EVALUATION OF A MODELING AND AUTOMATIC C CODE GENERATION TOOLSET AS AN OPEN SOURCE ALTERNATIVE SOLUTION

William Fotso Kom, Xavier Querol

EVALUATION OF A MODELING AND AUTOMATIC C CODE GENERATION TOOLSET AS AN OPEN SOURCE ALTERNATIVE SOLUTION

Authors: William FOTSO KOM, Xavier QUEROL

1. INTRODUCTION

This paper is focused on the model based design (MBD) approach, more particularly on the automatic C code generation. The goal of our project consists in evaluating how far the toolset called Scilab-Scicos and GeneAuto can be used as the open source alternatives to other solutions.

1.1. Context and goal of the project

Nowadays, in automotive and aerospace industry, control systems are intensively used in embedded control units, for running critical functions like spacecraft trajectory control, power steering, thermal and electrical engines. The complexity of these systems has lead to the emergency of a specific new approach for their design process, more appropriate than the classic V-model. It is called MBD (Model-Based Design).

The MBD approach is a concept which aims to transform the process of building embedded systems and software using mathematical models. It is based on representing the system/software with a visual and executable model, which is built, simulated and validated throughout the design process, using high level description language and tools. The activity of implementing the related code is performed automatically by some code generator tools, instead of writing it manually. Thus, engineers can locate and correct errors early in system design, when the time and financial impact of system modification are minimized.

1.2. Presentation of the tools

Before entering in the details of our work, some tools used for the study are presented. On the one hand, we introduce the open source tools like Scilab, Scicos and GeneAuto. On the other hand, we deals with the commercial tools developed by MathWorks® like Matlab® and Simulink®.

Scilab is an open source software, initiated in 1994 by the INRIA. It is now promoted by the Scilab Consortium which is composed of many actors from industrial and academic world (INRIA, DASSAULT-AVIATION, RENAULT, THALES, CNES, PSA PEUGEOT CITROEN, ENPC …). It is a cross-platform computational package and a high-level, numerically oriented programming language. It can be used for signal processing, statistical analysis, image enhancement, fluid dynamic simulations, numerical optimization and modeling and simulation of explicit and implicit dynamical systems.

Scicos is a graphical dynamical system modeller and simulator. It is used to create block diagrams, to model and simulate the dynamics of hybrid dynamical systems (continuous and discrete time), and to compile such models into executable code. It is used for signal processing, systems control and to study physical and biological systems.

GeneAuto is an open-source toolset for converting Simulink®, Stateflow® and Scicos models into executable program code. Output to C is currently available. It was developed in the context of an ITEA European project that ended in December 2008, but it continues to be maintained by Continental Automotive (formerly Siemens VDO) and some partners from aerospace and automotive industry.

Matlab® is a software provided by MathWorks®, for running technical and scientific computing. It offers a lot of modules called “toolboxes”, which can be used for designing and programming a wide variety of complex numeric calculations and algorithms.

Simulink® is an extensible block-diagram environment, also provided by Mathworks®. It is used as an extension of Matlab®, for multidomain simulation and Model-Based Design of dynamic and embedded systems. It provides an interactive graphical environment and a customizable set of block libraries that let the user design, simulate, implement, and test a variety of time-varying systems, including communications, controls, signal processing, video processing, and image processing. The Matlab®-Simulink® software package is currently the most known and the reference toolset for MBD purposes.

2. TOOLS GENERAL STRUCTURE

The two kinds of tools that we have to compare have the same structure, based on three parts:

- a software which provides an interactive technical computing environment, and functions for algorithm development, data analysis, data visualisation, and numeric computation
- an extensible block-diagram environment for simulation and model-based design, used to describe and implement the behaviour of control, signal processing, image processing, communications, and physical systems
- a set of blocks and libraries, which can come from the tool programmers or from some third-party packages

	MathWorks	Open source
Blocks and libraries (Native and third-party)	Toolboxes	Modules
Block-diagram environment Simulator	Simulink	Scicos or Xcos
Numeric solver Basic primitives Language Interpreter	MATLAB	ScicosLab or Scilab

3. TOOLS COMPARISON

The comparison will be based both on our own experience of the tools, on their user manual, and on any reliable documentation from the web. We have considered some differencing criteria.

3.1. Modeling and simulation

3.1.1. Physical domain

This section deals with the physical domains (electrical, mechanical, hydraulic, thermal …) which are applicable to the tools, in order to evaluate which kinds of physical phenomena and systems can be designed with each of the tools.

