Incremental Compilation in OpenModelica

Contact: Peter Fritzson (peter.fritzson@liu.se, tel: 0708-281484) or Lena Rogovchenko-Buffoni (olena.rogovchenko@liu.se) or Martin Sjölund (martin.sjolund@liu.se)
PELAB – Programming Environment Lab, Institutionen för Datavetenskap
www.openmodelica.org

PELAB, together with the Open Source Modelica Consortium (an international open source effort supported by 45 organizations, see www.openmodelica.org) develops OpenModelica, an open-source Modelica-based modeling and simulation environment intended for industrial and academic usage. OpenModelica, includes the OpenModelica Compiler (OMC) of the Modelica language, an equation-based and functional language aimed at modeling of cyber-physical systems. like cars, aircraft, power plants, wind power, etc.

An important goal for the OpenModelica compiler is to have very short turn-around time, around 2 seconds, for re-compilations with small code changes. This is important for more interactive usage of the Modelica language. It should even be possible to stop a program, make small changes, and continue execution in many cases.

The master thesis project should develop incremental compilation for OpenModelica. This is partly based on earlier work of incremental dynamic loading of functions and separate compilation at the module level of the OpenModelica compiler (OMC). This task can be divided into three or more sub-tasks:

- Incremental compilation after a code change in the body of the function. The function is recompiled and dynamically loaded into the executable image in memory to replace the original function.
- Incremental compilation after changes to declarations. A cross-reference dependency analysis keeps track of dependencies, which functions depend on which declarations.
- Editor interface. The editor keeps track which files or functions are changed. Alternatively, a fast diff mechanism can detect which parts of files have been changed.

The work is suitable for one or two students, and can be divided according to sub-task. Most of the development is in a functional language called MetaModelica, including support for pattern matching and AST transformations. It has Modelica-style syntax but functionality similar to functional languages like F# and OCAML. A short introductory course is available. Some run-time system development will be in C. The whole OpenModelica compiler is written in about 200 000 lines of MetaModelica. The compiler is bootstrapped, i.e., it compiles itself. The student should be interested in advanced programming and compiler construction.