

OPC Server for P-NET using VIGO

Martin Wollschlaeger

Abstract

OPC (OLE for Process Control) is a new technology in the market of automation and control systems. Based on the OPC specification, concepts for the mapping of P-NET systems to OPC elements are discussed. Aspects of an implementation of an OPC server for P-NET are described. The implementation uses VIGO and its basic concepts, in order to achieve a bus system independent OPC solution. In addition, the integration of the OPC server into the VIGO tools set is described. First results of the implementation are discussed, comparing the implementation efforts and the results to traditional solutions.

1. Introduction

The trends in software technologies play an important role for software strategies in the area of fieldbus systems, too. These trends are characterised to be integrating between applications. An excellent example for that is the Internet - it is integrating different hardware and software platforms as well as information of different sources and structures. These integration tasks require unique software interfaces to handle the information.

A very important technology for interface construction comes from the "Windows-world", introduced by Microsoft as OLE (Object Linking and Embedding) /1/. OLE has become a standardised interface and mechanism for data exchange within Windows. OLE is based on a technology named COM (Component Object Model) /2/. The main principles of COM rely on reusable objects (components) with well-defined interfaces, based on object-oriented concepts. Using COM, applications may be partially or completely reusable and may reuse previously developed components themselves.

The acceptance of the de-facto standard operating system MS Windows - first of all in its 32-bit implementation of Windows NT or Windows 95 - in the area of process control application has introduced OLE and its technology to those kind of applications. However, some special subjects have to be considered in order to make OLE technology really useful for process control. A special interest group defined the add-ons to standard OLE and created OPC (OLE for Process Control). In 1995, this group went public as the OPC Task Force at the ISA Show in New Orleans. Its members consisted of Fisher-Rosemount, Intellution, Intuitive Technology, OPTO 22, and Rockwell Software. Microsoft was to be involved in a supportive and consultative role.

The first draft version of the OPC specification was released in December 1995, a second draft specification followed in March 1996, and at August, 29th the final release of the OPC specification was published /3/. The success of OPC may be recognised by the number of products using OPC as well as by the number of OPC Task Force members (currently 127 from all over the world). Local offices - like OPC Foundation Europe - have been established, in order to provide information and technical support.

Dr.-Ing. Martin Wollschlaeger
Institut für Prozeßmeßtechnik und Elektronik (IPE)
Otto-von-Guericke-University Magdeburg
P.O. Box 4120
39016 Magdeburg
GERMANY

Tel.: +49 (391) 67-1 46 53
Fax: +49 (391) 5 61 63 58
e-mail: mw@ipe.et.uni-magdeburg.de

2. OPC Overview

2.1. Objectives

Today manufacturers face the task of integrating plant floor data into their business systems. This task is made difficult because of the almost unlimited number of methods plant floor devices use to communicate to other systems.

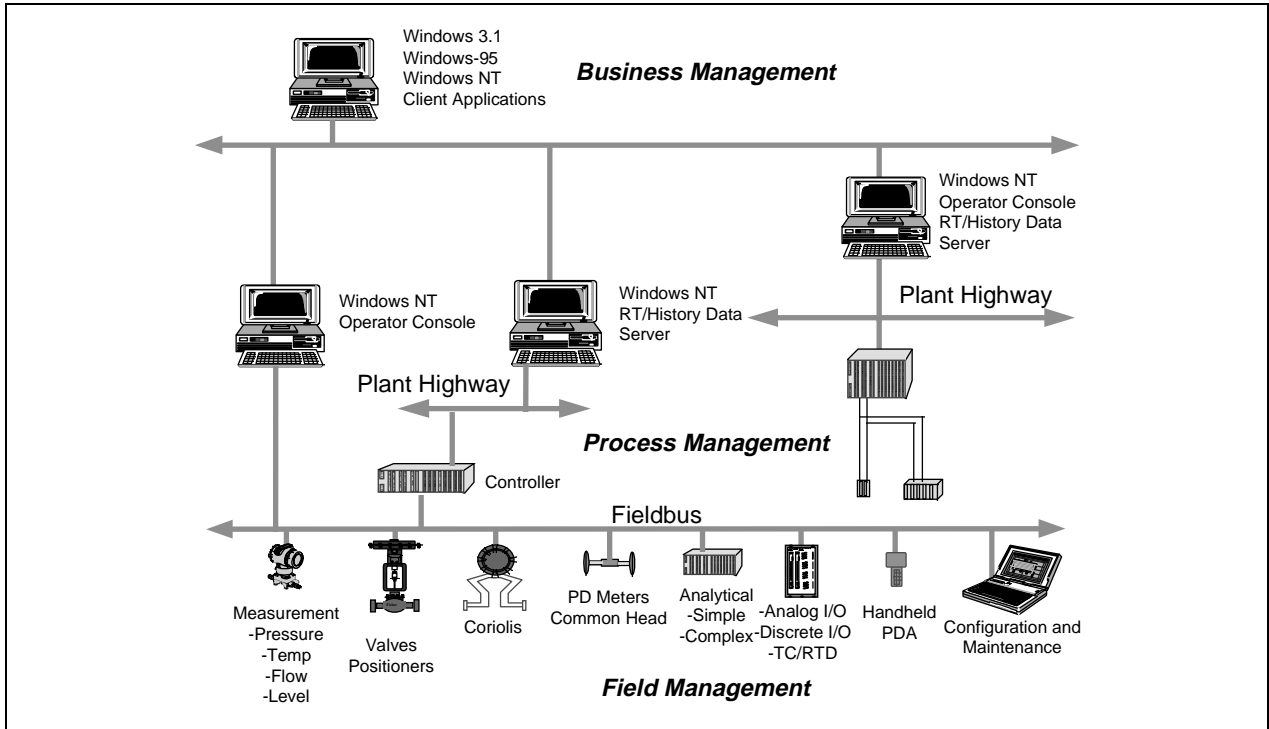


Figure 1: Process Control Information Architecture

The information architecture for the Process Industry shown in **Figure 1** involves the following levels:

❑ **Field Management.**

With the “smart” field devices, a wealth of information can be provided concerning field devices that has not been available in the past. This information provides data on the health of a device, its configuration parameters, materials of construction, etc. All this information must be presented to the user, and any applications using it, in a consistent manner.

