Open Architectures to Improve Plant Visibility and Performances

by Integration Objects

Fulvio Roveta
Integration Objects Srl
Via Unità d'Italia 33 – Arenzano (GE)
Tel. 010 9113076 / 347 5839347
froveta@integ-objects.com

TABLE OF CONTENTS

TABLE OF FIGURES

About Integration Objects

Integration Objects is a leading provider of OPC software products for industrial automation solutions. Our OPC software products are employed in a variety of industrial environments, including process plants, pipelines, electrical grids, transportation infrastructures and utility equipment. Using the most advanced Microsoft development technologies, Integration Objects' products are highly scalable and specifically designed for security, reliability, and interoperability with the lowest overhead computing. Details about Integration Objects can be found on the web at www.integrationobjects.com or contact our sales team at sales@integrationobjects.com.

Summary

Manufacturing companies have invested large amounts of money in industrial automation infrastructures, but many have yet to realize the full value from their investment as many of these infrastructures remain islands of automation; information silos disconnected from other systems, including business systems.

Enormous amounts of data may be available, but very often companies lack the proper infrastructure that would allow them to adequately share data so they can use the information to improve process efficiency and profitability. Integration between process control, plant-level and enterprise systems is no longer an option for manufacturing and energy companies if they want to remain competitive.

Convergence of the IT world with automation and process controls greatly helps this effort by providing economically and technically competitive solutions. However, the sharing of process information within and between networks must be done in a way that aligns with the overall security policies of the enterprise.

This paper will discuss some of the options available for integrating automation islands and for distributing information at any level in the organization. In particular, two main players in the system integration arena will be examined: OLE for Process Control (OPC) and Service Oriented Architecture (SOA). Some drawbacks that such technologies may have will be mentioned and alternative solutions will be explored. Real world examples will be presented to show benefits provided by open architectures.

Coping With Automation Islands

“Island of automation” is a popular term used to describe the inability of automation systems to communicate easily with each other. Industrial communication protocols, network technologies, and system integration have been providing tools to improve this situation. However, manufacturing companies frequently have to cope with many different automation and control systems, each one using different proprietary standards. That makes information sharing a difficult task to accomplish. A rational response is to suggest that companies change the various components of their systems so that they are uniform, but for practical and economical reasons, it is rarely possible to standardize all the control systems to a single vendor or to the same model. Old systems have to coexist and cooperate with new ones and, especially in aged systems, the opening to the external world is not always easy.

Often the workaround for such barriers is represented by developing point-to-point solutions that satisfy urgent requirements but are frequently inefficient and also require high costs for the development and the maintenance of the new applications. Research firms Gartner Group and Forrester found that approximately 35 percent of IT budgets are spent building point-to-point interfaces.

An ideal solution for this issue would be an infrastructure allowing the posting of messages on the network from any point which would safely transport it to its disparate destinations. Integrating each automation island into a common infrastructure is not necessarily less expensive to implement than point-to-point, but this solution produces fewer interfaces and the maintenance and servicing is typically less expensive [1]. Additionally, if the common infrastructure is based on widespread standards, the whole system will benefit from several other advantages:

- **Robustness**: Standard interfacing solutions are widely tested, optimized, well known by a large developer base, and the off the shelf products are reliable and supplied by multiple vendors. Robustness of the whole architecture is improved while development and maintenance costs are reduced.

- **Flexibility**: The automation islands can keep on producing and reading data in their native format. The interface software will be in charge of translating proprietary data according to the selected standard and exposing them in a platform independent way. This means that changes in the automation islands will not impact the whole architecture, but will just require adjustments in the interface.

- **Communication**: data can be transported in real-time. The same version of data will be accessible at the same time for all users, even if they are in different locations. No polling mechanisms (to refresh data to be acquired) are requested. That reduces the data load on the network and lessens the number of real-time updates of the variables.

