Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 49-3 (2016) 465–470

# Reliable Code Generation and Test Environment demonstrated on a PI-Controller Design

**Schwarz M.H.*., Sheng H.***, Üstoglu I. **, Chabaan W. *** & Börcsök J.*****

** Safety Computer Technology, University of Kassel, Germany.*
** Control and Automation Engineering Department, Yildiz Technical University, Istanbul, Turkey*
**** Computer Architecture and System Programming, University of Kassel, Germany.*

**Abstract:** This paper focuses on the latest version of a tool developed within the department to guide developer through the various steps of requirements, test-procedures and documentations to finally download the derived algorithm on a safety programmable logic controller (PLC). Requirements on how to develop and how to test software are steadily increasing due to international standards, which have to be obeyed in industries in general, and in the process- and automation industries in particular. A simple model is derived to be used to develop a PI controller that has to pass several steps to be qualified for execution on a PLC. The paper explains the different steps and describes the various problems that might occur.



*Keywords:* safety, reliability, automated code generation, PI control, international standards.

## 1. INTRODUCTION

Novel and advanced algorithms, particularly developed in academia, are often not adopted in industries, which does not indicate a lag of usability or complex procedures, but those algorithms are often either demonstrated in simulation or using self developed hardware with components of the shelf. In industries, programmable logic controllers (PLC) are the most used and widespread hardware systems. Those derived algorithms have either to be operated on such PLCs or are not of interest for industries. One way to overcome this problem is to re-develop the algorithms in a PLC conform language to be used on such systems. However, this procedure can be time consuming and fault-prone, then the original algorithms and the adopted algorithms have to be kept consistent. The other way is to use software interfaces, if they are applicable and made public to the user, to incorporate the developed algorithms into an industrial accepted environment.

One advantage of using programmable logic controllers as a target system is that the developer has not to consider safety and reliability issues, at least at the first glance. Safety, reliability and availability in the sense of international standards such as the *IEC 61508 (2010)* are important issues in industries and manufactures have to develop their systems in accordance to relevant safety standards. Such standards provide guidance on how to develop safe and reliable systems, this includes hardware development and software development and generally follows the development schematic shown in Figure 1. Industries have to put much effort into safety and reliability development and that is why it is obvious that the development of applications, which are the algorithms derived by academia, has also to follow stringent rules, in such a way that the overall reliability is not decreased due to malfunctioning or bad written code. The programming languages normally used in industries are defined in the international standard *IEC 61131-3 (2010)*.

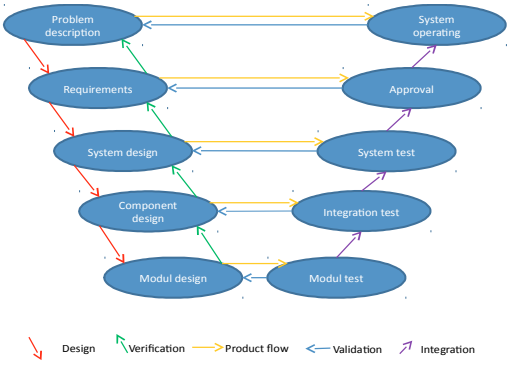


Fig 1: Development schematic

Programmes, algorithms and functions can be written in one of the five described languages, three are graphical oriented and two are textual languages. However, original algorithms for example developed in *Matlab®*, *Simulink®* or *LabView®* have to be re-written.

A very interesting publication by *Valencia-Palomo and Rossiter (2011a)* is a good example for using a conversion to apply research on a PLC. In this case a model predictive strategy was developed and converted to *Structured Text* to operate the development on an Allen Bradley–Rockwell Automation PLC. The authors argue that at the end the derived development has to be operated on a system to be trusted and used in industries. Additionally, the paper presents a robust, constraint but simple design of a *Model Predictive Controller (MPC)*, which outperforms a *Proportional-Integral-Derivative* (PID) Controller. In many cases, PID control is still the preferred strategy in use. However, MPC control was established in industries and was analysed in academia and possesses potential to outperform a PID controller.

Another paper by *Valencia-Palomo and Rossiter (2011b)* uses a combination of *Ladder logic* and *Structured Text* to

2405-8963 
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2016.07.078

implement their controller system and demonstrate the use of IEC 61131-3 languages in research. However, this example shows that a more reliable automated code converter would be beneficial and necessary.

