ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2005, Vol. 34, No. 1

USING DESIGN PATTERNS FOR DESIGN AND PROGRAMMING OF COMPLEX AUTOMATION CONTROL ALGORITHMS

Darius Ezerskis, Rimvydas Simutis

Institute of Automation and Control Systems, Kaunas University of Technology
Studentų St. 48, LT – 51368 Kaunas, Lithuania

Abstract. In contemporary automation systems the complexity of control algorithms is increasing significantly, accordingly the structuring and reusability of control algorithms is becoming more important. This article suggests a decomposition of controlled process into elementary control sets. For development of the elementary control sets an approach using design patterns is proposed and some typical design patterns are presented; their advantages and difficulties of practical application are discussed as well. The proposed design patterns are used for developing real industrial control systems. The paper in addition discusses the benefit of using the proposed design patterns for development of the control systems with additional safety requirements.

1. Indroduction

A significant part of process control problems are of universal nature and do not depend on control hardware or industry branch. For instance, control algorithms solutions for transport, circuits of technological protection, communication among PLC's are general for all industry branches and all programming systems. It is useful to make such design tasks possibly general. A typical example of such generalization for the control tasks is well known function block (FB) libraries. Such libraries are implemented in nearly all process control programming systems. Moreover, there are standards such as EN 61131-3 [1] for function block libraries. The use of function blocks improves the quality and safety of the control algorithms considerably. However, it is recommended to implement in function blocks only for those functions that can be characterized very specifically and can be applied repeatedly without modifications.

In case a control algorithm is complex and needs to be modified for each application, function blocks are unsuitable solution. For example, it is difficult to design a universal function block for closed loop controller, because for this controller many extra blocks such as signal filtering and linearization, set-point processing and etc. are needed. It would be very difficult to implement such functionality into one universal FB because each installation of closed loop controller can differ in its control algorithms and its signal, set-point or manipulated value preprocessing. However, the structure of controller algorithms, in fact, is the same and can be made according to one pattern. Therefore some automation equipment suppliers offer templates for closed loop control algorithms [9] [11].

In this paper we propose a method how to find and design the typical patterns in a technological process. In addition we present design patterns for development of control algorithms for typical control tasks. Referred algorithms for automation control of the technological process were actually built according to proposed design patterns. Additionally, the paper discusses practical benefits of the patterns, their influence on safety of the programs, and the difficulties that can emerge in the course of implementation.

2. Design patterns

The design patterns, first introduced by Christoph Alexander [5], are used in informatics for algorithm design and analysis. There were attempts to apply design patterns for design of the control algorithms [3], [7]. However, it was attempted to use only design patterns already existing in computer science. Using these methods in practice is quite complicated for the reason that automation engineers don't have sufficient theoretical informatics backgrounds to use such design patterns. Moreover, the automation control has specific requirements for control algorithms such as effecttive usage of resources, on-line change of program, on-line adding and removing of technological or control equipment, etc [7]. For the above-mentioned reasons this article introduces special design patterns for control algorithms.

Design patterns can be used for structural design of programs (structural patterns) as well as for

algorithm design (behavior patterns) [5]. Using the design patterns has several advantages:

Supplements methods of object modeling by enabling to use experience of designing and programming from old to new projects.

Enables the use of specified (generalized) behavior models for controlled objects as well as models for communication with other objects.

Enables the use of the proven algorithmic solutions in new projects, similar in their problematic. This increases the safety of the programs.

In the following sections we introduce a procedure for determining typical patterns in technological process and we suggest some patterns for common design problems in automation control.

3. Determination of design patterns

3.1. The decomposition of a system

During design of the process control algorithms it is required to decompose the whole controlled object into smaller components. Commonly this can be done using top-down decomposition of the process into smaller technological objects [10]. The decomposition leads to *elementary control segments* ECS, as shown in Figure 1.

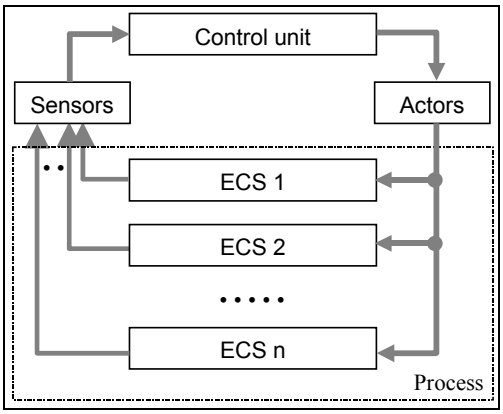

Figure 1. Decomposition of process

This kind of decomposition leads to smaller subsets of controlled object that are better to understand and highlights the interfaces among objects. Such elementary control segments (ECS) can be:

Transporting way (for example system of Screw conveyors, elevators etc.),

- Reactor,
- Boiler protection system etc.

