EURIWARE

Developer Guide

Integration of new meshing algorithm as plug-in to SALOME Mesh module

Rédaction	Vérification	Approbation
J DOROVSKIKH	V SANDLER	G DAVID

Date : 17 Septembre 2010

Avant-propos:

This document is a developer guide that provides a details description of new meshing plug-in integration procedure.

SOMMAIRE

1. Introduction

This document describes how to add custom meshing tools to the SALOME application.

SALOME Mesh module provides a plug-in mechanism that allows using custom hypotheses and algorithms for mesh computation. Each meshing plug-in is as a set of hypotheses and algorithms. Each algorithm is intended to create a set of mesh elements from initial data:

- Given topology (TopoDS_Shape to be meshed).
- Results of other algorithms work (existing mesh elements). Usually (but not always) algorithm of higher dimension uses mesh elements and nodes, generated by algorithms of lower dimensions. For example, Quadrangle_2D algorithm builds faces grid, basing on edges discretization, done by any 1D algorithm.
- Discretization parameters, defined via hypotheses, compatible with the algorithm.

This document is a developer guide that provides a details description of new meshing plug-in integration procedure, step by step.

2. Integration of meshing plug-in

2.1 Create your plug-in module

Create on disk a directory structure like for usual SALOME module. In the `src` directory you will create directories for server and, optionally, client libraries of your plug-in. For example, see structure of `NETGENPLUGIN_SRC`.

- Server library provides your algorithms and/or hypotheses implementation.
- Client library provides implementation of graphic interface for your hypotheses creation.
- Choose a name for your plug-in. Here after in this document we will mention it as <MyPluginName>.

2.2 Implement server plug-in library

2.2.1 Implement functionality of the hypotheses and algorithms

Server library is the main part of your plug-in. Here you have to implement the way your algorithms will generate mesh and the data your hypotheses will keep.

Inherit corresponding classes from SMESH, for example:

```
class NETGENPlugin_NETGEN_3D: public SMESH_3D_Algo
```

For algorithm it is necessary to
- Define its name and suitable hypotheses in the constructor.
- Redefine methods `CheckHypothesis()`, `Compute()` and operators.

For hypothesis implementation you have to

- o Define its name.
- o Set parameter `_param_algo_dim` to dimension of mesh elements, which will be generated by your algorithm (which will use this hypothesis).

See an example in:

```
SMESH_SRC/src/StdMeshers/StdMeshers_*.*
NETGENPLUGIN_SRC/src/NETGENPlugin/NETGENPlugin_*.*
```

2.2.2 Define CORBA interfaces for your hypotheses and algorithms

Create an IDL file in the `idl` directory of your module folders structure. Define there your algorithms and/or hypotheses interfaces. Inherit corresponding SMESH interfaces, for example:

```
interface NETGENPlugin_NETGEN_3D : SMESH::SMESH_3D_Algo
```

This interface should define API methods for your hypotheses edition. You will call them from your client (GUI) library or from python scripts to set your hypotheses parameters.
See an example in:

```
SMESH_SRC/idl/SMESH_BasicHypothesis.idl
NETGENPLUGIN_SRC/src/NETGENPlugin_Algorithm.idl
```

2.2.3 Implement CORBA interfaces

Inherit your classes, implementing declared IDL interfaces, from the corresponding classes of `SMESH_I` package and from your CORBA interfaces, for example:

```
class NETGENPlugin_NETGEN_3D_i:
  public virtual POA_NETGENPlugin::NETGENPlugin_NETGEN_3D,
  public virtual SMESH_3D_Algo_i
```

Include your IDL files. For example, if you have described your interface in file `NETGENPlugin_Algorithm.idl`, put the following two lines at the beginning of your header file:

```
#include <SALOMEconfig.h>
#include CORBA_SERVER_HEADER(NETGENPlugin_Algorithm)
```

The common header file `SALOMEconfig.h` provides macro `CORBA_SERVER_HEADER` that is used to include IDL interfaces to the C++ code.

