W. Freieisen, R. Keber, W. Medetz, D. Stelzmüller:
Testing PLC programs with Modelica.
Modelica Workshop 2000 Proceedings, pp. 109-110.

Testing PLC programs with Modelica

Wolfgang Freiseisen, Robert Keber, Wilhelm Medetz, Dietmar Stelzmüller
SCCH – Software Competence Center Hagenberg, Austria
www.scch.at

Abstract

This paper describes the current status of the VirtMould project. The goal of the project is to develop an injection moulding machine simulator, that allows an offline simulation and testing of IEC 1131 based plc programs.

Introduction

ENGEL is a leading manufacturer of injection moulding machines, producing more than 1000 machines per year. Only 30% of these are standard machines, all others are configured individually according customer requirements. This also requires individual plc software development (based on IEC-1131) for each of these machines.

The final machine is usually only two weeks available for testing (including software test). This places a very high time constraint on software development, as the software test (and fixing potential software bugs) must be finished within this time. As software functionality and complexity continuously increases, also this bottleneck is continuously increased.

In order to improve this situation, each plc programmer should have a "virtual injection moulding machine" on his desktop. It must be integrated with the IEC -1131 programming environment and should allow interactive testing and debugging of plc programs. A prerequisite therefore is, that also the IEC-1131 programming environment supports execution (simulation) of plc programs on the desktop (usually plc programs can only be executed on a real plc). It must also provide an interface for coupling the plc simulation with the machine simulation.

The main goals of the VirtMould project are:

- Increase plc software quality
- Decrease plc software development time
- Increase customer support and satisfaction

In order to achieve a high user acceptance it is essential that the plc programmer has no extra work to define the simulation models. The simulation models are generated automatically based on existing hydraulic and electronic CAD models of the machine.

The testing environment will also provide the possibility to test program changes for already delivered machines. To reduce errors due to side effects of program changes a regression test environment will perform an automatic test run.

The project will be realized in several phases. In the first phase, only very approximate simulation models will be used to simulate the behavior of the injection moulding machines. The main goal of these models is not quantitative but qualitative correct simulation. Depending on the achieved results the models will be refined later.

Modelica

Several commercially available simulation systems were analyzed for usability within this project. The main requirements were:

- Integration with plc simulator through a COM interface. The simulation model execution must be triggered from the plc simulator.
- Generation of simulation models from CAD models.
- Possibility to add a custom graphical editor. The graphical editing should be similar to the CAD system already in use.
- Possibility to add a custom visualization that supports special debugging and testing functionally like assertions and breakpoints
- A standard hydraulic library should be available.

Due to these requirements the final decision was to implement a Modelica subset compiler for this project. This provides an easy integration with the other components and later will also provide the possibility to use a third-party Modelica compiler or hydraulic library.

Architecture

A parser implemented with ANTLR converts standard Modelica files to an intermediate parse tree representation based on XML (figure 1) . These ModelicaXML files contain all the information of the Modelica language, but provide easier access to the simulation models. The graphical editor is based on these XML files and provides simple interactive editing of the simulation models. The automatic model generator, that converts CAD models to Modelica models, imports XML based CAD models and outputs ModelicaXML models. In a next step the code generator uses these ModelicaXML files to generate executable C++ simulation code.

