Modelica as a host language for process/control co-simulation and co-design

Filippo Donida, Alberto Leva
Dipartimento di Elettronica e Informazione, Politecnico di Milano
Via Ponzio, 34/5 – 20133 Milano, Italy
{donida,leva}@elet,polimi.it

Abstract

The manuscript describes a project, currently under development at the Politecnico di Milano, the aim of which is to create an integrated environment for the modelling and simulation of process control systems, where the plant(s) are described according to the Modelica object-oriented paradigm, while the control systems are specified in an IEC 61131.3-compliant language, and automatically translated into algorithmic Modelica. Preliminary results will be reported, given the vast scope of the project, but even at the present stage, interesting discussions are possible on the potentialities and pitfalls of Modelica (and even of object-oriented modelling at large) when it comes to describe control algorithms of realistic complexity and size.

1. Introduction

A significant experience is nowadays available on the use of Modelica to model, simulate and assess control systems in the process domain [15, 16, 12, 13]. As witnessed by many references (samples will be given in the final manuscript, including some directly related to the authors' experience) there is a correspondingly vast *corpus* of libraries, models, and system studies [7, 6, 15, 16, 14, 4].

Based on that experience, a critical point when dealing with applications of realistic size is invariantly the "correct" representation of the control system. The object-oriented paradigm can be suitably exploited to allow for various, interchangeable control representations of different, scalable complexity, and such a possibility is definitely *a plus* of Modelica. However, when it comes the time to describe the control system in full detail, the most effective way to do so is not only algorithmic, but compliant with the industrial standard accepted in that domain, the IEC 61131-3 being the most important one [8, 18, 17, 9, 5, 2, 11]. Adhering to an industry standard is beneficial not only in terms of acceptability of the developed simulators on the part of people who know much more about their processes than about simulation (a problem worth addressing in any case, however) but also in terms of reduced ambiguity in the realisation of controller models [3, 4].

After several years of experience on the matter, the authors are strongly convinced that Modelica is very well suited as a host language for the representation of realistic-scale process controls, but that to do so it is highly desirable to allow for the specification of such controls in IEC-compliant languages.

Based on the above idea, the AutoEdit (the name may change in the future) project was started. The aim of the project is to set up a tool composed of

- a graphical Modelica editor, aimed at writing the "plant model",
- an editor for IEC 61131.3 languages (at present the Ladder Diagram, Sequential Functional Chart and the Functional Block Diagram are being considered), aimed at writing the "control model",
- a "compiler" capable of translating both the "plant" and "control" model in a single Modelica file, to be fed to any Modelica translator for simulation (the term "compiler" being used here for analogy and compatibility with the IEC terminology, albeit the Modelica jargon would most likely advise something like "pre-translator"),
- and a simulation output browser.

To the best of the authors' knowledge, such a tool is the only one allowing to couple Modelica process modelling with IEC (i.e., industry standard) control system representation, greatly facilitating the creation of simulators of process control systems.

AutoEdit is fully written in java (hence cross-platform), uses the XML language as internal data format for maximum openness and transparency, and is entirely free software, released under the terms of the GPL license. It is the authors' intention to allow AutoEdit to operate with any Modelica translator, so as to maximise its use and to have the maximum amount of feedback for improvement. At present, the AutoEdit site is hosted at the URL http://home.dei.polimi.it/donida/projects.php?project=AutoEdit

The paper organised as follows. First a minimal review of the background. Then, a discussion is carried out on the opportunity of generating event-driven Modelica code with an *ad hoc* tool, instead of describing control system components, as already attempted, with Modelica (continuous time based) models. The outcome of such discussion, as can be guessed, is that the "best" approach depends on the size of the considered application, direct generation of algorithmic code being preferable in the case of large (control) systems. The AutoEdit project is then described, illustrating its goals, structure, organisation, present state, and future developments.

2. Background

Recent advances in object-oriented modelling allow to tackle the simulation and the computer-aided control system design of industrial plants in a unified framework. Traditionally, however, the plant study and design, the following design assessment simulations, the control system design, the overall system validation, and the operator training, are not developed in a coordinate way within a single environment. By vastly acknowledged opinion, doing so is a waste of time and resources, not to say a possible source of errors, because the involved environments are frequently not compatible each other, requiring manual intervention to transfer information from one tool to another..

The Modelica multi-physics approach allows *per se* to perform a first integration of two of the involved frameworks: the plant model and its control are defined with an *equation* section for the plant and an *algorithm* section for the control code, and then the two sections are unified in a single model and simulated simultaneously.