❑ **Process Management.**

The installation of Distributed Control Systems (DCS) and SCADA systems to monitor and control manufacturing processes has made data available electronically which had in the past been gathered manually. This data must be provided in a consistent manner to the operators and engineers responsible for making decisions based on that data.

❑ **Business Management.**

Additional benefits can be gained beyond those typically cited when justifying the installation of control systems, accomplished by integrating the information into the business systems managing the financial aspects. Providing this information in a consistent manner to client applications minimises the effort required to provide this integration.

To do these things effectively, manufacturers need to access data from the plant floor and integrate it into their existing business systems. Manufacturers must be able to utilise off the shelf tools (SCADA packages, databases, spreadsheets, etc.) to assemble a system to meet their needs. The key is an open and effective communication enabler concentrating on data access, and not

the types of data. So there is the need for a common way for applications to access data from any device on the plant floor, allowing compliant applications to seamlessly access data in a manufacturing environment. OPC is designed to be a method to allow business applications access to plant floor data in a consistent manner and will provide many benefits:

- ❑ Hardware manufacturers only have to make one set of software components for customers to utilise in their applications.
- ❑ Software developers won't have to rewrite drivers because of feature changes or additions in a new hardware release.
- ❑ Customers will have more choices with which to develop World Class integrated manufacturing systems.

With OPC, system integration in a heterogeneous computing environment will become simple. Leveraging OLE/COM the environment shown in **Figure 2** becomes possible.

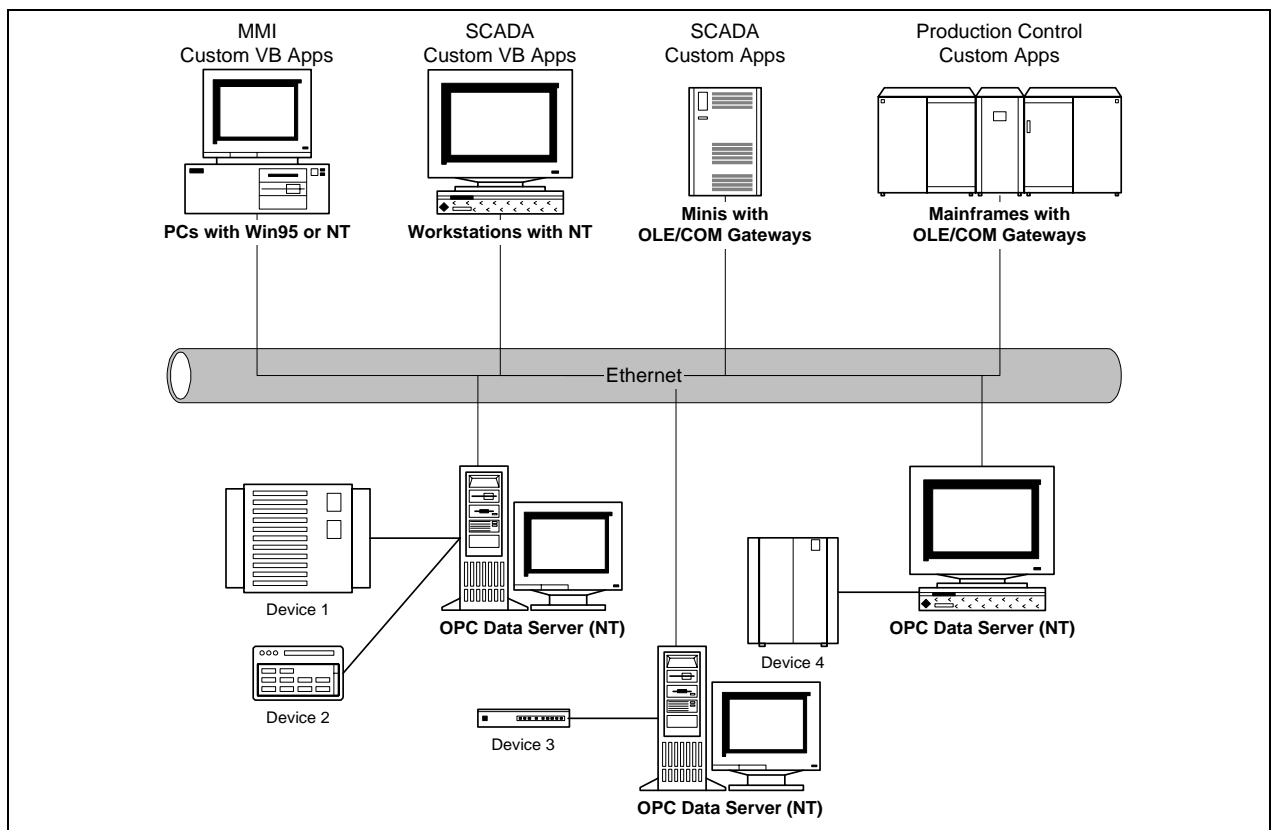


Figure 2: Heterogeneous Computing Environment based on OPC

Today, there are numerous existing applications which access plant floor data (MMIs, SCADA, SPC, etc.), written by various companies. These companies gain access to the data by independently developing drivers for their own packages. This has led to the following problems:

- ❑ Much duplication of effort: Everyone must write a driver for a particular vendor's hardware.
- ❑ Inconsistencies between vendors drivers: Hardware features not supported by all driver developers.
- ❑ Support for hardware feature changes: A change in the hardware's capabilities may break some drivers
- ❑ Access Conflicts: Two packages generally cannot access the same device at the same time since they each contain independent Drivers.

