MCPI – a HMI Application for Monitoring and Controlling Industrial Processes

Vasile Gheorghiță GĂITAN, Mihai Gabriel DĂNILĂ, Mihai Gavril ROBU, Nicoleta Cristina GĂITAN
University "Stefan cel Mare" Suceava, GENPRO 07 SRL Suceava
Str. Universitatii13, RO-720229
bld.George Enescu nr.38, RO-720253 Suceava

***Abstract*—A component of a complete SCADA system used to monitor and control industrial processes is represented by an application which allow human – machine interaction. This paper briefly describes special features, advantages, structure and functional behavior of such as application. Data acquisition software systems are not complete if they do not offer the functionality to supervise and control online values. Since the industrial process automation has passed the stage when devices were working independent, without gathering process's parameters, a distributed system for data monitoring and control (frequently used today) is composed of applications which connect to devices, read and write data, and expose data to other applications which allow interaction of the operator with the industrial process. The main goals of such as applications are: expose process's parameters to the final user, allow the process to be controlled by an user, to set control rules, warn the user regarding alarms and events that need to be acknowledged or only announced.**

***Index Terms*—alarms and events, HMI, industrial automation, Ole for Process Control, SCADA**

I. INTRODUCTION

HMI applications have an important role in monitoring and control industrial processes for the reason that they create human – machine interface. As friendly and easy to use this interface is, as easy the process will be supervised and controlled [3]. The application's functional behavior which has to be prompted, accurate and customizable depending on process's properties, should not be omitted.

MCPI (Monitor and Control Industrial Processes) is a HMI application which, as its name denotes, allows interaction with industrial processes. This interaction is achieved using an OPC data server, in order to create a distributed system, or, if it is necessary, the application may have an individual behavior in which the communication with devices is made using some internal components (drivers – ethernet, serial, USB).

II. FEATURES

MCPI represent a HMI application within a SCADA system. In order to run, this application requires a Windows operating system and it communicates with devices from the industrial process using OPC data servers [2] or internal drivers. MCPI was designed as an OPC client, but it is easily adapted for other types of servers. The main roles are:

- continuous industrial process monitoring;
- process controlling by the human operator;
- automatic process controlling using block schemes;
- data acquisition from wide areas processes (by connecting to data sources places in different points of interest);
- databases creating with time evolution of a process.

MCPI has the following features:

- Graphical objects – there are an extensible library with graphical objects, used to create a customized interface;
- Alarms – alarms can be generated, filtered, shown and printed;
- Security – various security levels can be set to control the process and data viewing. Each user can have his own security level;
- Data and event logging – real-time data acquired from process and MCPI events can be stored. The following user's actions may be logged by MCPI: windows closing, switches closing, parameters modifying, etc. An event is stored on the disk with the additional data: user name, date and time when event was triggered, name of the object which triggered the event, old and new value of the object;
- OPC – MCPI is Ole For Process Control compatible;
- Edit mode and Run Mode – application operates in two modes: edit and run. Switching from one to another is made by selecting a menu option or by using a keyboard shortcut. Menus and application's tools are available depending on the current operating mode and security level of the active user.

III. SOLUTION ARCHITECTURE

The solution architecture consists of an ensemble of objects, processes and connections. The user's task is to create, configure and connect objects between them. A process is a group of independent or interconnected objects between them, and the user creates processes to accomplish certain tasks. The user can open and close processes without affecting other running processes. The solution architecture is presented in **Figure 1**.

IV. PROCESSING TRIGGERED BY EVENTS

The application run is totally conducted by events. A

certain sequence of code is executed only when one of the input variables is changed. In order to save processor time the looping execution of the code is avoided.

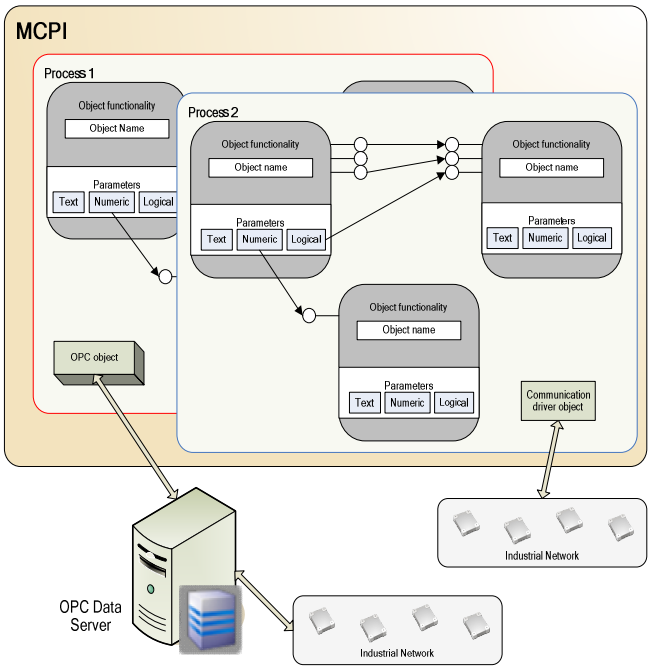

Figure 1. Solution architecture

Each object stays inactive until an event is triggered on one of its connections. The object processes the value according to its internal logics (given by the object type) when an input signal is modified. The objects signal events only when the processing result is changed. This application approach will consume less processor time rather than a solution based on a loop that waits for input signal to change. A close to reality simulation is obtained and therefore MCPI will run faster.

