MFront for the Code-Aster’users and beyond

Code-Aster users meeting — Thomas Helfer\textsuperscript{1}, Jean-Michel Proix \textsuperscript{2}, Olivier Fandeur\textsuperscript{3,4}, François Curtit\textsuperscript{5}, Charles Toulemonde\textsuperscript{5}, François Hamon\textsuperscript{2}, Vincent Faucher \textsuperscript{3,4}, Michel Casella\textsuperscript{1}

26 March 2015

\textsuperscript{1} CEA, Département d’Études des Combustibles
\textsuperscript{2} EDF R&D, Département Analyses Mécaniques et Acoustique
\textsuperscript{3} CEA, Département de Modélisation des Systèmes et des Structures
\textsuperscript{4} IMSIA, UMR 8193, CNRS-EDF-CEA-ENSTA
\textsuperscript{5} EDF R&D, Département des Matériaux et Mécanique des Composants
Forewords

A brief tour of MFront

Material knowledge management

Conclusions

Appendix
## APPLICATIONS FILIÈRES

<table>
<thead>
<tr>
<th>REP</th>
<th>REP</th>
<th>RNR-Na</th>
<th>MTR</th>
<th>RNR-G</th>
<th>HTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALCYONE</td>
<td>CYRANO3</td>
<td>GERMINALV2</td>
<td>MAIA</td>
<td>CELAENO</td>
<td>ATLAS</td>
</tr>
</tbody>
</table>

### ÉTUDES NON STANDARDS

- **LICOS**

### Application µe

- **V.E.R.**
A brief tour of MFront
MFront is an open-source code generator based on C++ developed within the PLEIADES platform for:
- material properties
- mechanical behaviours
- models
MFront is an **open-source** code generator based on C++ developed within the PLEIADES platform for:

- material properties
- **mechanical behaviours** of particular interest for Code-Aster
- models
MFront

MFront is an **open-source** code generator based on C++ developed within the PLEIADES platform for:
- material properties
- **mechanical behaviours** of particular interest for Code-Aster models

MFront provides several *domain specific languages*:
- ease of use, expressiveness, etc..
  - focus on physical content
  - low programming skills requirements
MFront is an open-source code generator based on C++ developed within the PLEIADES platform for:

- material properties
- mechanical behaviours of particular interest for Code-Aster models

MFront provides several domain specific languages:

- ease of use, expressiveness, etc..
  - focus on physical content
  - low programming skills requirements

MFront tries to comply with Éric Lorentz’s rule of three:

- reliability (gives the correct result)
- robustness (gives a result)
- numerical efficiency (is as fast as possible).
MFront

- **MFront** is an *open-source* code generator based on C++ developed within the PLEIADES platform for:
  - material properties
  - mechanical behaviours of particular interest for Code-Aster
  - models

- **MFront** provides several *domain specific languages*:
  - ease of use, expressiveness, etc..
    - focus on physical content
    - low programming skills requirements

- **MFront** tries to comply with Éric Lorentz’s rule of three:
  - reliability (gives the correct result)
  - robustness (gives a result)
  - numerical efficiency (is as fast as possible).

- Interoperability (rule of four? See ”Interfaces” below).
MFront

- MFront is an **open-source** code generator based on C++ developed within the PLEIADES platform for:
  - material properties
  - **mechanical behaviours** of particular interest for Code-Aster models
- MFront provides several *domain specific languages*:
  - ease of use, expressiveness, etc..
    - focus on physical content
    - low programming skills requirements
- MFront tries to comply with Éric Lorentz’s rule of three:
  - reliability (gives the correct result)
  - robustness (gives a result)
  - numerical efficiency (is as fast as possible).
- Interoperability (rule of four? See ”Interfaces” below).
- **Strong emphasis on Quality Assurance** (rule of five? See ”Material knowledge management” below).
MFront

- MFront is an **open-source** code generator based on C++ developed within the PLEIADES platform for:
  - material properties
  - mechanical behaviours of particular interest for Code-Aster models
- MFront provides several *domain specific languages*:
  - ease of use, expressiveness, etc..
    - focus on physical content
    - low programming skills requirements
- MFront tries to comply with Éric Lorentz’s rule of three:
  - reliability (gives the correct result)
  - robustness (gives a result)
  - numerical efficiency (is as fast as possible).
- Interoperability (rule of four? See ”Interfaces” below).
- **Strong emphasis on Quality Assurance** (rule of five? See ”Material knowledge management” below).
- PLEIADES goals: building high quality fuels performance codes
A first example: material property

@DSL MaterialLaw; // treating a material property
@Material UO2; // material name
@Law YoungModulus_Martin1989; // name of the material property
@Output E; // output of the material property
@Input T,f; // inputs of the material property
@Function // implementation body
{
  E = 2.2693e11*(1-2.5*f)*(1-6.786e-05*T-4.23e-08*T*T);
}

\[ E(T,f) = 2.2693 \times 10^{11} (1 - 2.5 f) (1 - 6.786 \times 10^{-5} T - 4.23 \times 10^{-8} T^2) \]

