Concept of the Virtual Distributed Control System

Maximilian Stremy, Andrej Strasiftak, Pavol Zavacky

***Abstract*—Article deals with design and implementation of modular concept of the distributed control system, where properties of systems with different elements (control elements, network communication, virtual object of control, different control approaches), hierarchical structure and control levels could be analyzed, implemented and verified. In the system could be also compared standardized approaches to the creation of functional blocks (IEC 61131-3 vs. IEC 61499). In the created system connections of virtual objects of control and models (for example WinCC, Control Web, Matlab) and real or virtual control systems (PLC-Logo, S7-300, Simulator S7-300 and similarly) can be implemented through a communication dispatcher. Implemented system is possible to use in education process or in e-learning as well.**

***Index Terms*—distributed control system, PLC, virtual object, standards.**

I. Introduction

THE realization of the virtual distributed control system is built on the software implementation of small PLC Logo 12/24 RC. Students of UIAM MTF STU in Trnava participated especially in the implementation phase by the whole system development. Partial solutions have been and are defended in their thesis or dissertations.

Basic concept of solution has three parts:

1) Virtual and real control elements – there was software model of PLC LOGO created in this part (design and implementation of core and graphic interface). Due to innovations and updates the communication dispatcher can use the real control elements (e.g. S7-300).
2) Communication interface – design and implementation of the communication dispatcher, which is realizing communication between several elements of distributed control system.
3) Virtual objects/process of control – design and implementation of the virtual and real objects of control for whole system with using SCADA systems (Control Web, WinCC, connections to real objects using communication cards and similarly.)

Communication between the main elements is based on the current use of ethernet and TCP/IP protocol (Fig. 1).

Manuscript received June 2, 2012; revised August 10, 2012.

M. Stremy is with the Institute of Applied Informatics, Automation and Mathematics, Faculty of Materials Science and Technology in Trnava, Slovak University of Technology in Bratislava, Trnava, 917 24, Slovak Republic (e-mail: maximilian.stremy@stuba.sk).

A. Strasiftak is with the Institute of Applied Informatics, Automation and Mathematics, Faculty of Materials Science and Technology in Trnava, Slovak University of Technology in Bratislava, Trnava, 917 24, Slovak Republic (e-mail: andrej.strasiftak@stuba.sk).

P. Zavacky is with the Institute of Applied Informatics, Automation and Mathematics, Faculty of Materials Science and Technology in Trnava, Slovak University of Technology in Bratislava, Trnava, 917 24, Slovak Republic (e-mail: pavol.zavacky@stuba.sk).

II. Implementation of PLC

Model for the implementation of virtual programmable logic controller was a logic module LOGO!12/24 RC, selected especially with regard to the scope and complexity of programming functions. It is a relay module with and integrated real-time clock in practice management technology suitable for less demanding processes, such as interior and exterior lightning, ventilation control system, signal processing and control and communication links with the local module for distributed process control. It is compact model of the programmable logic controller class micro PLC.

LOGO! 12/24 RC can realized:

- 16 timers,
- 24 counters,
- 8 switching clock
- 3 counters,
- 42 pulse current relays,
- 42 flip-flop circuits
- functions for analog signal processing and text messages.

Mentioned logical controller offers:

- digital inputs I1 and I24,
- analog inputs AI1 and AI8
- digital outputs Q1 and Q16,
- analog outputs AQ1 a AQ2,
- blocks of digital flags M1 and M24, M8: boot flag
- blocks of analog flags AM1 and AM6,
- scroll register bits S1 and S8,
- cursor keys for operator (C▲, C►, C▼ and C◄),
- 16 virtual (software) outputs X1 and X16.

This type of PLC can be programmed in the software environment or directly by with the operator buttons and display.

ISBN: 978-988-19252-4-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

The aim was to create software representation as closely as possible to real controller in terms of GUI, control, operation and programming. The entire module development process was divided into two independent parts, which provide information each other:

- Graphic interface – design, movement of the screens and menu elements, folding program and similarly. All parts and functionality corresponds to the real machine.
- The core of the programmable logic controller – the implementation of the control program, communication with other elements, settings, etc.

