Automatic Generation of Graphical User Interfaces for Simulation of Modelica Models

Clemens Schlegel
Schlegel Simulation GmbH
Meichelbeckstr. 8b
D-85356 Freising
cs@schlegel-simulation.de

Reinhard Finsterwalder
Universität der Bundeswehr
München
D-85577 Neubiberg
reinhard.finsterwalder@unibw.de

Abstract

For a certain class of applications simulation models are developed and then rolled out for standalone usage without the tool with which they have been developed. The user is intended to perform simulation runs, to inspect results, to change selected parameters within given bounds, but not to inspect or even change the model itself.

The reasons for such a usage scenario are manifold: The simulation is intended to be used as a black-box tool by non simulation specialists, a component vendor (electric drives, pneumatic or hydraulic components, etc.) likes to demonstrate the performance of his components in the context of a simulation or the model developer may hide model details.

If a model development tool includes code generation the model specific simulator can be setup fully automatic. However, a GUI (graphical user interface) for such a simulator must be developed manually. We developed a tool which automatically generates a simulator GUI from a Modelica model and data definition.

Keywords: graphical user interface generation; Modelica parser; standalone simulator

1 Introduction

The core functionality of a model-specific simulator is to run a simulation experiment, to inspect trajectories and / or scalar results and to change, store and retrieve parameters. The computational part of such a simulator may be set up easily using a code generating model development tool. However, to our knowledge, there are no tools publically available for automatic generation of the graphical user interface (GUI) part. On the other hand nearly all information needed to set up a simulator specific GUI is available in the model code. This GUI, which may be used independent of a general simulation environment, can be set up automatically by parsing a Modelica model. We used Dymola [1] for model development and model specific C-code generation and developed an own tool for GUI generation.

1.1 Limitations

Since the generated model specific C-code can't be changed anymore the GUI's usage is restricted to operations which do not require to change variable dimensions or to replace parts of the model. Arrays must have a fixed dimension or be handled via an external C-function with dynamic storage allocation.

1.2 Core requirements

Focusing on the black-box simulation case we define the following desirable functional requirements for the simulator and the GUI. Some of this requirements map directly to Modelica parameter and record declarations.

- The simulation executable is driven by a parameter and simulation control input file and saves computed trajectories in a result file.
- The GUI is generated at run time by parsing Modelica files, the GUI structure is not stored.
- Only model components and output variables declared on the top hierarchy level of the model are available in the GUI. Protected, inherited, modified and replaced declarations are taken into account, no restrictions on the Modelica language apply.

- Only parameter record classes and top hierarchy model classes (including all inherited and redeclared classes) must be available as Modelica code for GUI generation. Thus the model may contain references to confidential libraries without disclosure of the corresponding Modelica code, external function calls and encrypted classes (if allowed by the modeling tool).
- Names and attributes of Modelica parameters and output variables and restrictions on parameters (protected, read only, min/max values) are retained in the GUI.
- Based on parameter record class parameterization drop-down lists for parameter record selection are set up. This means a model provider (or even a GUI user) may later add parameter records which automatically show up in the corresponding selection list without rebuilding the simulator executable. These records may contain any data type including arrays, optionally a whole record can't be modified and / or its content is not visible. This feature facilitates the usage of datasheet libraries without disclosure of data details.
- It must be possible to read data from external files at runtime without fixed file names.
- All model parameters including records, record names and modified records may be stored on and read from file. The tool must check the consistency of the model, the read in parameter file and the parameter record definitions.

1.3 Additional functionality

Apart from the model parameterization and simulation experiment settings some more information is needed for automatic generation of a handy GUI:

- The top model file name and class name
- The name of the simulator executable
- The user may specify predefined trajectory plots
- The user may specify predefined reports containing descriptive text and trajectory final values

All this information is stored in a configuration file.

2 Parsing the Modelica model

For parsing Modelica [2] files the parser generator tool PCCTS [3] has been used. Based on a grammatical description of a formal language, PCCTS generates C++ code of a corresponding parser which we integrated in our application.

