Oil & Gas Science and Technology – Rev. IFP, Vol. 60 (2005), No. 4, pp. 617-627

Dossier

Software Interoperability for Petroleum Applications
Interopérabilité logicielle : les applications dans l'industrie pétrolière

Interoperability and Integration of Industrial Software Tools

R. Matania[1]

1 Gensym SA, 31, boulevard des Bouvets, 92741 Nanterre Cedex - France
e-mail: rmatania@gensym.com

*Interoperability is the ability of two or more systems or components to exchange information and to use the information that has been exchanged**

* Institute of Electrical and Electronics Engineers. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York, NY: 1990.

Résumé — Interopérabilité et intégration en ligne de logiciels industriels — De nombreuses applications logicielles ont été installées dans l'industrie au cours des années quatre-vingt et au début des années quatre-vingt-dix.

Ces applications étaient généralement dédiées à une tâche spécifique et présentaient des capacités limitées d'interopérabilité. L'échange d'information entre applications était alors réalisé par transfert de fichiers ou par l'intermédiaire de passerelles spécifiques, solutions le plus souvent coûteuses et techniquement compliquées.

Alors que l'exigence d'interopérabilité devenait de plus en plus forte, l'échange d'informations pertinentes se trouvait donc considérablement freiné par ces architectures rigides conçues à l'origine pour des applications indépendantes. Ce constat a conduit à une remise en question des techniques employées afin de répondre à ce nouveau besoin.

Cet article présente dans un premier temps l'évolution des techniques d'interopérabilité jusqu'aux techniques les plus récentes d'échange de messages mises en œuvre dans le cadre du projet CHEM et propose ensuite une analyse d'*OPC (Object Linking and Embedding for Process Control*), le standard émergeant pour l'interopérabilité des applications industrielles.

Abstract — Interoperability and Integration of Industrial Software Tools *— During the eighties and early nineties, a multitude of software applications became available that addressed specific industry needs. These applications generally had very limited scope and operated independently. They were either unable to communicate with other applications or, in order to share data and resources, employed cumbersome and costly techniques such as file exchange or ad hoc application to application bridges.*

The consequences were that useful information would not always be available where it was most needed, information would be duplicated and the scalability of applications difficult and costly. This article firstly presents the trend from traditional methods of application interoperability to the more recent messaging techniques with the example of the recent European CHEM project. Secondly, it focuses on Object Linking and Embedding for Process Control (OPC) as one of the newer industry standards for communication.

INTRODUCTION

Traditionally, many companies selling industrial software tools and hardware, such as distributed control systems, SCADA (Supervisory Control And Data Acquisition) packages, programmable controllers, etc., provided external interfaces for interoperability but employed proprietary protocols requiring custom software development. This meant significant investments and maintenance overheads for those deploying these systems. As a result, end-user companies often became locked in to specific vendors due to the high costs and risks of switching.

1 TRENDS

With the proliferation of proprietary interfaces, integration of applications meant substantial development effort to provide adapter layers so that applications could operate collaboratively and share data, typically through a point-to-point communication architecture *(Fig. 1)*.

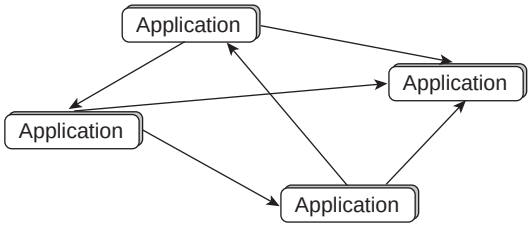

Figure 1

Point-to-point communication.

The advantage of a point-to-point architecture is that it is easy to understand for the developer and can be quickly implemented when there are only a few applications. However, as the number of applications grows, the quantity of connections quickly becomes overwhelming and leads to what is described as a "spaghetti architecture." With point-to-point integration, applications are generally "tightly coupled" to each other by adhering to application-specific interfaces. As a result, any change in a single application can easily have consequences on the other applications with which it communicates.