A. GRAPHIC INTERFACE OF SOFTWARE PLC

The idea is to create faithful functional and graphical copy of the LOGO! 12/24 RC and implementation of a display unit divided into two parts – display and control. The control is realized with the cursor, if necessary pre-specified keyboard keys. Text to display and particular items vary exactly as the real PLC. The displayed text and a letters are also true physical copy. In this part of module is implemented creating a control program in terms of linking particular functional block and input their input parameters.

The actual GUI of the virtual PLC comprises of two parts: [1]

- display area,
- Control area.

The displaying part is depicted as a dynamic picture (Fig. 2).

The display of virtual PLC is divided as a matrix containing 14 columns and 6 rows, where the indexing starts at the top left corner with coordinates (0,0) and ends in the lower corner with coordinates (13,5) [1]. One field to display a character is matrix 8x5 pixels size. A character is displayed by lightning points of this matrix (Fig. 3)

Control of virtual PLC is implemented using 6 buttons and their functions depend on the menu of the selected item. This means that control is depending on the menu item or screen element. Functionality of buttons and called features can be fundamentally different.

B. THE CORE

In the core is implemented definition of functional blocks with their parameters and procedures for creating the mutual linking of individual blocks. There is also implemented conversion of inputs to outputs of the blocks and the additional functionality necessary for the operation of the PLC and its connection with graphical interface. PLC implemented functions are represented by their own class containing the peculiar properties (variables) and function (procedures) that are used to convert input signals into output signals. In general, Tblock [2] was created as the main class that contains properties common to all functions, such as number of inputs, input value, output value, the block size, block title, block type, etc. (Table 1)

Each class in mentioned structure is derived by inheritance from the main class. The class hierarchy is shown in the figure below (fig.4).

Running program is implemented and controlled by the threads. One Thread in an endless cycle which goes through a field in witch is stored a program with specific instances of classes. One part of this cycle is called a procedure called Calculate() to calculate the outputs of individual blocks. The second thread records the current time (how long the timer runs) for the delay function – this is particularly important in terms of synchronization functions and their implementation within the control program.

Input and output processing is designed in the core. From the communication dispatcher comes a string of characters as a predefined format. Then parser splits the chain and assignment of the inputs. The actual conversion output is realized by passing a two-dimensional array, that stores formed blocks, and calling the function Calculate. This ensures the calculation of the output of a preventive block on its inputs and (or) internal parameters. These will be served as inputs to the next block in the generated control program. Once you pass the entire field of created blocks, outputs are sent to the communication dispatcher. The string of characters that are forwarded is realized by autonomous function (SendString()). If any output does not change the value, string is not sent. It is sent only in case of an update to one of the outputs. This avoids overload of communication channel. The implementation was made by many features with their characteristic attributes and functionalities (conversion time in timing blocks

TTimeBlock, treatment and reduction of data fields, blocking, write and load generated program to a XML etc.)

Difference against real PLC is saving control program to pre-defined XML format, so there is no loss at the end of the application. For configuration and connection of inputs and outputs there has been developed own window, in which user assigns links to inputs and outputs with other elements in the distributed control system.

C. COMMUNICATION INTERFACE AND DISPATCHER

Communication between elements within the system is ensured by the communication dispatcher based on client-server architecture. Clients represent a virtual PLC or controlled object and procedures necessary for communication with the dispatcher must be implemented in them. The dispatcher is a server that provides communication between the virtual regulator and a virtual process (fig. 5).

The existing library in the development environment was used, with functions and procedures covering the TCP/IP transmissions.

The principle is as follows. Server is running and is waiting to initiate connection with clients. Client stations initialize connection to the server and identify themselves with necessary data (description, number of I/O). Implemented multithread architecture provides always creation of a new thread to communicate with each client. It is checked every 60 seconds if they are alive – if no response from client, thread is closed automatically by the dispatcher. This will prevent buffer overflows and saves CPU resources on the computer. The dispatcher keeps

ISBN: 978-988-19252-4-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

a table with their clients´ identities. Each of them can ask for list of already registered clients in the network and establish communication with them. The dispatcher can then manage all inputs and outputs links in a separate table (IO connection) and in fact act as a router.