- usable in Code-Aster through the python interface.
- also in C, C++, fortran, excel, Cast3M, etc..
- usable in MFront’s mechanical behaviours!
A first example: material property

```plaintext
@DSL MaterialLaw; // treating a material property
@Material UO2; // material name
@Law YoungModulus_Martin1989; // name of the material property
@Author T. Helfer; // author name
@Date 04/04/2014; // implementation date
@Description // detailed description
{
  The elastic constants of polycrystalline UO2 and (U, Pu) mixed oxides: a review and recommendations
  Martin, DG
  High Temperatures. High Pressures, 1989
}

@Output E; // output of the material property
E.setGlossaryName("YoungModulus");

@Input T, f; // inputs of the material property
T.setGlossaryName("Temperature");
f.setGlossaryName("Porosity");

@PhysicalBounds T in [0:*]; // Temperature is positive
@PhysicalBounds f in [0:1.]; // Porosity is positive and lower than one
@Bounds T in [273.15:2610.15]; // Validity range

@Function // implementation body
{
  E = 2.2693e11*(1.−2.5*f)*(1−6.786e−05*T−4.23e−08*T*T);
}

\[ E(T, f) = 2.2693 \times 10^{11} \ (1 - 2.5 \ f) \ (1 - 6.786 \times 10^{-5} \ T - 4.23 \times 10^{-8} \ T^2) \]
```
A simple $J_2$ (isotropic) plastic behaviour:

\[ f(\sigma_{eq}, p) = \sigma_{eq} - Hp \leq 0 \]

example of specific behaviour implementation

automatic computation of the consistent tangent operator
A simple $J_2$ (isotropic) plastic behaviour:

$$f(\sigma_{eq}, p) = \sigma_{eq} - H p \leq 0$$

example of specific behaviour implementation

automatic computation of the consistent tangent operator

various domain specific languages are available to cope with:

- general small strain behaviours, general finite strain behaviours, cohesive zone models
- explicit or implicit integration schemes
  - various algorithms available
Code generation and interfaces

- Finite elements solvers: Code-Aster, Cast3M, ZeBuLoN.
- Fast Fourier transform solvers: TMFFT, AMITEX_FFT
- Fuel performance codes: Cyrano3
**Code generation and interfaces**

- **Finite elements solvers:**
  - **Code-Aster**, **Cast3M**, **ZeBuLoN**, **EuroPlexus**, **Abaqus**, **Ansys**.

- **Fast Fourier transform solvers:**
  - **TMFFT**, **AMITEX_FFT**, **CraFT**

- **Fuel performance codes:**
  - **Cyrano3**, **Galileo**
Material knowledge management
One of the main benefits of MFront is to make the link between:

- Solvers:
  - Finite elements solvers (Code-Aster, Cast3M, etc..)
  - Fuel performances codes (PLEIADES), etc...
- Material knowledge management projects
One of the main benefits of MFront is to make the link between:

- Solvers:
  - Finite elements solvers (Code-Aster, Cast3M, etc..)
  - Fuel performances codes (PLEIADES), etc...

- Material knowledge management projects

With those projects, users are/will be granted access to:

- Checked implementations (expert judgement, unit tests)
- Technical notes, experimental data, etc...
- Material knowledge versionning
Studies quality assurance

One of the main benefits of MFront is to make the link between:

- Solvers:
  - Finite elements solvers (Code-Aster, Cast3M, etc..)
  - Fuel performances codes (PLEIADES), etc...

- Material knowledge management projects

With those projects, users are/will be granted access to:

- Checked implementations (expert judgement, unit tests)
- Technical notes, experimental data, etc...
- Material knowledge versionning

Those projects allow much deeper interactions/discussion/feed-backs between simulation and experimental people.
Studies quality assurance

One of the main benefits of MFront is to make the link between:

- **Solvers**:
  - Finite elements solvers (Code-Aster, Cast3M, etc..)
  - Fuel performances codes (PLEIADES), etc...

- **Material knowledge management projects**

With those projects, users are/will be granted access to:

- **Checked implementations** (expert judgement, unit tests)
- **Technical notes, experimental data, etc...**
- **Material knowledge versionning**

Those projects allow **much deeper interactions/discussion/feed-backs between simulation and experimental people.**

**Stronger quality assurance for the end user studies**
 Sirius and Cadeex: some unformalised perspectives

Two material knowledge management tools: Sirius database in PLEIADES, Cadeex at EDF.
- Those projects have followed complementary paths.
  - many experience to share
  - would benefit from a standard file exchange: MADNEX?

End-user would greatly benefit from a Salome interface
Solver inputs files would greatly benefit from allowing access to the underlying material data (Salome again)
Two material knowledge management tools: Sirius database in PLEIADES, Cadeex at EDF.

- Those projects have followed complementary paths.
  - many experience to share
  - would benefit from a standard file exchange: MADNEX?

End-user would greatly benefit from a Salome interface

Solver inputs files would greatly benefit from allowing access to the underlying material data (Salome again)
Conclusions
Acknowledgements

- J. M. Proix, J. Hure, F. Hamon, I. Ramière, É. Castelier, O. Fandeur, V. Blanc, J. Julien, B. Michel, B. Bary, F. Milliard, A. Courcelle, and all the others for various contributions.

- all the persons who contributed to the open-source release of TFEL: J. P. Defain, D. Banner, T. De Soza, V. Marelle, É. Lorentz, C. Toulemonde, F. Curtit, R. Masson, and all the others.

- the authors are grateful to J. Besson and S. Quilici for their valuable help in building the ZeBuLoN interface.

This research was conducted in the framework of the PLEIADES project, which was supported financially by the CEA (Commissariat à l’Énergie Atomique et aux Énergies Alternatives), EDF (Électricité de France) and AREVA and in the framework of the Simu-Meca2015 project held within EDF R&D.
Thanks your for your attention. Questions?

MFront: a code generation tool dedicated to material knowledge

New users/contributors are welcomed!
http://tfel.sourceforge.net
To meet CEA and EDF needs, TFEL 2.0 is released under a multi-licensing scheme:

- **open-source licences**:
  - **GNU Public License**: This licence is used by the Code-Aster finite element solver.
  - **CECILL-A**: License developed by CEA, EDF and INRIA, compatible with the GNU Public License and designed for conformity with the French law.

- CEA and EDF are free to distribute TFEL under custom licences: Mandatory for the PLEIADES platform.