Further decomposition of ECS leads to the following elementary controlled units (ECU):

- Heater,
- Motor,
- Screw conveyer,
- Solenoid valve etc.

Accordingly the two substantial structural elements of control system are:

1. Elementary control unit (ECU),
2. Elementary control segment (ECS).

Though the amount of ECS is not so high, the amount of elementary controlled units can be very high.

Continuously applying object-modeling techniques for each controlled unit and segment a control object will be built. This control object represents status of the real controlled object and according to control algorithm and input signals will generate output signals to the real object [4].

3.2. The control of ECU

It is useful to implement the control of elementary units by using function blocks FB (like EN 61131-3). For example, it is quite simple to create a function block for motor control, as shown in Figure 2. The inputs and outputs of this FB are easy to understand and document themselves. The internal functionality of FB can be described with some sentences or logical equations. Each programmer knows well FB's for PI or PID control or function block library EN 61131-3.

If a recurrent control algorithm can be implemented repeatedly in various projects without modifycations it should be programmed as a FB.

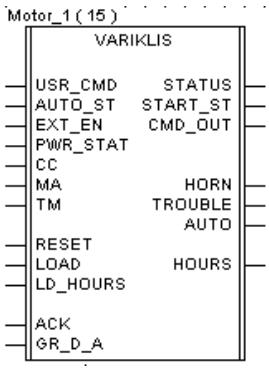

Figure 2. FB for motor control

If a control algorithm can be implemented only with several parametric inputs, which modify algorithm, it should be implemented as a set of FB's or a design pattern.

3.3. The control of ECS

The modeling and programming of elementary control segment is much more complex task than designing an elementary control unit.

The ECS for the same technological function can:

- Vary for the similar processes. For example a water transfer in one line can be done with one pump and in another line – with two pumps of lower

capacity. Thus the amount of controlled units differs, as functionality and protection rules stay the same.

- The functionality of control algorithm can vary in the start-up stage of the technological equipment.
- Technological properties of ECS can require different control algorithms, measuring methods and etc.

Due to the higher level of complexity a realization of the control algorithms for the ECS as a FB is unsuitable. It is recommended to use design patterns for design of these algorithms. The design pattern can be applied to:

- Program structure,
- Material and products transporting tasks,
- Selection of transporting ways,
- Starting sequences of technological units,
- Design of protection circuits,
- Communication between subsystems and operator,
- Methods of user interface.

There are no special and specific rules for use of design patterns. The authors, who write about the patterns, suggest to use design patterns when design problem is recurrent in several objects, however the actual implementation of algorithms may vary from one implementation to another [5], [8].

The below-set examples show the design patterns for control algorithms. It will be shown how implementation flexibility and safety can be achieved for such algorithms.

4. Typical examples of ECS in process automation systems

4.1. The transporting way

The problem:

All transporting systems have the following typical properties and requirements:

1. The transport way may consist of an equipment set such as:
   a) pumps, valves, filters for gas and liquid transport,
   b) elevators, conveyers for solid materials,
   c) air compressors, airlock valves and diverters for powder materials.
2. This equipment starts operation according to the given starting sequence to avoid product jams. It is usually operated from begin to the end of transport way in liquid transportation and vice versa in solid and bulk materials transportation.
3. In trip case of any unit, the whole equipment in a transportation sequence must be stopped in a specific safe manner. For example, in the conveyer transport all units that are before fouled-up units stop immediately and all units behind must operate for a while to clean themselves.
4. Transport units must have so called “cleaning” or “testing” mode to bypass all interlocks of equipment – in this case it is the manual mode.

The solution:

All the equipments in the transport way are to be implemented as objects. These objects must have the following properties:

1. An unambiguous trouble signal.
2. An input - *External disable* (Ext_DIS). The high level of this input disables the operation of the equipment.
3. An operation mode AUTO/MAN. In AUTO mode the signals are interlocked and the AUTO_ST are active. In MAN mode only manual start is possible. AUTO_ST and Ext_DIS are ignored.
4. The “cleaning” time for automatic stop must be specified for each object.