Every client has a TCP/IP interface that provides translation of posted messages to logic or analog inputs and outputs. Kernel elements in the network (e.g. software PLC) is not overloaded with data transfer and in addition is allowed easy process of integration and control elements which implements their own TCP/IP interface with minimal modifications of existing solutions.

For two-way communication is used message system sent between paired elements in the network. The structure consists of 4 parts: [4]

1) Identification – the client sends the message to the dispatcher with their identification data
2) List of equipment – after a successful connection dispatcher will send a list of equipment to new client
3) The values – all values (binary as well) are sent as analog. This allows the implementation of voltaic hysteresis for digital inputs and outputs and integrate the virtual with the real elements
4) Inputs and outputs assignment – provides client linking and the assignment of communication channels (inputs of one client are outputs of another client and vice versa.

These four types of messages are providing all communication between clients, network settings, as well as send and receive messages. The communication dispatcher can process multiple parallel connections at once and independently of each other (fig. 6).

D. TECHNOLOGIES

To implement virtual PLC and the communication interface was determined environment Lazarus with support of Pascal language belongs in class Rapid Application Development tools. Libraries which should depend on the operating system were not used, thus the condition of platform and hardware independence was kept. The implementation of communication was made by using synapse library making procedures and functions based on TCP/IP.

For implementation of virtual objects was selected the real-time environment Control Web This environment offers possible opportunities of communication with more than one interface (Lazarus, C++, C#, Delphi, etc.). Must be maintained defined/required block structure of the data necessary for communication with the dispatcher. To save settings, control algorithms, or temporary conditions in each virtual means serves XML files with defined structure.

E. INNOVATION AND UPDATE

In the system was designed and implemented a new communication protocol, linking between the real object with virtual logo by dispatcher and its control have been tested. Currently working on new implementation main elements to the C language platform – Lazarus freeware environment (and applications developed in) has proved to be inadequate and particularly in some specific cases unstable (more threads within the application, components communication of distributed systems in real time, etc.). In this regard, we proceeded to change the platform to increase the reliability and features implemented in the teaching (e-learning) and demanding architectures of distributed control systems. Changes, new features and modules were mainly carried out on the Linux platform in NetBeans and C++ using libraries Comedi (drivers for communication with the plug-in devices integrated as a module in Linux kernel) and GTK + version 2.0 (GUI for an application). The implementation arrangements were:

1. Software PLC:
   - Complete reconstruction of controls in the setup form.
   - Added functionality for the integration of PLC expansion (added inputs and outputs).
   - Checking function in disconnection, to the client is connected to some other client.
2. Communication dispatcher:
   - Added a list of commands for server whitch can respond.
   - Increased stability of created connection – revision of checking and working with checking thread.
   - Added sending parameter information between clients.

III. REAL PROCESS CONNECTION TO VIRTUAL DCS

As the expanding application was implemented and tested communication between software PLC and real process using Advantech card PCI 1711. The card is connected to the computer via serial port and using drivers there is communication with the real process. Reading and writting values is carried out by procedures and functions which use the library comendi.

The initialization of communication and connecting to each input-output channels is possible to set various parameters such as scale used in the channel (eg. 0 is the range +/- 10V, 1 is the range +/- 5V) or type of analog device (eg. AREF_GROUND – is for I/O connected to the ground, AREF_COMMON – lower inputs of all channels are connected , but insulated of GND). The set range calculates the value from the card and the process to the Volts through a programmed module „LINEX“. The card has 12-bit converter, data provides after convert from 0 to 4095. If +/-10 range is set, then -10V is 0 and +10V is 4095. Similarly, it also works for other ranges. [5]

This method and linking of the real process to virtual PLC using communication dispatcher with the simple control program has been verified and validated. It represents one of the options to link the real with the virtual elements in proposed concept of a distributed control system using software or real programmable logic controllers.

