The Uniform Engineering of Distributed Control Systems Using the OPC Specification

Vasile Gheorghiță GĂITAN[1], Valentin POPA[1], Cristina TURCU[1],
Nicoleta Cristina GAITAN[2], Ioan UNGUREAN[2]

[1] *Stefan cel Mare University of Suceava, 13, University Street, RO-720229 Suceava*
[2] *SC ProIng SRL Suceava, 38, G. Enescu Street, RO-720253*
[1]*gaitan@eed.usv.ro*

Abstract—**OPC specifications have considerably contributed to the uniformization and standardization procedures for the software applications gathering process data and exchanging it in a unitary manner. However, this specification does not provide instructions or guidelines on how to achieve the interconnection with field devices. The present article proposes a standardization solution in field networks, which will enable users to gain access to a server with a communication component and several network drivers. Consequently, all engineering aspects related to implementation will be given a uniform interpretation. We will get a uniform engineering of distributed control systems.**

Index Terms—**OPC, distributed systems, local industrial networks, communication component**

I. INTRODUCTION

Most research in the field of distributed automation systems has been concentrated upon: 1) the uniformization of design standards with implementation standards; 2) the establishment of standardization procedures for the software applications gathering process data and exchanging it in a unitary manner. In this article, we will only focus upon the process of uniformization and approach the following aspects: gaining access to local industrial networks, and the description of field devices.

The level of today's technological advances has determined us to consider industrial systems with complex communication structures [6] as the one suggested in figure 1. OPC Specification just allows the interconnection of these different types of networks.

Initially, the original OPC specification [1], [4] was meant to solve the problem of integrating client applications with personal computers (PCs) and devices. The automation industry was eager to standardize connectivity and adopted the OPC specification for a wider range of applications than initially intended.

Most producers in the field of automation technologies based on PC, HMI (Human Machine Interface), SCADA (Supervisory Control and Data Acquisition) applications and DCSs (Distributed Control System) or softPLCs accompany an OPC client and/or and interface for an OPC server [2].

After much feedback received from industrial users, the OPC specification has improved considerably in the last ten years and the OPC UA has become the state-of-the-art technology in SCADA applications. The OPC Unified Architecture (UA) is an independent platform standard, which allows the communication among different systems and devices.

Following our expertise gained in the design, implementation and exploitation of OPC servers in several research contracts, we have gathered the observations below:

1. When a user purchases an OPC server and wants it connected to a new industrial network, he faces several situations:
   - To get a new server whose interface complies with the desired network;
   - To get a driver for the desired network to be attached to the server;
   - To design its own server and driver complying with the new network;

2. When a user purchases a new device and wants to include it in the application, he is confronted with several situations:
   - The network is DDL-compliant and the device is accompanied by a description file (or the user is expected to write this file following the guidelines in the instruction book), which ultimately ensures the plug-and-play feature;
   - The network is not DDL-compliant, but it recognizes the plug-and-play device;
   - The network is not DDL-compliant and it does not recognize the device; consequently, the user either orders the corresponding driver, or implements the whole application by himself. Both solutions are costly enough to be discouraging.

3. Whenever a client application requires a new revision to be improved, the user either orders the driver or decides to implement the whole application by himself. Again, both solutions are costly enough to be discouraging.

The situations mentioned above occur because the OPC specification *fails to define* in any possible way the interface with the local industrial network (LIN) or with any other application for data acquisition; moreover, the client application does not come with an SDK to allow the introduction of further facilities.