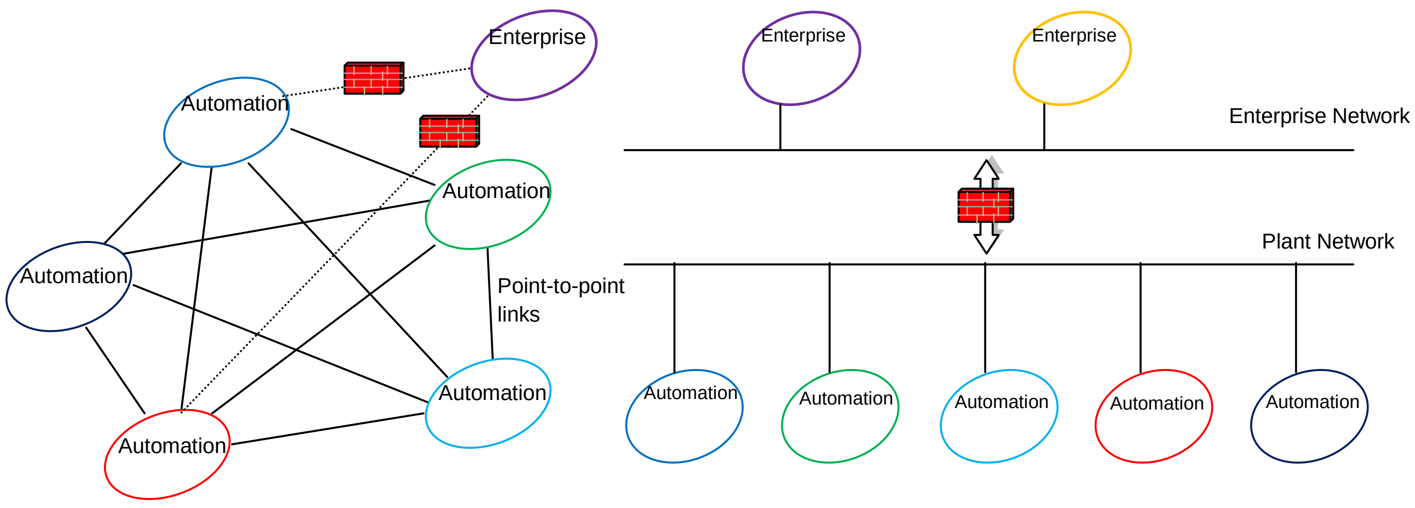


Figure 1: Point-to-point vs. infrastructure approach

The complexity of a large point-to-point integration is given by the following formula: for "n" interconnected systems, the total number of point-to-point interfaces equals n(n-1)/2. Following such a formula, we can assume that to fully interconnect 5 different automation islands, it will require 10 interfaces. Yet, when using the “infrastructure” approach, we would require just 5. Additionally, given the quadratic relationship, the gap will greatly increase with the increasing number of points.

It can be argued that the systems rarely need full interconnection, but even if you decide to integrate just a few systems, the common infrastructure approach will result in fewer interfaces. Sometimes those interfaces are complex to develop and the difference between costs and savings on the number of interfaces can reasonably cancel one another. A considerable help would be an industrial standard that could simplify system interconnections and reduce development time for new interfaces.

Other important savings come from maintenance. It is a fact that a large part of IT budgets is used to maintain automation islands which, according to some research, can reach up to 80% of IT budgets. By bringing all the information from automation islands into a common infrastructure based on standard communication protocols and software architectures, companies can redirect the funds typically reserved for maintenance expenses towards funding innovation.

Benefits from the open infrastructure model can be summarized as follows:

- Lower total cost of ownership for the open infrastructure compared to point-to-point solutions.
- Improved operational performance, system response, and load and processing times.
- Extended life of legacy systems because they can integrate new functionalities rather than being completely replaced when an important change is necessary. This also helps avoid costly investments required to replace IT systems.

- Can satisfy both current and future business requirements and anticipate business growth.

Events such as changes in the system requirements, upgrades of legacy systems, and company organizational changes (acquisitions, merges) will be easily managed thanks to the flexibility of the open infrastructure model, thus increasing the return on investment brought by such a model.

Several standards have been defined and many tools are available to build an open infrastructure. In this paper we examine two main players in the system integration arena: OPC and Service Oriented Architecture (SOA).

OPC is the most widely used real-time connectivity standard for the industrial automation and process control industries. Using a client-server architecture, it enables the sharing of real-time information, historical data, and alarms & events. OPC helps bring this data from the process level to the enterprise level, thus improving operational efficiency, lowering interface costs and reducing development time.

SOA is a major trend in the ICT industry and it is rapidly becoming the actual foundation for enterprise architectures. Its main strengths are the independence from the vendor, interoperability, and loose-coupling between service providers and consumers.

OLE for Process Control

Defined by the OPC Foundation, OPC™ (originally known as OLE for Process Control) is an open, non proprietary specific, connectivity standard used in industrial automation and IT systems. The OPC Foundation is dedicated to ensuring interoperability in automation by creating and maintaining open specifications that standardize the communication of acquired process data, alarm and event records, historical data, and batch data to multi-vendor enterprise systems and between production devices.