In the present software engineering arena, translators and cross-compilers are well diffused,

but basically such tools are available for the software development only. To the best of the authors' knowledge there are no similar examples in *simulation for control* area, except for some *ad hoc* solutions pertaining to the micro-controller real-time applications.

The AutoEdit is an attempt to fill the gap sketched above. It is in the first place an integrated IEC61131.3 compliant environment for the graphical development of the control programs, having (algorithmic) Modelica as the target language. Moreover, it proposes new standard for the Ladder Diagram (LD) and Sequential Function Chart (SFC) file representations, using the XML language and DTD validation. AutoEdit also encompasses a converter from the SFC XML to LD XML format, managing different level of variables' scope, as required to be compatible with the way IEC-compliant projects are organised. In one word, AutoEdit is an attempt to allow developing the model of a complete control application (process and control system) in a single environment, and having as final output a complete simulator of the overall application.

3. Modelling control code in Modelica

Consider the way a control application is typically developed in an IEC-compliant environment. The application is composed of programs, written in one or more of the supported languages, and linked together by the development tool. The programs of an application are organised into sub-applications, that in turn are deployed to one or more CPUs and arranged into threads, each one composed of programs that share the cycle time. i.e., the temporal cadence for the update of inputs and outputs.

The goal of AutoEdit is to take as inputs

- a model of the plant written in standard Modelica
- and some description of the control application (the term “application” being intended in the IEC sense summarised above,

producing as output a single Modelica model, to be fed to any Modelica translator for subsequent simulation.

The question, then, is how to describe the control application.

Basically, one can follow two strategies. One is to describe the IEC languages' elements as Modelica models: this is somehow tempting especially if one considers the graphical IEC languages (FBD, LD, and SFC). Doing so allows to take profit from the manipulation capabilities of the adopted translator, to the apparent advantage of simulation efficiency.

The other strategy is to translate the IEC programs into Modelica algorithms, to be assembled conveniently in blocks, and connected to the plant model in the usual way.

AutoEdit takes the second way, for the reasons summarised in the following. First, especially large can easily lead symbolic manipulator to deal with thousands and thousands of variables: many of them are managed trivially, but the overhead remains. Then, many problems in IEC-specified control systems reside in the incorrect synchronisation of control threads and applications, and therefore – for a credible validation of the control system – representing that timing (e.g., and typically, with when clauses) is very important; if this is done, given the limitations of when-equations, describing the code as algorithms starts looking advisable. In addition, the organisation of the code in threads and sub-applications is typically functional, thus better reflected in algorithms than in equations.

Finally, and in some sense as a by-product, if a tool like AutoEdit generates algorithmic Modelica code starting from an IEC source, then the same tool can easily be extended to generate – from the same source – code in virtually any procedural programming languages. Exploiting that possibility is in the future plans of the AutoEdit project, and will lead to a single tool for the simulation of a complete system (avoiding the "how-to-close-the-loop" problem of IEC development environments) and also for the generation of the control code to be actually deployed to the system's CPU(s).

4. An example

A very simple example is now reported to better illustrate the ideas of section 3. In this example, a home irrigation plant is introduced. The plant has an accumulating tank, a pump, two level sensors, and three valves, each one connected to an irrigation line. A schematic figure of the plant is reported as figure 1.

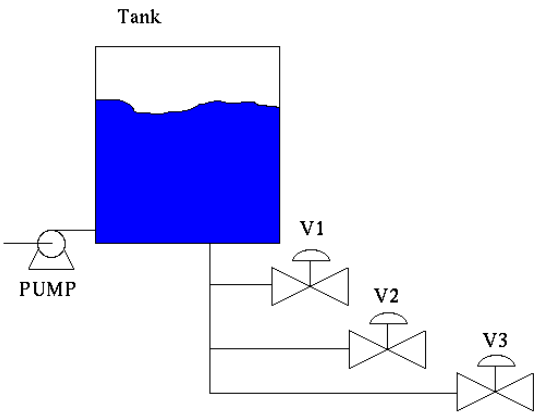

Figure 1: the example plant.