For the MathWorks® tools, there are specific toolboxes for almost all types of physical systems: mechanical (SimMechanics®), electrical power (SimPowerSystems®), hydraulic (SimHydraulics®), and multidomain (Simscape®) physical systems. Each of these toolboxes is very rich, so that it is possible to model a complete hybrid system. For example, a system including both electrical parts and mechanical parts can be build easily. Some appropriate blocks are used for modeling energy transfers and conversions between the parts of a system.

For the open source tools, there are some modules available for modeling electrical, mechanical or hydraulic systems (using the Coselica library in Scicos). But, there is no module provided for building a multidomain system. Thus, if a system is composed of an electrical generator supplying voltage input to a rotational motor, it won't be possible to design the whole system on the same model.

In conclusion, the MathWorks® tools are more complete than the open source ones considering the hybrid systems modeling.

3.1.2. Control systems design and analysis

This section deals with the possibility to have a simple interface that allows the user to quickly view the response of a given transfer function on different close loop gain, and to adjust the parameters of that transfer function.

Specific toolboxes are available on Matlab®-Simulink® for designing control systems that are commonly used for aircrafts, spacecrafts, propulsion systems, and plants. Some other toolboxes are focused on system identification, which allows creating linear and non linear dynamic models from measured input-output data of a real system. Fuzzy logic systems can also be designed and simulated. These toolsets provide workflow-based GUIs to manage the entire control design process, and lets the user access to the graphical and automated tuning capabilities of the tools. Some command-line interface for developing automated linearization scripts and performing batch linearization are also available.

There is a spacecraft library (CelestLab) for Scilab that has been developed by CNES, for mission analysis purposes. It is used for trajectory analysis and orbit design for various types of missions (around Earth, interplanetary...). CelestLab includes about 200 functions that allow mission designers to perform various tasks such as: orbit propagation, manoeuvre computation, change of reference frames and coordinates, etc …

Scilab provides an identification toolbox, which is used to construct mathematical models of systems from measured input-output data sequences. The tool allows pre-processing of signals, identification of systems and validating of constructed models. The tool is aimed particularly at black-box identification.

In conclusion, the MathWorks® tools are more exhaustive than the open source ones considering the control systems design.

3.1.3. State machines and control logic

This section deals with the possibility to develop state machines and flow charts.

Stateflow® is a toolbox which extends Simulink® with a design environment for developing state machines and flow charts. Stateflow® provides the language elements required to describe complex logic in a natural, readable, and understandable form. It is tightly integrated with Matlab® and Simulink®, providing an efficient environment for designing embedded systems that contain control, supervisory, and mode logic.

There is no specific module in Scilab or ScicosLab for designing state machines. There are some existing discussions about this subject on online forums, but for the moment, Scilab is restricted only to the capability to implement an automatic switching between different implementations of the same algebraic function.

In conclusion, the MathWorks® tools are again more complete than the open source ones considering the state machines development.

3.2. Code generation

This section deals with the possibility for the user to generate code from a model described inside the tool, so that it can be compiled and executed on PC targets or on embedded real-time targets. Programming language that has been focused on is the C language. For both tools, the user can describe model using blocks, or the high-level scientific language (Matlab® or Scilab), or some other standard languages like C and FORTRAN. On the other hand, they also provide the possibility to convert a model into C code for PC targets and for embedded targets.

Gene-Auto is an open-source C code generator for real-time embedded systems. It works both with Matlab®-Simulink® and ScicosLab-Scicos. It takes as input a functional description of an application specified in a high-level modeling language and produces C code (in close future also Ada) as output. Gene-Auto converts different types of models into imperative C. The model types supported by Gene-Auto are data-flow models (from Simulink® and Scicos) and state models (Stateflow® diagrams from Simulink® only).

In addition to Gene-Auto, there are two possible methods for generating C code from Scilab or Scicos. The first method is the native Scicos code Generator. In this case, the generated code can be only used on a computer where Scicos is already installed, because this code contains function calls to some Scicos modules when executed. The second method for generating C code from Scilab blocks is to use the “Scilab2C” toolbox, which is a standalone toolbox of Scilab, developed as part of the hArtes European project, for providing an interface between the Scilab scientific software package and some software design flow. In this case, the code is platform-independent, but Scilab2C need Scilab code as input data. It is then intended for people who have good skill for describing systems using Scilab code instead of Scicos visual models.