The evolution of the Internet has played a major role in the area of application interoperability by providing an infrastructure that links applications, businesses, and users. It has promoted the adoption of standards, such as XML for data representation, and the use of common software protocols, such as TCP/IP and SOAP (Simple Object Access Protocol). These advances combined with the adoption of industry standards, such as OPC, by major equipment and system vendors has given end-users the means to greatly simplify the sharing of data at all levels of an enterprise—from the production level up to the executive level.

While the Internet and the convergence towards a number of industry standards have simplified interoperability, a variety of architectures continue to exist for integration of applications.

Middleware-based integration *(Fig. 2)* has become an increasingly popular solution for interoperability of heterogeneous applications. It provides generic interfaces that send and receive messages between applications through a central message server that takes charge of routing the messages. Message-Oriented-Middleware (MOM) "loosely couples" applications *versus* "tightly coupled" point-to-point integration.

Employing middleware architectures cuts interapplication spaghetti and reduces application maintenance. Such architectures also enable the addition of new applications with minimum impact on existing ones. One tradeoff is the overhead associated with installing and maintaining the middleware itself.

The business information technology markets have been addressing the challenges of interoperability and application integration for many years through Enterprise Application Integration (EAI). The Internet has advanced the progression of EAI.

The industrial computing markets have been slower to adopt these recent technologies. However, with the increasing

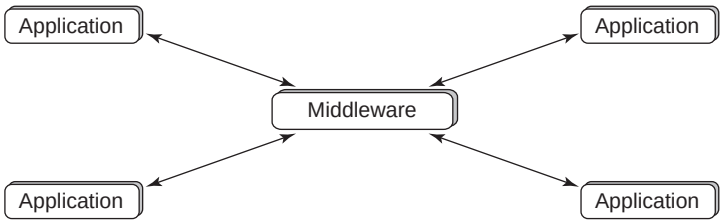

Figure 2

Middleware integration.

requirement for data to be available across all levels within an enterprise, the trend is strongly towards distributed architectures with increased interoperability among applications from multiple vendors.

CHEM was a three-year project funded by the European Community under the Competitive and Sustainable Growth program of the Fifth RTD Framework Program (1998-2000). The project demonstrated how the challenge of interoperability between industrial software systems can be solved through innovative techniques. The following chapters describe the principles of the CHEM architecture and the different software components built as part of the project to enable integration.

2 APPLICATION INTEROPERABILITY – THE EXAMPLE OF CHEM

The objective of the CHEM project has been to develop and implement a flexible advanced Decision Support System (DSS) for process monitoring, data and event analysis, and operation support in industrial processes. The resulting system is a synergistic integration of innovative and specialized software tools, such as process trend analysis, fault diagnosis, and reactive scheduling, designed to improve safety, product quality, and operational reliability as well as to reduce economic losses due to faulty process states within refining, chemical and petrochemical processes. A large part of the project scope was concentrated on integration aspects of the software tools in order to achieve the goal of system flexibility.

2.1 The Challenge

With more than twenty CHEM software tools to be integrated, together with the diversity of programming languages used to develop them (Java, C++, G2, Prolog, etc.), and the variety of data formats, a traditional point-to-point approach to integration was rejected for the reasons evoked above. In addition to the large amount of effort that would have been required to develop all the specific interfaces, subsequent modifications to tools, or the addition of new tools, would have impacted the interface adapters of existing applications and required significant development. Also, integration flexibility had added importance to this work because the various tools were initially in different stages of development ranging from designs and prototypes through to commercial industrial applications.

The strategy of CHEM integration was therefore to provide a common software infrastructure (Integration Platform) that would allow a wide variety of application tools to communicate and share data while minimizing the specific development for each tool. The methodology and technology choices were made to facilitate the integration of not only the existing CHEM tools, but also to facilitate adding applications in the future.