II. THE PROPOSED ARCHITECTURE OF THE OPC APPLICATION

The general structure proposed for the OPC application is presented in figure 2. It may be easily extended for OPC UA. As figure 2 shows, the application contains five basic components:
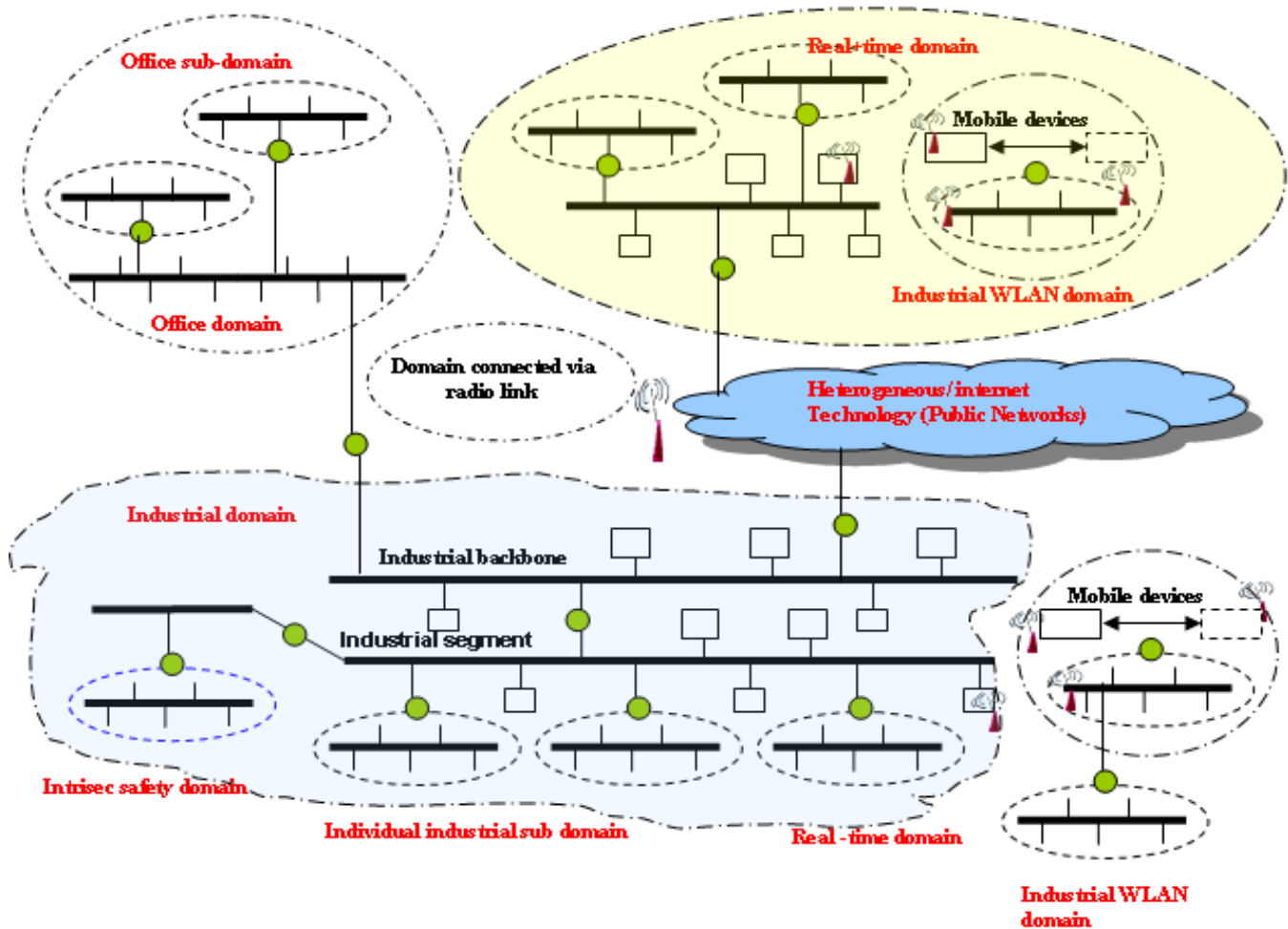
Figure 1. A complex communication system in industry.

1. Three servers: the data server, the history server, and the alarm and event server
2. The two-leveled communication component:

- The acquisition component which defines the object dictionary (a collection of objects and member data typical of every device and defined by an attached EDS file) and comprises a utility program called the network manager
- The communication module, which contains drivers typical of every communication protocol, required by the field devices, and, in some cases, a network set up compliant with a well-defined protocol.

3. The database
4. The client of the SCADA application
5. The middleware is meant to ensure the distribution of information within the widely distributed network and provide support for the implementation of the client-server architecture.

