Tutorial at Modelica'2009: Using the MDT-ModelicaML Eclipse Plugin for Modelica Development and UML-Modelica Systems Engineering.

by

Adrian Pop (LIU) and Wladimir Schamai (EADS)

Bring your laptop –hands-on exercises

The OpenModelica MDT (Modelica Development Tooling) Eclipse plugin and UML profile for Modelica, named ModelicaML, is presented in this tutorial, including hands-on exercises.

MDT is primarily aimed at development of large Modelica models or libraries. It has support for browsing, editing, code completion, type information in popup windows, automatic indentation, model refactoring, building executables, and debugging (debugging currently only for MetaModelica). It also allows simulation and plotting from a special command window. Thus it provides a rather complete integrated development environment, and it is also the first available Eclipse plugin for an equation-based language. It is currently used regularly for Modelica model development as well as for development of the OpenModelica compiler.

The ModelicaML UML profile is an open-source implementation being developed by the Open Source Modelica Consortium (www.openmodelica.org), It is based on the OMG UML and reuses concepts from the SysML (Systems Modeling Language) profile and required for system specification. UML diagrams are also extended to support all Modelica constructs. With ModelicaML system engineers are able to specify entire systems, starting from requirements, continuing with behavior and finally perform system simulations based on the Modelica simulation technology and standard libraries.

A presentation of MDT/ModelicaML, its integration in Eclipse, and a demonstration will be given. Workshop participations will be able install the software and use it for some hands-on exercises.

Notes:

The Unified Language (UML) has been created to assist software development processes by providing means to capture software system structure and behavior. This has evolved into the main standard for Model Driven Development in software modeling.

The System Modeling Language (SysML) is a graphical modeling language for systems engineering applications. SysML was developed by systems engineering experts, and was adopted by OMG in 2006. SysML is built on top of UML and tailored to the needs of system engineers by supporting specification, analysis, design, verification and validation of broad range of systems and system-of-systems.

ModelicaML is a graphical notation that is based on the OMG UML and which reuses SysML concepts (e.g. Requirement). It extends the UML/SysML and is designed towards generation of executable Modelica code to be simulated using Modelica simulation tools. ModelicaML is currently implemented as an UML profile using the Eclipse UML2 technologies. This allows usage of the ModelicaML Profile in any Eclipse-based UML/SysML tool, such as Topcased or PapyrusUML.

Short Biographies

Adrian Pop is currently working with model-driven development tools, including the OpenModelica compiler, the ModelicaML profile, and the MDT Eclipse plugin. He is currently technical coordinator at the Open Source Modelica Consortium and is also part-time at Linköping University. He received a PhD on the topic of Integrated Model-Driven Development Environments from Linköping University in 2008.

Wladimir Schamai is a researcher in the Systems Engineering department of EADS Innovation Works (the corporate research center of the EADS company) and a PhD student at the Linkoping University. After studying computer science he has been a performing software developer for few years. Since 2005, he is actively involved in research concerning Model-Based Systems Engineering. He is a member of INCOSE and OMG.

Preprint submitted to 7th International Modelica Conference.
Received June 30, 2009.