OPC Specifications define a standard communication method between field devices and generic software applications that is founded on Microsoft's OLE/DCOM technology. The specifications are based on a client-server model in which the OPC server provides an interface to the managed OPC objects while client applications use standard access methods to the OPC objects for controlling devices and managing data. For this reason OPC-compliant applications can communicate with any OPC server independently from its manufacturer and the type of information it represents.

OPC servers provide a way for many different software packages to access data from a process control device, such as a PLC or DCS. Traditionally, anytime a package needed access to data from a device, a custom interface, or driver, the program had to be specifically written. The purpose of OPC is to define a common interface that is written once and then reused by any business, SCADA, HMI, or custom software packages. Once an OPC server is written for a particular device, it can be reused by any application that is able to act as an OPC client. OPC servers use Microsoft's OLE technology (also known as the Component Object Model, or COM) to communicate with clients. COM technology provides a standard to exchange real-time information between software applications and process hardware.

Many industrial users have standardized on OPC as their backbone for all industrial connectivity solutions, thus enabling true interoperability between multi-vendor systems and devices. The deployment of OPC-based applications is easy when both clients and servers are installed on the same machine. It becomes somewhat more challenging when both systems are installed within the same process control network, but on two separate machines. Such configurations are possible and present little security risk when functions are limited. However, major challenges are faced with this setup when attempting to connect a process control network to an enterprise or business network(s). These challenges are primarily due to the fact that OPC specifications are natively based on COM/DCOM technologies which are difficult to implement and present security vulnerabilities [2]. Users' concerns for DCOM include problems with configuration, robustness, integrity of the data, and performance of the data transfer and are repeated themes heard from the industrial community.

We may classify issues related with DCOM use in industrial environments as:

- **Security**: DCOM is a Microsoft proprietary technology that was designed for general IT (back office) applications. Developed from a general IT perspective and not from real-time process control, the vulnerabilities of DCOM are well-understood, inside and outside the process control community. DCOM requires, for example, many ports for finding other hosts, resolving names, requesting services, authentication, sending data, and more. If these ports are not available, DCOM will automatically search for others. Any port and service used by DCOM is a target for viruses and worms. A capable hacker can create services directed at these ports that query which services are running and that match to the hacker's toolkit of exploit scripts. For example, Port 135 must be opened for the initial communication handshake, plus an additional range of ports whose numbers depend on the quantity of running processes hosting DCOM objects. The default port range is typically large, although it can generally be reduced. DCOM makes dynamic port allocation, i.e. it chooses random ports numbered beyond 1024 when opening connections. DCOM also cannot work across Network Address Translation (NAT). Callbacks are not carried out on the same port, but a new one is opened. This means that when a callback is set up, the client and server roles are reversed so that the firewall sees the server as a client trying to open a network communication to the outside—and this will likely fail in a default firewall configuration.

- **Robustness**: DCOM can take an unreasonably long time to fail an activation request as it tries each available network protocol (TCP, UDP, IPX, NP, etc.) in turn, until they all fail. Timeouts of 3 minutes are not uncommon. (The DCOM designers claim they chose robustness over performance, but that contention is unsupportable.) Few IT professionals will want to wait 3 minutes for a LAN connection to respond -- 3 seconds is more like it!

- **Set-up Complexity**: Most of the support calls that OPC product manufacturers receive are due to the DCOM set-up configuration challenges faced by end-users. We attribute this issue primarily to the complexity of setting up DCOM and Windows Security. Set-up configuration is relatively manageable when both OPC clients and servers are installed on the same machine, but the challenges start when both systems are installed remotely, configured as NT services, or running over a WAN. To perform such configurations, the user has to be an expert on Windows Security. The set-up can be so complex that it can take weeks for some users, even with a lot of assistance, to get the connectivity between the OPC Clients and Servers working.

New OPC-based software technology is now available to avoid DCOM challenges when seeking to send process data from one system to another, while ensuring fast, reliable, and secure OPC remote communications.

Service Oriented Architecture

The Service-Oriented Architecture (SOA) paradigm is a major trend in the Information and Communications Technologies (ICT) industry. Based on open standards and protocols and adopted by a large base of applications in the industry, SOA is changing the computing and communication landscapes. SOA is an enterprise-wide architectural approach that promotes vendor neutrality, interoperability, and loose-coupling between service providers and consumers. Thanks to such features, it is rapidly becoming the de-facto foundation for enterprise architectures and is also expanding towards the plant floor. The SOA paradigm is spreading in the embedded-device area, sensors and actuators, thus making it possible to build completely new architectures where the same communication paradigm can be used from the shop floor to the top management level.

