PLANIFICATION ET ANALYSE D’EXPÉRIENCES NUMÉRIQUES
AVEC SALOME ET PRÉREQUIS URANIE

DESIGN AND ANALYSIS OF NUMERICAL EXPERIMENTS
WITH SALOME AND URANIE AS PREREQUISITE

Francis Kloss
CEA Saclay, DEN/DM2S/STMF/LGLS

SALOME USER DAY | 26 NOV 2015
Planification et analyse d’expériences numériques avec SALOME et prérequis URANIE -
Design and analysis of numerical experiments with SALOME and URANIE as prerequisite

Francis Kloss
CEA Saclay, DEN, DM2S, STMF, LGLS
Journée des utilisateurs SALOME
26 novembre 2015
1. Goal
2. Data model for application based on SALOME
3. Define an application for convection studies
4. Build a geometry for an office
5. Generate a 3D mesh with hexahedra
6. Launch a calculus using TrioCFD
7. Visualize temperature, pressure and velocity with PARAVIS
8. Parametric data model
9. Design of experiments with URANIE inside SALOME
10. Analysis of results with URANIE inside SALOME
11. Visualization of statistic results with Matplotlib
12. Roadmap for 2016
1. GOAL

- Need to build business-specific applications:
  - With pre-processing
    - CAD (GEOM, SHAPER)
    - Meshing (SMESH, HOMARD)
  - Data setting
  - With calculus (JOBMANAGER, YACS)
  - With visualization (PARAVIS, Plot2D)

- And with such application, make numerical experiments
1. GOAL

- CEA feedback about a lot of application:
  - With python: ALAMOS, MATIX, PPGP, ILMAB, ALLIANCES, etc.
  - With C++: PAREX, OSCAR, WPROCESS, SALOME MONTE-CARLO, etc.
    - An approach for small or medium size application

- Make a choice: object programming approach
  - Development in C++ or Python
  - Generic data model by programming classes
  - Model-view-controller for programming application
2. DATA MODEL

- **Class Core::Data**
  - ✓ Root class to define data by derivation
  - ✓ Name attribute
  - ✓ Send messages for listeners like interactive program
  - ✓ Management of deletions

- **Class Core::Model**
  - ✓ Main class to define a data model by derivation
  - ✓ Automatic serialization in standard format: XML and JSON
  - ✓ Also capability to serialize in specific format file
2. DATA MODEL IS A PART OF MVC PATTERN

- **Class Core::Engine**
  - Define a C++ API or a Python API for algorithms or treatments
  - Need a data model

- **Class Core::Command**
  - Like YACS commands (or QAction class of Qt technology)
  - Automatic dump python
  - Undo-Redo functionality
  - Capability to compose an application by some elements from other applications

- **Class Core::Client**
  - For the graphic user interface (and for also the desktop)

- **Class Core::Controller**
  - Based on model-view-controller software architectural pattern
2. DATA MODEL

- **Elementary data class of the model:**
  - Class Core::Bool
  - Class Core::Int
  - Class Core::Double
  - Class Core::String

- **Containers of the model:**
  - Class Core::List
  - Class Core::Array
  - Class Core::Map
  - Class Core::Vector
  - Class Core::Matrix (based on boost)
2. DATA MODEL

- **Others data class:**
  - Class Core::Choose
  - Class Core::Reference (to another model)

- **Specific data class for simulation:**
  - Class Core::Geometry (issue from SHAPER or GEOM)
  - Class Core::Mesh (based on MEDCoupling)
  - Class Core::Field (based on MEDCoupling)
3. DEFINE AN APPLICATION FOR CONVECTION STUDIES

- Explain this talk by a realistic enough application
  - Model an office with SALOME
    - With a window
    - With a convector
  - Calculus of the temperature by convection with TrioCFD
    - In particular, find the temperature at the desktop position
  - Visualize the result with SALOME

- Explain how to pass to one calculus to an experimental plan
  - New generic data model for SALOME
  - In particular, SALOME use URANIE
3. DATA MODEL

- **Geometric elements**
  - Office
  - Window
  - Convector

- **Meshing elements**

- **Physical data**
  - Exchange coefficient
  - Outdoor temperature
  - Power of the convector

- **Sensor**
  - To analyze the temperature at a place
namespace Convection {

    class Office: public Core::Model {

        Constructor(Office, Core::Model);
        ~Office();

        private:
            Room room;
            Convector convector;
            Window window;
            Door door;
            Sensor sensor;
    }

}

namespace Convection {

    class Room: public Core::Data {

        Constructor(Room, Core::Data);
        ~Room();

        private:
            Core::Double x;
            Core::Double y;
            Core::Double z;
            Core::Double dx;
            Core::Double dy;
            Core::Double dz;
    }

}

3. DATA MODEL

namespace Convection {

class Convector: public Core::Data {

    Constructor(Convector, Core::Data);

