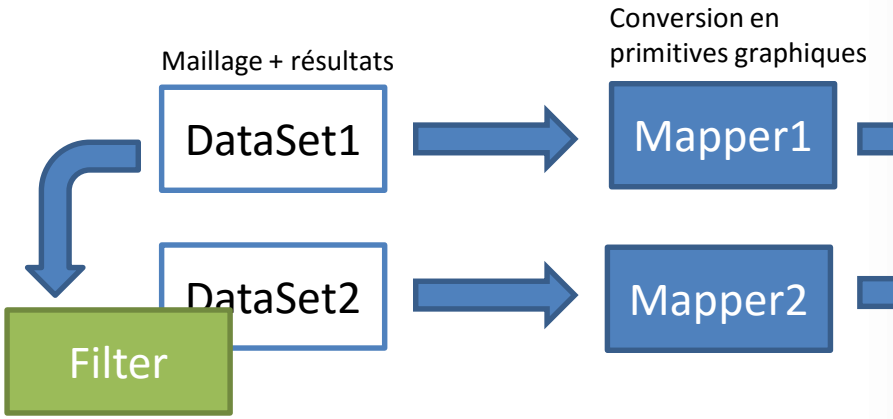
Affichage 3D des résultats sur maillage avec VTK (Kitware)

Points forts de VTK:

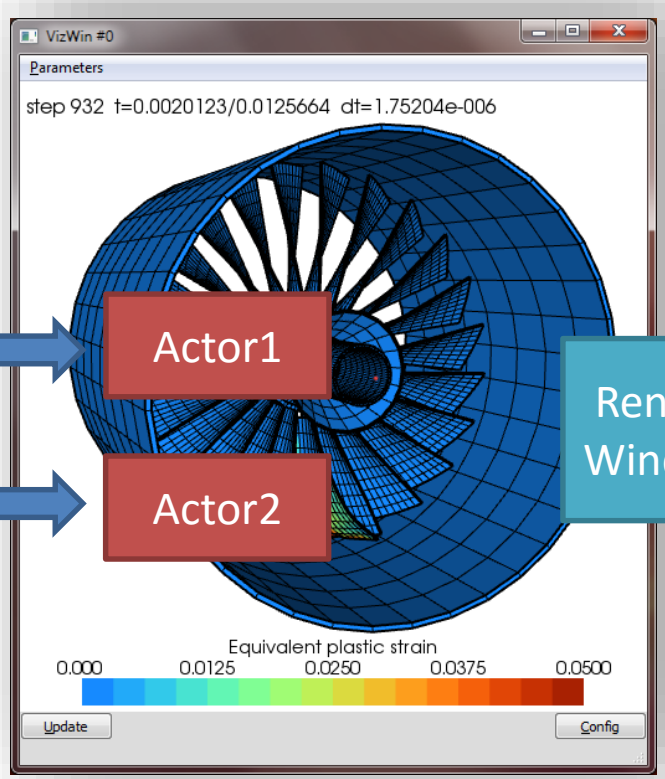
- voir Qt
- + interface Qt/python

http://www.vtk.org/

« Pipeline » de visualisation (principe):



- Extrusion,
- Extraction d'iso-valeurs,
- Numérotation, …



Conclusion et perspectives

Ca y est: je suis docteur!
J'ai pu réutiliser les acquis du passé.
La collaboration a porté ses fruits.

Merci pour ton travail. Les projets suivants sont déjà en route!

Espèce de geek!

images de http://geek-and-poke.com/

Conclusion et perspectives

Exemple de collaboration plus large (département)