2.1 Parsing the model and the parameter input

All Modelica files in a directory structure are parsed starting from the directory where the configuration is stored. The parser builds an abstract syntax tree for the Modelica code of the complete model hierarchy. The parser fully supports inheritance, modifications and redeclarations within the model hierarchy. Only the data needed for setup of the GUI are retained: model components, model outputs, component parameters, and parameter records. All other tokens are skipped, e.g. algorithm clauses, connect statements, equation clauses, arithmetic expressions.

Since only classes in the model hierarchy are parsed, no information is available from model libraries outside of that hierarchy. In order to generate a complete parameter input file, the remaining parameters are found by parsing the default parameter input file of the simulator (for Dymola it may be generated by the simulator itself).

2.2 Consistency check

It must be checked whether the simulation executable has been generated from the parsed model. Therefore it is verified that all parameters of the model are consistent with the parameters of the parameter input file which may be read in to retrieve previous parameter settings.

2.3 GUI setup

By walking through the abstract syntax tree the graphical user interface is built: tabs for components and controls for model parameters are created and drop-down lists are set up for selecting replaceable parameter records. Thus the user can switch records without having to rebuild the simulation executable. This feature is beyond the functionality of most Modelica tools. Since record names are also stored on the parameter file record selection settings (and not only parameter settings) of previous simulation runs can be retrieved. Modified parameter values are marked red in the GUI.

3 The generated simulator GUI

3.1 General description

At runtime the GUI generator needs the following files:

- Configuration file

- Modelica code of parameter record classes and top hierarchy model classes including all inherited and redeclared classes
- Model specific simulator executable

To start the GUI generation the configuration file is read in. After parsing the tool displays a menu and icon bar and three areas: a model and output navigation tree, a multi document view area for model component parameters, trajectory and report display, and a message area (figure 2, underlying Modelica model see figure 1). Concerning model parameters the GUI has the same functionality as a general simulation tool. For each top level model component a tab is displayed which contains all parameter and record selection controls. Parameter name, default value, unit and descriptive text are shown. Clicking the symbol right of a record selection control the contents of the chosen record is shown. If not protected or read-only parameters may be edited within the defined limits. If a parameter contains a file name the contents of that file is displayed by clicking the respective control (figure 3).

3.2 Performing simulation experiments

The complete parameter setup may be stored on or read from file. To retain the user's parameter changes a new parameter input file is generated before starting a simulation run. That parameter file is enriched by the names of the selected data records. The default simulation experiment setup is contained in the parameter file, but may be overridden using the GUIs simulation menu. After a simulation run the simulation log file is shown in the GUI message area.

3.3 Reading data from file

Since arrays must have a fixed dimension (see section 1.1) a model developer will make use of external tables stored on file handled via an external C-function with dynamic storage allocation. If a model development tool does not support change of string parameters without C-code rebuild a workaround has to be implemented to allow reading from different files. To do so the simulation model has to define fixed file names while the corresponding data records contain a parameter with the specific file name. When starting a simulation run, the specific parameter files are copied to the current directory and renamed to the predefined fixed file names.

3.4 Trajectory and report display

The GUI supports the display of multiple diagrams and reports in multiple windows. While multiple y-axes are supported, trajectories with same unit and similar range are automatically displayed with a common y-axis. Numeric values of a selected trajectory are displayed in addition in a list box (figure 4). Predefined reports show up in a separate window of the multi document view area (figure 4). Trajectories from a previous simulation run may be read in from file and displayed for comparison or used for additional reports.

4 Conclusion

In order to facilitate the development of black-box simulators with a fixed model the automatic generation of simulator specific graphical user interfaces is desirable. A practical approach for building the GUI of such a simulator for Modelica models has been presented. Parsing the model and the data definitions a GUI is set up at run time. The main issues are to check the consistency of the model, the parameter file and the parameter record definitions and to map a smart limitation of the parameter space to the GUI in order to ensure that the simulator can be used as a black-box.
Future development will focus on additional consistency checks and user comfort. Pre-parsed model and data definitions may be integrated directly in a binary code format of the GUI, thus avoiding to read in the model at runtime. The documentation embedded in a Modelica model may be made available in the GUI. Simulation log files may be evaluated automatically to detect convergence problems and event chattering.