IV. EXPLOTATION IN DCS

With implemented components, control, communication and procedural elements listed in the previous chapters can be created different structures of distributed control system. Moreover, the modular concept of DCS can integrate the real elements with the virtual ones within a single

ISBN: 978-988-19252-4-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

architecture. Software PLC may be a major element in the hierarchy of DCS, which would link with the real stations allowed to implement and analyze different approaches [6] and concepts:

- Alternative event processing in the combined discrete dynamical system (stochastically generated events and interrupts in time-driven system).
- Auto-correction of process cycle under the defined algorithms (using artifical intelligence or expert systems in e.g. event-driven systems)-
- Application of standard IEC 61499 in the treatment of Profinet CBA and comparison with the classical approach in complex architectures and software implementations.
- Real-time Profinet communication (RT,IRT, NRT) in RTC 2 class.

Transfer and communication between elements of distributed systems realizes communication dispatcher via a standard TCP/IP protocol, so e.g. S7-300 station (or simulator) can be also connected to the system. In terms of fidelity of the system compared to the real situation, the utilization of the network Profinet or standard Profinet CBA, functionality or interruption station Simatic S7-300 for processing events in the combined (hybrid) discrete dynamic systems, is design of unified model indicated in the block diagram (fig.7).

In that single DCS model, using a standard IEC 61499, can be implemented master-slave (producer-consumer) control architecture of PLCs. Also as determined conditions can be verified on the virtual, real or simulation equipment and analyzed behaviour of designing control algorithms as well. In special cases the model can be used for testing and verifying of the security solutions for protection of process control networks [7].

Virtual models linking (created in th Control Web, WinCC etc.) and communication with the real stations Simatic S7-300, S7-200 is available through OPC server or through MPI interface and software PRODAVE MPI V6.0. There is possible to create models of safety-critical processes with PLC as master element of control and analyze states and potentially dangerous transitions in the process [8]. Concept of model was also used for an experiment of authorization through Trust Chains in Ad hoc Grids [9].

Virtual model (Fig. 8) created in the control Web provides connection also to the real PLC stations or their simulators as well as possibility of extension and communication with virtual part of designed distributed system. Communication with the virtual models (heating system model or production line in Control Web) can be realized by the embedded communication dispatcher with the use of available drivers (ASCII driver, a virtual serial port plus TCP/IP interface, in the newer version has a direct TCP communication) and the creation of TCP/IP connection.

Similarly, it is possible to implement complex systems as part of virtual distributed control system, where control and processing of signals corresponding to parts of the virtual model will .be realized on multiple PLC simultaneously. This makes it possible to compare standards such as IEC 61499 and IEC 61131 (IMAP vs. STEP7).

V. Conclusion

The article deals with design and implementation of modular concept of the distributed control system, where properties of systems with different elements (control elements, network communication, virtual object of control, different control approaches), hierarchical structure and control levels could be analyzed, implemented and verified. In the system could be also compared standardized approaches to the creation of functional blocks (IEC 61131-3 vs. IEC 61499). Through the implemented communication dispatcher is possible to include and integrate to the mentioned system the virtual objects of control (e.g. WinCC, Control Web, Matlab) and real or virtual control systems (PLC-Logo, S7-300, Simulator S7-300 and similarly). The concept of virtual DCS provides space for the realization of chosen methods, technologies and approaches, their analysis and comparison in unified model of DCS at approximately the same conditions. Selected parts of the system were implemented by students of the University.