OPC will help to overcome these problems by providing components with specified interfaces, that allow data access from clients in a standard manner.

A growing number of custom programs are being developed in environments like Visual Basic (VB), Delphi, Power Builder, etc. OPC must take this trend into account. Microsoft understands this trend and designed OLE/COM to allow components (written in C and C++) to be utilised by a custom program (written in VB or Delphi). This is why OLE is an integral part of OPC. This model fits the needs of Process Control exactly. Developers will write software components in C and C++ to encapsulate the intricacies of accessing data from a device, so that business application developers can write code in VB that requests and utilises plant floor data. To achieve this, all the VB programmer needs to know is the interfaces to the OPC objects.

2.2. General OPC Architecture and Components

OPC interfaces can be potentially be used in many places in an application. They can be used at the lowest level to get raw data from the physical devices into a SCADA or DCS. They can also be used to get data from the SCADA or DCS system into the application (**Figure 3**). Using DCOM (Distributed COM) in the future, it will be possible to construct an OPC Server which allows a client application to access data from servers provided by many different OPC vendors and possibly running on different nodes via a single object.

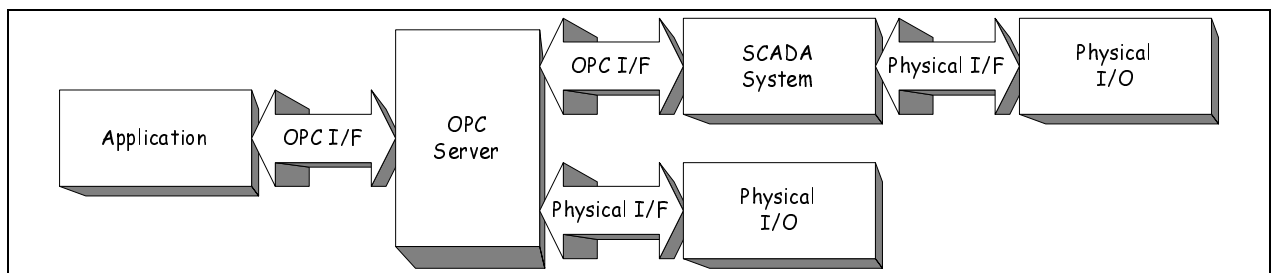


Figure 3: Example for OPC Clients and Servers with specialised interfaces (physical I/F) and common interfaces (OPC I/F)

The OPC specification defines two sets of interfaces - the OPC Custom Interfaces and the OPC Automation interfaces (**Figure 4**). The OPC Specification specifies COM interfaces, not the implementation of those interfaces. Like all COM implementations, the architecture of OPC is a client-server model where the OPC Server component provides an interface to the OPC objects and manages them. An OPC Server will generally be a local or remote program (EXE-file) which includes code that is responsible for data collection from a physical device.

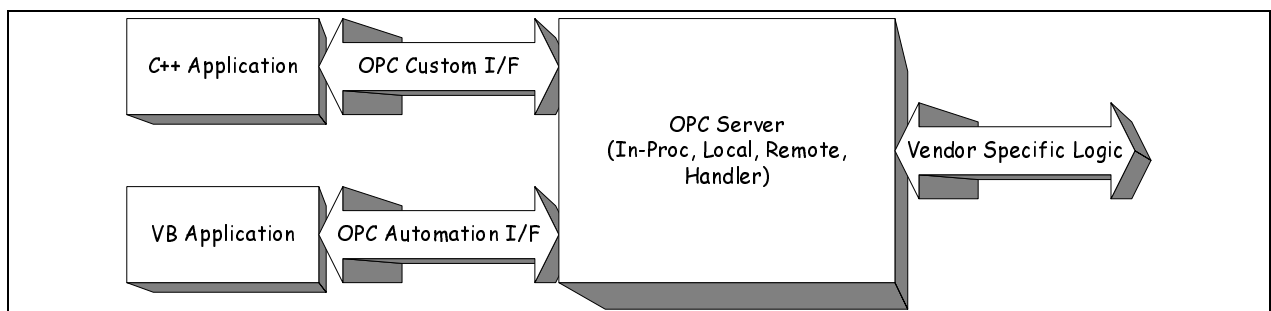


Figure 4: OPC Interfaces

2.3. The OPC Object Model

2.3.1. Server Object

OPC Servers are provided by different vendors (**Figure 5**). The code written by the vendor determines the devices and data to which each server has access, the way in which data items are named and the details about how the server physically accesses that data. The OPC Server object (**Figure 6**) provides a way to get data from a particular class of data sources. The number and types of sources available depend of the server implementation.