References

[1] Dymola Version 7.4. Dassault Systèmes, Lund, Sweden. www.dymola.com.

[2] Modelica Specification, Version 3.1, May 2009. www.modelica.org/documents.

[3] Parr, T.J., Language Translation using PCCTS and C++. Automata Publishing Company, San Jose, 1993.

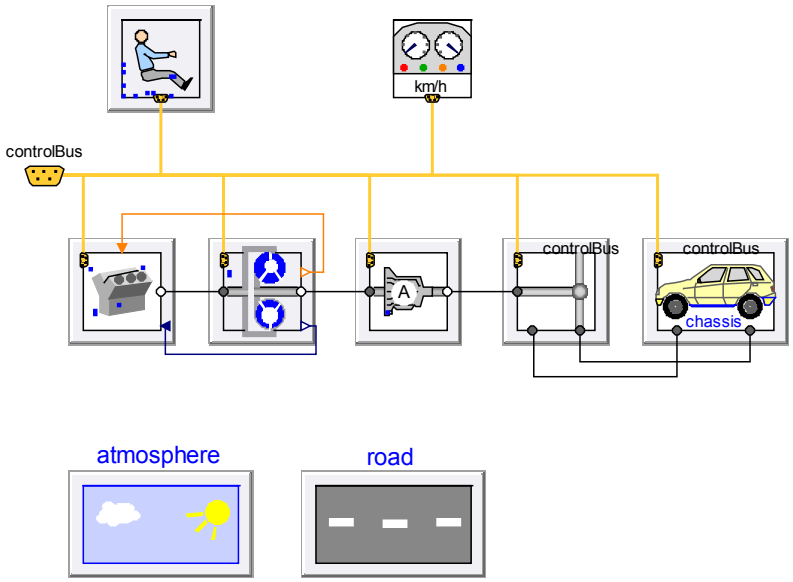

Figure 1: A Modelica simulation model

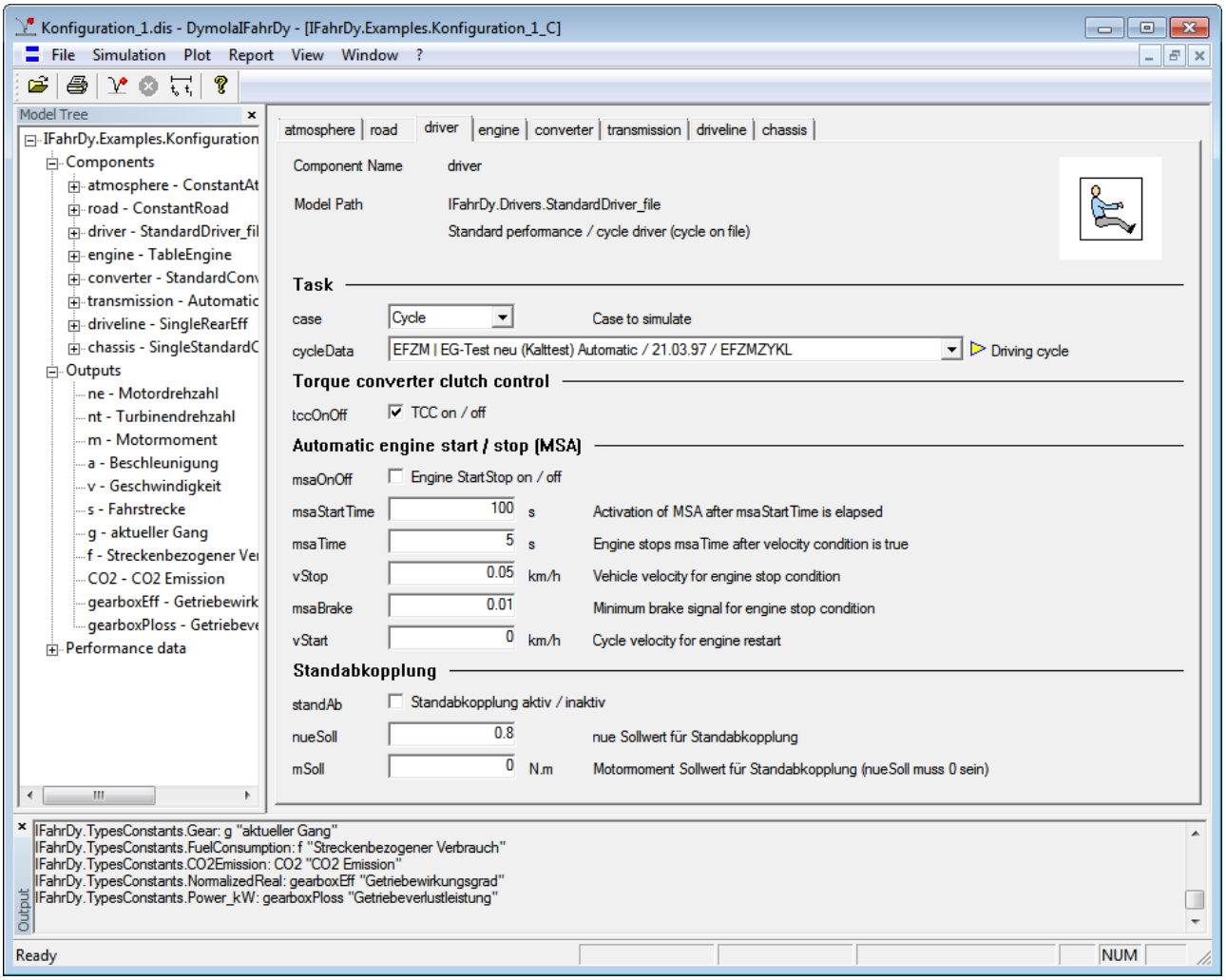

Figure 2: GUI generator main view (model parsed)