The pump starts pumping water in the tank when the level of the water is lesser than a OK_LEVEL (boolean sensor that returns true if covered by water) level since the water reaches the level FULL_LEVEL (similar boolean sensor). Everyday, say at 20:00 (event launched by a START_CYCLE variable, assumed here to be managed by some clock external to the program), each of the valves (V1 to V3) has to be opened. Each valve, one for each zone, remains opened for 10 minutes and then is closed. There is also a ON/OFF command: if ON is true then the plant works as described below, otherwise all the valves are opened, and the pump is stopped.

Figure 2 shows the overall control program, written in the SFC language, as is appears in the AutoEdit window. It is possible to recognise the various elements of the (very simple) control logic, and to appreciate the similarity of the user interface to that of the typical IEC-compliant environments (to the advantage of acceptability on the part of control system developers). We do not report simulations here since the plant and control operation in this example are very simple, and would not contribute to the purpose of this paper.

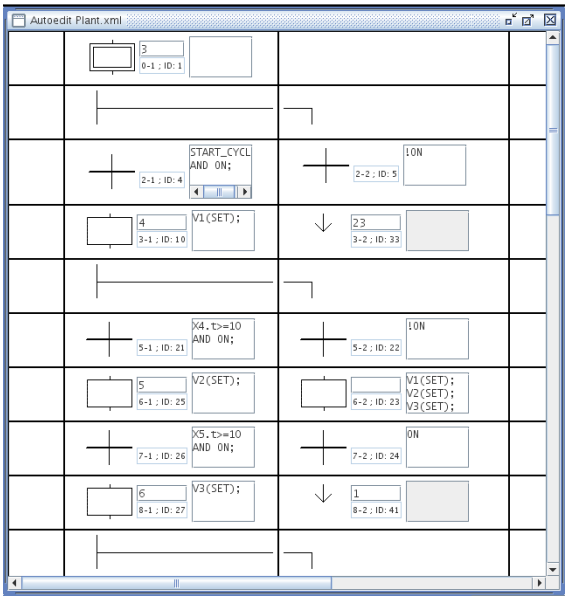

Figure 2: the example plant control in SFC.

In the example the translation was very simple but, when considering industrial applications of realistic size, the number and length of the lines

of control code would increase dramatically, and automatic generation of the algorithmic code would prove necessary. In addition to this if we consider the possibility to have heterogeneous IEC-compliant programs a mixture of ST, LD and SFC implementation, the complexity further increases.

Thanks to the AutoEdit conversion utility, it is possible to translate the SFC programs into LD and then, automatically, to Modelica algorithm-based models. The translation of an heterogeneous IEC control program is perfectly transparent to the AutoEdit user.

4. The AutoEdit project

The project started in the 2004 with the intent to realise a Java graphical application to support graphical programming for the LD, SFC and Structured Text (ST) languages.

From 2004 to 2006 a graphical application was therefore developed to graphically support the SFC and LD programming.