2.2 State-of-the-Art Choices

The decisions for the integration architecture of the CHEM project were strongly influenced by software industry standards, such as XML, MOM, CORBA and Java. These choices have proved judicious if one judges by the successful results obtained during the project. Demonstrations of the CHEM integrated solutions have been made on live industrial processes with encouraging results. While it should be noted that CHEM's aim was not to develop a full commercial application, coupling these standards with proven industry software greatly simplified the process of integrating the project's many and diverse software tools.

2.2.1 G2

The heart of the CHEM Integration platform is G2, a real-time, object-oriented development environment with powerful reasoning capabilities from *Gensym Corporation* (www.gensym.com). G2 is employed by major industrial organizations throughout the world in a wide variety of sectors, with an important presence in the chemical, oil, and gas industry.

G2 is a software environment for developing on-line mission-critical applications designed to operate 24 hours a day, 365 days a year. G2 is not only a powerful tool for building real-time decision-support applications such as those for alarm management, diagnosis, and high-level supervisory control, it is an ideal information federator. It readily provides the "glue" to enable heterogeneous applications to operate together and share data through powerful real-time interfacing and data management capabilities. Moreover, development with G2 is extremely fast and flexible, and encourages a rapid, incremental methodology for development. Buildling the Integration Platform in C++ would have significantly increased the required amount of effort.

It is noteworthy that G2 was also used as the development environment for many of the CHEM decision-support tools.

2.2.2 XML and the Information Model

XML is a powerful universal standard for representing data in a flexible format that facilitates interoperability between applications. Even though XML has primarily been designed as an improvement over HTML for Web-based applications, it has proved highly suitable for data exchange between heterogeneous applications, including the CHEM tools. This is primarily due to the fact that it allows applications to decode the semantics of data structures without the need to understand the meaning of the data.

```
<PV ID="Var-T-1004">
        <v>10.2934</v>
        <u>bars</u>
        <t>2002-07-30T13:20:00.000</t>
    </PV>
```

Figure 3

Example of a process variable in XML form.

XML allows specialized data exchange languages to be defined for a variety of domains and this feature was leveraged in the CHEM project to allow the tools to communicate in a unified manner. However, simply applying the XML standard to data structures does not solve all the problems related to exchanging data between applications. It was necessary to define a grammar that the CHEM tools would adhere to. As a result, a common information model was developed that would define a common vocabulary and terms that would be used for XML elements, tag names, etc. For example, the XML structure that represents a process data variable containing elements such as the current value, timestamp, sensor name, etc. *(see Fig. 3)*, needs to be defined in a unique way for all the CHEM tools.

The common information model provides these definitions. The XML format allows generic decoding of the data (parsing) for interpretation by the receiving applications.

2.2.3 Message Oriented Middleware

If a point-to-point communication architecture had been used to integrate the CHEM tools, apart from the development overheads stated above, the resulting implementation would have been a "spaghetti" of applications.

The recent trend towards the use of a "message bus" architecture, employing middleware techniques has the following advantages *versus* earlier integration techniques:

– The specific development effort required for each application to send and receive messages is greatly reduced compared to the adapters necessary for point-to-point communication.
– It allows incremental implementation in that a small set of XML messages can be defined when few applications are installed and then the model expanded as more applications are integrated.
– The functionalities provided by messaging architectures (point-to-point, publish/subscribe, etc.) allow adoption of flexible communication strategies.
– Message-oriented middlewares are particularly well adapted to exchanging XML format messages.

2.3 The CHEM Integration Architecture

Figure 4 gives an overall view of the CHEM architecture featuring the Integration Platform components, the specific software tools and the industrial process supplying real time data.

The following types of components or component groups can be identified.

2.3.1 Application Set

The application set corresponds to a complete integrated application comprising a number of specialized software tools, an interface to the industrial process and finally, a number of components provided in the Integration Platform. The tools assembled into an application set are typically chosen because they each fulfill a particular function and complement each other. They are designed to operate in a cooperative manner by exchanging data and commands so as to form a single coherent application that replies to a particular need.

2.3.2 Integration Platform

The Integration Platform, centered around G2, regroups the components that support the interoperability of tools. It essentially provides both data centralisation and communication services that allow tools to share data and communicate between themselves.