We may define a service-oriented architecture (SOA) as a set of architectural principles for building systems that are autonomous, but still able to interoperate**.** This definition includes two key characteristics of SOA: *autonomy* and *interoperability*. The autonomy of systems implies that they can be created independently of each other, they operate independently of their environment and they provide self-contained functionality. The abstraction between service interface and service implementation supports interoperability among applications from diverse vendors and platforms. Applications know how the service should be queried and how it is expected to reply. Details about implementation are completely hidden and have no relevance. Autonomy and interoperability can be seen as contrasting properties, but thanks to its architectural principles, SOA is able to reconcile them. [3]

Interfaces for services are designed according to an outside-in approach. The interface is defined according to the exigencies that may arise in a business process context and isn't influenced by details of its implementation. Hence we may define SOA as a "business centric" rather than a "technology centric" paradigm. As a consequence of that, service interfaces are coarse-grained and based on the exchange of documents.

Communication entities are loosely coupled, since a service's functionality is defined in terms of contracts (a communications agreement, as defined collectively by one or more service description documents) and schemas describing the documents exchanged by the service in a standardized format, completely platform-agnostic. The service implementation isn't visible to external users and it can be changed without affecting the service's users.

Wide utilization of SOA cuts development costs since the re-use of services is facilitated and application programming is done at the highest possible level of abstraction. Scalability, manageability and maintainability are also greatly enhanced.

Some criticisms of SOA are based on the assumption that SOA applications may run slowly and may require more processing power, which would thus increase costs. Most implementations do incur these overheads, but SOA can be implemented using technologies which do not depend on remote procedure calls or translation through XML, which are considered to be the main causes of inefficiencies in SOA.

Another concern is that Web Services standards and products are still evolving (e.g., transaction, security), and SOA can thus introduce new risks to your system unless properly managed. Precautionary measures to avoid such security threats require additional room in the budget and contingency for additional proof of concept work.

With the advent of .NET and the .NET Framework, Microsoft introduced a set of new technologies in the form of Web Services and .NET Remoting [4, 5]. .NET Remoting and ASP.NET Web services are powerful technologies that provide a suitable framework for developing distributed applications.

The Web Services technology enables cross-platform integration by using HTTP, XML and SOAP for communication thereby enabling true business-to-business application integration, even across firewalls. Because Web services rely on industry standards to expose application functionalities on the Internet, they are independent of programming language, platform and devices.

Remoting is an enabler for application communication which addresses many of the objections described above. It is a generic system that different applications can use to communicate with one another. .NET objects are exposed to remote processes, thus allowing inter-process communication. The applications can be located on the same computer, different computers on the same network, or even computers across separate networks. This enables applications to take advantage of remote resources in a networked environment.

Both Web Services and .NET Remoting support the development of distributed applications and help enable application integration, but for any implementation it is necessary to consider their differences to decide which one best fits the requirements.

In the next paragraphs, case studies of industrial applications will show how such technologies can support information distribution and how they can be used to improve the communication of the OPC standard.

Automated Emergency Advisory System

The automated emergency advisory system described in this paragraph was implemented for a large oil producer to improve operations on the distribution network for refined products.

Main objectives of the project were to:

- Capture over 30 years of operational knowledge about refined products
- Provide online decision support for operators
- Standardize emergency actions

- Reduce the time cycle of problem resolution

The other main feature requested for the advisory system was the ability to integrate information coming from very different sources that were acting as independent automation islands:

- SCADA system's OPC servers (DA/HDA)
- Different relational databases and Plant Historians
- Expert system for fault diagnosis

Under normal conditions, the Advisory System is used to monitor the supply chain and produce charts and reports. As soon as symptoms of possible problems are identified, it notifies the users, which provides early detection of outage scenarios. Using the fault models available on the expert system and the operational knowledge stored internally, the Advisory System can identify the root causes of alarms and guide the operator towards the best solutions. Having operating procedures defined in the Advisory System grants consistency in operations and allows the application of best practices in all emergency situations, thus ensuring more efficient and safer operations.

The Advisory System is able to expose such knowledge and operational data through the network using a client-server architecture shown in **Figure 2**. The architecture is composed of 3 layers that are independent from each other:

- The data sources (the Expert System, the Plant Historian and the relational databases)
- The Knowledge .Net Server handles the communication with the data sources and hosts the operational knowledge used to monitor the pipelines and to suggest actions to the operators based on the expert knowledge that was built in the system
- The HMI applications are the client applications that display real time information to the end user

