Solution Based on IEC 61499 for Standardized Representation of Components in a Real-time Library of Reusable Algorithms for Process Control

Oana Rohat, Luiza Ocheana, Dan Popescu

University "Politehnica" of Bucharest, Romania, (e-mail: oana.rohat@gmail.com, luiza.ocheana@yahoo.com, dan_popescu_2002@yahoo.com)

Abstract: This paper describes several solutions for a standardized, generic representation of the algorithms in a library of complex multi-functional process control algorithms. After presenting the library architecture, the paper describes the library design guide that will help the library administrators to manage the components that will be added to the library. Also we defined the performance criteria for algorithm representation and we presented a comparison of different standards based on these criteria so that we can find the best solution for the standardized representation of the components in the library. In the end we apply a transformation method from Simulink to the selected representation in two process examples. This method can be used for algorithm verification and validation purposes.

Keywords: standardized algorithm representation, open process control library, function blocks, IEC 61499, IEC 61131-3, Matlab Simulink.

1. INTRODUCTION

Currently the industry is based more and more on automated controllers, managing from simple functions like process monitoring and standard commands, to PID regulation and up to complex control applications based on artificial intelligence, predictive methods and automated system analysis.

In order to increase their competitiveness, process control engineers need to find solutions for improved efficiency and faster development times for control logic programming while keeping in mind the safety and reliability aspects. Their focus should be on reusability, the possibility of implementing the same control logic on similar applications, and on using a distributed control logic, that will allow decreasing the complexity by distributing the control functions on several PLCs. The first aspect is hard to be implemented because the control logic depends greatly on the PLC manufacturer. The last one is applied today (usually while still keeping a high complexity level) in most large plants, where a dedicated dispatcher is used for monitoring and/or controlling specific units.

Advanced control algorithms have been developed in order to respond to the new industrial requirements. When adding hardware and software components to the system infrastructure, disturbances may occur, and this is why it becomes necessary to optimize the existing control algorithms and to create new ones that are able to provide efficiency at reduced costs.

Developing an on-line library of advanced control and optimization algorithms for large – scale industrial plants, written in a standardized format based on IEC 61499, will provide the needed support for process control engineers to develop and implement complex solutions using modular, scalable, reusable, standardized functions.

Matlab Simulink and LabView are commercial tools widely used for process control that have implemented libraries of a large number of functions with focus on process modeling, simulation and optimization, in the first case, and on measurement problems in the second one. A widely used library was SLICOT (Benner et al., 1999) which provided numerical algorithms for high complexity computations in control theory and systems. It used BLAS (Dongarra et al., 1990) and LAPACK (Anderson et al., 1999) libraries with numerical linear algebra routines to provide methods for the design and analysis of control systems.

Since these are more oriented to academic research, also libraries and collaborative platforms that are oriented to the practical implementation of control functions were formed. ThinkCycle was a web-based industrial design project with the purpose of creating an open-source collaborative platform for engineers, designers, researchers and other professionals from a wide range of industrial areas. (Strasser et al., 2004; Wei, 2001) developed IEC61499 function block libraries for embedded closed loop control and IPMCS (Industrial measurement and control systems) and their results can be integrated as part of the open web library presented in this paper.

The most common way of algorithm representation in current process control applications is by using function blocks (FBs). They represent drag-and-drop blocks that can be linked to each other and execute a specific function or algorithm based on the input variables and provide the result as output variables. Simulink and LabView also use this kind of representation as a more intuitive way for process control programming and design. DCS (Distributed Control Systems) manufacturers (like Honeywell, Emerson, Yokogawa, etc.) have their own representation of FB that is dependent on the control programming application. Most PLC (Programmable Logic Controllers) manufacturers chose for representation the IEC61131 standard (IEC, 1993) that was created with the intention on providing code portability between different vendors, but this aspect could not be entirely materialized because of the hardware dependency. Still, commercial programming applications (like PLC-Prog) or Matlab PLC coder module provide the possibility of compiling an IEC 61131-3 based algorithm for different controllers. The IEC 61499 (Lewis, 2001) representation further develops the rules from the IEC 61131-3 standard by adding events to control the execution of the function blocks.

The main objective of the presented work is to identify the structure and best representation format for the library components keeping in mind the reusability and openness characteristics. The library must ensure the user that the algorithms provided are reliable and respect certain performance criteria. This is why we will focus on methods for the verification and validation of the algorithms that need to ensure they have the correct behavior in a real process operating environment.

The main aspects that need to be considered in the algorithm representation are:

- following a well-documented design guide;
- choosing the best implementation standard that fulfills the library needs;
- establishing performance criteria for the algorithms that will be added.

Other aspects regarding the library development like storage mechanism and process interfaces development were detailed by (Ocheana et al., 2012).

The rest of the paper is structured as follows. Section 2 presents a comparison between different common representation standards based on function blocks like: IEC 61499, Matlab Simulink, IEC61131-3 and IEC61850. These several representations are considered as possible solutions for algorithm representation in the library. To better understand the needs in terms of implemented functions, data representation, reusability and process integration, section 3 shows how the library will be integrated into an on-line application. Section 4 details the function block design guide, the performance criteria and the needed characteristics to represent function blocks so that the objective of providing an open source library with standardized, reusable algorithms for advanced control is met. In section 5 we implemented the simple algorithm of the PID function blocks in the different representations presented in section 2. Based on the advantages and disadvantages relative to the defined guide, we chose the best solution for the library components representation as being the IEC 61499 standard. Section 6 applies a verification and validation method for the selected IEC 61499 representation and exemplifies it on designing the controllers for two different processes: one standard PID controller for a second order system and a cascade controller for a heater tank. Conclusions and future work are presented in section 7.