Moreover, there are four C code generators available for Matlab®-Simulink®, which are:

- Simulink Coder® (formerly Real-Time Workshop®): it is a MathWorks® product, and it uses target template files to translate Simulink® models into ANSI/ISO C code. The target templates specify the environment on which the generated code will run. The user can develop his own custom targets or use the ready-to-run configurations and third-party targets supported by Simulink Coder®. Simulink Coder lets the user control the way model data appears in the generated code, and provides user-selectable code optimizations to improve code efficiency.

- Embedded Coder® (formerly Real-Time Workshop Embedded Coder®): it is a tool which Generate C and C++ code optimized for embedded targets.

- Stateflow® coder: it is a The MathWorks® product, and generates C code from Stateflow® charts.

- TargetLink®: it is a dSPACE® product, and produces high-quality code directly from Simulink/Stateflow models. It is widely used in Automotive and Aerospace industry. It offers support for AUTOSAR, give a direct link to debugger of Microsoft® Visual Studio

4. APPLYING OPEN SOURCE TOOLS TO THE DEMO APPLICATION

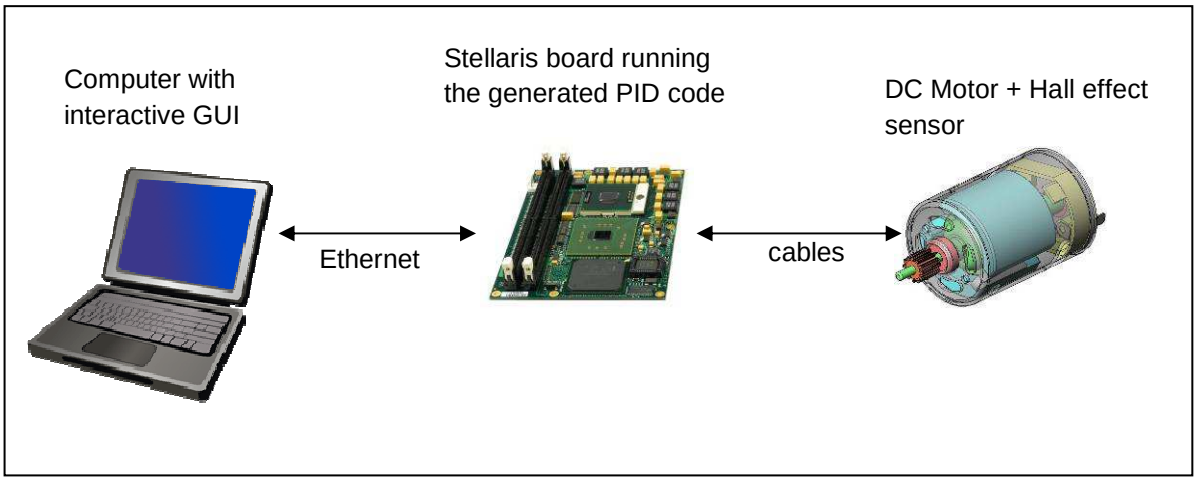
4.1. System overview

The system we built is the combination of:

- a control board: it is an embedded Stellaris evaluation board, with a 32-bit ARM Cortex M3, running the motor controller we had to build. It receives commands and parameters (target speed, Start/Stop, PID parameters ...) from a computer through its Ethernet interface; it controls the motor, and returns the motor status (temperature, speed, current…) to the computer.

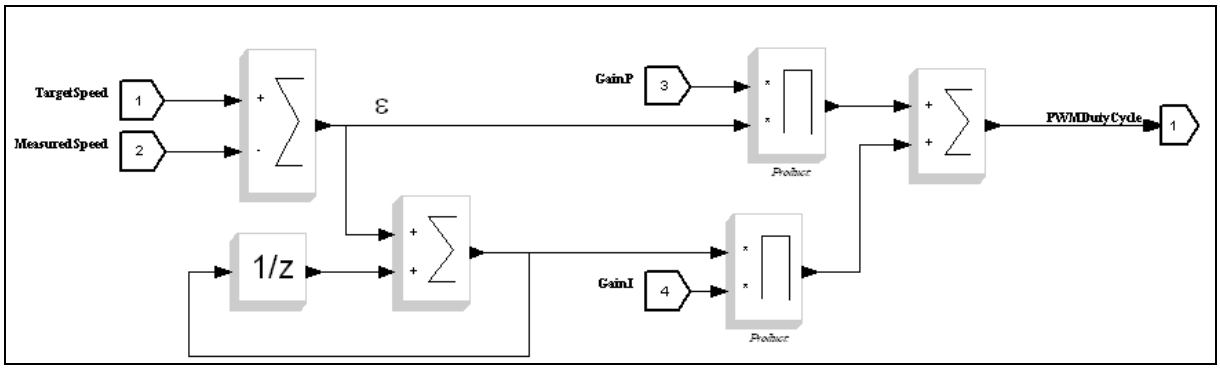
- a computer running an interactive GUI, for sending commands and parameters to the control board; it also receives and displays status returned by the control board.