Labo Prof. J.-P. Ponthot

metafor.py

Metafor
Solveur solide

python

Solveur fluide – structure

Labo Prof. V. Terrapon

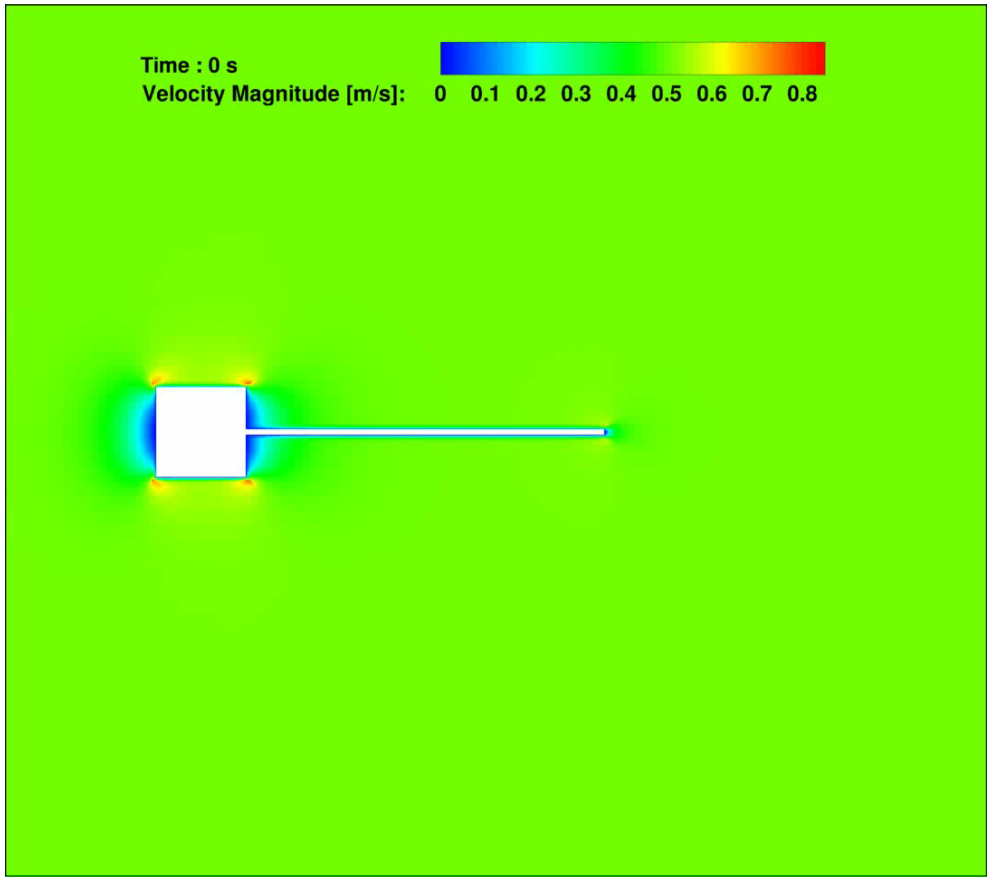
SU2.py

SU2
Solveur fluide

python

David Thomas

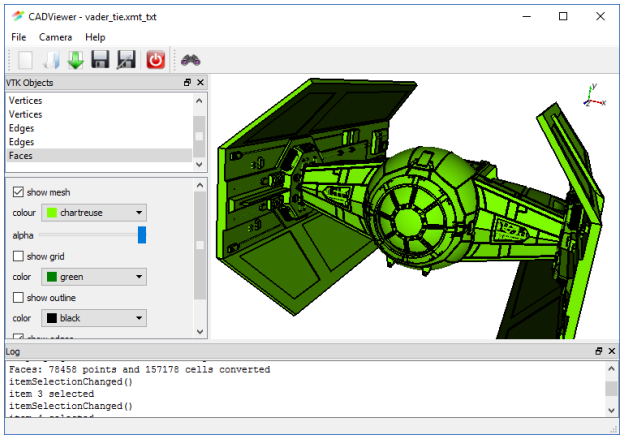
Couplage fluide-structure



Conclusion et perspectives

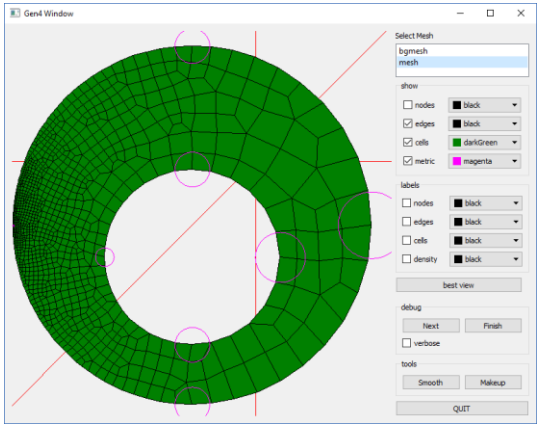
Metafor en Open Source?