Before a thorough discussion of how access may be gained in local industrial networks, several implementation details are worth mentioning. Thus, the *data server*, which functions as system data collector, carries two interfaces:

1. An interface for the *devices in the field* or for other devices employed in semi-manual data acquisitions;
2. Another interface with a middleware, which ensures the distribution of data within certain networks such as the INTERNET (following the TCP-IP protocols, which also permit serial connections via SLIP and PPP protocols). This interface allows *i* SCADA station clients *(i=1÷j)* to be connected to the application.

The other server types (i.e. data, *history, and alarm and event*) can get both local and remote clients via the middleware. The history server and the alarm and event server can be the clients of some local data server or of any data server on OPC-SCADA stations.

The acquisition component generates the object dictionary. The objects therein are made available to the data servers, which, in their turn, make them available to system clients (provided they are given access rights).

The acquisition component creates the object dictionary following the instructions received from the communication module and from a description file of the devices in the field. This description file is called the *device electronic data sheet*, which is an EDS file (Electronic data Sheet) equivalent to a DDL.

The drivers associated to the communication module employ the protocol required by every local industrial network (such as M-bus, Modbus RTU/ASCII, TCP-IP, or CANOpen).

The network manager administrates the object dictionary and "visualizes" the network in order to perform tests, apply configurations and take maintenance decisions.

At times, certain operations can be directly performed by a network *set up* provided by the producer of a network adapter or of a network device.

III.   A NEW SOLUTION FOR UNIFORM ENGINEERING ACCESS TO LOCAL INDUSTRIAL NETWORKS

As figure 3 illustrates, the communication component comprises two levels. The *acquisition component* is designed to collect the whole amount of data carried by the drivers at the level of the communication module and then make it available to the server data.

Using protocol-specific drivers, the communication module carries out the communication protocols in order to obtain data from the devices in the field. Here is a list of reasons that account for our structural decisions:

- To offer a single interface for the data server, one comprising a set of well-defined functions (methods) (a user may have one data server and one acquisition component);

- To permit the creation of an object dictionary ready to integrate any other devices, irrespective of their producer or communication protocol;

- To allow the description, administration, configuration and maintenance of all listed devices and of their corresponding protocols; the description file is user-friendly, and the specialized software component is designed to manage networks and devices;

- The device driver is the only element specific to any given local industrial network; using a wrapper, it will be attached to the standard function interface offered by the communication module. Network configuration is not entirely dependent upon the original software of the driver: a special simulation driver is also provided.
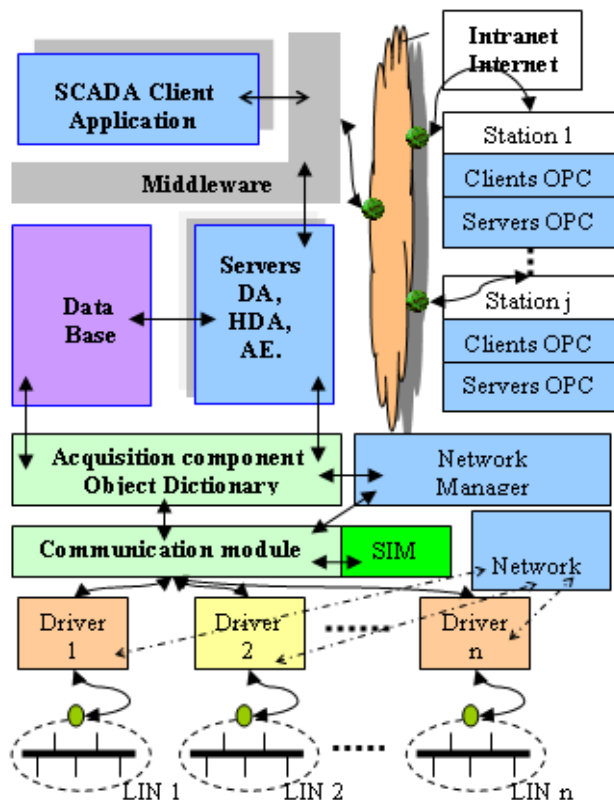

Figure 2. The architecture of the OPC-SCADA application.

The communication component must comply with the following requirements:

- It is supposed to run entirely on the host system if the adapter to the local industrial network is passive (e.g. RS232 to RS485 passive converter). Real-time facilities may be obtained only for WINDOWS CE.