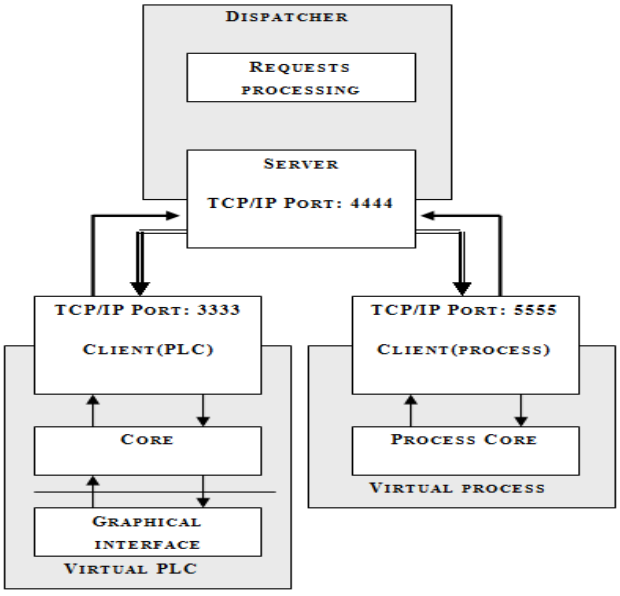


Fig. 1. Communication model

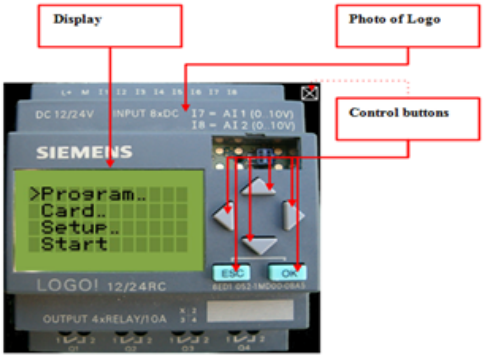
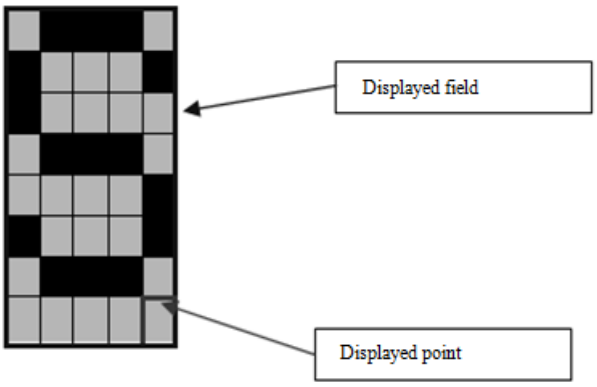


Fig. 2. Virtual LOGO and layout of controls/display

ISBN: 978-988-19252-4-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)



[0,0]= FALSE	**[0,1]= TRUE**	**[0,2]= TRUE**	**[0,3]= TRUE**	[0,4]= FALSE
[1,0]= TRUE	[1,1]= FALSE	[1,2]= FALSE	[1,3]= FALSE	**[1,4]= TRUE**
[2,0]= TRUE	[2,1]= FALSE	[2,2]= FALSE	[2,3]= FALSE	[2,4]= FALSE
[3,0]= FALSE	**[3,1]= TRUE**	**[3,2]= TRUE**	**[3,3]= TRUE**	[3,4]= FALSE
[4,0]= FALSE	[4,1]= FALSE	[4,2]= FALSE	[4,3]= FALSE	**[4,4]= TRUE**
[5,0]= TRUE	[5,1]= FALSE	[5,2]= FALSE	[5,3]= FALSE	**[5,4]= TRUE**
[6,0]= FALSE	**[6,1]=TRUE**	**[6,2]= TRUE**	**[6,3]= TRUE**	[6,4]= FALSE
[7,0]= FALSE	[7,1]= FALSE	[7,2]= FALSE	[7,3]= FALSE	[7,4]= FALSE

Fig. 3. Displaying of „S” character. [1]

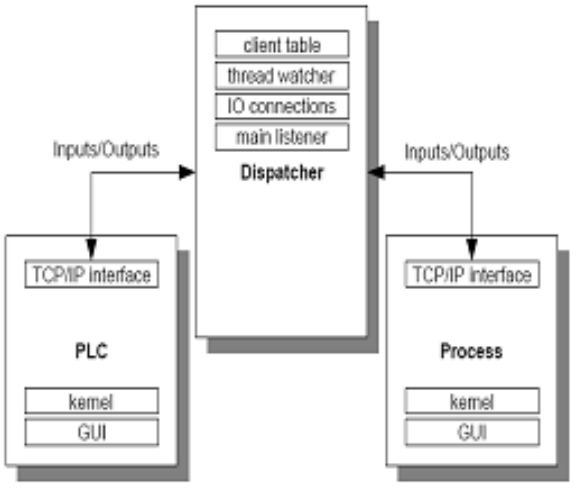


Fig. 5. Communication dispatcher and linking with other elements. [3]