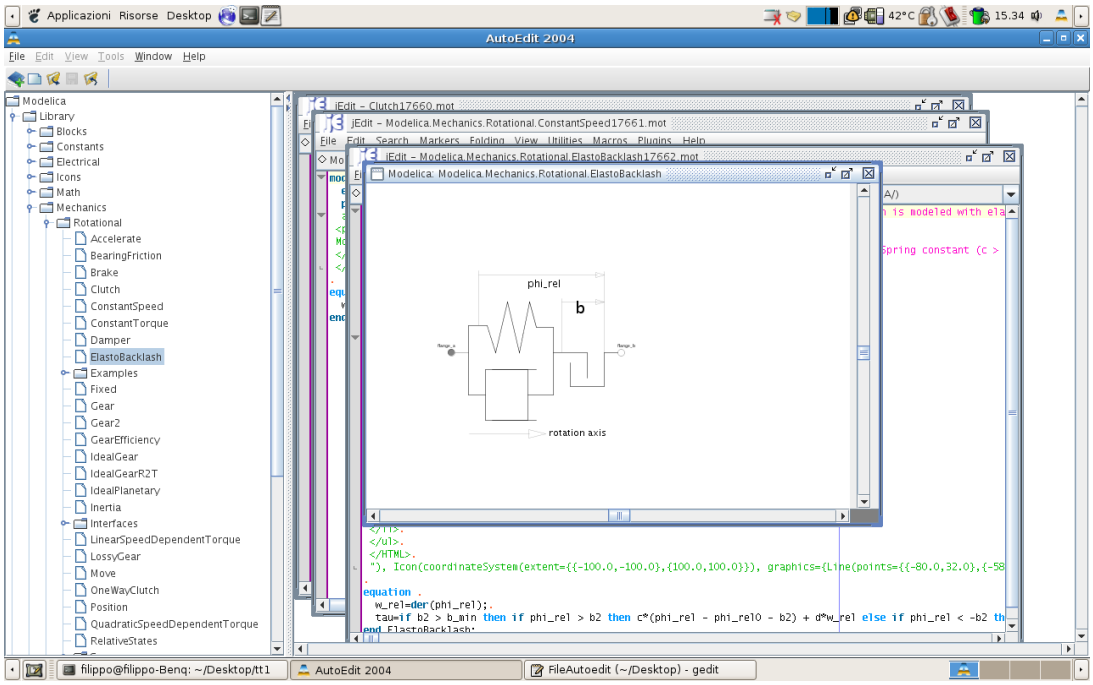

Figure 3: The AutoEdit main window.

Starting from 2006, the target was widened as illustrated in section 3, so as to integrate the AutoEdit environment with a Modelica editor, and then (starting in 2007) to create a converter from SFC, LD XML and Modelica algorithmic-based .mo files. This is – more or less – the present state of the project. Notice that a high development effort is being spent on AutoEdit, so that the mentioned state is continuously changing. The reader is referred to the project site for up-to-date information.

Here, just some samples of the AutoEdit operation are given. Space limitations prevent from reporting here any technical detail, that can anyway be figured out from the site, and will also be available in the system documentation.

Figure 3 shows the main window of AutoEdit with a Modelica model open for editing. It is

possible to see the multiple subwindows scheme, allowing simultaneous editing of multiple (process and/or control) models. The AutoEdit text editor, thanks to the integration of the JEdit software, offers many functionalities, among which syntax highlighting, bracket highlighting, text folding (also for annotations), word auto-completion, auto-indentation and many others utilities. of multiple models.

Figure 4, on the other hand, shows the conversion from SFC to LD, namely of the pump control program in the example introduced above. It is possible to appreciate the usefulness of having simultaneous views of the same code with different representations, a facility offered by several IEC-compliant environments, and of high usefulness according to the opinions of the industrial community.

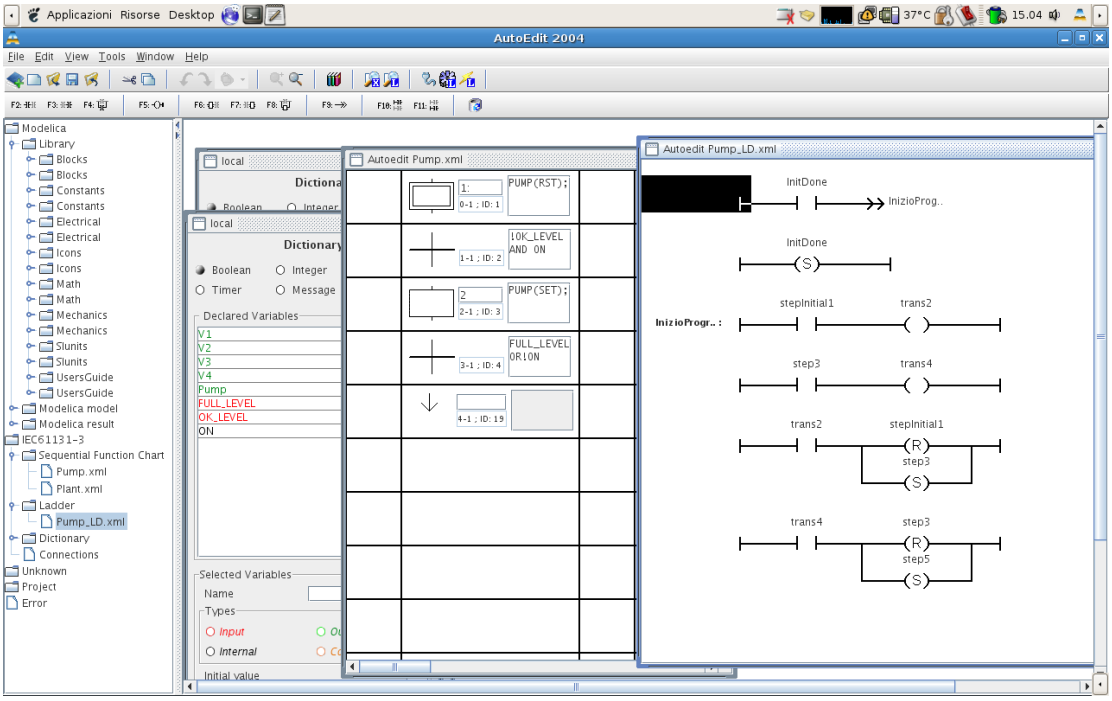

Figure 4: the pump control program converted from SFC to LD by AutoEdit.