The idea of algorithm for a transport system is schematically detailed in Figure 3. At first the command from master algorithm starts *obj_1*. The start of the successive objects is possible only in the case if the preceding object already operates. This is realized by *Ext_Dis* input of the object. This manner of object interlocking guaranties that in trip case of the device all objects behind this object will stop automatically. The TOFF block enables the cleaning stop sequence of the system. If the system is not in AUTO mode all objects won’t be interlocked.

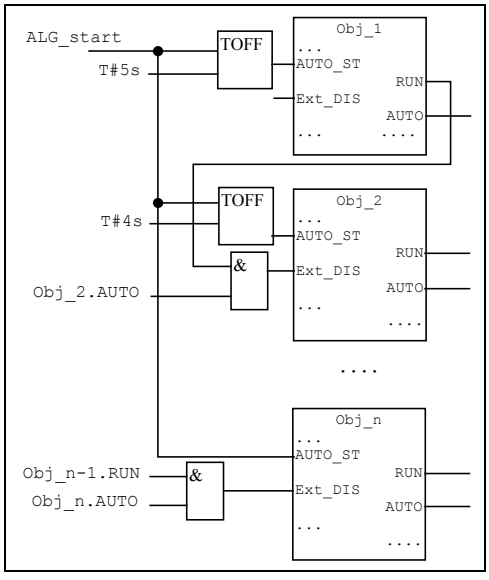

Figure 3. The algorithm of transport way

There is also another method to implement inter location. It can be done by adding new objects that already have additional logical functions, for example Ext_DIS_AUTO, which functions only in AUTO mode. This method could be implemented in traditional

way using object programming procedures. However, in this case it is not advisable for several reasons:

1. The efficiency of control algorithm will decrease, because in the place of AND operation subroutine CALL function would be used.
2. These starting interlocks and "cleaning" times are specific only for transportation algorithms. If new function blocks would be created for each specific application, the library would become huge and not practicable.
3. Finally, the interlocks and stop delays are properties of the technological process and not of the technological unit.

The Implementation area:

PLC and Process station algorithms; independent of the programming language; desirable that there would be possibilities to structure the program and to create FB .

The consequences:

Advantages

- Easy to understand and simple. It is suitable to solve nearby all transportation tasks.
- All necessary functions for normal and abnormal operation are formulated.
- No limitation for implementation area. Each type of objects – motor, pump, valve or elevator – can be controlled.
- Simple inserting and removing of objects in transportation sequence.
- All interlocks among objects can be easily monitored and modified.
- Typical structure of program is easy to detect in a program code. This helps significantly to understand and maintain the program.

Disadvantages and possible implementation problems

- For the reason that it is not an object or FB algorithm it must be programmed individually in each specific case.
- Errors during the programming may emerge, however they would not be of structural nature – and can be easily found and corrected [6].

4.2. The way map

The problem:

Usually the transportation of materials in industry is implemented as a transportation network. This means there is more than one source and destination of transportation way. Examples are shown in Figure 4.

Specifically, for this task the authors saw the utmost variety of algorithmic solutions, all of which could realize the same functionality.

The algorithms for transport way selection must meet the following requirements:

1. The ways shall be easy to modify,
2. The check of selected way and actual way shall be possible.
3. The amount of way combinations has to be unlimited.

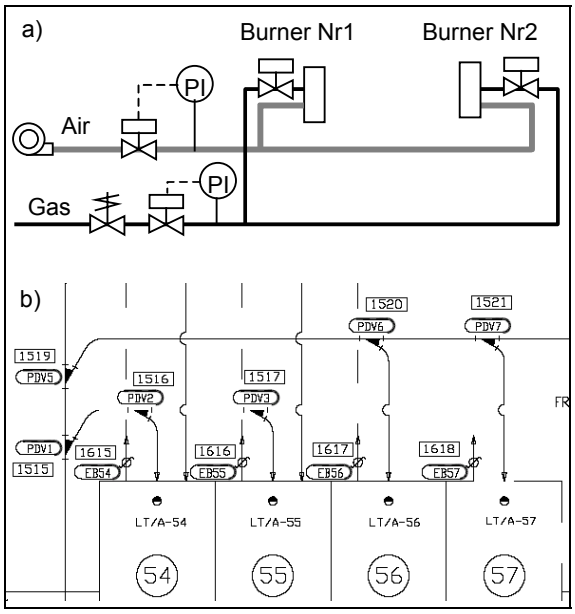

Figure 4. Examples of transport ways

The solution:

The command variables of all transportation units are collected in a structure *transport_set* (see **Table 1**):

Table 1. Structure transport_set (according to Figure 1b)

Member	Type	Comment
PDV1_A	BOOL	PDV1 position A
PDV2_A	BOOL	PDV2 position A
PDV3_A	BOOL	PDV3 position A
PDV5_A	BOOL	PDV4 position A
PDV6_A	BOOL	PDV5 position A
PDV7_A	BOOL	PDV6 position A

The *Map* is array, whose length is the number of specified transport ways. Array elements, which are of the type *transport_set*, represent the map of transport ways. Each array element has to be initialized in the method to represent one of the desired transport ways.

```
Map ARRAY [1..n] OF transport_set =
        {0,0,x,x,x,x, (* 54 *)
         0,1,0,x,x,x, (* 55 *)
         1,x,x,1,x,x, (* 56 *)
         1,x,x,1,0,x, (* 57 *)
         .......}     (* others *)
x - any value.
```

To each transportation object the corresponding members from the command structure *transport_cmd*, as shown in Figure 5, shall be assigned. For selection

the desired transport way it is enough to assign an adequate element of the array *Map*.

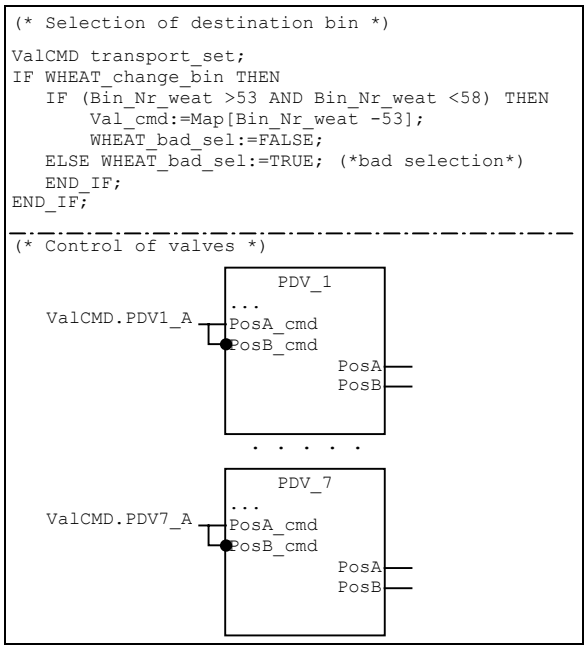

Figure 5. Selection of transport way

If the actual states of transportation objects are collected into a structure of the same type *transport_set* (for example with the name *ValPOS*), comparison of structures ValCMD and ValPOS enables monitoring if the right way of transportation will correspond to the one that was chosen.

The Implementation area:

Algorithms with necessary configuration of objects' status according to specific requirements, for example transportation systems. A typical implementation example is transport way selection in control algorithms for milling products silo storage.

The consequences:

Advantages:

- The algorithm is very simple. It is enough to fill an array with values.
- It is possible to add new way or modify an existing way at runtime.

Disadvantages:

- In some programming systems it is complicated to add new objects to the structure at runtime.
- In the testing phase all way combinations must be tested.

4.3. The Starting Sequence

The problem:

Operation of a technological process usually starts in several steps. Each technological step has explicit start conditions.

For example, to start a cleaning section for grain cleaning, first it is necessary to give a sound signal, then an air compressor and filters must start. Subsequently all cleaning equipment – unit after unit – can start.