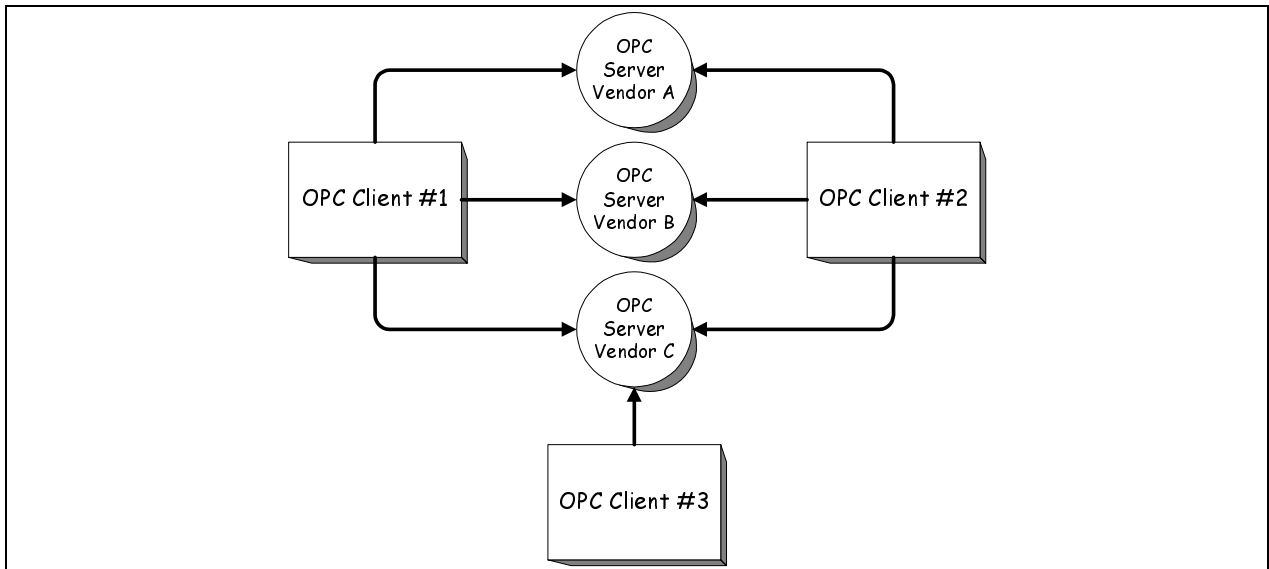


Figure 5: OPC Client/Server Relationship

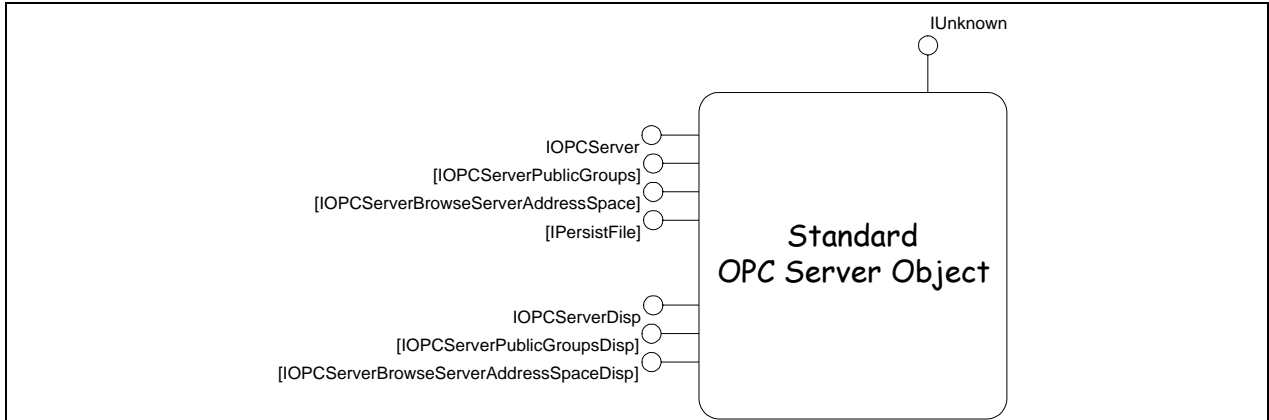


Figure 6: Standard OPC Server Object

2.3.2. Group Object

The OPC Servers provide the capability to define ‘groups’. These groups allow clients to organise the data they want to access. The group can be activated and deactivated as a unit. The group also provides a way for the client to ‘subscribe’ to the list of items so that it can be notified when they change. The OPC Groups (**Figure 7**) provide a way for clients to organise the data in which they are interested. Exception based connections can also be created between the client and the items in the group and can be enabled and disabled as needed. The time resolution of the data in the group can be specified.

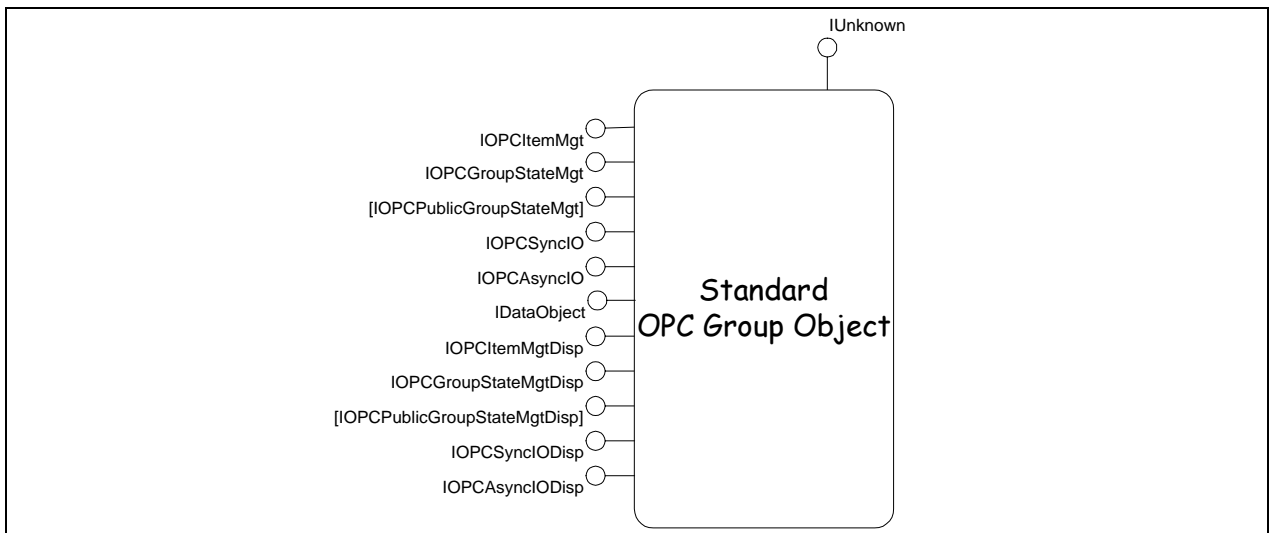


Figure 7: Standard OPC Group Object

2.3.3. Item Object

Within each Group the client can define one or more OPC Items (**Figure 8**). The OPC Items (**Figure 9**) represent connections to data sources within the server. Associated with each item is a value, a Quality Mask and a Time Stamp. The value is represented as a VARIANT. Those items are not the data sources - they are just connections to them.