- a three phases brushless DC motor, combined with a Hall effect sensor for high resolution speed measurement.



4.2. Building of the PID controller

No atomic block was provided by Gene-Auto as PID controller. We then had to create it ourselves by aggregating basic blocks (additions, subtractions, products, unit delay, input and output blocks) in a proper way. Because we couldn't get the motor characteristics from the supplier of the evaluation kit, we were not able to estimate the gains of the PID controller. So, we decided to just model a PI controller, expecting that it will be possible to reach some good values of the P and I gain when running the controller directly on the board. The Scicos model of our PI controller is given by the following figure:



4.3. Generating and analysing the C code

4.3.1. Code generation with Scicos

After creating the diagram, and before launching the Gene-Auto code generator, we have to put the desired region to generate in a SuperBlock. The Scicos model is now ready for the code generation. When the generator is called, the SuperBlock is first translated into the Gene-Auto System Model (GASM) which will be saved as an xml file. After that, the Gene-Auto tool chain is automatically called to generate the associated C code and a new Scicos block based on the generated C code is then created in the Scicos diagram.

4.3.2. The generated files

In the output directory, we find three folders and two scripts organized with the following structure:

- three folders:
  - \macros: contains the Scilab interfacing function of the generated block
  - \src: contains the C source files generated by the Gene-Auto code generator as well as the Scicos block simulation function. This folder also contains the Scilab scripts builder.sce and loader.sce used for building the dynamic library of the C functions

- \Xml: contains the files Controller_PI.xml.mdl created by the Scicos to Gene-Auto translator and a sub directory named tmp which contains the Gene-Auto chain log file and the intermediate versions of the Gene-Auto System Model

- two scripts:
  - builder.sce, used by Scilab to build all the generated stuff
  - loader.sce, used by Scilab to load it.

4.4. Compiling and testing the generated code

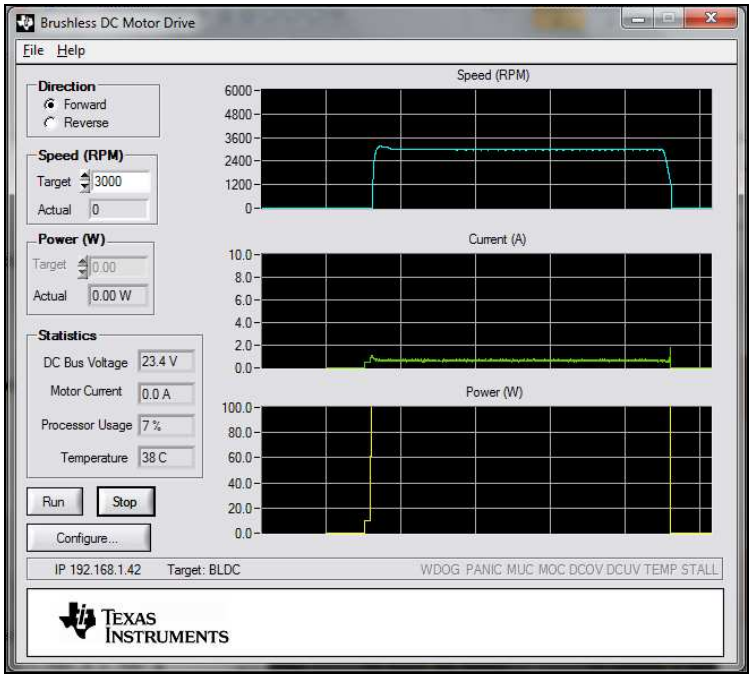
For compiling we needed to take one C-file (Superblock.c) and 3 header files (GATypes, Superblock.h, and Superblock_type.h) are needed from the "src" folder. These files were then integrated into the firmware files of the Stellaris, so that the speed controller calls the generated "SuperBlock_compute" function for regulating the motor speed with the designed PI algorithm.

The whole project was compiled without any error with the *RedSuite* IDE, running a GNU C compiler for ARM Cortex M3 processors.