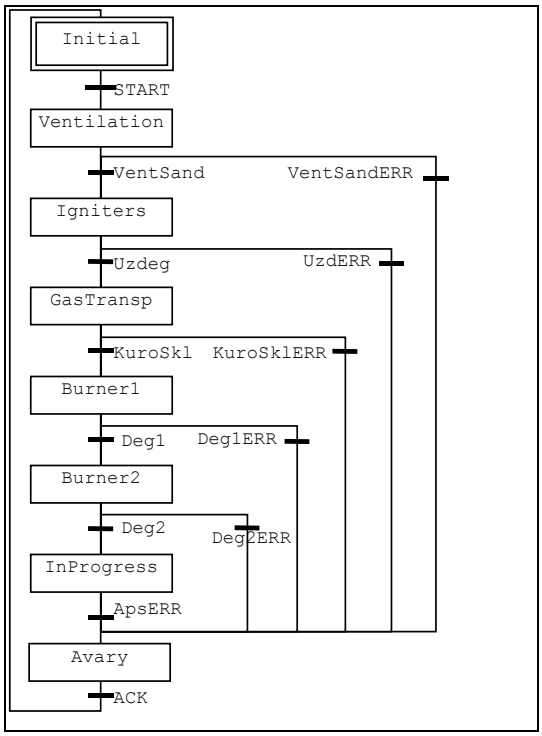

Figure 6. The Algorithm for boiler startup

It is advisable to represent such starting sequence in the control algorithms well. Step-condition programming languages, for example Sequential Function Chart according to EN 61133-3 can be used. SFC step can represent a technological step and generate control signals for operations at this step, and SFC transitions represent conditions of transition from one technological step to another. It makes start-up of the system easier. The deficiency of this solution is that all transitions for abnormal situations in a process must be predicted and programmed as transitions to "alarm" step. Otherwise the algorithm can "lock" in some step and does not return to the initial state. In general case, there can be much more transitions for abnormal conditions than for normal process operations. This significantly reduces readability and usability of the algorithm – especially at start-up, because all transitions of the algorithm must be tested.

An additional problem is to detect the primary condition, which led the algorithm to the alarm state. If the technological process stops, it is essential to detect the original reason of the stop, because during the process shutdown more process parameters can be in abnormal state. Detection of primary stop condition is necessary in the boiler, turbine and power distribution control algorithms.

The solution:

All technological protections shall be aggregated into one set and expressed as a value. In each starting step of technological unit there must be defined which protections are necessary for this step and corresponding value must be calculated. In this case the transition condition to protection state is:

Actual_protection_value < *calculated_value*

The real application of such algorithm is presented in Figure 6. The primary reason for abnormal situation in a controlled unit is *Actual_protection_value* at the entry point to "alarm" step.

The real implementation example:

The start-up of a boiler is performed in a following sequence:

1. Ventilation of boiler and leakproof test of gas pipeline (STEP: "Ventilation")
2. Burning-up the igniters (STEP: "Igniters")
3. Gas supply to burners (Opening of gas protection line) (STEP: "GasTransp")
4. Sequential burning-up of burners (STEP: "Burner1" and "Burner2")
5. Boiler runs (STEP: "InProgress")

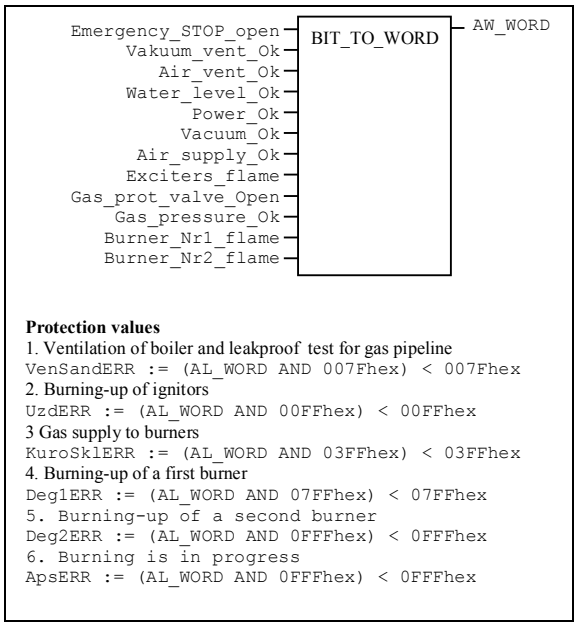

Figure 7. Calculation of protection values

In each step value AL_WORD is evaluated and compared with the value required to this step, as shown in **Figure 7**. If the corresponding protection input becomes FALSE, the AL_WORD will be less than the calculated value and the algorithm will switch to step "Avary". At the entry to this step actual value AL_WORD must be stored to save the primary reason (image of protections) of boiler stop.

The implementation area:

SFC programs, which realize starting sequences.