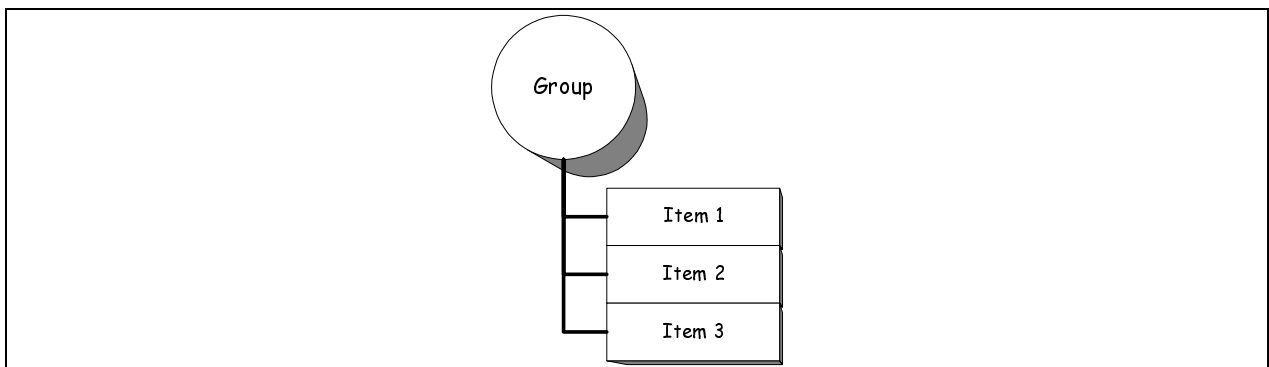


Figure 8: Group/Item Relationship

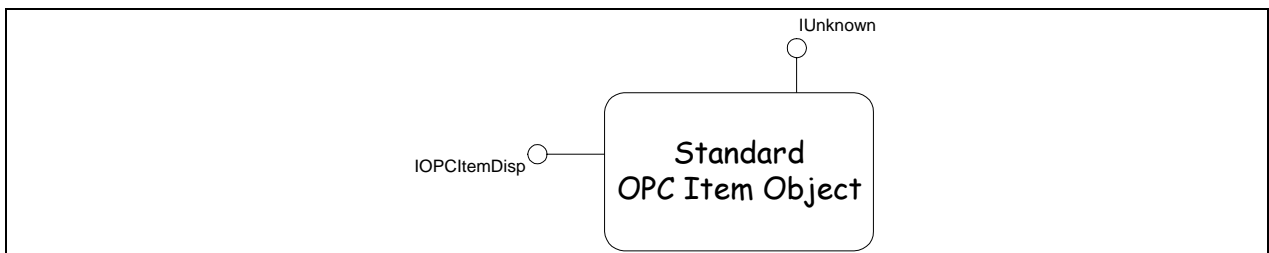


Figure 9: Standard OPC Item Object

3. VIGO and OPC

3.1. General Aspects

VIGO itself is an OLE-based system [4]. Data access using VIGO is performed by establishing connections to an OLE2-Automation interface. So an OPC server can use that VIGO automation interface to connect itself to the data sources for OPC items. The Manager Information Base (MIB) is the central point in VIGO containing description information regarding a specific pro-

ject. An OPC server should be an integral part of the MIB-based structure, in order to provide unique data exchange methods between the OPC server and any other tools (**Figure 10**) /5/. The tools can be used to build up projects, pre-defined initial values for an OPC server.