    ~Convector();

    private:
        Core::Double x;
        Core::Double y;
        Core::Double z;
        Core::Double width;
        Core::Double height;
        Core::Double power;
    }
}

namespace Convection {

    class Window: public Core::Data {

        Constructor(Window, Core::Data);

        ~Window();

        private:
            Core::Double x;
            Core::Double y;
            Core::Double z;
            Core::Double width;
            Core::Double height;
            Core::Double exchange_coef;
            Core::Double outdoor_temp;
        }
    }
}
4. BUILD A GEOMETRY FOR AN OFFICE
5. GENERATE A 3D MESH WITH HEXAHEDRA
5. GENERATE A 3D MESH WITH HEXAHEDRA
More about the CEA open-source code TrioCFD

- Sourceforge.net/projects/triocfd

Data set for a Navier-Stokes solver (extract)

```plaintext
Convection_Diffusion_Temperature {
  convection { amont } diffusion { }
  conditions_initiales { temperature Champ_Uniforme 1 20.0 }
  conditions_initiales { temperature Champ_fonc_reprise xyz cas_ferme_init.xyz pb temperature last_time }
  boundary_conditions {
    door paroi_adiabatique
    window paroi_echange_global_impose h_imp
  }
Champ_Front_Uniforme 1 30. T_ext Champ_Front_Uniforme 1 10.
  convector paroi_flux_impose Champ_Front_Uniforme 1 200.
  walls paroi_adiabatique
  ceiling paroi_adiabatique
  floor paroi_adiabatique } }
```
7. VISUALIZE TEMPERATURE WITH PARAVIS
8. PARAMETRIC DATA MODEL

- **Generic parametric data model**
  - All classes that derivate from Core::Data have a general parametric attribute
    - All data could be parametrized
  - All classes that derivate from Core::Model have a general parametric serialization needed to build the data for a plan of experiments

- **Parametric data model for URANIE**
  - The probability distributions defined by URANIE are used for this parametric attribute
  - Specific serialization by a “XML” file for URANIE
  - Specific dialog box to enter these distributions
8. PARAMETRIC DATA MODEL

- **Object Browser**: Convection

<table>
<thead>
<tr>
<th>Data Model</th>
<th>Value</th>
<th>Parametric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Office</td>
<td>myOffice</td>
<td></td>
</tr>
<tr>
<td>room</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Where</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>door</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Where</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Width</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Height</td>
<td></td>
<td></td>
</tr>
<tr>
<td>window</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Where</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Width</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Height</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Physic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Exchange coef</td>
<td>30.0</td>
<td>law=uniform min=10.0 max=70.0</td>
</tr>
<tr>
<td>- External temperat</td>
<td>10.0</td>
<td>law=uniform min=-10.0 max=+10.0</td>
</tr>
<tr>
<td>convector</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Where</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Physic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Power</td>
<td>500.0</td>
<td>law=uniform min=200.0 max=400.0</td>
</tr>
<tr>
<td>Meshing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Mesh size</td>
<td>0.05</td>
<td></td>
</tr>
<tr>
<td>Sensor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- SONDE_X</td>
<td>0.730268</td>
<td></td>
</tr>
<tr>
<td>- SONDE_Y</td>
<td>1.31915</td>
<td></td>
</tr>
<tr>
<td>- SONDE_Z</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>General</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
8. PARAMETRIC DATA MODEL

- More about URANIE
  - See: sourceforge.net/projects/uranie

- The “xml” file generated for URANIE

```xml
<!DOCTYPE Problem SYSTEM "uranie.dtd">
<Problem>
  <Header name="Office" title="Convection with TrioCFD" debug="5">
    <Application name="uranie" version="1.0"/>
  </Header>
  <DataDictionary>
    <DataField name="temp" law="uniform" min="-10.0" max="10.0"/>
    <DataField name="power" law="uniform" min="200.0" max="400.0"/>
    <DataField name="coef" law="uniform" min="10.0" max="70.0"/>
  </DataDictionary>
  <Sampler method="SRS" N="100" export="sampler_SRS_100.dat"/>
  <Sampler method="LHS" N="100" export="sampler_LHS_100.dat"/>
</Problem>
```
Launch URANIE to generate a design of experiments
- A plan with a total of 100 cases

Generate all data set for TrioCDF
- Note: a “template” data set must be defined

Launch several TrioCFD
- 10 machines with 10 jobs
- 2 hours for all calculus
After all TrioCFD calculus we have 100 results

- Particular: temperature values are dumped at the sensor position

URANIE aggregates the 3 input parameters and the output data of interest (temperature)

- Ready for analysis
COBWEB with blue lines for temperatures in [+18°C : +21°C]
12. ROADMAP FOR 2016

- Finalize the data model
- Continue the Model-View-Controller pattern
- Build new small or middle size applications
- Migrate some existing applications
- And operate more functionalities of URANIE from SALOME:
  - Metamodel
  - Uncertainty propagation
  - Sensitivity analysis
  - Optimization
  - Etc.