See an example in:

```
SMESH_SRC/src/StdMeshers_I/StdMeshers_*_i.*
NETGENPLUGIN_SRC/src/NETGENPlugin/NETGENPlugin_*_i.*
```

2.2.4 Implement factory function

The goal of the factory function is to create hypotheses and algorithms objects by request from SMESH module.

```
extern "C"
{
  GenericHypothesisCreator_i* GetHypothesisCreator(const char* aHypType)
  {
    // Hypotheses
    if (strcmp(aHypName, "LocalLength") == 0)
      aCreator = new HypothesisCreator_i<StdMeshers_LocalLength_i>;
    else if (strcmp(aHypName, "NumberOfSegments") == 0)
    …
    // Algorithms
    else if (strcmp(aHypName, "Regular_1D") == 0)
      aCreator = new HypothesisCreator_i<StdMeshers_Regular_1D_i>;
    else if (strcmp(aHypName, "MEFISTO_2D") == 0)
    …
    return aCreator;
  }
}
```

Note: when this method is called, parameter `aHypType` is initialized by value of `<type>` attribute from your plug-in XML file (see chapter 2.5 for plug-in XML file description) and this is also a value of parameter `_name` of your algorithm/hypothesis class.

See an example in:
```
SMESH_SRC/src/StdMeshers_I/StdMeshers_i.cxx
NETGENPLUGIN_SRC/src/NETGENPlugin/NETGENPlugin_i.cxx
```

2.2.5 Write Makefile.am

Let you want to call your server library `MyServerLib`, then set variable `lib_LTLIBRARIES` value to libMyServerLib.la. You can choose any name for your server library, just specify it correctly in your plug-in XML file as `libMyServerLib.so` (how to do it will be described on page 7).

If implementation of your server library is separated into several packages, or you have other reasons to make some of your header files visible outside concrete package (you want to use them in some other module implementation), do not forget to list them in `salomeinclude_HEADERS` section.

List your source files in section `dist_libMyServerLib_la_SOURCES`.

Specify `libNETGENEngine_la_CPPFLAGS` and `libNETGENEngine_la_LDFLAGS` as required compilation and linkage flags.

See an example in:
```
SMESH_SRC/src/StdMeshers_I/Makefile.am
NETGENPLUGIN_SRC/src/NETGENPlugin/Makefile.am
```

2.3 Implement client (GUI) plugin library

This step is required only if your hypotheses/algorithms need specific GUI for their construction.

It is usually required only for hypotheses, which provide some parameters for their construction. Algorithms are usually created without any specific parameters.

Note, that hypothesis can be also created without any parameters; in such a case algorithm behavior depends just on the hypothesis presence/absence.

2.3.1 Implement the required GUI

GUI consists of a set of the dialog boxes which are used to enter hypotheses parameters by the user.

See an example in:

`SMESH_SRC/src/StdMeshersGUI/StdMeshersGUI_*Creator.*`
`NETGENPLUGIN_SRC/src/GUI/NETGENPluginGUI_*Creator.*`

In order to create your own dialog box:

- If your dialog will contain only one field you can inherit your `Creator` class from `StdMeshersGUI_StdHypothesisCreator` and redefine the methods `storeParams()`, `stdParams()`, `attuneStdWidget()`, `hypTypeName()`.

- In the other case you should inherit it from `SMESHGUI_GenericHypothesisCreator` and redefine methods `buildFrame()`, `storeParams()`, `retrieveParams()`

Example:
`SMESH_SRC/src/StdMeshersGUI/StdMesherGUI_NbSegmentsCreator.*`

All data from your plug-in XML file is accessible in your GUI via `HypothesisData` class:

```
HypothesisData* data = SMESH::GetHypothesisData( aHypType );
```

See `HypothesisData` class definition details at:
`SMESH_SRC/src/SMESHGUI/SMESHGUI_Hypotheses.h`

2.3.2 Provide graphic and textual resources for GUI

Optionally you can implement the GUI resource files `<MyResourceKey>_images.ts` and `<MyResourceKey>_msg_en.ts`. These files are the part of the Qt internationalization system used in SALOME.