2. COMPARISON BETWEEN DIFFERENT ALGORITHM IMPLEMENTATION STANDARDS

The library components must follow a standardized representation based on an open specification. Also, the main characteristics of the algorithm are the reusability and the possibility of deployment on a large number of industrial controllers or PLCs, independent of their manufacturer. The main components of the library will be provided as functions. This involves a clear definition of the capabilities of each component, as well as their interfaces, meaning the input and output parameters. In order to comply with these requests, we analyzed different representations of function blocks: IEC 61499, Matlab Simulink and IEC 61131-3. The main characteristics of each representation and also a comparison of their advantages and disadvantages are detailed in the following subsections.

2.1 The IEC 61499 Standard

The IEC 61499 standard is based on function blocks and has evolved from the IEC61131-3 standard (see section 2.3), widely used as a programming language by many PLC manufacturers (Lewis, 2001; Vyatkin, 2011). The standard is starting to win the interest of more and more researchers, process control engineers and manufacturers because of the important benefits it brings: the possibility of implementing fully distributed applications, the modularity and reusability of the function blocks, reconfiguration, the hardware platform independency etc.

As presented in (FBDK, 2008) IEC 61499 function blocks are made of 2 areas, one for execution flow control, and one for data flow control which also runs the function block algorithm (see Fig. 1).

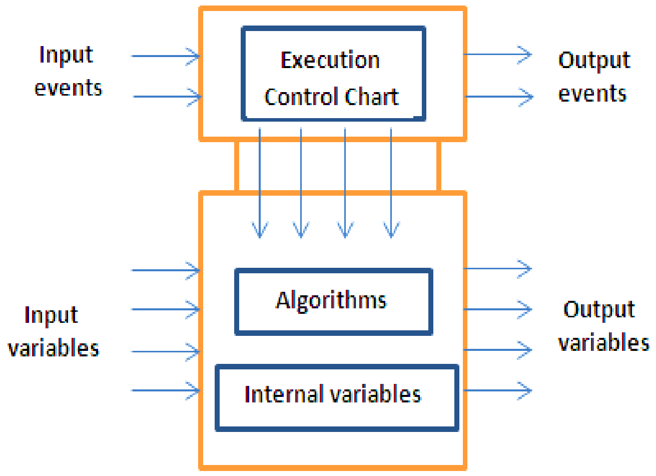

Fig. 1. IEC 61499 function block.

The execution sequence is as follows: when an event arrives, the execution control function is activated and notifies the resource scheduling function. This starts the algorithm execution that uses the input data flow to compute the values for the output data flow. The resource scheduling function is notified when the algorithm ends and it activates the execution control function that generates the output events.

According to the IEC 61499 standard, function blocks can be basic (when algorithm equations are defined inside the FB, as described before), service interface (a type of basic FB used for developing graphical user interfaces GUI elements, communication services, or interfaces to hardware components) or composite (when the FB is made of several basic FBs chained at both event and data levels).

Keeping in mind the representation requests of the library components, this representation provides the following advantages:

- it is based on an open standard which makes it possible for unified algorithm development for all users that want to contribute to the library development;
- it is hardware independent which results in the possibility of implementing the developed functions on any PLC supporting this standard, no matter the manufacturer;
- reusability and reconfigurability are easily implemented by the standard specifications;
- it allows the process engineers to divide the controlled object into several less complex subsystems that are more easily monitored. Each of these subsystems can therefore be treated separately and the control algorithms employed are more generic and can be reconfigured to work with similar dynamic performances for any of the subsystems;
- the standard's modular structure contribute significantly to the generic nature of these algorithms in the sense that the same algorithm can be used in different system components with a minor input/output and parameter reconfiguration effort;
- it simplifies the design and understanding of complex control systems by including encapsulation (each function block is presented as on input/output box, thus hiding its internal algorithms) and hierarchy (composite function blocks can be developed to accomplish specific functions, thus offering the possibility of using a top-down decomposition approach);
- it allows the definition of specific communication and process interface function blocks;
- free development and runtime environments are available for function block design and testing ((FBDK, 2008); FBench, 4DIAC etc.);
- the researchers' and manufacturers' grown interest in the standard development from the last years;
- allows defining the FB execution order since the functions are event-triggered and are executed based on the chain of events. This allows designing more efficient, both synchronous and asynchronous applications.

On the other hand, there are some disadvantages of the IEC 61499 standard, like:

- the free development and runtime environments are not mature and have some issues that may make some algorithms to block at execution. This can be overcome by a rigorous design of the algorithms or by using a commercial tool like ISaGRAF;
- the verification and validation of the function block applications can be difficult. One solution is presented in section 6;
- possible un-deterministic behavior because of poor standard information regarding the management of multiple input events that may come in a short period of time;
- lack of devices from which can run this standard (including some from manufacturers like Allen Bradley, Centris Technologies, Motorola, TCS, iMonitor etc.) resulting in a low industrial adoption;
- difficult to commission and maintain because of the primitive development tools. This can be overcome by using a commercial tool like ISaGRAF. Still we must keep in mind that there are differences in FB interpretation between the free tools and ISaGRAF.

2.2 *Matlab functions*

Matlab is a tool for mathematical computation, with a large number of implemented functions for numerical analysis, process control, optimization etc. It is widely used especially for modeling, simulation and analysis of process control systems. The Simulink tool from Matlab allows implementing block diagram applications based on function blocks with the generic representation presented in Fig. 2.

Also, Matlab has a specific component called "Simulink PLC coder" that allows converting algorithms to a function block representation based on IEC 61311-3.