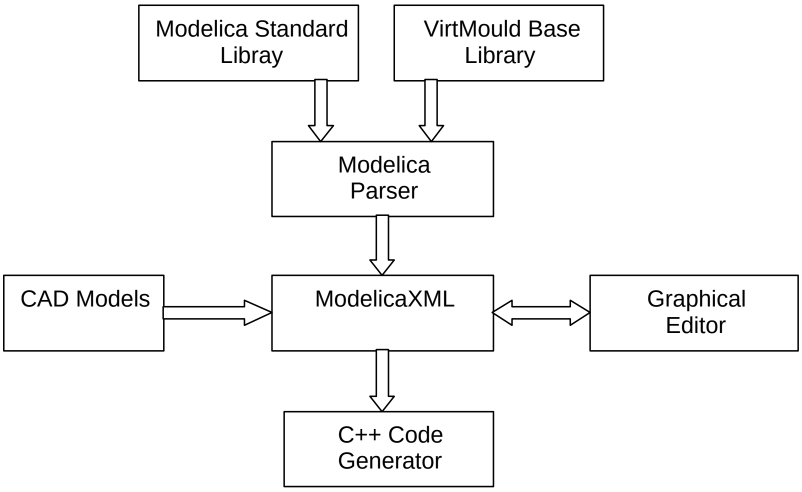

Figure 1: Architecture

Modelica Compiler

The implemented Modelica compiler currently only supports a subset of the Modelica language, mainly input output blocks. The main focus of the compiler is on an efficient integration of discrete simulation with continuous simulation. Therefore an event based discrete scheduler has been implemented. It uses the information of the connected input output blocks as a directed graph and provide an efficient change propagation, thus minimizing the need to recomputed the whole simulation model at each simulation step.

Modeling

In the first phase simple simulation models of the complete machine (hydraulic, mechanic, electronic, pneumatic, logic) are created. These are built upon input output blocks using discrete simulation logic and simple ODE´s. The main goal of these models is not quantitative but qualitative correct simulation. The overall simulation model can be viewed as a large input output block, where the input are the ou ports of the plc, and the output are the inports of the plc (figure 2).