See an example in:
`SMESH_SRC/src/StdMeshersGUI/StdMeshers_*.ts`
`NETGENPLUGIN_SRC/src/GUI/NETGENPlugin_*.ts`

Note:

- `ICON_SMESH_TREE_HYPO_<MyHypType1>` specifies an ID of the icon for the Object Browser for the hypothesis `<MyHypType1>`.
- `ICON_SMESH_TREE_ALGO_<MyAlgType1>` specifies an ID of the icon for the Object Browser for the algorithm `<MyAlgType1>`.

See the chapter 2.5 for more details about meaning of the `MyResourceKey`, `MyHypType1`, `MyAlgType1`.

2.3.3 Implement Hypothesis Creator and factory function

Below is a typical code of the factory function that creates and export new instance of the hypothesis creator class. This method is automatically invoked from SALOME.

```
extern "C"
{
  SMESHGUI_GenericHypothesisCreator*
      GetHypothesisCreator( const QString& aHypType )
  {
    if( aHypType=="NumberOfSegments" )
      return new StdMeshersGUI_NbSegmentsCreator();
    else
      return new StdMeshersGUI_StdHypothesisCreator( aHypType );
  }
}
```

Here aHypType parameter is used to pass the value of the <type> attribute in your plug0in XML file (see 2.5).

See an example in:

```
SMESH_SRC/src/StdMeshersGUI/StdMeshersGUI.cxx
SMESH_SRC/src/StdMeshersGUI/StdMeshersGUI_StdHypothesisCreator
SMESH_SRC/src/StdMeshersGUI/StdMeshersGUI_NbSegmentsCreator
```

2.3.4 Write Makefile.am

Let you want to call your client library `MyClientLib`, then set variable `lib_LTLIBRARIES` value to `libMyClientLib.la`. This is the default name of the library. You can choose any name for your client library; then specify it correctly in your plug-in XML file as `libMyClientLib.so`.

If you want to export some of your header files (e.g. you want to use them in some other module implementation), do not forget to list them in the `salomeinclude_HEADERS` section.

List your source files in the section `dist_libMyClientLib_la_SOURCES`.

Define `MOC_FILES` variable as a list of files, which will be generated automatically for Qt dialogs.

Set `nodist_libMyClientLib_la_SOURCES` variable value to the `$(MOC_FILES)` since these files must not be included in a distribution.

Also set `nodist_salomeres_DATA` to the list of *.qm files; these files will be automatically generated from the corresponding resource *.ts files
.
Specify `libMyClientLib_la_CPPFLAGS` and `libMyClientLib_la_LDFLAGS` as required compilation and linkage flags.

See an example in:

```
SMESH_SRC/src/StdMeshersGUI/Makefile.am
NETGENPLUGIN_SRC/src/GUI/Makefile.am
```

2.4 Create icons for the Object browser

Icons are image files which are used for the displaying of the hypotheses and algorithms in SALOME Object Browser. If your hypotheses/algorithms do not need specific GUI, but you want to provide icons for object browser, see 2.3.2 chapter.