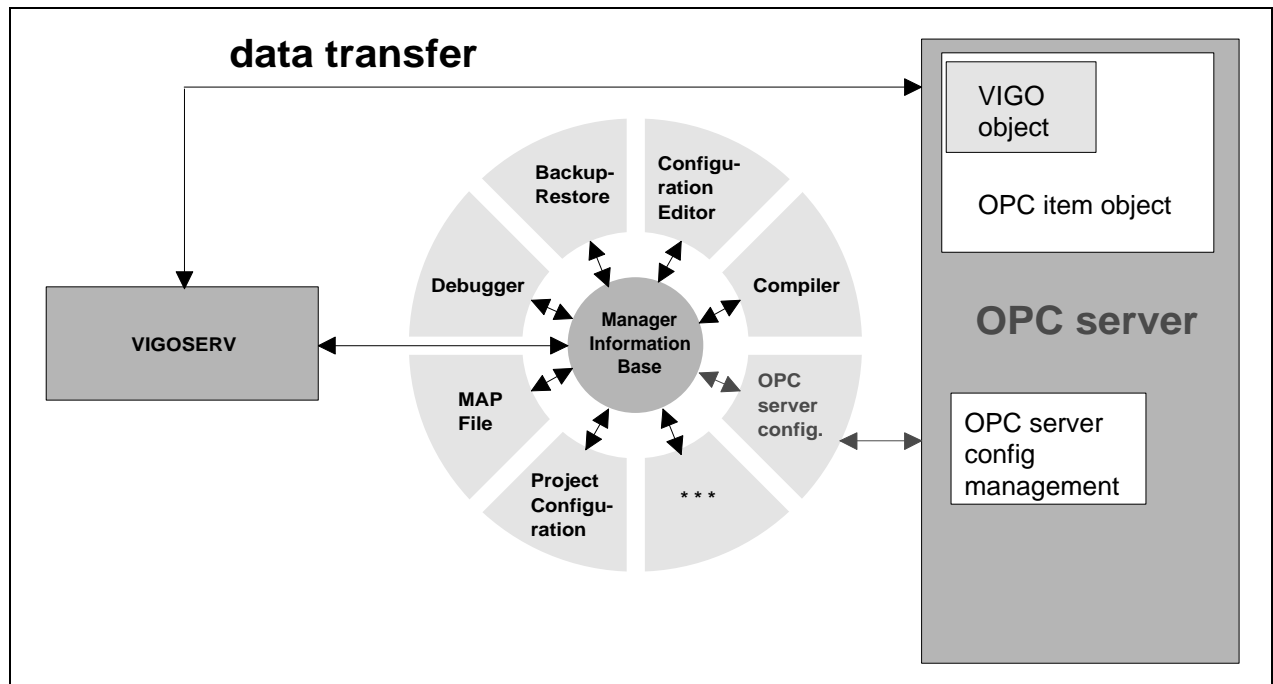


Figure 10: OPC server and its relations to the MIB

Using VIGO as the basic data access method, it acts as the physical interface from **Figure 3**. It would be possible to use generic interfaces to a P-NET system, but that would prevent the use of the integrative solution VIGO, providing unique interfaces for different communication systems. In addition, the efforts of RACKS and NOAH towards defining a fieldbus application programming interface /6/ may be integrated into an OPC implementation based on VIGO. Another interesting reason for the choice of VIGO as a physical interface is its internal object oriented structure.

3.2. Mappings, objects and interfaces

The implementation of an OPC server requires a mapping between an OPC object and VIGO objects. This mapping should be implemented based on VIGO's object model. The item object in OPC represents a variable in VIGO. Using the principles of OLE, an OPC item object may be implemented as a new object aggregating a VIGO object. The VIGO object performs data access and error handling, the item object implements the OPC interfaces (**Figure 11**).

If there is a request sent to the OPC interface, the data can be handled directly by the VIGO object, as they are transferred as a variant. The data descriptions for the OPC client (regarding type, structure etc.) are provided by the MIB, that is accessible through the VIGO object. So any OPC client may use the OPC interfaces without recognising the VIGO object itself. This is inheritance defined in the COM specification.

The assignment of an item object to a VIGO object can be controlled by passing the appropriate parameters to the VIGO object. The main parameter is the PhysId, which selects a VIGO object. As the creation of a VIGO object instance at runtime is possible, the OPC item objects may be created dynamically. The assignment of multiple OPC item objects to one physical VIGO object is possible, however, transparency and uniqueness of data has to be considered.

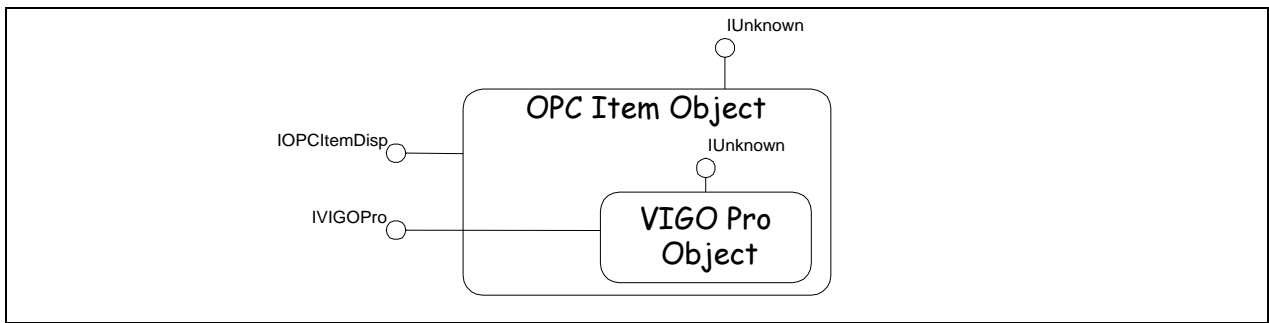


Figure 11: An OPC item object with an aggregated VIGO object

OPC group objects can be handled as arrays of VIGO objects aggregated within OPC item objects. There are no special group structures required. However, it follows the idea behind group objects, to map a channel to a group. A P-NET channel describes a complex measurement value, containing the data itself, error information and status, limits, scale and configuration information. Mapping the channel to a group, its data can be accessed at a glance. Of course, there are any other possible group definitions allowed. Special interest in group definitions may be recognised concerning synchronisation issues. It is possible, to define initial groups based on a MIB project, simply by creating an array of strings containing PhysId values for the single VIGO objects aggregated within OPC item objects. In addition, time parameters (update cycle, etc.) can be defined to be handled by the group object.

The OPC server object itself can contain information concerning the VIGO project, appropriate data access paths and so on. This information may be requested by the client. On start-up, the server object requests the start-up of VIGO, loading the actual project and connect VIGO to the net. For OPC, there is no difference between VIGO variables connected directly, via gateways or even via a VIGO-supported local area network.

4. Implementation

4.1. OPC Server

A test version of a VIGO OPC server was developed running on a 32-bit Windows platform. MS Visual C++ and Microsoft Foundation Classes (MFC) were used as development platform. The OPC specific parts are implemented following sample code provided by Intellution, one of the founding companies of OPC /7/. The server implementation is currently under test using OPC specific test environments. Those environments are provided for example by Fisher-Rosemount or Intellution and consist of OPC clients and analysers, which perform special tests of the implemented interfaces using defined operating conditions /8/.

The server is currently implemented as in-process and local server. That means, that it has to be implemented on the same computer, where an OPC client resides. In addition, this is a computer, that is connected to VIGO. However, since VIGO supports connections over a local area network, the computer has not to be the one with a P-NET card implemented. Future versions of the server implementation will be able to be used as remote servers. **Figure 12** gives an overview of appropriate scenarios.