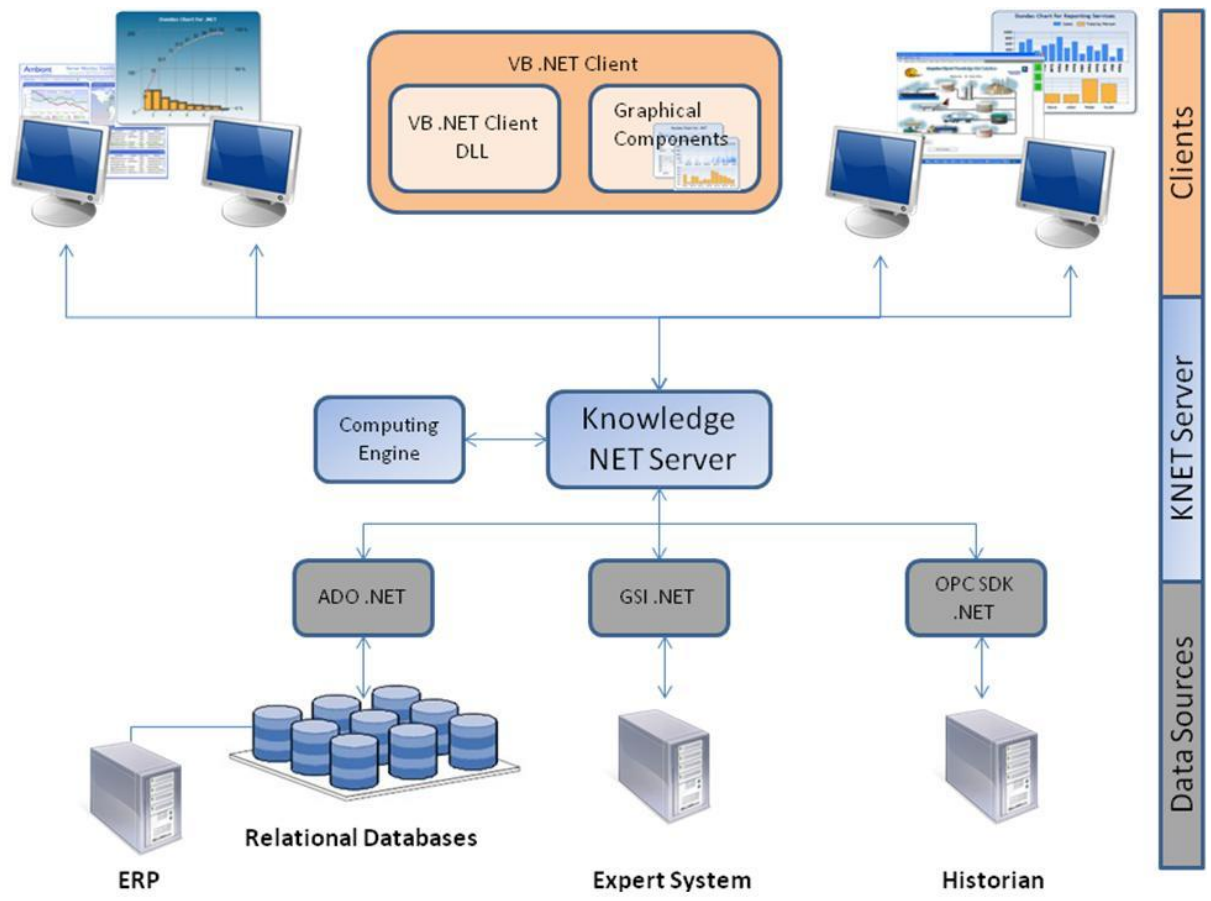


Figure 2: The architecture of the Automated Emergency Advisory System

The Client/Server architecture is based on .Net Remoting, one of the most advanced and recent Microsoft .Net technologies. Microsoft .Net Remoting provides a powerful and high performance way of working with remote objects- a perfect fit for accessing resources across the network.

In fact, Microsoft .Net Remoting provides a wide range of communication options. The Remoting features included in the Microsoft .Net Framework are built to provide distributed object services over the most common communication channels such as TCP and HTTP. However, if there are any custom communication requirements, the pluggable architecture enables users to create new protocols and to use them with the Remoting system.

For what concerns system security, Microsoft .Net Remoting, unlike COM/DCOM, limits the number of open communication ports so it won't have any conflict with the implemented security policies.

The communication between the KnowledgeNet Remoting Server and the Expert System is assured by a GSI .Net Wrapper. The interface for the Expert System is based on a proprietary protocol of the manufacturer (GSI). Using a .NET wrapper around the Expert System allows the exchange of data using standard methods, thus bypassing any proprietary solution.

The communication between the KnowledgeNet Remoting Server and the Plant Historian Server is possible through an OPC Client .Net Server Development Kit. The communication between the KnowledgeNet Server and the relational databases is done through Microsoft ADO.Net.