2.5 Describe your plugin in special XML resource file

You should create an XML file named `<MyPluginName>.xml` which should describe all the algorithms and hypotheses, implemented by your plug-in package. See sampleof such a file below:

```
<meshers-group name="MyName"
               resources="MyResourceKey"
               server-lib="libMyServerLib.so"
               gui-lib="libMyClientLib.so">
  <hypotheses>
    <hypothesis type="MyHypType1"
                label-id="My beautiful 1D hypothesis"
                icon-id="my_hypo_1_icon.png"
                dim="1"/>
    <hypothesis type="MyHypType2"
                label-id="My beautiful 3D hypothesis"
                icon-id="my_hypo_2_icon.png"
                dim="3"
                need-geom="false"
                auxiliary="true"/>
  </hypotheses>
  <algorithms>
    <algorithm type="MyAlgType1"
               label-id="My beautiful 1D algorithm"
               icon-id="my_algo_1_icon.png"/>
               hypos="MyHypType1,…"
               opt-hypos="MyHypType5,…"
               input="VERTEX"
               output="EDGE"
               dim="1"/>
    <algorithm type="MyAlgType2"
               label-id="My beautiful 3D algorithm"
               icon-id="my_algo_2_icon.png"
               input="QUAD"
              need-geom=”false”
              support-submeshes="true"
               dim="3"/>
  </algorithms>
</meshers-group>
<hypotheses-set-group>
  <hypotheses-set name="Automatic Tetrahedralization"
                  hypos="MaxLength"
                  algos="Regular_1D, MEFISTO_2D, NETGEN_3D"/>
  <hypotheses-set name="Automatic Hexahedralization"
                  hypos="NumberOfSegments"
                  algos="Regular_1D, Quadrangle_2D, Hexa_3D"/>
</hypotheses-set-group>
```

See an example in:

```
    SMESH_SRC/resources/StdMeshers.xml
    NETGENPLUGIN_SRC/resources/NETGENPlugin.xml
```

Attributes of the `<meshers-group>` tag:

- Value of the `<name>` attribute is used to collect hypotheses/algorithms in groups, when they are displayed in the algorithm/hypothesis creation dialog box in GUI. You can also use this attribute for short description of your meshing plug-n (implementing your GUI).
- Value of the `<resources>` attribute ("MyResourceKey" in above example) is used to access resources (messages and icons) from the GUI (see chapter 2.3.2). It should coincide with the name of plug-in.
- Value of the `<server-lib>` attribute is a name of your meshing plug-in’s server library (see chapter 2.2.5).
- Value of the `<gui-lib>` attribute is a name of your meshing plug-in’s client library (see chapter 2.3.4).

Attributes of the `<hypothesis/algorithm>` tag:

- Value of the `<type>` attribute is an unique name of the hypothesis/algorithm.
  - It is a value of the _name field of your hypothesis class (see chapter 2.2.1, implementation of the constructor of `StdMeshers_LocalLength` class: _name `= "LocalLength"`).

- It is a key to each certain hypothesis class (see chapter 2.2.4, implementation of `GetHypothesisCreator()` method in the `StdMeshers_i.cxx`).
- It is a key to each certain hypothesis GUI (see chapter 2.3.1, for example implementation of `StdMeshersGUI_StdHypothesisCreator` class, usage of method `hypType()`).
- It is a key to algorithm/hypothesis icon in the Object Browser (see chapter 2.3.2).

- o Value of the `<label-id>` attribute is displayed in the GUI in the list of available hypotheses/algorithms ("Create Hypothesis/Algorithm" dialog).
- o Value of the `<icon-id>` attribute is a name of the icon file, which is displayed in the GUI in the list of available hypotheses/algorithms ("Create Hypothesis/Algorithm" dialog).
- o Value of the `<dim>` attribute means algorithm/hypothesis dimension and is used in GUI as algorithms/hypotheses of different dimensions are created separately.
- o If `<need-geom>` attribute is set to “true”, a hypothesis/algorithm can be used for creating mesh without underlying geometry.

Specific attributes of the `<hypothesis>` tag:
- o If `<auxiliary>` attribute is set to “true”, a hypothesis can be used in addition to an assigned “main” hypothesis (such auxiliary hypothesis can be selected in “Create Mesh” dialog in a special area).