- It is distributed on the host system and on the Intelligent Master when the type of the adapter is also Master Intelligent, ensuring real-time facilities.

- The *object dictionary* represents the key of communication component.

- The object dictionary allows full access to an industrial network station; each station has its own object dictionary.

- The object dictionary is defined by an EDS (Electronic Data Sheet) or DD (Device Description) file, specific to each station.

- The object dictionary contains at least PDOs (Process Data Object) that allow read/write operations from/at analog and numeric inputs/outputs and the detection of other active values (to be subsequently defined) of various stations, as well as SDOs (Service Data Object) that allow the reading of states or the reading/writing of station parameters.

- the EDS files will be used to create the object dictionaries, and the communication component will update these on Intelligent Master;

- it will ensure the definition of a cycle of data acquisition/reading from/into the process. The cycle will be divided in equal time quanta. These quanta will be employed in the cyclic communication employing PDOs, as well as in the acyclic communication using PDOs or SDOs. At least one time quantum will be reserved for the acyclic communication. Each station may receive one or several quanta for the acyclic communication.

- The communication component will provide a utility program designed to ensure full access to object dictionaries and to create the object dictionary for virtual channels. This utility program can switch the dictionary into the simulation mode. Thus, the values will not be read from the stations, which will display programmable constant (or evolving) values.

- The communication component will provide an utility program designed to define, manage and test the acquisition cycle.

- The communication component will provide a set of methods to allow user applications gain access to the object dictionaries.

- The communication component will update the PDOs defined in the acquisition cycle; any PDO, which is not part of the acquisition cycle, will be read/written in an acyclic manner.

- The communication component governs the acquisition cycle

- The communication component allows for the automated or manual definition of industrial network characteristics (e.g. station scans, introduction/ elimination of news stations, etc.)

- The communication component will provide a utility program designed to monitor station response time included in the catalogue.

IV. ACQUISITION COMPONENT

The acquisition component uses EDS files and active field-network information received from the communication module to create the object dictionary. To put it differently, this dictionary is a collection of data gathered from the process; depending on their need and implementation solutions, the data server or other servers will connect to this dictionary.

The acquisition component will memorize the acquisition module values in a cache. In fact, it memorizes the values displayed by each device in order to make them available to the data server.

Besides memorizing process values, the acquisition component is supposed to perform the following functions:

- Provides a network manager.
- Provides a set of methods to get information on the items (i.e. the properties of monitored devices) offered by the acquisition component.
- Present a set of methods for item writing and reading.
- Present a set of methods for moving within the tree data structure (networks, devices, device properties). These methods are meant to:

  - Modify the current position within the tree data structure.
  - Ask for the items and item groups on the current position.
  - Ask for a unique item name to be subsequently added to a client-created group in order to obtain values for the item in question.
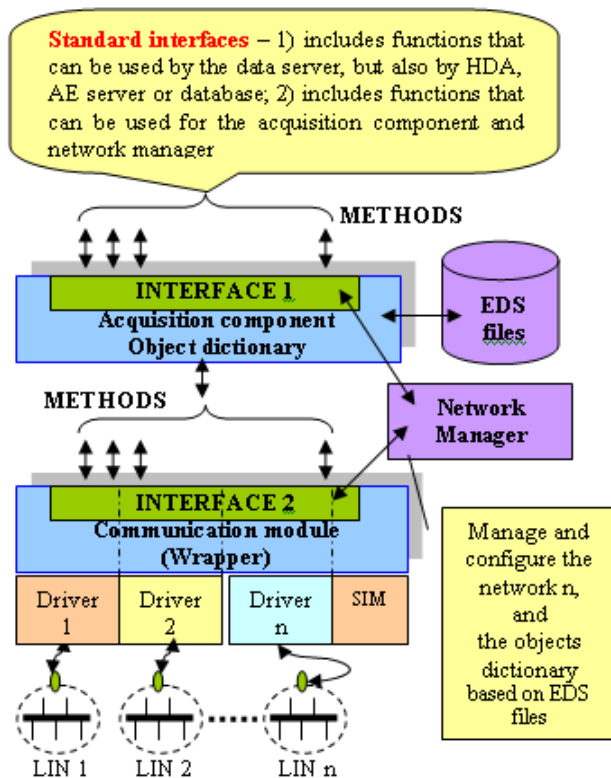

Figure 3. The architecture of the two-leveled communication component.