The implementation of the Advisory System allowed the oil producer to reach and sustain their original objectives. In particular, the ability to collect information coming from different and previously non-communicating systems and to integrate those systems with a common standard able to hide their internal implementation has allowed the building of a flexible data infrastructure. When using the 3-tier architecture, any minor or major modification in a given layer won't affect the other layers, which facilitates the maintainability of the application and provides flexibility to the solution. Accessing important data became an easy task which has enhanced grid supervision and the ability to apply the operational knowledge stored in the KnowledgeNet Server has ensured prompt and reliable decisions by the operators.

OPC-Based Infrastructure to Integrate Process Control

This project was realized to enable more rational and efficient communication and data sharing among 10 different DCSs and the plant historians. Communication among DCSs was originally achieved using point-to-point proprietary interfaces; specifically, 13 interfaces were running to allow data exchange. Additionally, another 10 point-to-point interfaces were installed to collect data from DCSs and to store the data in the plant historian. The resulting architecture is displayed in **Figure 3**.

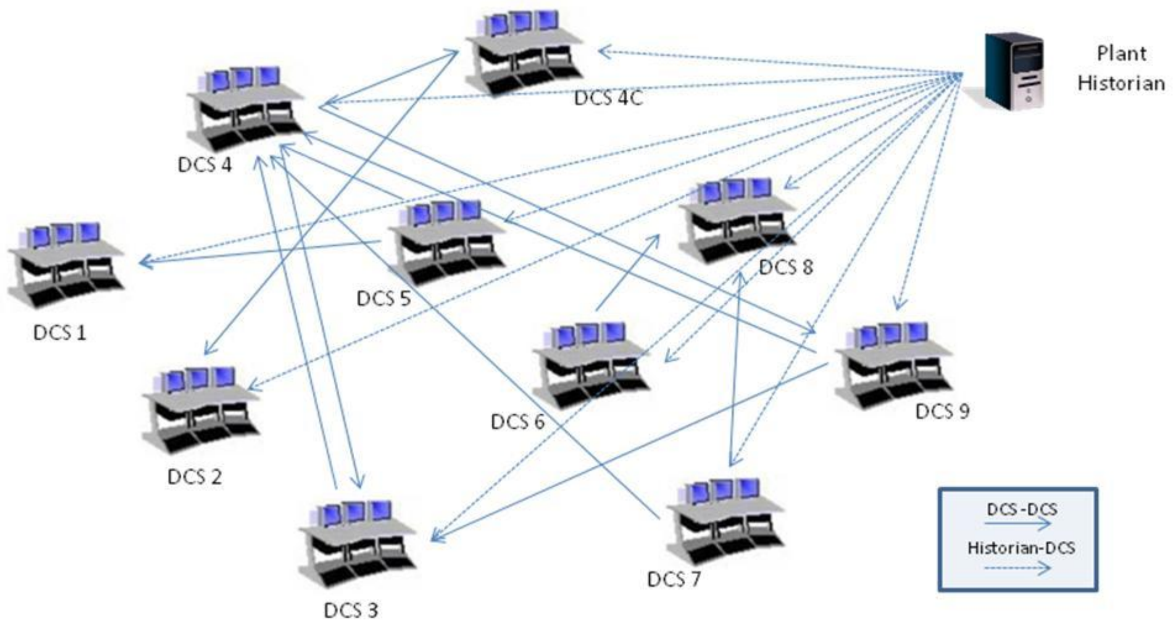


Figure 3: Original architecture of the site

Looking at the chaotic architecture, it is understandable that the implementation and maintenance of such a system was costly and that it was prone to communication failures. Additionally, changes in the DCSs (upgrades, replacements) would have impacted communication- potentially requiring new point-to-point interfaces.

It was decided to design an alternative solution to satisfy the following requirements:

- Uses industrial standard solutions

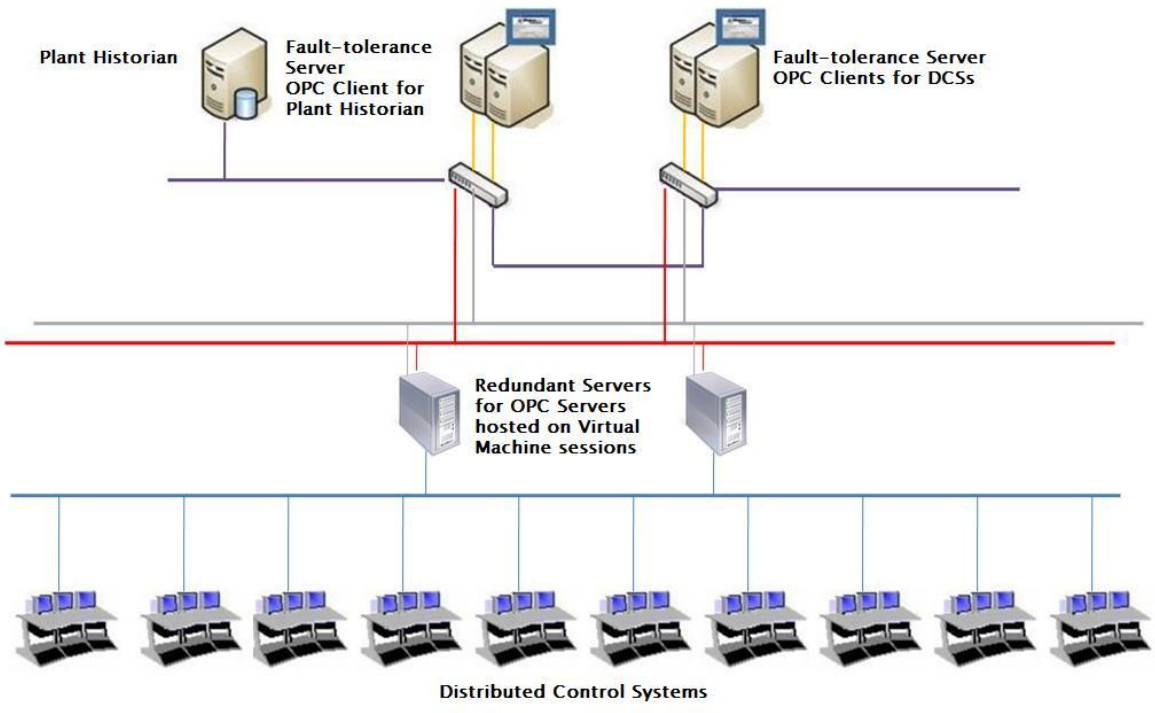
- Simplifies the system architecture
- Makes the architecture more flexible and able to follow changes in the plant control systems
- Improves reliability and robustness

OPC was identified as the standard solution able to simplify the whole system architecture. All data exchanges (DCS to DCS, DCS to Plant Historian) are now based on this standard. Each DCS exposes its variables on an OPC server provided by the DCS' manufacturer. Inter DCS data exchange is performed by OPC clients accessing data on the servers. The OPC client-server approach also makes the system more flexible for any future changes. Replacing a DCS with a different one simply requires installing the new OPC server and client, without having any impact on the other control systems or on the Plant Historian.

The choice of the communication standard was the first step in the project. The other main requirements were then considered. In order to simplify the system architecture, it was decided to install all the OPC servers on one machine and the same for all the OPC clients. This decision simplified the hardware architecture, but introduced some concerns about the reliability: what if one of the 10 OPC servers crashes and causes a shutdown or a deadlock for the whole Windows operating system? A redundant hardware configuration was installed by running a virtual infrastructure. Two virtual servers were deployed on a redundant configuration. Each server hosts 10 different virtual machines- one for each OPC Server. In total, 20 virtual machines are running the OPC Servers for the DCSs. These servers are configured as two units where the first acts as a primary and the second (geographically located at another site) as a backup. Each VM session runs all the necessary software applications needed to communicate with the DCS while also using the manufacturer's standards to expose the DCS' variables through OPC specifications.

In addition to the Virtual Machine servers, a Virtual Center Management console was installed on a remote desktop to allow remote configuration and setup of the virtual infrastructure. Such a solution was initially tested for 3 DCSs then was extended to all the control systems.

OPC clients used for inter-DCS communication and for collecting data for the Plant Historian were installed on a fault tolerant computer, providing full duplication of hardware and software. Redirection of clients to the appropriate server is automatically performed.



Plant Historian
Fault-tolerance Server OPC Client for Plant Historian
Fault-tolerance Server OPC Clients for DCSs
Redundant Servers for OPC Servers hosted on Virtual Machine sessions
Distributed Control Systems

Figure 4: Final architecture of the site

Another important subject to address in the architecture design was related to safety and efficiency of OPC communication. As already discussed in paragraph 2, meaningful concerns can be brought by the DCOM layer used by OPC. The new architecture displayed in **Figure 4** graphically showed the integrators how having 20 OPC servers and 11 clients distributed on 3 different networks could have imposed significant set up complexities and could have introduced major delays when trying to properly configure DCOM at any level. Additionally, weaknesses in DCOM security could have exposed the process network to external attacks. To solve the problem, an OPC tunneling package was installed. [6] The OPC tunneling software converts COM calls to .Net, which solves the concerns that COM/DCOM use can bring. To better understand how the tunneling software works, we can consider an example where an OPC client communicates to an OPC server. The OPC tunneling package is installed both on the server and client side and the two instances act as .Net peers and communicate through .Net Remoting. Their role is to redirect COM (and .Net) calls to .Net (to COM, respectively) calls.