V. MCPI OBJECTS

The solution architecture is based on objects and connections between objects. The user's task is to create, configure and connect objects between them. The objects can be software representations of real components (switches, relay, PLC). Each object encapsulates a specific functionality. Each object has a set of parameters (based on them the object can be defined and configured) and a set of data members (considered input/output points). The following diagram shows the functionality, the input/output data and the object's parameters:

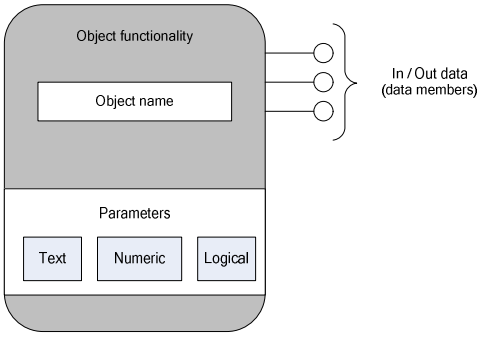

Figure 2. Object structure

Object parameters define the object's characteristics and the possible functionality limits of the object. The parameters are displayed in a dialog box particular for each object. Examples of object parameters: communication speed, high-high limit. Many parameters may contain an expression, which means that one parameter containing an expression will be re-evaluated when one of the operands is modified. Some parameters for better flexibility, are doubled as input/output data (can be read/written from outside the object).

Each parameter will be identified by:

- unique name inside the object;
- data type (the type is encapsulated in the value, which is an Object type variable);
- value – can represent a value or an expression

Data members contain information about the current state of an object such as: value, text color displaying. Input/output data represents the object communication interface to the outside. Input/Output data aliases can be created, but with another configuration (other filter constants). Data member are defined by:

- name – an unique name. May be formatted as following:
  - unique name, any kind of name.
  - a kind of name such as: „input 1 – input 50" which actually represents 50 data members;
- data type – logical, numeric, text;
- access type – read or/and write;

The following things can be achieved using data members:

1. Creating connections with other objects. Only the WRITE access type data members can be connected. For such type of data member a READ access type will be selected from the entire set of objects created.
2. Certain data member characteristics modification. Thus, for a specified data member can be defined an alias (nickname) which is used to create connections between objects. A scaling from one interval to another and a deviation (it is used to update or not the data member value) can be defined for numeric values. Alarm conditions can be defined for numeric and logical members, but only for READ type access members.

Connections: Objects can be interconnected, allowing to the signals to pass from one object to another. In this way the data members from one object can be connected with data members of another object.

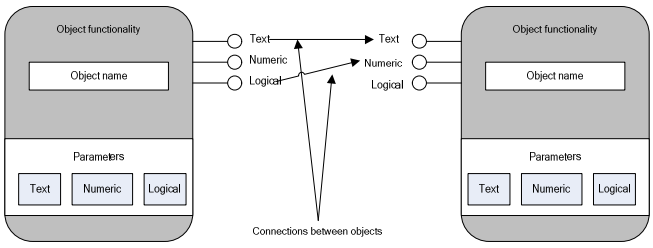

Figure 3. Connections between objects

Each object must have a unique name in the process/folder (each object will belong to one process/folder). The name of the object is specified in properties window, where parameters can be configured. There are a set of rules that must be followed when a name is given to one object:

- the name can include the following characters: A – Z, a – z, 0 – 9 and underscore (_);
- the name must begin with a letter,
- the name must have at most 32 characters;
- it should not contain space or tab character;
- the name mustn't represent keyword of the application (the name of objects or mathematical functions). For example it cannot be used names like switch, if, now, sin, cos, etc.

In the process, the objects can be grouped in folders, for a good organization of them.

This sort of grouping permit to objects to have a unique name at folder level and also permit a good organization of process structure. The folders must have a unique name at process level or at parent folder level, and the set of the rules for names is the same like objects.

In this way can exist objects with the same name in the process, but those objects must be in different folders.

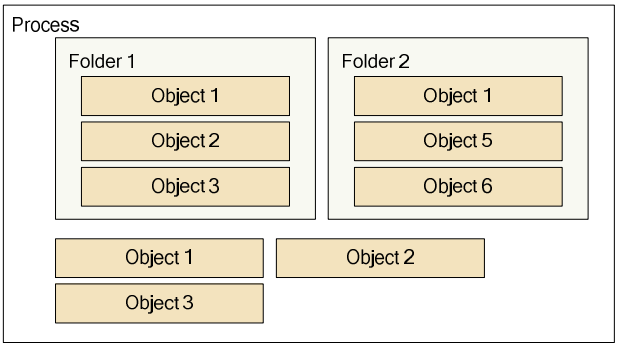

Figure 4. Structure of folders and objects from process

VI. ALARMS AND EVENTS

An *event* can represent any modification in a MCPI application. For example: modification of a potentiometer value, reception of a message from serial port, window opening or closing.