5. Future developments

Many interesting "future works" arise for the AutoEdit project from the scenario synthetically described above. Among those possible developments, those that seem more promising, and are therefore scheduled as work to be done in the near future, are

- the development of a 3d viewer for the simulation data,
- the addition of other advanced editing functionalities,
- the exploitation of interaction/integration possibilities with other IEC-compliant tools,
- the output of *ad hoc* real-time code in several languages, the C languages being for obvious reasons the first to be considered,
- the addition of multitasking support.

6. Conclusions

A Java-based integrated environment for the

development of complete object-oriented simulation models of controlled plants, namely the AutoEdit project, was presented.

The goal of AutoEdit is to allow the user to create both the plant model, using the power of the Modelica language, and an algorithmic model of the control program, adhering to the IEC61131.3 industry standard,

As such, AutoEdit not only proposes a software solution, but also tries to suggest new standards and ideas for unifying two of the most important activities of the computer-aided engineering tasks: model and control co-simulation.

References

[1] T. Sato, E. Yoshida, Y. Kakebayashi, J. Asakura, N. Komoda, Application of IEC61131-3 For Semiconductor Processing Equipment, Emerging Technologies and Factory Automation. Proceedings. 2001 8th IEEE International Conference on, 2001.

[2] J. Huang, Y. Li, W. Luo, X. Liu, K. Nan, The Design of New-Type PLC based on IEC61131-3, Proceeding of the Second International Conference on Machine Learning and Cybernetics, Xi, 2-5, November 2003.

[3] A. Leva, A. M. Colombo, Method for optimising set-point weights in ISA-PID autotuners, IEE Proc-Control The09 Appl., Vol. 146, No. 2, March 1999 .

[4] H. Takada, H. Nakata, S. Horiike, A Reusable Object Model for Integrating Design Phases of Plant Systems Engineering, Proceedings of the Fourth International Conference on Computer and Information Technology (CIT'04).

[5] H. Taruishil, S. Kajiharal, J. Kawamotol, M. Ono, H. Ohtani, Development of Industrial Control Programming Environment Enhanced by Extensible Graphic Symbols, SICE-ICASE International Joint Conference 2006 in Bexco, Busan, Korea, Oct. 18-2 1, 2006.

[6] Y. Qiliang, X. Jianchun, W. Ping, Water Level Control of Boiler Drum Using One IEC61131-3-Based DCS, Proceedings of the 26th Chinese Control Conference, Zhangjiajie, Hunan, China, July 26-31, 2007.

[7] M. Bonfe', C. Fantuzzi, L. Poretti, PLC Object-oriented programming using IEC61131-3 norm languages: an application to manufacture machinery, in Proc. of IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics, vol. 2, pp. 787-792, 2001.

[8] [Online]. Available http://www.plcopen.org.

[9] J. Roger Folch, J. Pérez, M. Pineda, R. Puche, Graphical Development of Software for Programmable Logic Controllers, 12th International Power Electronics and Motion Control Conference.

[10] [Misc]. DeltaV: Monitor and control software.

[11] [Misc]. Labview: http://www.ni.com/labview.

[12] [Online]. Dymola: http://www.dynasim.se

[13] [Online]. Openmodelica: http://www.ida.liu.se/labs/pelab/modelica/OpenModelica.html

[14] A. Nobuo, I. Kenichi, Y. Eiji, Application portfolios for stardom, 12th International Power Electronics and Motion Control Conference.

[15] M. Otter, K. E. Årzén, I. Dressler, StateGraph-A Modelica Library for Hierarchical State Machines, 4th International Modelica Conference, March 7-8, 2005.

[16] O. Johansson, A. Pop, P. Fritzson, Engineering Design Tool Standards and Interfacing Possibilities to Modelica Simulation Tools, 5th International Modelica Conference, September 4-5, 2006.

[17] E. Tisserant, L. Bessard, M. de Sousa, An Open Source IEC 61131-3 Integrated Development Environment, Industrial Informatics, 5th IEEE International Conference on, 2007.

[18] [Online]. ISaGRAF: http://www.icpdas.com/products/PAC/i-8000/isagraf.htm