V. COMMUNICATION MODULE

The communication module ensures the communication with the acquisition devices in keeping with their protocol. If there is no Master network with (real-time) acquisition facilities, the module implements even the automatic system generating the acquisition cycle, which will be further referred to as the acquisition engine.

For simple adapters such as RS232/ RS485 or RS232/ M-bus, RS232/ CAN, USB/CAN., the engine is implemented on the host computer. There is no real-time data acquisition for operating systems such as tip Windows 2000, XP, or Vista. Real-time acquisition is valid only for Windows CE, Microsoft. The acquisition drivers may be classified as follows:

- *simulation drivers*. They are meant to simulate the gathering of process data and they may be used in testing servers, clients and the networks manager.
- *driver with a PC-based acquisition engine*. The acquisition may be performed locally on the PC using an acquisition card connected, for instance, to an internal PCI bus, or to one or several field devices connected to a local industrial network via a simple adapter (with or without local intelligence, the latter characteristic being valid only for the implementation of the network protocol). The PC connection may be performed using standard interfaces such as COMi, LPTi, USBi, Firewire, or Ethernet.
- *driver with no acquisition engine*. The engine is placed on the intelligent adapter (with internal or external PC connections). As these types of adapters may carry local data histories, data base updates must be handled with extreme care.

VI. DEFINED INTERFACES

OPC specifications have considerably contributed to the uniformization and standardization procedures for the software applications gathering process data and exchanging it in a unitary manner. However, this specification does not provide instructions or guidelines on how to achieve the interconnection with field devices.

The acquisition component functions are briefly presented in table 1. These functions are designed to create the interface with the data server. The functions proposed for standardization for the communication module are presented in table 2. These functions are designed to create a standard interface between the acquisition component and the communication module; this interface allows for the standard connection of drivers specific to the network protocol.

TABLE 1. FUNCTIONS PROPOSED FOR STANDARDIZATION
FOR THE ACQUISITION COMPONENT

No	Function (method) - Description
1	*LoadNetworkConfiguration* Loads the saved configuration at the last OPC server shutdown (into the edf.ini file)
2	*ShowNetManagerDlg* Launches the network manager implemented in this library (displays the window for the network manager)
3	*Sds* This function is used for user identification (seeting up the security level)

4	*QueryAvailableProperties* This function is employed to obtain the ID list and the property description of items.
5	*GetItemProperties* This function reads the values for the property list received as parameter.
6	*LookupItemIDs* This function is used to obtain the connection string list for the item properties.
7	*ChangeBrowsePosition* This functions allows to move inside the tree structure with the items on the server
8	*BrowseOPCItemIDs* This function is used to obtain a name list of the items in accordance with a received filter.
9	*GetItemID* This functions returns the connection line for an item
10	*BrowseAccessPaths* This function initiates the browse operation starting with an item whose path is set as parameter (the browse operation does to start from the root)
11	*SyncRead* The function is used for the synchronous reading of data (the reading may be performed from the cache or directly from the device)
12	*SyncWrite* The function is used for the synchronous writing in the network devices
13	*GetItemAttributes* This function is used for reading of item attributes
14	*GetState* This furnctions returns data on server status
15	*GetItemResult* This function is employed to read access rights and identify the type of the item sent as parameter
16	*ItemExists* This functions verifies whether an item is present or not in the server address list
17	*FreeDllMemory* Frees the resources for the library while executing the application that uses the library
18	*ShowConnManagerDlg* The function launches the connection manager implemented by the library called „gpcc_ODV.dll".