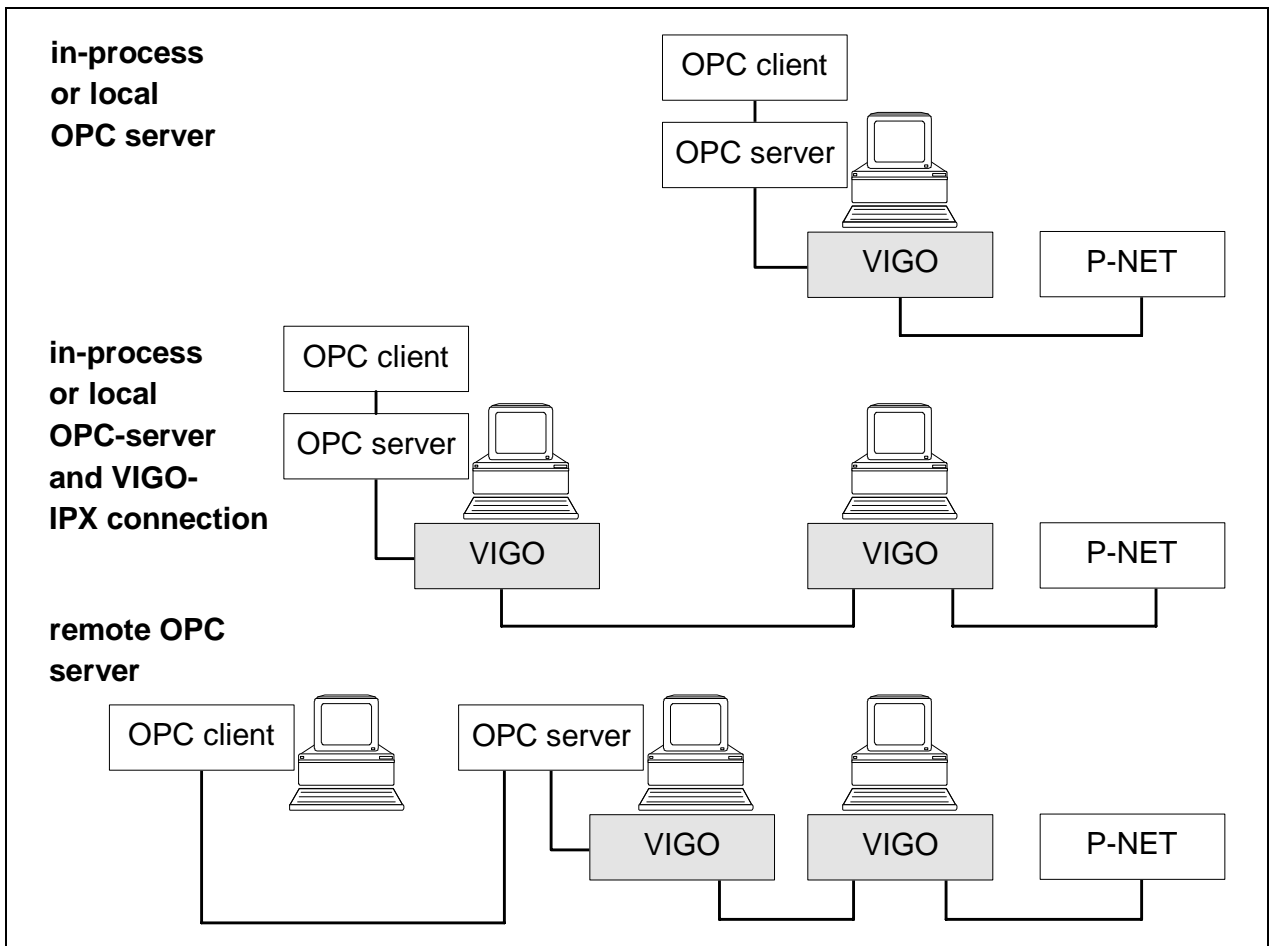


Figure 12: Scenarios for using the OPC server

The server specific objects with relations to VIGO are defined following the concepts described above. The item management interfaces are mapped to management functions of VIGO objects. The objects can be defined by an initialisation file at start-up or dynamically at runtime. Update information and other timing parameters for groups are handled by a special group object implementing the OPC interfaces and mapping them to arrays of VIGO objects. The aggregation of VIGO objects within the OPC objects provides direct access to the VIGO object interfaces for an OPC client, because the OPC object can redirect any access directly to the VIGO interface. In addition, existing clients that use the VIGO interface can use the VIGO interface of a VIGO based OPC item object in the same way as the original VIGO object.

For test purposes, the group and item structures of the current OPC server instance are displayed in the OPC server's window. In addition, menu items are inserted to manage the server object itself. Using these items, a configuration file can be read or the current configuration can be stored. A server reset may also be performed. As described later, the window provides a starting point for management tools. In order to allow debugging, the values read from VIGO may be displayed.

4.2. Management Tools

The purpose of the management tools for the OPC server is to define initial values for group and item management. An OPC "session" everytime depends on an active VIGO project. Therefore the OPC server must know the project name. The groups may be defined based on an active project. The OPC items themselves may be selected from the project and are saved within the OPC server's initial value file.