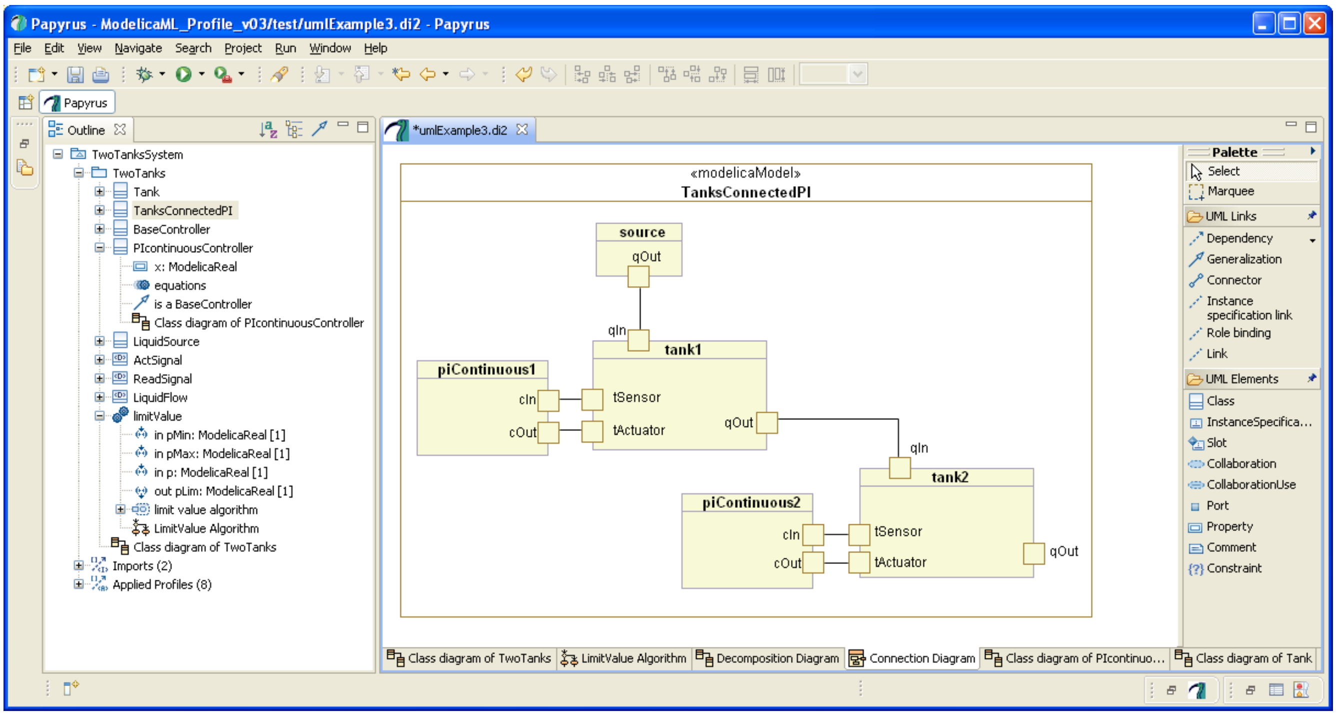
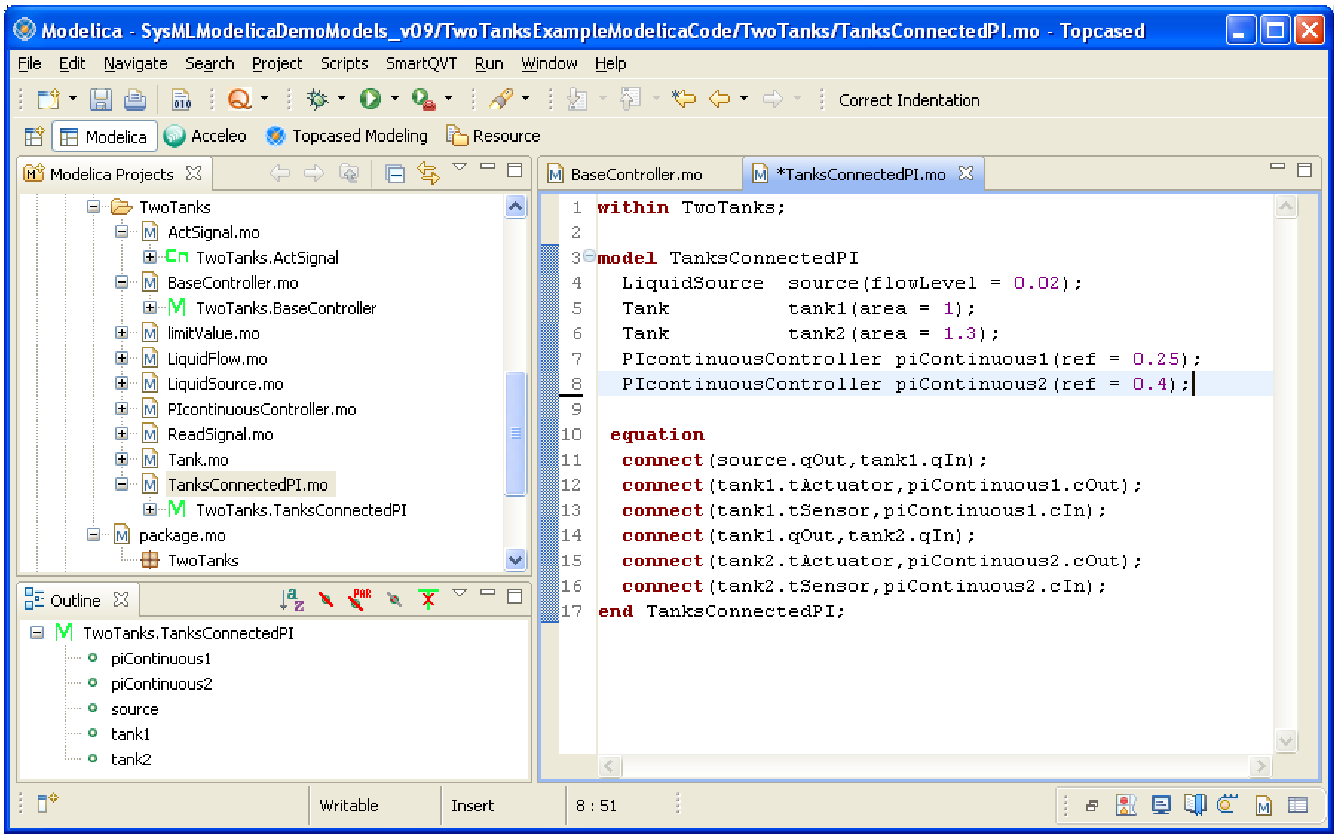


Figure 1: Example of a ModelicaML model in PapyrusUML modeling tool



```
within TwoTanks;

model TanksConnectedPI
  LiquidSource   source(flowLevel = 0.02);
  Tank           tank1(area = 1);
  Tank           tank2(area = 1.3);
  PIcontinuousController piContinuous1(ref = 0.25);
  PIcontinuousController piContinuous2(ref = 0.4);

 equation
  connect(source.qOut,tank1.qIn);
  connect(tank1.tActuator,piContinuous1.cOut);
  connect(tank1.tSensor,piContinuous1.cIn);
  connect(tank1.qOut,tank2.qIn);
  connect(tank2.tActuator,piContinuous2.cOut);
  connect(tank2.tSensor,piContinuous2.cIn);
end TanksConnectedPI;
```

Figure 2: Example of a Modelica code in Modelica Development Tooling (MDT)

Preprint submitted to 7th International Modelica Conference.
Received June 30, 2009.