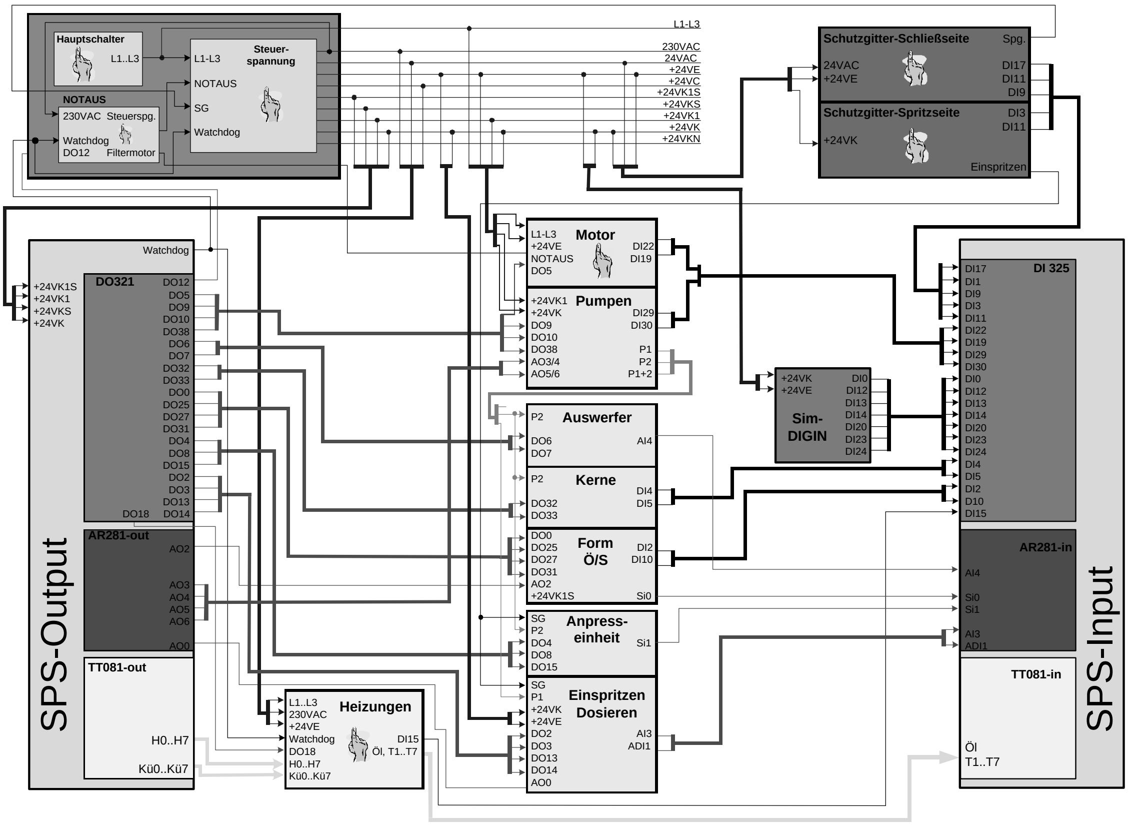

Figure 2: Overall moulding machine model

Model Execution

The interface between the plc simulator and the machine simulator is realized by a COM interface. It supports both synchronous (offline) and asynchronous (real-time) simulation. For the synchronous simulation the simulation steps are triggered by the plc program simulator. This provides an ideal debugging environment as the simulations can be stopped, analyzed, and then continued at any time step. The asynchronous simulation provides some kind of soft real-time simulation where both simulations are executed in real-time. It is also planned to use some Windows NT real-time extension to provide a hardware in the loop environment, where the machine simulator is coupled with a real plc. Figure 3 shows first results of simulating a complete injection cycle.

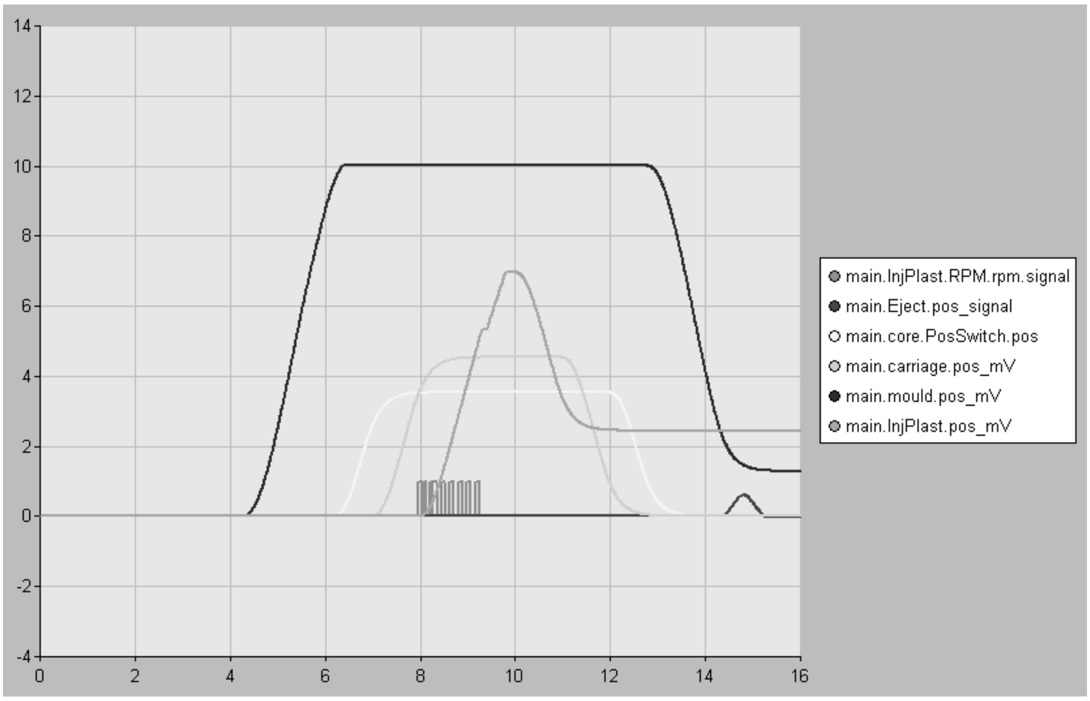

Figure 3: Injection cycle results

Graphical Editor

The graphical editor is based on Objective Views, an MFC based C++ graphical editor framework. It provides standard graphical editing features like add, delete, move, connect and editing of sub models. To provide an easy navigation it will also support a tree view of the simulation model and of available simulation libraries.

Visualization

For the visualization OPC (OLE for process control) is used as a standard interface. This allows the integration of available process visualization tools. All Modelica variables are mapped to OPC tags. OPC provides an easy mechanism to browse available tags and to query the current value. It also provides an event based notification mechanism for tag value changes.

Results and outlook

The current implementation provides a first working injection moulding machine simulator and looks very promising. In addition to the main application area of program testing it might also be used for computer-based training.