TABLE 2. THE FUNCTIONS PROPOSED FOR STANDARDIZATION FOR THE COMMUNICATION MODULE

No	Function (methods)	Description
1	*AddCommunication* Adds a network saved at application shutdown.	
2	*DefCommunication* Defines and adds a new network	
3	*ModifCommunication* Modifies the communication settings for a network	
4	*GetCommString* Gets the configuration string for a network	
5	*StartCommunication* Starts the communication for a network	
6	*StopCommunication*	

	Interrupts the communication for a network
7	*SetPointerEvFct* Sends a pointer to the furntion in use in order to receive the events from the libraries involved in communication
8	*ScanNetwork* Scans a netoowrk
9	*GetAchisitionPARAM* Reads the acquisition parameters for a network
10	*SetAchisitionPARAM* Sets up the acquisition parameters for a network
11	*StartTest* Initiates the testing of a network
12	*StopTest* Stops the testing of a network
13	*StopScan* Stops the scanning process
14	*DelCommunication* Deletes a network
15	*AddDevice* Adds a device to the network whose handler is received as parameter
16	*GetCommunicationDescription* Reads the network description
17	*AcquisitionParamsToString* Takes over the string of acquisition parameters
18	*StringToAcquisitionParams* Sets up the acquisition parameters
19	*FreeDllMemory* Frees the dll memory
20	*AddAsincronObject* Adds asynchronous objects to the bottom list of asynchronous objects
21	*DeleteAllDevicesNetwork* Deletes the devices of a network
22	*ScanNetworkEthernet* Verifies the Ethernet connexions
23	*ShowConnectionsManager* Displays the connection manager
24	*GetTypeComunication* Returns the communication type
25	*StartConnectionsManager* Creates the thread for the connection manager and initiates the connection check
26	*StartCheckConnections* Starts the connection check (starts the connections manager)
27	*StopCheckConnections* Closes the connections

VII. EXPERIMENTS AND RESULTS

The current version of the SCADA system was developed after the observations that we have made for a period of eight years. In this section we provide a comparison in terms of performance between the new version of OPC-SCADA application and the earlier version. The architecture of the old OPC-SCADA application is presented in fig. 4. The main role of the communication component from the old OPC-SCADA application is to ensure the connection between the RS-485/232 master and the various OLE device components. The concurrent access to the COM interface exposed is handled using the Apartment threading mode.
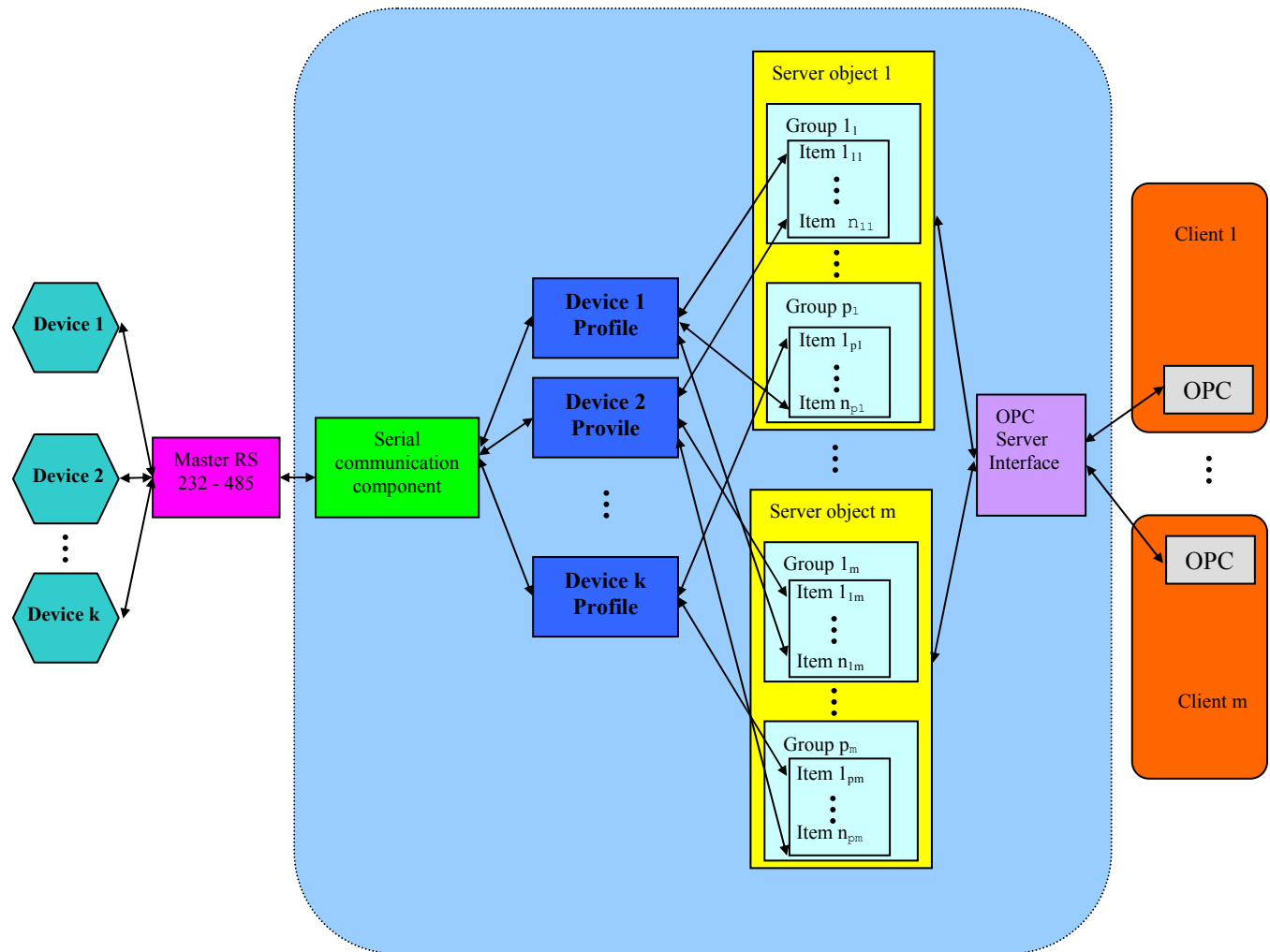
Figure 4. The architecture of the old OPC-SCADA application.