It is useful for other step/transition algorithms

The consequences:

Advantages

- Flexibility. The algorithm can be used even for boiler, mill or reactor control as well.
- Tiny and clear SFC network.
- Transition to the "alarm" state is always possible.
- Process state at each moment is easy to identify.
- Reliable identification of the primary stop reason.

Disadvantages:

- Calculation of protection value reduces readability of algorithm
- As calculation of a protection value is in a different routine, the start-up of the algorithm may be more complicated.

Implementation examples:

1. Boiler start-up algorithms. ABB ProContic and Modicon PLC families
2. Milling stations – start-up of all technological sequences. Modicon family.

4.4. The dynamic check

The problem:

Distributed control systems require health check of each system component. It is necessary to detect the controller stop or communication break. Usually it is not enough to read data from the controller, because communication coprocessor can work even if the main controller program is stopped.

The solution:

To use the so-called dynamic check. The control unit generates the periodical pulses. These pulses are read from another device and checked if they are periodical. If they are not, an alarm will be generated.

The implementation area:

In the systems where detection of device health or line health is necessary.

The consequences:

Advantages:

1. Simple to implement
2. Guaranties the detection of communication or program fail

Disadvantages:

1. Additional loads to the communication channel
2. The period of pulses must be adjusted to the communication channel reaction time.

The implementation example:

This simple pattern (see Figure 8) is used to identify communication fault with another PLC. The variable *BH_MX_ethernet_imp* is read from another PLC and represents periodical signal.

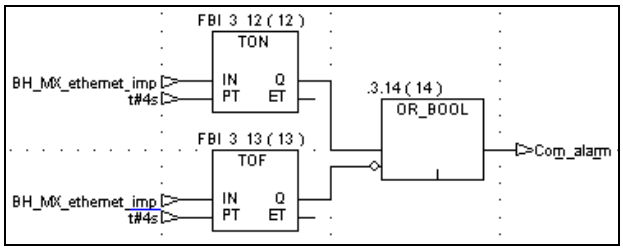

Figure 8. Example of a dynamical communication check

5. Composition of Design Patterns

By using the presented design patterns it is possible to build the major part of automation programs, as shown in Figure 9.

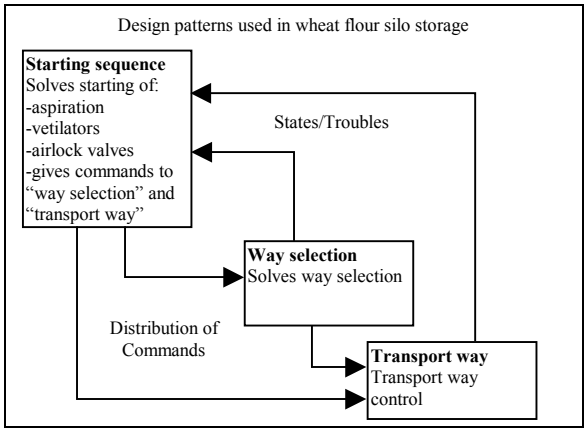

Figure 9. Using composition of patterns

The design patterns *Transport Way, Starting Sequence, Way Map* can be composed to implement nearly full transportation algorithm for production storage or air and gas supply to boiler. Moreover, the program structure and program solutions remain nearly the same. Only unsubstantial technical details are different.

If the control system comprises several control units for one technological unit, dynamic control pattern can be applied to monitor communication among devices.

6. Implementation of design patterns

Practical application of design patterns was employed in several real projects. The first application was to use design patterns in boiler start-up algorithms [2].

More design patterns were used to design a control program for the mill AB "Malsena", Vievis, Lithuania. This technological system consists of:

- 7 big technological subsystems such as wheat cleaning, rye cleaning, wheat and rye milling, storages for both types of products and screening grinding.
- 23 technological groups, those consist of several technological units.
- More than 350 motors, 70 valves and diverters, 20 roller-mils.
- More than 1000 IO signals, 5 PLC's, two control rooms.

All program design and programming tasks were done by the team of three programmers. The leader of the team was one of the authors of the design patterns proposed, for other programmers it was the first try to design the process automation programs in such technique.

The usage of design patterns has significantly improved communication inside the team, because most algorithmic solutions were unified. Using the same algorithmic solutions implied better interfaces among program parts from different programmers, consequently this led to reducing programming errors.