An *alarm* represents an abnormal state of work and must be notified by operator. It can be defined alarm conditions for objects and for data members. The alarms can be filtered, displayed, stored and printed.

VII. DATA AND EVENTS STORAGE

Data storage: MCPI stores data in the moment when value of a node is changing. The data is not logged at set times. In this way it can be fallow the accurately evolution of a parameter. Of course it can be establish filter constants and dead zones to not burden the data base if it is not necessary.

Events storage: It is useful if it is necessary to follow the MCPI actions or modifications made by human operator. It can be stored events like: events produce by objects, system events (application closing), events produced by modification of users (potentiometer adjust, etc). When an event is stored the following items are also written: the operator name, the data and time when the event was produced, the name of the objects that was produced the event, the old and the new value of the object. These information are useful when a post damage analysis of industrial process is required.

The access to the data from data base for visualization or processing is made by a consulting module of logged data. On data from history can be applied the following filters: time filter (for example: it will be read the data from an interval time), numeric filter (for example: it will be read the data from an interval value), data logger type (data, alarms or events).

The data from history can be exposed to the user in graphics form or table form. Also, the data can be visualized in raw form (like they are in the data base) or can be made processing on them, like: average, sum, or complex mathematical expressions (created by the user) [4]. The data from history are followed by the time stamp. It can be visualized the occurrence moment of alarms and events, if the alarm is active or was confirmed, the type of alarm/event, number of alarms/events in a time interval, etc. If an alarm was acknowledged, the data base contains the user that acknowledged it. These information will be displayed in a table form.

VIII. COMMUNICATION WITH DEVICES

The current application represents an HMI that assure a friendly modality for human user to control and monitor the industrial processes. Communication with process devices is established by OPC data servers [1]. To connect to the server, an OPC object that manages connection with data server must be created within the application. If connection with many servers is required, then many OPC objects will be created.

The application architecture is based on objects and connections between them. Therefore, an object can connect to any other objects that belong to the same process as soon as created. These objects may be graphic displays, control objects that are used by a human operator to access process values. The connection creation is shown bellow.

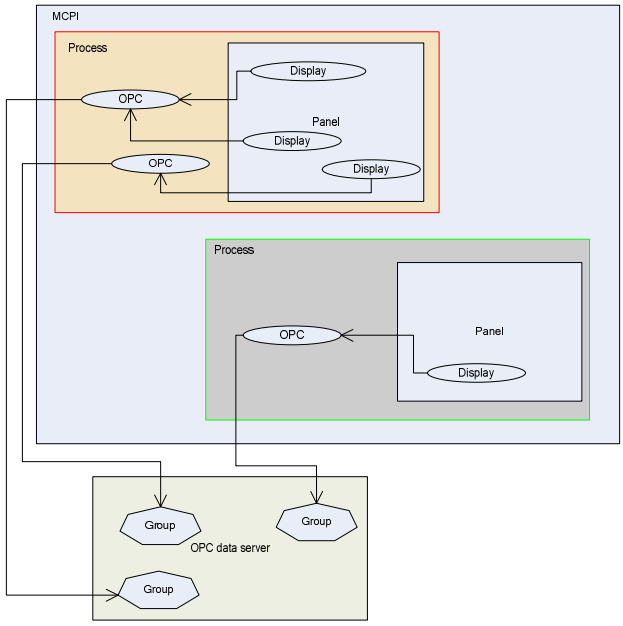

Figure 5. Connection to the OPC data server

This is the typical communication mode with real process devices, but internal communication objects are created when required (for serial communication, ethernet, wireless).

IX. SECURITY

The client application has its own security system based on users, group of users and passwords. A security level is configured by default for each possible operation performed by the application, but it is also possible to associate other security levels for objects and application options. The security level association can be made only by the administrator.

A system security was necessary due to information and commands received/transmitted within MCPI which have a major importance for good industrial process controlled work [4].

X. CONCLUSIONS

Due to internal structuring mode of the application and its functionalities, it offers the following advantages:

- complete functional control of the monitoring process;
- scalability;
- possibility to interconnect with different systems;
- event triggered processing. The code is not executed until input data is modified. It results in high execution speeds;
- after getting familiarized with the application, the user can build a graphical interface in short time. As the complexity of the process grows, the friendly graphical interface allows better understanding of the data acquired from process not only by those familiarized but also by others.

REFERENCES

[1] OPC Foundation Specifications, www.opcfoundation.org
[2] Frank Iwanitz, Jurgen Lange, OPC Fundamentals, Implementation, and Application 2nd rev. Ed., 2002
[3] Bela G. Liptak, Instrument Engineers' Handbook: Process Software and Digital Networks, 2002
[4] Lingfeng Wang, Kay CHen Tan, Modern Industrial Automation Software Design, 2006