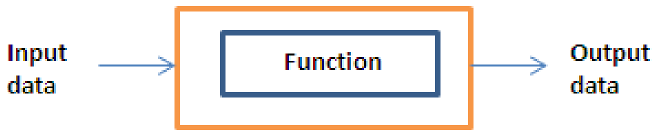

Fig. 2. Simulink block.

Compared to IEC 61499, the Simulink representation provides the following advantages:

- strong computational power and a large number of implemented functions;
- possibility of testing and validation of the functions by using process simulation blocks;
- generation of IEC 61131 code (using the PLC coder module) compatible with the integrated development environment (IDE) used by most important PLC manufacturers (including B&R Automation Studio, PLCopen XML, Rockwell Automation RSlogix 5000, Siemens SIMATIC STEP 7, and 3S-Smart Software Solutions CoDeSys).

Still, there are important disadvantages that sustain that this representation can't be used in the algorithm library:

- it is not open;
- the reusability features are not strongly sustained by the block definition as any modification of a block needs to be done in its every occurrence;

- developing distributed applications can be done only between communication computers, using an additional Matlab tool;
- it requires greater computational resources;
- it is oriented more on research use rather than industrial use which results in little or no interest in the development of process interfaces and communication objects;
- cyclic execution of all the functions.

A transformation model between Simulink and IEC 61499 was presented in (Yang and Vyatkin, 2010; Yang and Vyatkin, 2012). As was presented in (Yang and Vyatkin, 2012) the advantages from the Simulink representation can be used in the IEC 61499 representation by bridging the two function models using a transformation method. This provides an easy verification and validation method for the applications developed based on the IEC 61499 standard.

2.3 The IEC 61131-3 standard

The IEC 61131-3 standard was the first attempt to create a unified programming language for all PLCs. Even if it was designed suggesting the portability of code, this was never materialized due to its hardware dependency. Like IEC 61499, the representation mode is based on function blocks but without the possibility of controlling the event flow. The main PLC manufacturers (like Allen Bradley, Siemens, HIMA, GeFanuc, Mitsubishi etc.) use for logic programming software tools that are compatible with the IEC 61131 standard.

The representation of the IEC 61131-3 function block is shown in Fig. 3. A comparison between IEC 61131-3 and IEC 61499 was done in (Dai and Vyatkin, 2009; Bezak, 2012). The studies conducted by (Gerber et al., 2008; Dai and Vyatkin, 2009) also present the steps needed for the migration from IEC 61131-3 to IEC61499, empathizing the limitations of such actions.

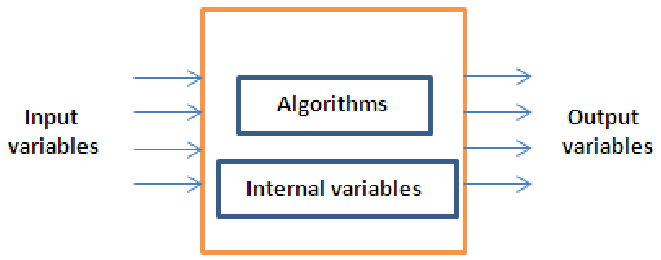

Fig. 3. IEC 61131-3 function block.

The IEC 61311-3 representation has many advantages that made it the most used PLC programming method (Lewis, 1995). They were kept also in the IEC 61499 representation. In addition, the following advantages are specific for the IEC 61311-3 representation:

- widely accepted by PLC manufacturers;
- there are many robust tools that allow programming based on this standard. As a result many applications, especially in discrete domains like a manufacturing, are programmed using on this standard.

As this is the IEC 61499 standard's ungraded version it is natural that it has important disadvantages over it:

- it doesn't support the development of distributed applications;
- the code reusability is limited (or not possible at all) because the IDE for application development used by the PLC manufacturers use different syntax or semantics of the certain programming structures (Dai and Vyatkin, 2009);
- the application is executed according to the cycle time depending on the control hardware. This may lead to different response times on different controllers;
- when developing an application, the function blocks have to be arranged in the sequential order of execution.

The comparison presented in this section empathizes the advantages of the IEC 61499 standard in terms of openness, reusability and implementation of distributed applications over the other two commonly used representations, IEC 61131 and Simulink. Also, the current increasing interest in the standard development and in its rapid industrial integration presented it as the best candidate for the library algorithm representation.

3. INTEGRATION OF THE LIBRARY INTO ONLINE APPLICATIONS

The library will include a large variety of mathematical functions and process control algorithms starting with simple ones (like the PID algorithm, functions for pump/motor start/stop etc.) and evolve towards more complex ones (like signal processing, system identification, strategies for hazard and risk management, fault analysis and assessment, etc.). In order to ensure the correctness and reusability of the functions provided, we designed a complex architecture able to provide support for all stages that are needed. The proposed application integration architecture is presented in Fig. 4.

The architecture involves two main components: a virtual platform for algorithms testing and validation and an on-line support of application that provides open access for the library components and also allows adding implementation results and process feedback.

The library will allow users to view and download different algorithms, to add new ones after they were tested and validated by the registered users, or to add process feedback that will help improve the algorithms and optimize them for specific processes. The algorithms will be written based on the IEC 61499 standard and will have associated descriptions that will give the user information about their functionality, the process where they can be applied and feedback from other users regarding their performance after the implementation in an operating plant. In order to allow reusability with no or little configuration changes, the algorithms will be provided as files containing the definition of the function block. This function block will be represented based on a guide and an open standard so that it can be included in the existing control logic of the plant. The access to the library will be possible using an online open application that will connect to a database for solving the user requests. Based on its functionality and the resources needed for its execution, this function block can either be integrated in the control logic of the process (offline integration) by use of specific process interfaces (PI) or can be run remotely in the library (online execution). If the FB is executed online, the communication to the process will be done through an OPC communication interface at the function block level that will send the required information to the process interface by the use of the Internet. The communication speed will depend in the selected operation mode, the algorithm complexity and the controller resources.