For program design the design patterns from sections 4.1, 4.2, 4.3, 4.4 and additional patterns *program structure* (structural pattern), *object control* (object-operator interface) were used.

Using of design patterns has significantly reduced the number of structural errors in the control software. It was sufficient to test implementation of the new pattern in detail in one technological branch. Following application of this pattern in other branches required only adaptation to new technological equipment and signal check. This considerably reduced the time of program design and implementation.

7. Advantages and disadvantages of patterns using

The usage of design patterns in practice gives the following advantages:

1. Effective reception of formally discovered algorithms. This advantage is particularly relevant for inexperienced programmers.
2. Generalization of algorithmic solutions and program architecture. It allows company, group or project wide unification of algorithms and program code.
3. Minimizing number of structural errors in the programs. The structural errors are the errors, identification and correction of which is labor consuming and in some cases impossible to correct [6].
4. Reduction of program support over the program life cycle. Programs from the known patterns are easier to understand for service staff.
5. Simple documentation of a program. It is sufficient to refer to the design pattern, which the program part comes from, and to document specific details of application.

The usage of design patterns causes the following disadvantages:

1. Up to now there is no suitable and comfortable notation and catalog form for design patterns.

Currently the most common pattern notation form is textual description. In most cases it is an html-format text with references to the relevant information and examples.

2. A good design pattern shall be of universal use and easy to understand. In some cases these two requirements are not easy to reconcile. A pattern author or a team of authors shall have particular experience and traditions in patterning.
3. There are no automatic tools for programming systems that would generate a program template directly from the pattern. This kind of tools would notably facilitate implementation of the design patterns.

The advantages listed above were actually tested in industrial applications. The practical utility of the patterns confirms expedience of their design. The authors believe that further analysis and developments of new patterns for new automation control fields is advisable. In addition, a common and comfortable notation form for design patterns would be further important improvement of this method.

References

[1] DIN Deutsches Institut für Normung e.V.: DIN IEC 1131. *Teil* 3 *Speicherprogrammierbare Steuerungen Teil* 3. *Bouth Verlag, Berlin*, 1992.

[2] **D. Ezerskis, R. Simutis.** Using of Design Patterns for Development and Evaluation of Computer Based Automation Systems. *Information technology and control, Kaunas, Technologija*, 2002, *No.*4(25), 7 - 12.

[3] **J.H. Christensen.** Design patterns for IEC 61499. *Konference papers: Distributed Automation* 2000. *Magdeburg, Germany*, 2000. *http://www.holobloc.com/papers/1499_despat.zip* [2003-01-29]

[4] **J. Fiedler, F.-K. Rix, H. Zöller.** Objekt orientierte Programmierung in der Automatisierung. *VDI-Verlag, Düsseldorf*, 1991.

[5] **E. Gamma, R. Helm, R. Johnson, J. Vissides.** Entwurfsmuster. *Addison-Wesley (Deutschland), Bonn,* 1996.

[6] **L. Hatton.** Programming technology, reliability, safety and measurement. *IEE Computing and Control Engineering*, 1998(9), 1, p. 23-27. *http://www.cs.ukc.ac.uk/people/staff/lh8/pubs/pubier298/PTRel_IER298.pdf.gz*

[7] **T. Honkanen.** Design Patterns in Automation. AS-116.140 *Postgraduate Seminar on Information Technology in Automation. Helsinki University of Technology*, 2002. *http://www.automationit.hut.fi/julkaisut/documents/seminars/sem_s02/Honkanen_paper.pdf*

[8] Industrial Experience with Patterns. *http://www.cs.wustl.edu/~schmidt/patterns.html*.

[9] **J. Müller.** Regeln mit SIMATIC. *Praxisfuch für Regelungen mit SIMATIC S*7 *und SIMATIC PCS*7. *Publicis MCD Verlag, München*, 2000

[10] **P. Rieger, M.S. Hoang, D. Ezerskis.** Lösung automatisierungstechnischer Aufgaben durch objektorientierte Vorgehensweisen. 28. *Jahrestagung des SAK*, *Dresden*, 1997.

[11] Schneider Electric. PL7 Junior/Pro. *Applikationsspezifische Funktionen der Steuerungen Premium. Regelung* TLX DS 57 PL7 xx *ger*, 2002.

DOI: 10.5755/j01.itc.34.1.11967