In addition to communication services, the integration platform contains a user interface module, a runtime coordination, and maintenance tools that contribute to integrated operation. These modules are described below.

Communications Manager

Communications Manager (CCOM) is one of the principal Integration Platform components. It allows the toolboxes and other components to communicate through the exchange of messages. CCOM provides a set of services for the emission and reception of messages. The data exchanged between components is in XML format but CCOM places no restrictions on the contents of messages. Its role is to assure the correct routing and delivery of messages CCOM verifies that messages are valid XML. The messaging approach to data communication allows more flexibility in respect to the overall application set architecture and places fewer constraints on the individual components. The integration platform is therefore more generic.

CCOM hides all aspects of transport protocols from the component that interfaces through a simple application interface (API). The CCOM API was built to support tools developed in C++, Java and G2 which covered the totality of the CHEM tools. CCOM was based on xmlBlaster, a message

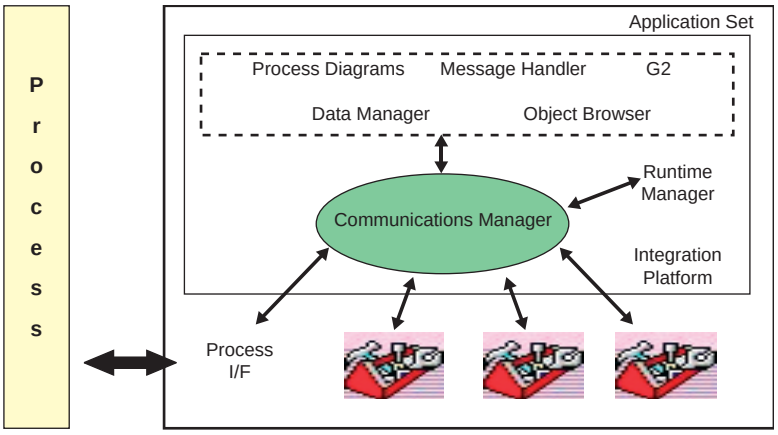

Figure 4

CHEM integration architecture.

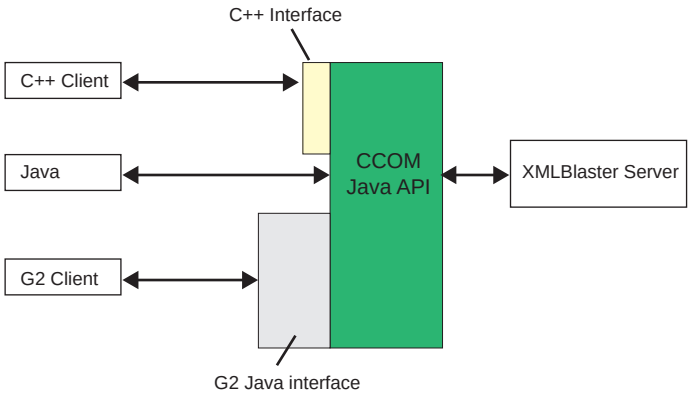

Figure 5

CCOM API.

oriented middleware server featuring both publish/subscribe and point-to-point message communication services.

In the CHEM implementation architecture, the CCOM API adds functionality to the basic messaging capabilities of xmlBlaster and further simplified the interface to the CHEM tools.

The CCOM API is in fact a layer above the xmlBlaster API, as shown in Figure 5.

The standard functions of xmlBlaster are:

– asynchronous messaging through publish/subscribe;
– asynchronous point to point messaging.

The CCOM API adds the following functionalities:

– XML message parsing to validate the contents of the messages for well formatted XML;
– adds synchronous point to point messaging so that the sender waits for a reply message from the receiver;
– manages timeout conditions for synchronous communications;
– provides an adapter layer to allow tools written in C++ and G2 to interface to the Java CCOM API;
– implements Remote Procedure Calls (RPC) through the synchronous point to point messaging so that a tool can invoke actions in other tools or for example, request information from another tool and wait for the information to be returned.