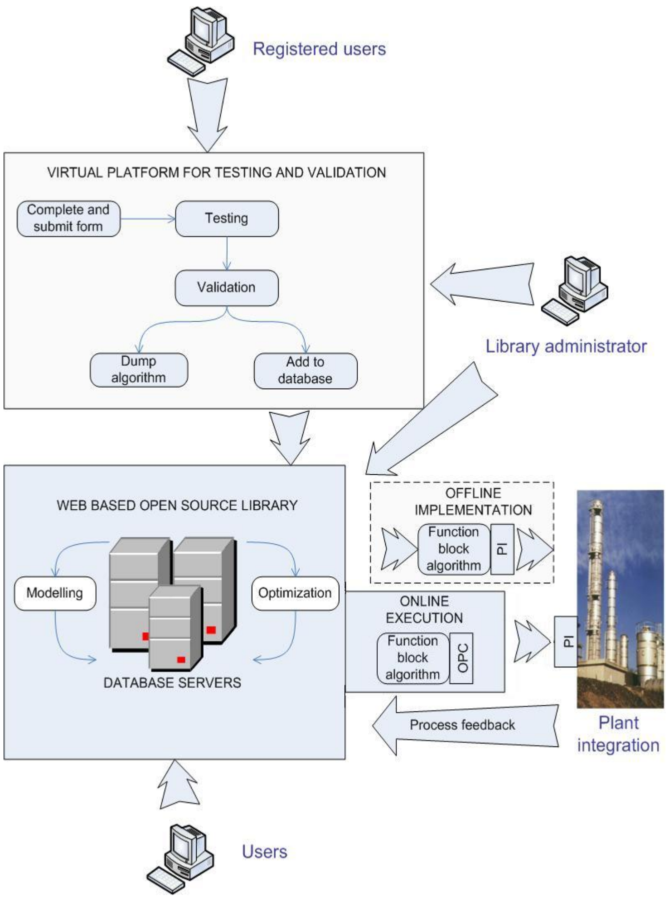

Fig. 4. Library integration into an online structure application

The library will also allow *registered users* to accept or discharge the uploaded control algorithms based on how they conform to the defined guide. These registered users will also have access to the virtual platform component of the library, consisting in a process simulation application, process modeling, optimization and system testing and validation tools. This component, along with the algorithm description of the implemented functions and the library guide will help registered users to decide whether an algorithm can be added to the library or not.

The library administrator will be the “bookkeeper” that will manage the database, taking care of both the structural aspects (database organization and search engine, structure, users etc.) and also the content aspects.

The algorithms can be downloaded directly on a process controller, or can be run on the web library. For this, process interface function blocks will be developed. This will allow accessing algorithms results through common communication standards like OPC, Modbus, Profibus, etc.

4. LIBRARY DESIGN GUIDE

Because of the great importance of the correct and efficient management of all possible use cases we must ensure that the algorithms added to the library are extremely reliable and efficient. For this it is necessary to consider the recent progress in the field of process automation from both theoretical and practical view, of computer science and of mathematical analysis.

The algorithm library must conform to specific documenting and implementation standards in order to ensure a uniform interface with the user, easy maintenance and the adaptability and portability needed for the execution on several computing platforms or applications. Such standards were defined for the libraries of international use BLAS (Dongarra et al., 1990), LAPACK (Anderson et al., 1999) and also for the automation library SLICOT (Benner et al., 1999).

The usability of the library will be enhanced if the user could run their components in a widely accepted software programs like MATLAB, Mathematica, Python, or Octave for the advanced theoretical applications or in a widely used automation standard like IEC 61131 (IEC, 1993) or IEC 61499 (Lewis, 2001) for the practical implementation. Such interfaces would allow increased flexibility and would simplify experimenting and combining different components.

The library will include at least two types of components: main components, fully documented, that involve a complex logic and can be used as they are in the process control application, and auxiliary components, which are used by the main components, involve less documentation and do not solve a process control problem on their own.

The library architecture presented in Fig. 5 is organized in sections, based on the problem solved and the type of industry on which they apply. Each section may have several subsections.

The sections for the algorithm type are:

- basic functions (refers to basic functions that are not yet implemented in FBDK or other free development environments and are needed in other algorithms);
- process interface (probably developed by manufacturers);
- communication interface (OPC, Modbus, Profibus etc.);
- control (PID, RST, cascade controller, bipositional controller, tripositional controller etc.);
- control sequence (for example an Enercon controller for wind turbine or a Schneider frequency converter need a certain sequence for sending set-points);

- safety (fault detection, fault accommodation, risk management etc.);
- modeling and optimization (Particle Swarm Optimization – PSO, Model Predictive Control - MPC, Genetic algorithms etc.).

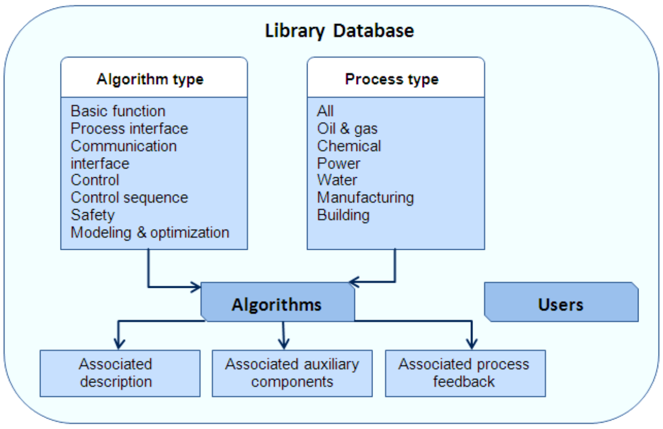

Fig. 5. Library architecture.