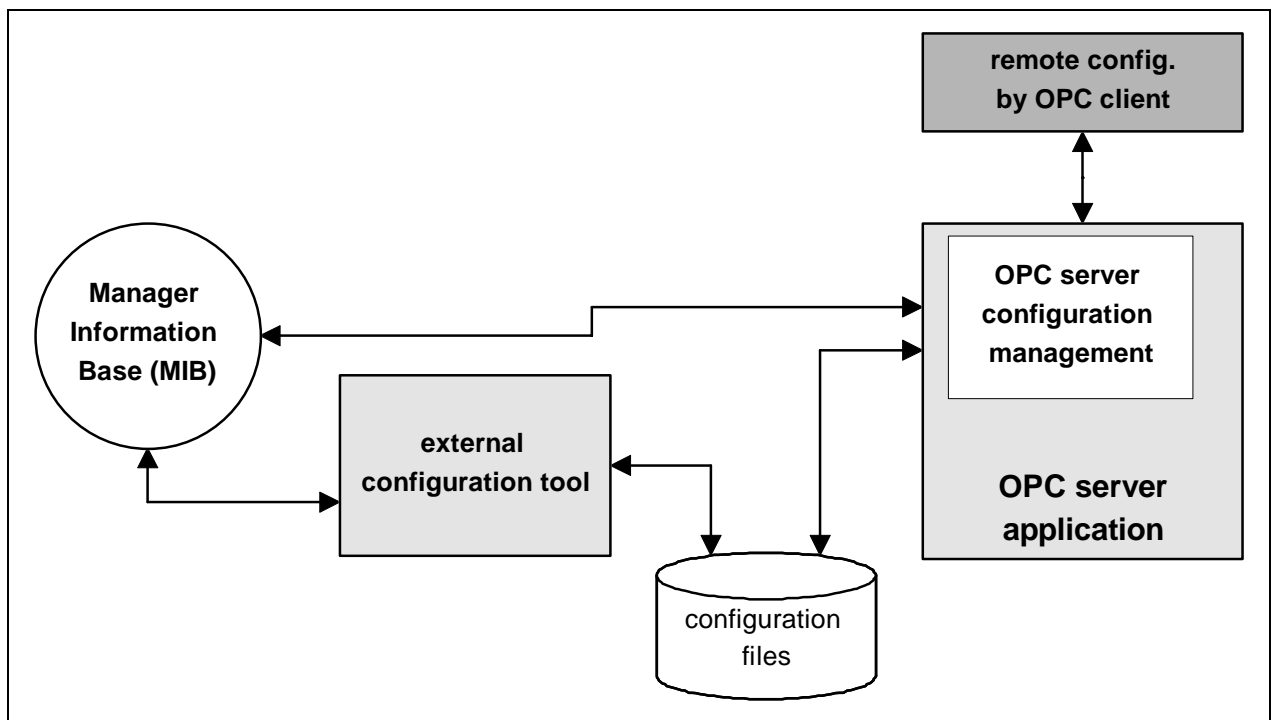


Figure 13: Implementation strategies for management tools

There are different strategies to implement the management tools (**Figure 13**). The first solution is to use the OPC server's application object. This object can use the functions provided by VIGO and can build up a connection to a MIBOCX Control. As that control refers to a project, it may be used to select the necessary PhysId strings, project name and so on. The application object has to arrange the values within its own data structure and may request additional values. The data may be stored by the application object itself. That strategy demands an on-line configuration, that means the OPC server has to run. So the configuration can only be performed on the computer with the OPC server implemented.

An off-line configuration tool may be integrated into VIGO using the "right mouse button" feature of VIGO (a context sensitive menu). Depending on the object highlighted in MibEdit, the configuration tool picks up the project name or a variable name (PhysId) and inserts it into the initial OPC values. When a project name is highlighted, it can be assigned to a current OPC project or a new one can be created. When a variable is highlighted, it can be assigned to existing OPC groups or to a new group. Group parameters (e.g. timing parameters) can also be defined. Using that implementation strategy, the configuration is independent from a running OPC server. In addition, the OPC server may be implemented without a visible window.

However, both strategies can be meshed together based on their identical file structure. Both implementation strategies show the efficiency of the COM based programming. No internal knowledge concerning VIGO is required, all is handled by objects provided by the developer of VIGO.

5. Further development

The current implementation shows the outstanding features of COM based programming. The existence of a VIGO object definition allows to insert an interface driven data exchange, without having to know internal details of VIGO. However, detailed tests have to be performed in order to specify timing relations. Based on those tests, profiling of the application has to be performed.

The introduction of DCOM to the server, that means the implementation of a remote server with the necessary proxy files is a project for the future. In addition, other VIGO-supported busses

than P-NET have to be considered. Special interest has to be paid to the result of the NOAH project. OPC should be included into the harmonisation of the fieldbus' applications.

Other interesting subjects are the integration of Intranet technologies and OPC, as well as the introduction of COM and DCOM to operating systems other than Windows. The expected OPC specification 2.0 will address DCOM related developments. ActiveX data objects are available to be included into container documents /9/, enabling those documents to integrate process control information and business data into plant-wide information systems.

6. Summary

The growing acceptance of OPC is an outstanding example for the success of the OLE/COM based programming technology in a wide area of applications. Especially in process control, where still a very conservative situation exists, OPC will introduce modern software development trends - with their benefits and their problems.

Using OPC techniques, the developers of process control software, SCADA and MMI packages can build their programs upon a standardised level of data acquisition and transport. It is expected, that the introduction of those interfaces will reduce the efforts in preparation of bus systems' use. The combination of OPC and projects like NOAH seems to be a big step into the direction of continuous, user-friendly, easy to manage automation solutions based on fieldbuses.

References

- /1/ Brockschmidt, K.:
Inside OLE.
Second Edition.
Microsoft Press, 1997.
- /2/ Chappel, D.:
Understanding ActiveX and OLE.
Microsoft Press, 1996.
- /3/ n.n.:
OLE for Process Control.
Industry Standard Specification, Version 1.0
OPC Task Force, August 29th 1996.
- /4/ Cramer, O.:
VIGO, a Fieldbus Management System.
4th International conference on the P-NET Fieldbus system.
Oporto, May 2nd -3rd 1996, proceedings.
- /5/ Cramer, O.:
VIGO TOOLS.
4th International conference on the P-NET Fieldbus system.
Oporto, May 2nd -3rd 1996, proceedings.
- /6/ Johansen, J.:
P-NET in the RACKS project.
4th International conference on the P-NET Fieldbus system.
Oporto, May 2nd -3rd 1996, proceedings.

- /7/ Chisholm, A.:
OPC Custom Server Test Program.
Intellution Inc., January 14th 1997.
- /8/ n.n.:
OPC toolkit.
Fisher-Rosemount Systems Inc., 1997.
- /9/ Chisholm, A.:
OPCDATA Control V1.0.
Intellution Inc., April 4th 1997.