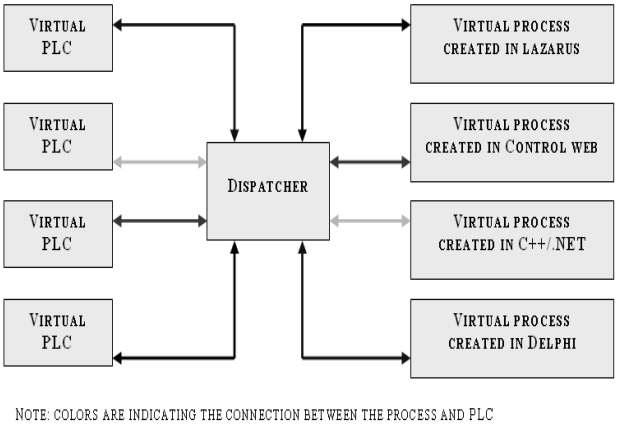


Fig. 6. Multiple parallel connections at once. [3]

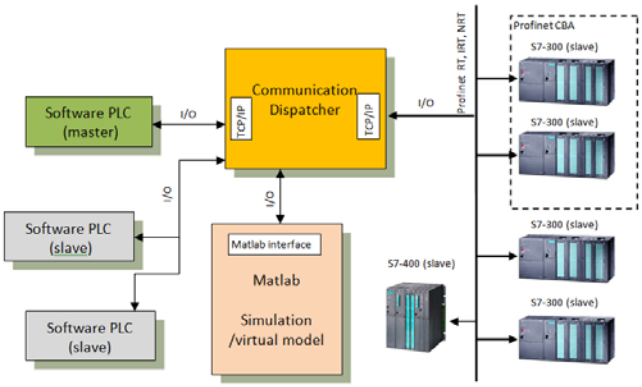


Fig. 7. Block diagram of unified model

Fig. 8. Virtual model of production line in the Control Web

ISBN: 978-988-19252-4-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

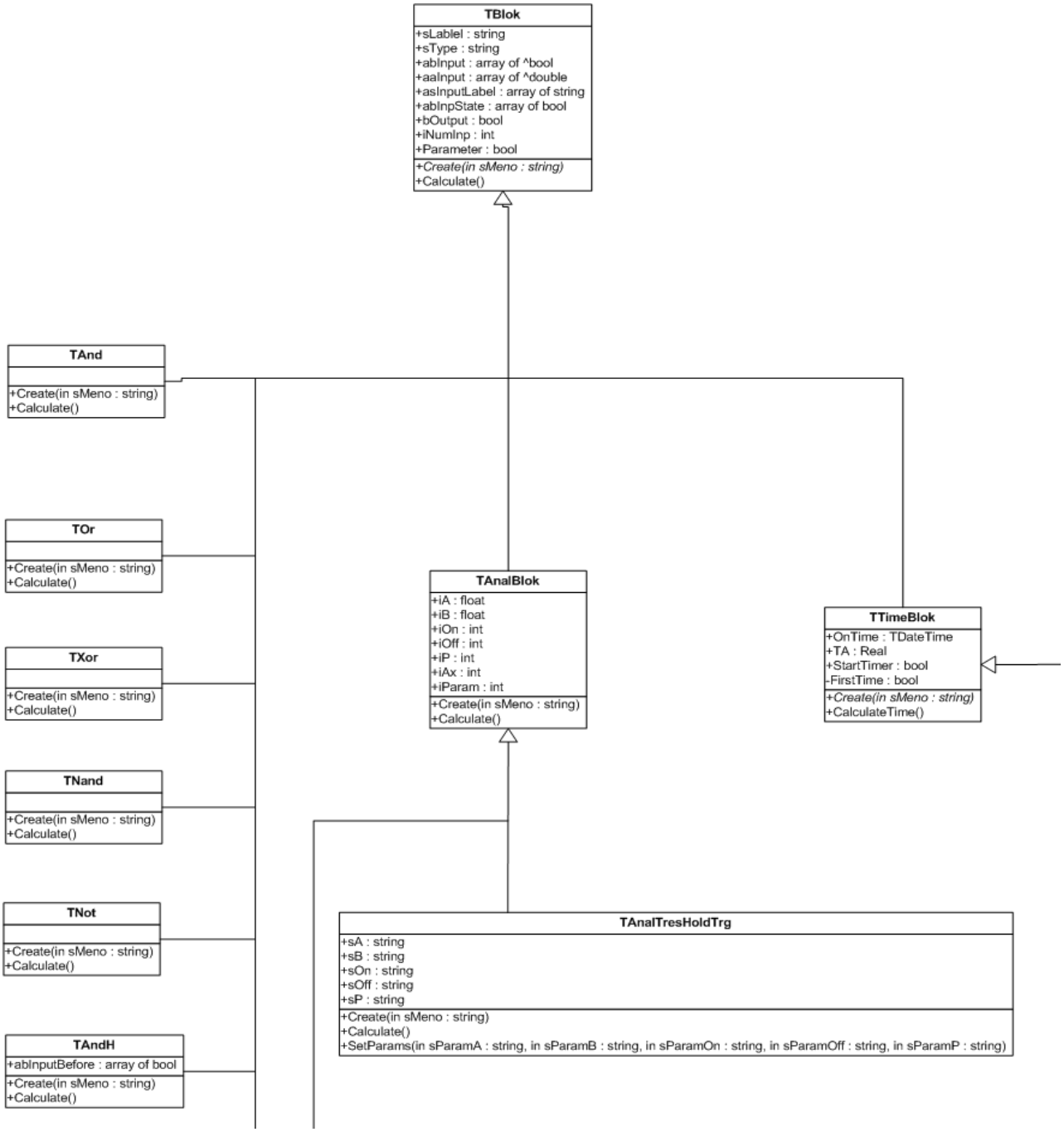


Fig. 4. Inheritance and class hierarchies. [2]

TABLE I: STRUCTURE OF TBLOCK MAIN CLASS [2]

TBlock		
Properties	*sLabel*	Name of created block (B1, B2, ...)
	sType	Block type (And, Or, ...)
	abInput	Pointer input array, refers to a Boolean value
	aaInput	Pointer input array, refers to an integer value
	asInpLabel	Array of strings which are stored the names of the inputs to the block
	abInpState	Array of flags that tell us whether the input is negated or not
	bOutput	Digital output value of the block
	iOutput	Analog output of the block
	iNumInp	Number of inputs
	Parameter	Determines whether the block has adjustable parameters
Functions	*Create(sName)*	Procedure to create an instance of the class
	Calculate	Calculate procedure to convert inputs to output

ISBN: 978-988-19252-4-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

References

[1] F. Michalik, "Interface design and implementation of the logics of virtual PLC," in Slovak, diploma thesis, UIAM, MTF STU, Trnava, SR, 2007.

[2] T. Sebela, "Design and implementation of core of virtual PLC," in Slovak, diploma thesis, UIAM, MTF STU, Trnava, SR, 2007.