During testing, we realized that there were some mismatches regarding variable types between the control board firmware and the generated files. It was not easy to find a solution, because there was no way to individually configure the type of each variable of the model. Finally, we decided to directly add a multiplier block into the model, and shift some values before calling the function (see figure below):

```
unsigned long MainSpeedController(void)
{

    IO_SuperBlock_compute param;

    param.B18 = (g_ulSpeed>>14);
    param.B20 = g_ulMeasuredSpeed;
    param.B22 = g_sParameters.lFAdjP>>16;
    param.B24 = g_sParameters.lFAdjI;
    param.B31 = g_lSpeedIntegratorMax;

    SuperBlock_compute(&param);

    return(param.B26);

}
```

We then used the interactive GUI for trying different values of P and I parameters without recompiling the project, and we could monitor the speed of the motor and see how efficient was the generated controller. The first tried values were giving a fast response time, but there were too many vibrations but finally we managed to set some P and I parameters which were efficient:



5. DIFFICULTIES FOUND

5.1. With the Stellaris evaluation kit

The main difficulty when working with this kit was to find the best way for downloading properly the code into the control board. In fact, the board was delivered with an Ethernet bootloader inside, but no tool for performing a classic JTAG download was present. After trying different methods, we finally decide to purchase a JTAG interface, and then we were able to flash the software to the board.

The other difficulty was to get some relevant information about the BL motor delivered with the development kit. In fact, according to the MBD approach, the first step for building the demo application was to build a model of the real motor, so that we can use it for virtually adjust the controller parameters. Then we decided to adjust directly these parameters on the running board later during the testing phase.

5.2. With Siclab-Xcos / ScicosLab-Scicos

First, we met some installation problems when trying to add the Gene-Auto plugin to Scicos. The installation required a C compiler that was not installed on the computer at the beginning. However, even if we have met some difficulties, it was the easiest way to use the tool instead of using it with Eclipse.

When working with third-party blocks, like COSELICA, it is quite impossible to get the description of some blocks, since almost all the related help files were empty. It may be due to the fact that people are working freely on this project and do not have enough time for achieving the documentation before publishing their work.

5.3. With GeneAuto

During the installation, in spite of the fact we followed all the instructions to install the Gene-Auto tool, we had some troubles because one step of the procedure was not described in the notice. We solved this problem going from a third-party forum where our problem and its solution were already described.

The Gene-Auto palette is only composed of generic blocks so the modeling of the corrector was complicated. Moreover, we have tried to use some blocks that were supposed to be supported by the Gene-Auto code generation but in fact, it was not the case.

6. CONCLUSION AND NEXT STEPS

6.1. Conclusion

The couple Scicos/Gene-Auto seems to be a good open source alternative to generate C code for embedded systems, when developing systems which do not require too many different blocks.

The generated code works fine. However, it could be better to control more parameters of the generator: there is no way to separately specify the data type of each data of the model, the variable names of the generated code do not come from the signal names which are in the model; there is no way to add comments from the Scicos model.

Going further with Scicos/Gene-Auto requires getting more skills about designing our own block in Scicos.

6.2. Next steps

6.2.1. Modeling systems

Considering the previous chapters, we have point that the open source tools ScicosLab/Scicos is less complete than Matlab®-Simulink®. On the one hand, Scicos does not offer the possibility to model multidomain systems. On the other hand, the same problem appears for modeling state machine and flow charts as we can do with Stateflow®. These kinds of models are more and more spread in industrial applications so it will be interesting to find and study an open source tool or a plug-in for Scicos that provides these capabilities.

6.2.2. Code generation

The Gene-Auto code generator is not complete enough to answer to all the industrial problems. In fact, it is not compatible with all Scicos blocks. Then, working further with Gene-Auto would require developing by our own some new blocks that can be supported by the

generator. This task demands time and our project duration does not enable us to develop anything.

However, using an open source tool is a benefit considering the point that we can modify its features freely. Thus, the user completely masters the use of its tool and it could be a saving of time for the future development of industrial applications.

7. REFERENCE DOCUMENTS

	Title	Reference	Author(s)	Version	Date
[1]	Implementation of Hybrid Automata in Scicos	TuC01.3	Masoud Najafi and Ramine Nikoukhah		October 2007
[2]	Gene-Auto: automatic software code generation for real-time embedded systems	DASIA08	Ana-Elena Rugina, Dave Thomas, Xavier Olive and Guillaume Veran		
[3]	hArtes - Publishable Final Activity Report	D7T0.1M42	hArtes consortium	3	20 July 2010
[4]	RDK-BLDC Firmware Development Package	SW-RDK-BLDC-UG-5450	Texas Instruments		December 02, 2009