Specific attributes of <algorithm> tag:
- o Value of the `<hypos>` attribute is a list of hypotheses, available for this algorithm.
- o Value of the `<opt-hypos>` attribute is a list of the optional hypotheses, compatible with this algorithm (such as `Propagation`, `QuadranglePreference`, etc.)
- o Value of the `<input>` attribute means type of mesh elements, required to present in mesh for this algorithm work.
- o Value of the `<output>` attribute means type of mesh elements, generated by this algorithm.
- o If `<support-submeshes>` attribute is set to “true”, the algorithm building all-dim elements supports sub-meshes.

Attributes of the `<hypotheses-set>` tag:
- o Value of the `<name>` attribute is a name of set.
- o Value of the `<hypos>` attribute is a list of hypotheses, which will be created automatically, if user selects this set.
- o Value of the `<algos>` attribute is a list of algorithms, which will be created automatically, if user selects this set.

Note: all attributes values from this XML file will be accessible in the GUI via `HypothesisData` and `HypothesesSet` classes (see 2.3.1).

2.6 Build your plug-in

Configure and build your plug-in (build_configure, configure, make, make install). Check, that all libraries (`libMyServerLib.so` and, optionally, `libMyClientLib.so`) are built; resource files are properly installed, etc.

2.7 Set up environment

Set environment variable `<MyPluginName>_ROOT_DIR` to your plug-in installation directory path. Add your plug-in to the list of all meshing plug-ins in your SALOME configuration file `~/.SalomeApprc.x.x.x`:

```
<section name="SMESH">
  <parameter name="plugins" value=
"NETGENPlugin,GHS3DPlugin,<MyPluginName>"/>
```

The Salome will automatically locate your XML file, searching for it in the following directories:

```
    ${<MyPluginName>_ROOT_DIR}/share/salome/resources/<mypluginname>
    ${SALOME_<MyPluginName>Resources}
    ${KERNEL_ROOT_DIR}/share/salome/resources
```

Add path to your plug-in resources, if you use some (see chapters 2.4 and 2.3.2 about it), in your Salome configuration file .SalomeApprc.x.x.x:

```
<section name="resources">
    ...
    <parameter name="StdMeshers" value=
"${SMESH_ROOT_DIR}/share/salome/resources/smesh"/>
    <parameter name="NETGENPlugin" value=
"${NETGENPLUGIN_ROOT_DIR}/share/salome/resources/netgenplugin"/>
    <parameter name="GHS3DPlugin" value=
"${GHS3DPLUGIN_ROOT_DIR}/share/salome/resources/ghs3dplugin"/>
    <parameter name="MyPluginName" value=
"${MyPluginName_ROOT_DIR}/share/salome/resources/mypluginname"/>
    ...
  </section>
```

2.8 Run SALOME

Run SALOME application, create new study, load Mesh module. Via menu *Mesh* → *Create Mesh* invoke “Create Mesh” dialog box and look at the available algorithms list. If everything is done properly, you should see your algorithms in this list.

Try to create a new hypothesis and check, if your hypotheses are available. Define complete set of algorithms and hypotheses; click “OK” in the mesh creation dialog. Compute the created mesh.

Check the result of computation.

Références documentaires

Documents de référence

Les documents cités dans le présent document ou utiles à la compréhension de son contenu sont :

Titre	Référence
[1] Contrat Marché C434C71440: TMA PAL/SALOME 2008-2010 : Année 2010	C434C71440 / OC2D07081

Historique des révisions

Les versions successives du présent document sont :

	Version	Rédacteur	Date	Objet de la révision
Version en vigueur	V1M6	J DOROVSKIKH	17/09/2010	Update for SALOME 5.1.4
Versions antérieures	V1M5	J DOROVSKIKH	20/06/2007	Update for SALOME 3.2.6
	V1M4	J DOROVSKIKH	07/02/2007	Update for SALOME 3.2.5
	V1M3	J DOROVSKIKH	30/03/2006	Additional remarks
	V1M2	M KAZAKOV	14/02/2006	Minor revision and remarks
	V1M1	J DOROVSKIKH	14/02/2006	Draft version for validation
	V1M0	J DOROVSKIKH	24/01/2006	Initial version