OMEdit - OpenModelica Connection Editor

Adeel Asghar
Motivation

- Modelica models were created using;
  - Textual editors
  - SimForge

- New Graphical User Interface was needed,
  - To overcome the deficiencies of SimForge

OMEEdit – OpenModelica Connection Editor
OMEdit

- OpenModelica Connection Editor
- Features
  - *Modeling* – Easy model creation for Modelica models
  - *Pre-defined models* – Browsing the Modelica Standard library to access the provided models
  - *User defined models* – Users can create their own models for immediate usage and later reuse
  - *Component interfaces* – Smart connection editing for drawing and editing connections between model interfaces
  - *Simulation* – Subsystem for running simulations and specifying simulation parameters start and stop time, etc.
  - *Plotting* – Interface to plot variables from simulated models
OMEdit - Workflow

model creation using Modelica Annotations

OMC Communication

OMC Proxy

OMEdit - OpenModelica Connection Editor
OMEdit - Windows

- Library Window
- Designer Window
- Messages Window
- Documentation Window
- Plot Window
Library Window

- Contains two tabs,
  - Modelica Standard Library
  - Modelica Files
Designer Window

- It consists of three views,
  - **Icon View** - Shows the model icon view
  - **Diagram View** - Shows the diagram of the model created by the user
  - **Modelica Text View** - Shows the Modelica text of the model
Messages Window

- Messages Window is located at the bottom in OMEdit. The Messages Window consists of 4 types of messages,
  - *General Messages* – Shown in black color
  - *Informational Messages* – Shown in green color
  - *Warning Messages* – Shown in orange color
  - *Error Messages* – Shown in red color
Documentation Window

- Shows the Modelica documentation of component models/libraries in a web view
Plot Window

- Shows a tree containing the list of instance variables.
OMC Corba Interface

- OMC is a short name for OpenModelica Compiler
- Two methods to invoke OMC,
  - As a whole program, called at the operating-system level, e.g. as a command.
  - As a server, called via a Corba client-server interface from client applications.
Invoking OMC through Corba

- Start `omc.exe` with special arguments,
  - `+d=interactiveCorba`
  - `+c=IOR-filename`
- A file with name specified in `+c` argument is created in temp directory.
- Read the Interoperable Object Reference (IOR) written in the file.
- Create the Corba object using the string-to-object method.

```cpp
QFile objectRefFile (path_to_IOR_File);
int argc = 2;
static const char *argv[] = { "-ORBgiopMaxMsgSize", "10485760" };
CORBA::ORB_var orb = CORBA::ORB_init(argc, (char **)argv);
objectRefFile.open(QIODevice::ReadOnly);
char buf[1024];
objectRefFile.readLine( buf, sizeof(buf) );
QString uri( (const char*)buf );
CORBA::Object_var obj = orb->string_to_object(uri.trimmed().toLatin1());
```
OMC API Enhancements

• Problems
  • Annotations for some models could not be retrieved correctly.
  • `renameComponent` command was very slow.
  • Package `Modelica.UsersGuide` does not have any icon/diagram annotation.

• Remedies
  • Instantiating (elaborating) the models.
  • `renameComponent` command goes through all the models and components and do refactoring. A new API command `renameComponentInClass` was introduced.
  • `getNamedAnnotation` command is added in OMC API. Which if returns true a predefined icon is used.
Modelica Annotations

- Annotations are used for storing extra information about a model such as graphics, documentation or versioning etc.
- OMEdit uses three types of Modelica annotations,
  - Graphical Annotations.
  - Connection Annotations.
  - Documentation Annotations.
Graphical Annotations

- Graphical annotation consists of two abstraction layers:
  - Icon Layer
  - Diagram Layer

- Graphical Elements
  - Line
  - Polygon
  - Rectangle
  - Ellipse
  - Text
  - Bitmap
Connection Annotations

- It defines graphical representation of a connection between two component models. An example of connection annotation string is,

  \[
  \text{connect (a.x, b.x)} \\
  \text{annotation(Line(points=\{-25,30\}, \{10,30\}, \{10, -20\}, \{40,-20\})))};
  \]

- \(N\) points = \(N - 1\) lines

- OMEdit provides,
  - A *Connector* class for each connection.
  - Keeps the track of all connections of a model.
  - Checking for incompatible types of connectors.
Connection Annotations (cont.)
Documentation Annotations

- Documentation annotation is used for textual description. The documentation annotation written as;

\[
documentation\_annotation: \\
\text{annotation}("\text{Documentation}\ "("\text{info}\ "="\text{STRING} \\
[","\text{revisions}\ "="\text{STRING}]\"")\"
\]

- OMEdit requests OMC for the documentation of a specific component/library through the `getDocumentationAnnotation` command.
- OMC returns the info annotation contained inside documentation annotation which is a string.
- The tags `<HTML>` and `</HTML>` defines the start and end of the string.
Qt’s QWebView class is used to display documentation annotation.

The HTML string of documentation annotation contains four types of links,

- Hyperlinks – Used to navigate to external websites.
- Image Links – Used to reference the local image files.
- Modelica Links – Used for linking to other component models.
- Mailto Links – Used to display email addresses that can be used for future contacts.

QWebView has built-in support for images.

Hyperlinks and Mailto links are handled through QDesktopServices class.

The Modelica links are special links which starts with Modelica:// and reference to some component model or a package.
// if url contains http or mailto: send it to desktop services
if (((url.toString()).startsWith("http")) or (url.toString()).startsWith("mailto:"))
{
    QDesktopServices::openUrl(url);
}
// if the user has clicked on some Modelica Links like Modelica://
else if (url.toString().startsWith("Modelica"))
{
    // remove Modelica:// from link
    QString className;
    className = url.toString().mid(10, url.toString().length() - 1);
    // send the new className to DocumentationWidget
    getDocumentationAnnotation(className);
}
Documentation Annotations (cont.)
Simulation and Plotting

- OMC API `simulate` command.
- Creates a simulation result file.
- The file contains,
  - List of instance variables with values over the time.
- Tree based on simulation result file.
- Existing OpenModelica Plot Window is used.
Thank You
Mahalo
Kiitos
Toda
Thanks
Merci
Obrigado
Gracias
Grazie
Takk