The major differences between these two versions are:

- For the first version, each type of device has a profile (an ActiveX instance) associated with it, while in the new version there is the acquisition component (as a dll library) which contain the objects dictionary and each device has an associated input in objects dictionary.

- In order to introduce a new device, we create an instance of an ActiveX control, while in the new version we create a new entry in object dictionary.

- In the OPC client, each component that reads data form OPC DA server will create its own group. In the new version it will create an OPC object that creates a group in the OPC server and other objects will retrieve data from the OPC server via OPC object (in this case it will create a single group for all objects from the client).

- Each component of the OPC client is an ActiveX control instance, while in the new version each component is an instance of a class from a dll library.

- In terms of software development, in the old version for each type of device we developed an ActiveX control, while in the new version we write a description text file (software development knowledge not required).

- The first version implements the OPC specification v1.05. The new version implements the OPC specification v2.05.

To demonstrate the increase of performance in the new version, we made the following experiment: for both versions, a project in the client (which displays a numeric value with a seven segments display) was created.

We monitored the performance evolution (CPU load and memory used) by doubling the display number systematically from one to 1024.

TABLE 3. THE OLD CLIENT. RESOURCE .
CONSUMPTION OF RESOURCES, CPU AND MEMORY,
FOR 1 TO 1024 GRAPHICAL CONTROLS

Application controls	CPU load %	Memory MB
1	0	12.804
2	1	12.864
4	1	12.892
8	1	12.952
16	1	13.044
32	2	13.212
64	3	13.492
128	9	14.212
256	16	15.68
512	34	18.922
1024	50	32.844

TABLE 4. THE NEW CLIENT. RESOURCE.
CONSUMPTION OF RESOURCES, CPU AND MEMORY,
FOR 1 TO 1024 GRAPHICAL CONTROLS

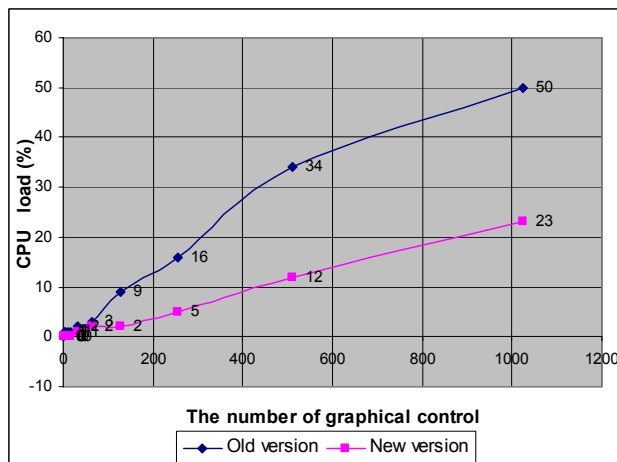| Application controls	CPU load %	Memory MB
1	0	26.488
2	0	26.6
4	0	26.08
8	0	26.08
16	0	25.116
32	1	25.116
64	2	25.58
128	2	26.34
256	5	27.5
512	12	30.048
1024	23	37.752