An example of the process type structure is:

- universal (sau general) (for example basic functions, process or communication interfaces do not depend on the type of process where they are used);
- oil and gas (refinery, warehouse, oil/gas loading and unloading terminal, etc.);
- chemical;
- power (wind power plant, photovoltaic park, nuclear plant, etc.);
- water and wastewater;
- manufacturing;
- building.

The library should not include an unnecessary large number of components. In order to implement similar tasks, often only one component can be used, with one or more functioning modes which can be set up by external parameters.

The library components that were identified in section 3 must be designed keeping in mind the correlation and compatibility between them in order to ensure the consistency of the results.

The main criteria for adding an algorithm to the library are:

- utility: to solve a problem of practical need;
- robustness: to provide either the correct results or an error or warning flag if the algorithm is used incorrectly (the problem is not well conditioned or does not apply to the class of problems for which that algorithm was designed);
- numerical stability and precision: to provide precise results as expected in the mathematical representation;
- execution speed: to be as high as possible, while keeping in mind the robustness, numerical stability and precision aspects.

In order to organize the data and algorithms in the library we need to define a unified strategy and rules for algorithm representation. These rules must ensure the performance criteria needed for the components of the library in order to be possible to accomplish the main objective. Considering the above mentioned we can say that the main objective of the paper is to establish the functionality of an open library of reusable algorithms for the advanced control of industrial processes. The following are the rules for the algorithm representation:

- reusability of the developed algorithms with minimum reconfiguration effort;
- open standard of representation;
- portability between different software development tools based on the selected representation;
- hardware independency (the same algorithm can be used on controllers from different manufacturers);
- adaptability to the process time.

The correspondence of the considered implementation standards to these criteria will help us find the most suitable solution for algorithm representation.

5. REPRESENTATION OF THE REUSABLE CONTROL ALGORITHMS IN THE LIBRARY

Due to the mentioned advantages, the algorithms in the library will be represented according to the IEC 61499 standard, based on the comparison presented in section 2. The exemplification of this model of representation will be made using a simulated discrete process having a PID regulatory. The process has the following transfer function:

$$H(z) = \frac{0.01316\,z^2 + 0.02632\,z + 0.01316}{z^2 + 0.1053\,z - 0.05263} \qquad (1)$$

The positional PID controller for discrete processes implements the following algorithm:

$$Y = Y_0 + K_p * E_n + K_p T_S * K_I * \sum_{i=1}^{n} E_i + \\ + K_P * \frac{T_D}{T_S} * (E_N - E_{N-1}) \qquad (2)$$

where K_P, K_I and T_D are respectively the proportional, integral and derivative parameters of the PID controller, T_S is the sampling period and E_k is the error between process value and set-point at different earlier algorithm execution moments.

The closed loop control of the process is illustrated in Fig. 6. The PID algorithm has as input the difference between the desired set-point and the process value. The output from the PID block controls the execution element (the exit valve in our example). The response of the process after this command is measured and compared to the set-point to analyze if the desired state was reached. When there is no error, the PID algorithm stops its command over the execution element and the process goes into a steady state. In

Fig. 6 we empathized how the PID algorithm will be integrated in the process through an interface. When the PID is developed according to the IEC 61499 standard, this interface is a SIFB (Service Interface Function Block).

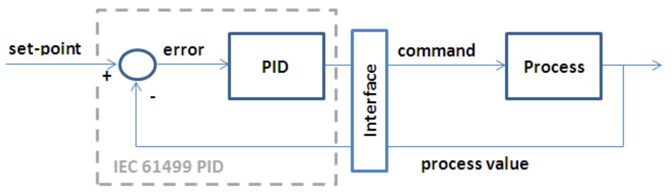

Fig. 6. Closed loop control.

For IEC 61499 representation of the PID algorithm we used a model presented in (Lewis, 2001) and implemented it in the free FBDK (Function Block Development Kit) environment (FBDK, 2008).

The algorithm is developed as a composite function block with the structure presented in Fig. 7 and Fig. 8. The component basic function blocks are PID_CALC, DERIVATIVE_REAL and INTEGRAL_REAL. The INIT event is used to initialize the internal variables of the corresponding function block. The PID algorithm is run when the RUN event is activated. In auto mode (MODE = 1) the PID_CALC FB computes the ERROR variable based on the PV and SP values and passes the result to DERIVATIVE_REAL and INTEGRAL_REAL. These blocks are executed when the EX event is activated. After that these two FBs send the values for the computed derivative and integral components of the PID algorithm and set the POST event in FB_CALC. Here, based on the KP, KI and TD parameters it calculates one step of the PID algorithm and outputs the value of the command through the XOUT variable. In manual mode (MODE = 0) the output is forwarded as the value of the MANOUT input. The CYCLE variable sets the sampling period for the algorithm execution.

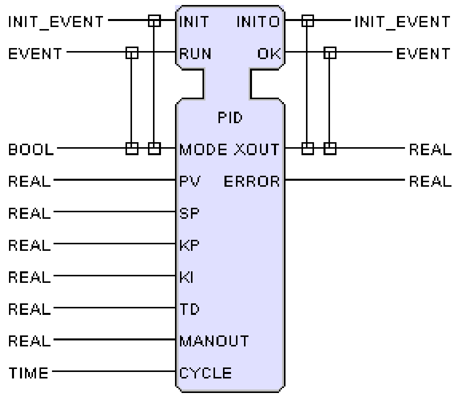

Fig. 7. PID composite function block.