But not only academia has its problems to get developed algorithms on reliable hardware, also industries increasingly realise the benefits of model-based design. Therefore, many additional analytical and verification functions and algorithms are provided by software environments such as *Matlab*® and *Simulink*® from *TheMathworks*® or *Labview*® from *National Instruments*® to be used in industries.

A very interesting example of modern model-based design is presented by *Snooke and Price C. (2012)*. They describe a model-based system that is functioning as a diagnostic model to determine whether the original process is working correctly or if faults occurred that might lead to a failure. The model has incorporated the FMEA (Fault Modes and Effect Analysis) and provides information about the actual behaviour of the system. This example shows a very advanced way to used model-based design.

The remaining paper is structured as follows: Section 2 will present a system that is going to be modelled and used as a process for system identification. Section 3 introduces a simple example to design a PID controller. Section 4 will briefly introduce the different IEC 6111-3 languages. In the following section this PI controller will be converted through several steps onto a PLC. The process-model will also be converted and has to pass all steps to be downloaded onto a different PLC. In section 6, both PLCs will interact with each other as process and controller and this will be compared with the original models derived and used in simulations. Conclusions are drawn and future work is presented in last section.

## 2. PROCESS MODEL AND IDENTIFICATION

In this section, a tank that can be filled with liquid should be modelled. The tank has one inlet ($I_I$) valve and can be described with Eq. 1. Additionally, the tank possesses an outlet valve ($I_O$) that can be described with Eq. 2. The volume can be calculated by calculating the difference between input and output and summed up, which is done in Eq.3. The tank itself is a simple model but can often be found in process industries.

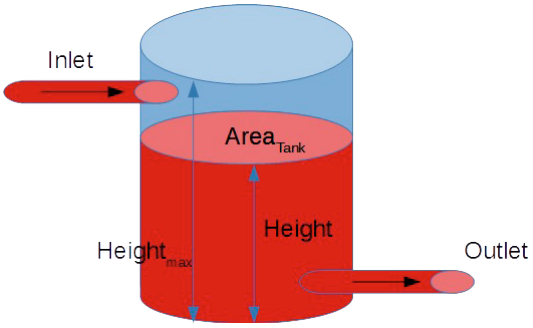


Fig 2: Schematic of a tank model

The inlet value can be defined as the volume flow:

$$I_I = \frac{Volume}{time} = \frac{V}{t} = \frac{A \cdot h}{t} = A_I \cdot v_I \tag{1}$$

The outlet value can be defined as the volume flow:

$$I_O = \frac{Volume}{time} = \frac{V}{t} = \frac{A \cdot h}{t} = A_O \cdot v_O \tag{2}$$

The current volume of the tank can be calculated as follows:

$$\int_{t_0}^{t_1} (I_1 - I_O)\, dt = \int_{h_0}^{h_1} A \cdot dh \tag{3}$$

The current height can be determined by:

$$H_{current} = \frac{\int_{t_0}^{t_1} (I_1 - I_O)\, dt}{A_{Tank}} \tag{4}$$

The current effluent velocity can be calculated using the Torricelli equation:

$$v_{current} = \sqrt{2 \cdot g \cdot h_{current}} \tag{5}$$

Using Eq. 2 and Eq. 5 results in:

$$I_O = \dot{V} = A_{valve} \cdot v_{current} = A_{valve} \cdot \sqrt{2 \cdot g \cdot h_{current}} \tag{6}$$

The outlet area is the diameter of the outlet valve and can be controlled from entirely open via a fraction of the diameter to fully closed.

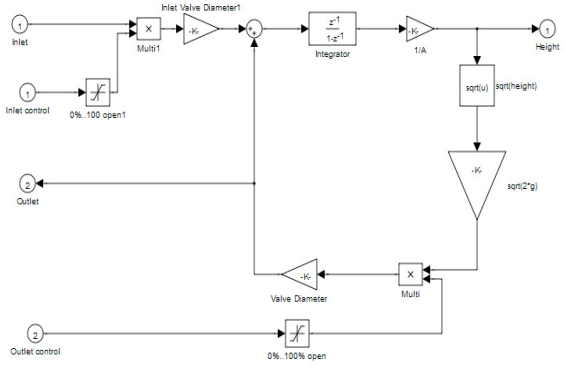


Fig 3: Simulink schematic

Figure 3 shows the model-based design of the tank with three inputs: the inlet volume, which can be controlled by the inlet valve control, the outlet control regulates the outlet valve from 0 % (closed) to 100% (fully open). The outputs are the current height of the liquid in the tank and the amount of liquid that leaves the tank. This model serves as a base to calculate the control parameters in the next section.