Figure 5. The tests were performed on a computer with Intel Pentium 4, 3.00 GHz, 512 MB RAM, Windows XP SP2.

VIII. CONCLUSIONS

The present article has addressed the issue of how to obtain a slight form of standardization in field networks. Thus, users might easily consider the use of a server with a communication component and of several network-oriented drivers. This solution results in the uniformization of all engineering aspects related to various implementation stages. The communication component was presented in detail.

The original contributions brought to the study of the OPC specification in distributed control systems refer, in author's vision, to the following aspects:
- the devise of the communication component and the object dictionary;
- the defining of the SCADA client;
- the defining of the distributed database.

The present article focuses only on major aspects related to the communication component. Our novel approach refers to:
- The two-leveled communication component.
- The creation of a standard set of methods ensuring

the connection to the data server.
- The defining of a device description language to allow the addition of devices to plug-and-play systems.
- The defining of a standard interface to connect device driver or local industrial network drivers to the communication module.
- The defining of an acquisition cycle based on time division, process data objects (PDOs) and service data objects (SDOs).

The proposed architecture has been put into practice and tested in various research contracts mentioned in [8], [9]. We have also implemented drivers for ASCII networks (using the RS485 standard line), as well as for MODBUS ASCII, RTU and TCP/IP, CANOpen and M-bus networks.

These results are obtained through practical experiments [10] and observations conducted during the 8 years. Always followed the increasing performance [11] and scalability of the system based on OPC servers.

In the future, we intend to create a mathematical model for the new implementation. Also, we want to develop a test scenarios to show superior performance of the new versions. In the figure 5 it is presented the first test made with the same client application. We can observe an improvement by 50% of performances.

REFERENCES

[1] Frank Iwanitz and Jürgen Lange / Hüthig OPC-Fundamentals, Implementation, and Application, 251 pages, 1 CD, ISBN: 3778529048.
[2] ISA EXPO2005 Chicago IL, OPC and OPC Unified Architecture
[3] Eric Murphy, MatrikonOPC - July 2006, OPC UA - How Deep Does Interface Standardization Go?
[4] OPC Foundation – OPC Common 1.10 Specification
[5] OPC Foundation – OPC UA Part1 – Concepts 1.00 Specification
[6] Dr.-Ing. Axel Klostermeyer Siemens A&D PT2 P, VAN – Developing the Future of Industrial Communication Presentation of the European R&D-Project "Virtual Automation Networks"
[7] Stefan-Helmut Leitner, Wolfgang Mahnke - ABB Corporate Research Center, OPC UA – Service-oriented Architecture for Industrial Applications
[8] Vasile Gheorghiță GĂITAN, Cornel TURCU, Alexandru GOLOCA, Renati POPA, An RFID and OPC Technology Based Distributed System for Production Control and Monitoring, RFID Eurasia 2007, September 5-6, Istanbul, Turkey.
[9] Cristina TURCU, Cornel Turcu, Valentin POPA, Vasile GAITAN, ICT and RFID in Education: Some Practical Aspects in Campus Life. 3rd International Conference on interdisciplinarity in education ICIE'07, March 15-17, 2007, Athens, Greece, ISBN 978-960-89028-4-8, ISSN 1790-661X
[10] Vasile GĂITAN, Using OPC technologies with the Highly Functional Distributed System Advances in Electrical and Computer Engineering, Suceava, Romania, ISSN 1582-7445, No 2/2003, volume 3 (10), pp. 35-46
[11] Valentin POPA,Vasile GĂITAN, Transponders in a Wireless Sensors Network, Advances in Electrical and Computer Engineering, Suceava, Romania, ISSN 1582-7445, No 1/2003, volume 3 (10), pp. 62-67