In this representation, the running of the algorithm can be easily adapted to the process time by the use of events. The RUN event can be set to either a cyclic execution signal or triggered by the change of the PV or SP values. This provides great flexibility and helps improve controller's execution performance. The reconfiguration of the algorithm can be done by modifying the basic function block network.

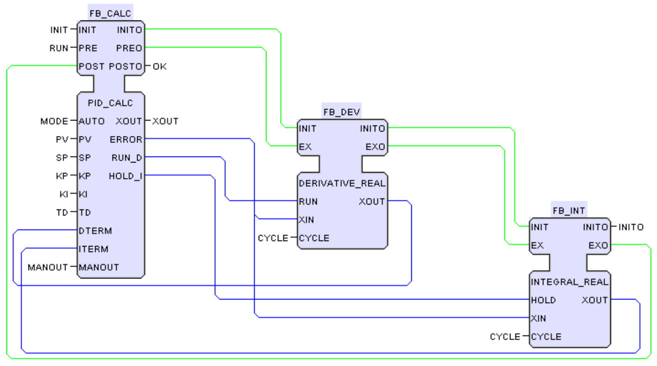

Fig. 8. PID Function block specification.

For comparison, we also represented the algorithm in Simulink using the existing PID function block. The Simulink representation is presented in Fig. 9.

The block allows setting the parameters for *Kp*, *Ki*, *Kd*. The block input is the error between set-point and process value. The output is the control signal for the valve. The block is executed at each sample time.

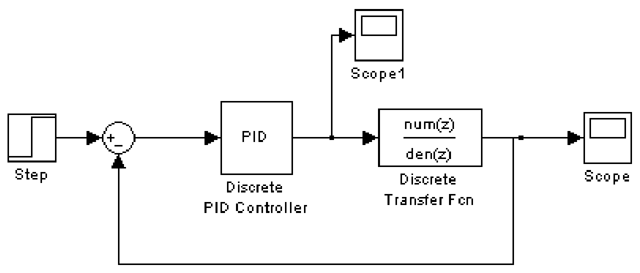

Fig. 9. PID algorithm in Simulink.

The main disadvantage there is no access to the structure of the algorithm, so no reconfiguration can be done in this form. Still, one can build a new PID algorithm using de blocks provided by Simulink (including derivative, integral and transfer function blocks).

Similarly, we represented the PID algorithm based on the IEC 61131 standard on a HIMA H51q controller using the ELOP II programming environment. ELOP II has a function block with the PID algorithm already implemented, as can be seen in Fig. 10.

This representation follows the discrete PID equation (1). The process value (Controlled variable), set-point (Reference variable), PID Structure (P, PI, PD or PID) and parameters, sampling period, processing pulse and the option for a correcting parameter (Y Value) must be set.

Even if the function block is accessed at each controller time cycle, the algorithm is only executed when the processing pulse is set, thus giving functionality similar to the IEC 61499 representation from the control of execution point of view.

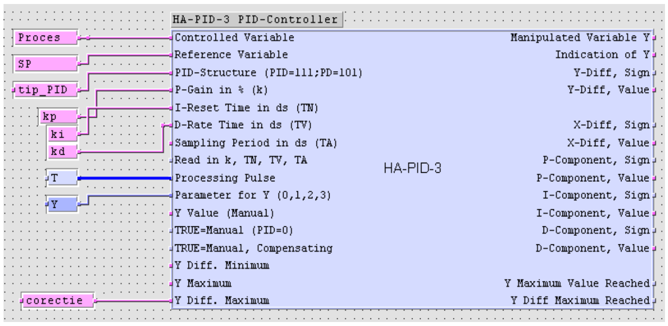

Fig. 10. IEC 61131 PID algorithm in ELOP II.

The block implemented in Elop II gives some options for controller reconfiguration, but this aspect may differ based on the development environment. For developing new algorithms that involve complex calculations little support is provided based on the limited library of existing function blocks.

After representing the PID algorithm in different formats, we concluded that from the development point of view, the IEC61131 and Simulink representations were easier to implement because of the availability of an already defined PID function block and of the use of an user-friendly environment compared to the rudimentary FBDK environment. Still, the Simulink PID block does not allow the user to reconfigure its parameters in run-time. From the reusability point of view, the blocks developed using Elop II does not allow exporting developed function blocks outside of the project application. The blocks developed in Simulink can be exported to other Simulink applications, with the limitations that come if using different Matlab versions. The IEC 61499 representation was more difficult to implement because of the lack of a process control library in FBDK. This aspect will be solved while developing the library, as both common and complex control algorithms will be represented as function blocks. This also adds the possibility of adding functionalities like dynamic reconfiguration or distributed execution to the algorithm. From the reusability point of view, the function blocks developed in FBDK can be exported to any other IEC 61499 compliant application. All these, added to the advantages mentioned in section 4.1. motivated us to choose IEC 61499 as the algorithm representation in the library.