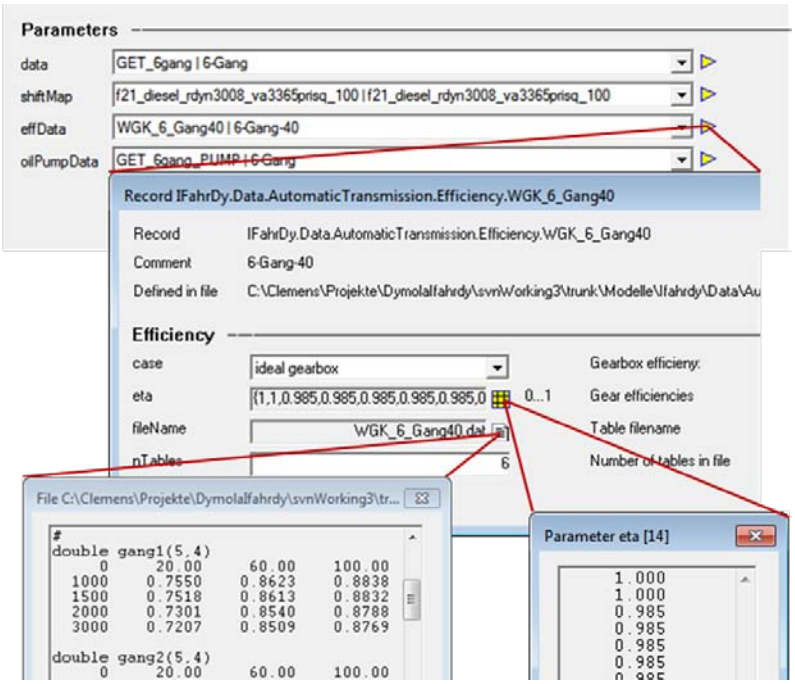

Figure 3: Display of record, file and array contents

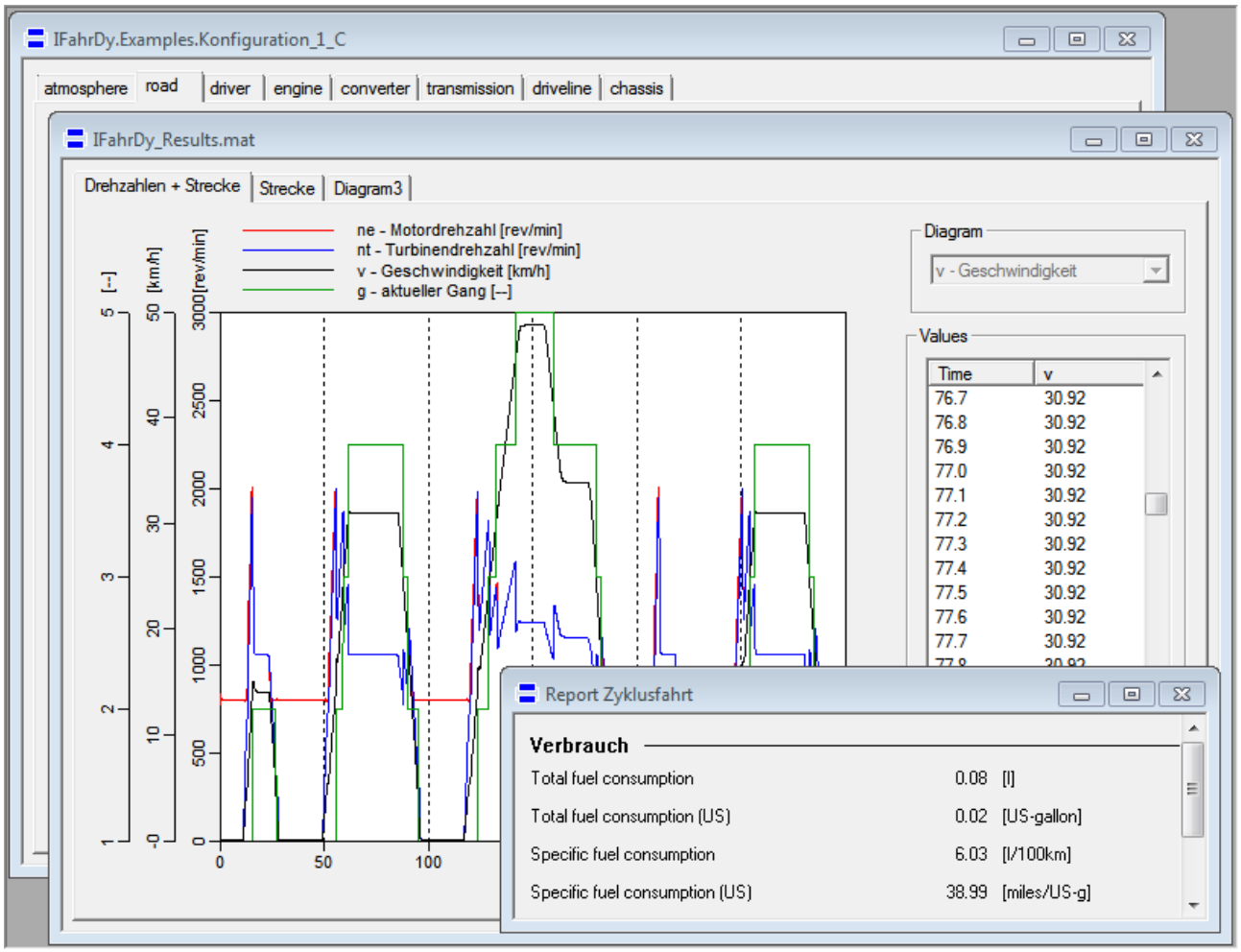

Figure 4: Trajectory and report display