[3] A. Elias and M. Stremy, "Usage of communication dispatcher for virtual devices," in *Process Control 2008: Proceedings of the 8th International Scientific-Technical Conference*. Kouty nad Desnou, Czech Republic: University of Pardubice, 2008.

[4] V. Minarovic, "Communication dispatcher for virtual automated systems," in Slovak, diploma thesis, UIAM, MTF STU, Trnava, SR, 2007.

[5] P. Zavacky, "Interconnecting of real processes with the virtual PLC," in Slovak, diploma thesis, UIAM, MTF STU, Trnava, SR, 2008.

[6] A. Trnka, "Classification and Regression Trees as a Part of Data Mining in Six Sigma Methodology", in *Proceedings of the World Congress on Engineering and Computer Science 2010 Vol I, San Francisco, USA, 2010.*

[7] M. Simon and R. Halenar, "Security solutions for protection of Process Control Networks," in *Journal of Information Technologies*, ISSN 1337-7469, p. 8-16, 2011.

[8] J. Zdansky, "Using PLC for control of safety-critical processes," in *Proc. OWD*, (pp. 421-426), 2004.

[9] L. Huraj and V. Siladi, "Authorization through Trust Chains in Ad hoc Grids," in *Proceedings of the 4th ACM EATIS annual international conference on Telematics and Informatics: New Opportunities to increase Digital Citizenship (EATIS '09)*, Prague, Czech Republic, June 2009, pp. 68-71, ISBN 978-1-60558-398-3.

ISBN: 978-988-19252-4-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)