6. VERIFICATION AND VALIDATION OF THE CLOSED LOOP APPLICATON

Verification and validation of the closed loop application refers to ensuring that the process response follows the system requirements. This implies modeling both the process and the controller and analyzing the closed loop response at a specific simulated input. While doing this in Simulink is an easy task, IEC 61131 and IEC 61499 require a greater effort. This is because most of the programming environments provide no support for system modeling. A solution for minimizing the effort of the verification and validation stages in IEC 61131 and IEC 61499 applications was presented in (Yang and Vyatkin, 2012; Yang and Vyatkin, 2010). This solution recommends the modeling and simulation of the process using Matlab Simulink and then applying a transformation method from the Simulink model to IEC 61499. A similar and even easier transformation method can be imagined also for the IEC 61131 standard and also specific modules like PLC Coder from Matlab or PLC Link can be used to automatically transform Simulink models into IEC 61311-3 function block networks.

We applied this method by creating a Simulink model of the PID controller similar to the IEC 61499 controller presented in section 5. We added the transfer function of the process according to (1) and analyzed the closed loop response.

Considering a sampling period of 0.05s we obtained the results presented in Fig. 1 for the following values of the PID parameters:

$K_P = 350$

$K_I = 300$

$T_D = 50$

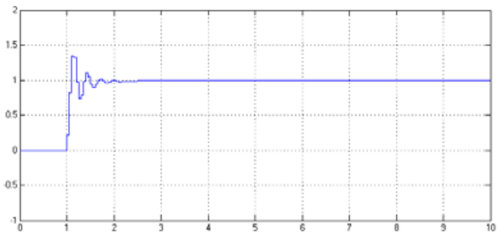

Fig. 11. Closed loop process response.

As this closed loop response satisfies the process performance requirements, we can say the PID controller was validated and we can also use the IEC61499 PID representation for process implementation.

Fig. presents the controller in its Simulink and IEC 61499 representation, and the correspondence between blocks. By applying the transformation, there is one IEC 61499 basic function block for each Simulink block. Every Simulink input and output parameter has a corresponding IEC 61499 input or output parameter. The State-flow charts implement the IEC 61499 ECC functionality, allowing block execution in specific conditions.

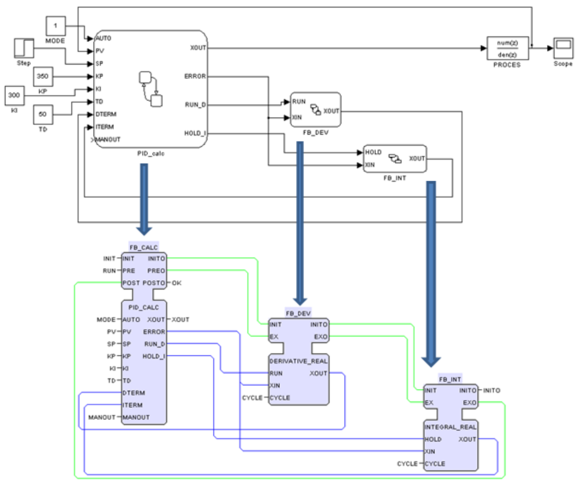

Fig. 12. Applying the transformation method between Simulink and IEC 61499.

A more complex control algorithm was implemented for the control of the level in a heating tank, by modifying the water input flow. For this, we chose a cascade control using two PID regulators. The outer loop will set the reference for the inner loop according to the heating tank level. The inner loop will control the valve for the tank input flow considering also the steam flow perturbations. The functional representation of the system is presented in Fig. . The red square represents the cascade controller. The blue square represents the inner control loop without perturbations.

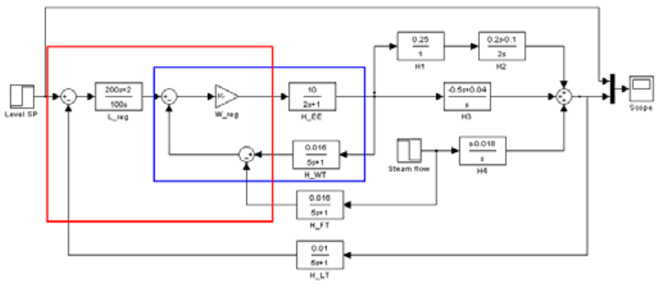

Fig. 13. Functional representation of the heating tank closed loop system.

Considering the inner loop system without perturbations we selected a P controller for the flow control. By setting up the system requirements in terms of performance and stability, we identified this system with a second order process and obtained the following value for the K_p parameter:

$K_P = 1.56$

For the outer loop we selected a PI controller. We computed the system transfer function in the absence of perturbations and obtained the following parameter values:

$K_P = 2$

$K_I = 200$

As this is a continuous system, we modeled the equivalent discrete controller and compared the result. The different closed loop responses are presented in Fig. .

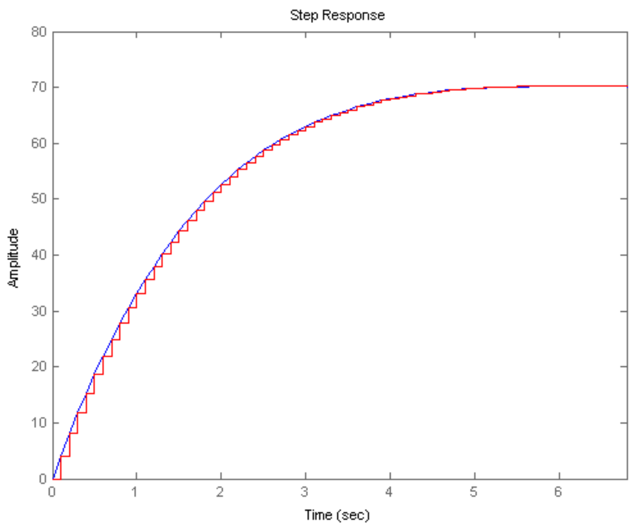

Fig. 14. Heating tank closed loop response.