Extraction de certains composants génériques de Metafor
en vue d'un dépôt sur github (en cours)



Interface CAD

Mailleur quad

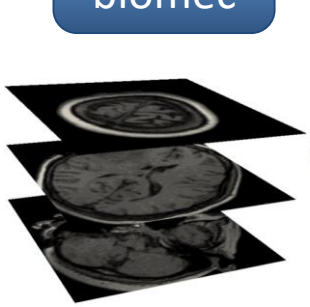


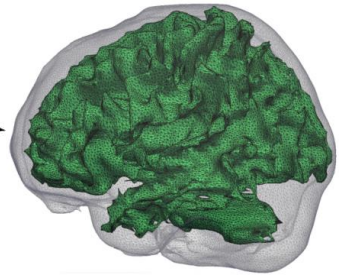
Metafor

Batterie de test

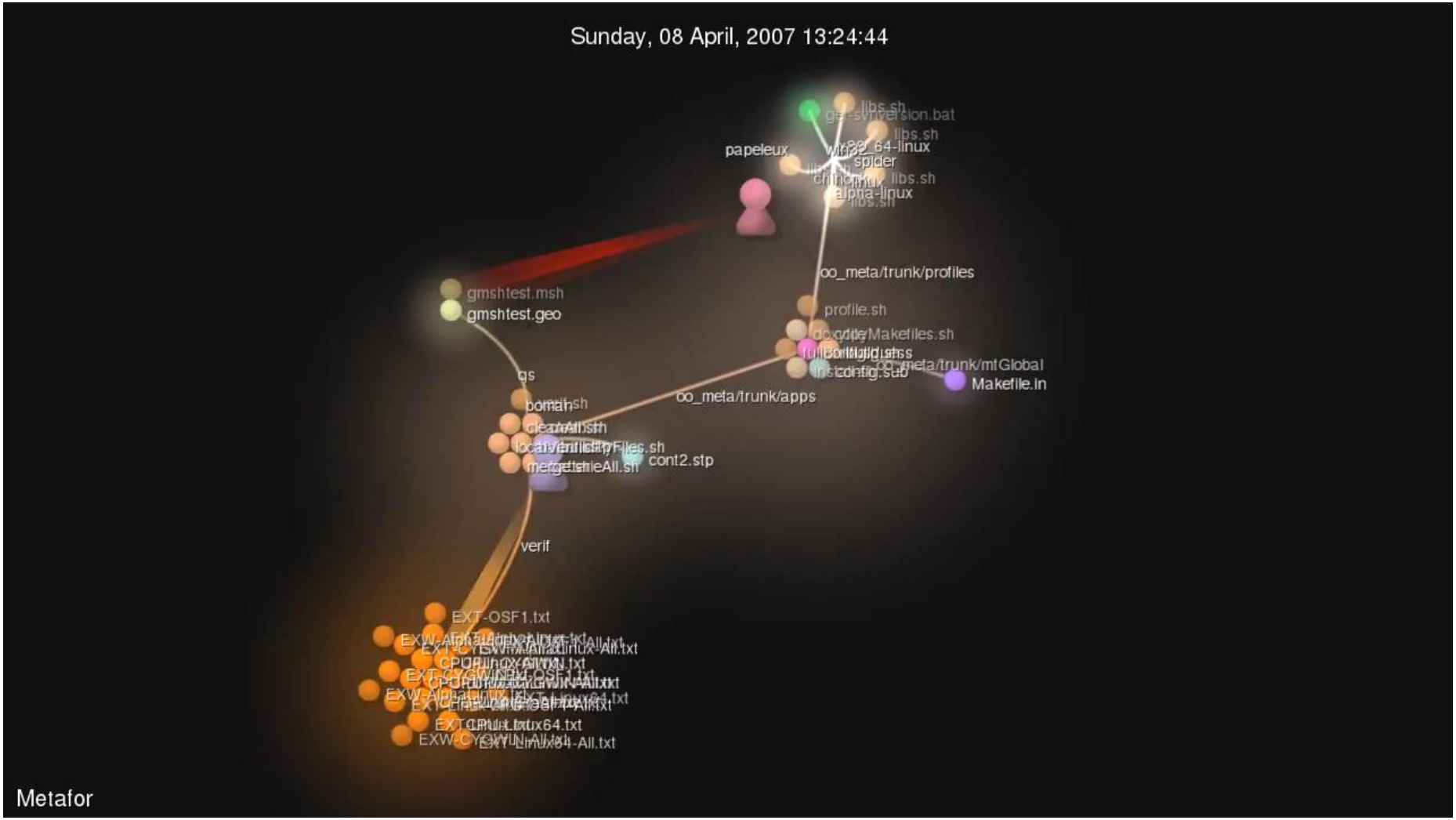
Solveur EF basique

Mailleur biomec





Merci!



gource -s .4 -p 0.001 -1280x720 --auto-skip-seconds .4 --multi-sampling --stop-at-end --highlight-users --hide mouse,progress --file-idle-time 0 --max-files 1000 --background-colour 111111 --font-size 20 --title "Metafor" --output-ppm-stream - --output-framerate 60 | avconv -y -r 60 -f image2pipe -vcodec ppm -i - -b 8192K movie.mp4