The OPC Server and Client in **Figure 5** are isolated by a firewall. To communicate with each other, they need to select the same channel protocol and formatter. Formatters are the objects that are used to encode and serialize data into messages before they are transmitted over a channel. At the other end of the channel, when the messages are received, formatters decode and de-serialize the messages. Channel protocols include TCP and HTTP. Formatters include SOAP and binary.

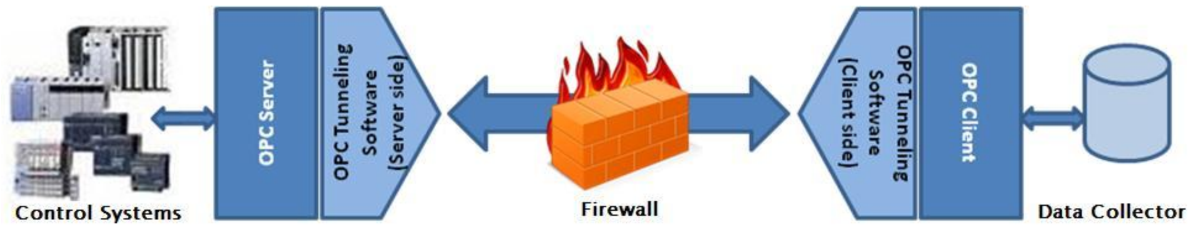

Control Systems
OPC Server
OPC Tunneling Software (Server side)
Firewall
OPC Tunneling Software (Client side)
OPC Client
Data Collector

Figure 5: OPC tunneling software

To access a server, the client just needs to specify the server IP address and the port on which the server will listen. It can get such information from a configuration file (XML file for example) and as said above, bidirectional communication on a unique port is possible. Here, the .Net Remoting architecture provides a way for applications on different machines/domains to communicate with each other and offers a powerful, yet easy way to communicate with objects in different app domains and to overcome domain controllers' security policies. Security barriers can be set using SSPI (authentication, etc.) or other custom developed modules.

Main benefits coming from the use of OPC tunneling software can be summarized as follows:

- **Security**: system security issues caused by DCOM weaknesses are solved
  - Communication across firewalls and Network Address Translators (NAT) through single ports to minimize security holes
  - User authentication at the server and tag levels for increased security
  - Data encryption for security reasons
  - Tracking of client/server communications
- **Robustness**: improved reliability and robustness
  - Configurable communication time outs
  - Automatic reconnection when the connection is lost
  - Data compression for better performance of data transfers across the network (from 2.5 to 10 times depending on the data type)
- **Set-up complexity**: communication issues caused by configuration errors are prevented and time is saved
  - Easy point and click configuration
  - Connecting OPC components from different domains, across VPN and through intranet and the internet (if required)

Conclusions

Integration between process control, plant-level and enterprise systems is no longer an option for industrial companies if they want to remain competitive, especially considering the current global context. Convergence of the IT world with automation and process controls greatly helps in this effort by providing economically and technically competitive solutions. There are different options available and it is important for companies to thoroughly understand their needs and objectives before making decisions.

This paper examined main concerns in system integration, with particular interest for the integration of real-time data with the enterprise level. Main trends have been described and some case studies demonstrate how a rational utilization of state-of-the-art software technologies can help improve operations and reduce budgets for project implementation and system maintenance.

Bibliography

[1] M. J. Schroeck: “Increasing ROI with Enterprise Application Integration”, DM Review Magazine, March 2000

[2] Digital Bond - British Columbia Institute - Byres Research, “OPC Security White Paper Series”, July 2007 -

[3] F. Jammes, H. Schmidt: “Service-Oriented Paradigms in Industrial Automation”, Parallel and Distributed Computing and Networks Conference, Innsbruck (Austria), pages 716-723, Februay 15-17, 2005

[4] M. Strawmyer: “.NET Remoting”, Developer.com

[5] T. Thangarathinam:”.NET Remoting versus Web Services”, Developer.com

[6] V.A., “OPC NetBroker^{TM} for Industrial Network Security and connectivity - Tunneling Process Data Securely Through Firewalls - A Solution To OPC - DCOM Connectivity”, Integration Objects White Paper, 2007