After the control rules were established in Simulink, we built the IEC 61499 model by applying the same transformation method described before. The resulting function block system can be seen in Fig.

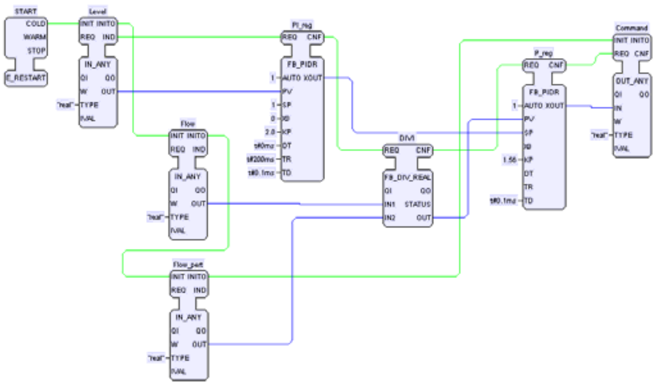

Fig. 15. IEC 61499 representation of the heating tank controller.

In the Simulink model, the execution is started by the step block that sets the level set-point. The order of execution depends on the linking of the blocks and is dictated by the Simulink solver. It can be displayed on each block. Each block is executed at each program cycle. This order of execution was considered when developing the IEC 61499 model. The IEC61499 application execution is launched by initiating the input and output blocks using the INIT event. A change in the level input activates the IND event. This executes the PI block. The end of the PI block execution provides a set-point for P and launches the DIV block. It computes the process value and activates the execution of the P controller. As can be seen, the IEC 61499 standard provides more efficiency and reduced total execution times as only when changes occur blocks are processed.

6. CONCLUSIONS

This paper presents the solution for the selection and representation of the components of an open library for process control algorithms. The comparison between the different representation standards empathized a series of advantages and disadvantages, relative to the established performance criteria. We concluded that IEC 61499 standard is the best choice for representing the algorithms. Starting from these advantages we proposed a design and a functional architecture for the library and a solution on how that library can be integrated in a real-time application. We showed how Simulink can be used for the verification and validation of the algorithms included in the library by applying a transformation method on the PID controller for a second order system and on the cascade controller of a complex system.

Future work will include the validation of the proposed solution on an industrial scenario, developing complex algorithms in the IEC 61499 standard, defining the library structure and use-cases so that in the end we can develop and manage a functional on-line open library for process control.

REFERENCES

Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Ostrouchov, S., and Sorensen, D.. (1999). *LAPACK Users' Guide: Third Edition*. Society for Industrial and Applied Mathematics.

Benner, P., Mehrmann, V., Sima, V., Van Huffel, S. and Varga, A. (1999). SLICOT - A subroutine library in systems and control theory. *Applied and Computational Control, Signals, and Circuits*, vol. 1, p. 499 – 539.

Bezák, T. (2012). Usage of IEC 61131 and IEC 61499 Standards for Creating Distributed Control Systems. Publ. Univ. Verlag.

Dai, W. W., and Vyatkin, V. (2009). A Case Study on Migration from IEC 61131 PLC to IEC 61499 Function Block Control. *7th IEEE International Conference on Industrial Informatics – INDIN 2009*, p. 79 - 84.

Dongarra, J., Du Croz, J., Duff, I. S., and Hammarling, S. (1990). Algorithm 679: A set of Level 3 Basic Linear Algebra Subprograms. *ACM Trans. Math. Softw*.

FBDK. (2008). Holobloc Inc. Resources for the New Generation of Automation and Control. Function Block Development Kit (FBDK), www.holobloc.com.

Gerber, C., Hanisch, H. M., and Ebbinghaus. S. (2008_. From IEC 61131 to IEC 61499 for Distributed Systems: A Case Study. *EURASIP Journal on Embedded Systems*, p.1-8.

IEC International Electrotechnical Commission. (1993) *Programmable Controller - Part 3: Programming Languages*, IEC 61131-3 Standard. International Electrotechnical Commission, Geneva.

Lewis, R. W. (1995). *Programming industrial control systems using IEC 61131-3.* IEEE Control Engineering, The Institution of Electrical Engineers.

Lewis, R. W. (2001). *Modelling control systems using IEC 61499: applying function blocks to distributed systems*. IEE, U.K. – Control Engineering Series 59.

Ocheana, L., Rohat, O., Popescu, D., and Florea, G. (2012). Library of Reusable Algorithms for Internet - Based Diagnose and Control System. *14th IFAC Symposium on information Control Problems in Manufacturing*, vol. 14, part 1.

Strasser, T., Auinger, F., and Zoitl, A. (2004). Development, implementation and use of an IEC 61499 function block library for embedded closed loop control. *2nd IEEE International Conference on Industrial Informatics INDIN 2004*, p.594 – 599.

Vyatkin, V. (2011). IEC 61499 as Enabler of Distributed and Intelligent Automation: State of the Art Review. *IEEE Transactions On Industrial Informatics*, vol.7, issue 4, p. 768 – 781.

Wei, Y. (2001). Implementation of IEC61499 Distributed Function Block Architecture for Industrial Measurement and Control Systems (IPMCS). *Total Enterprise Solutions Conference, ICAM.*

Yang, C. and Vyatkin, V. (2010). Model Transformation between MATLAB Simulink and Function Blocks. *8th IEEE International Conference on Industrial Informatics INDIN 2010*, p.1130-1135.

Yang, C., and Vyatkin, V. (2012). Transformation of Simulink models to IEC 61499 Function Blocks for verification of distributed control systems. *Control Engineering Practice*, vol. 20, issue 12, p. 1259 – 1269.