Communication with the xmlBlaster server was done using CORBA. The CHEM tool clients were however, totally isolated from the CORBA implementation through the use of a standard application programming interface (API) for each supported programming language. This meant that a Java based tool could send and receive messages with a C++ based tool without the need to understand the other tools implementation.

```
<CCOMEnvelope>
        <CCOMHeader Type="Data" Origin="TheSender" MustReply="yes">
        </CCOM Header>
                <CCOMBody>
                        Some valid XML Structure
                </CCOMBody>
</CCOM Envelope>
```

Figure 6

Example of a CCOM XML message.

The information contained within the XML message (see for example *Fig. 6*) provides all the information that is required for the tool to both process the message and provide any reply that my be required.

In addition to the specific data contained within the message body, the XML message header contains useful information such as the sender's name, whether a response is required, etc.

CCOM provides two ways for clients to communicate messages:

– point-to-point communication;
– publish/subscribe communication.

Point-to-point communication

In point-to-point communication, a client will addresses a message to another client by specifying the identifier of the destination client when it sends the message. In this case, the sender knows precisely to who the message should be delivered to. Throughout the rest of this document, the term SEND will refer to the emission of messages in point-to-point communication as opposed to PUBLISH that is described below.

Publish/subscribe communication

One of the most original and powerful features provided by the CHEM Integration Platform is the publish/subscribe communication.

In the publish/subscribe model a client calls a function in CCOM to publish a message and supplies a **Topic** name for that message. To receive messages that are published, other clients register interest for a particular message topic by informing CCOM that they are interested in that topic. When a client publishes a message for a given topic, CCOM dispatches copies of the message to all clients currently subscribed to that topic.

By using this method of communication, as opposed to point-to-point communication an application does not need to be aware of the applications that will receive the data messages. When a new application tool is added that is interested in receiving messages of given topic, there is no need to modify the application generating the messages since the new application will automatically receive them by subscription. Message dispatching is entirely managed by CCOM.

Data Manager

Data Manager (DTM) provides a simple centralized data repository that application tools can use to store and retrieve data (see *Fig. 7*). It is also the place where the real-time process data is held and made available to the application tools. In this way, applications do not need to be concerned with the specific details of the process interface itself.

DTM is a self-contained component built as a set of G2 modules that supplies this data storage and management function and is seen by the other components, in particular, the CHEM tools, as an application offering a number of services that are accessed through CCOM. In this way, it is possible for tools to make use of the publish/subscribe mechanisms of CCOM.

DTM allows applications to store data in objects known as "storage areas" which allows different types information to be grouped in the most appropriate ways. For instance, there could be at least one storage area for holding real-time process data and others for holding alarms and diagnostic information. In this way, data specific types of data can be easily be stored and made available to different applications that require it. Other applications can either specifically request data from given storage areas or subscribe to storage areas in order to automatically receive data periodically or when data is updated.

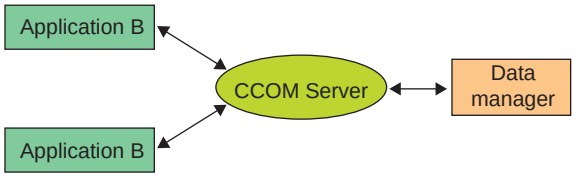

Figure 7

Service access through CCOM.

Common User Interface

The diversity of the application tools being integrated meant that the look and feel of their user interfaces was equally diverse. Although it was not feasible to impose the same UI specification for all the tools, there was a need to provide a common and relatively simple user interface that several application tools operating together in an integrated application could use to communicate with the end-user. A CHEM platform Common User Interface (CCUI), illustrated in Figure 8, was developed in G2 to provide application tools with a set of user interface components that they could use for interacting with end-users.

CCUI adopts the same philosophy as DTM in that it communicates with application tools through CCOM by providing a number of services as described below.

Message Browsers and Servers

The CHEM messaging system is based on the use of two types of objects *(Fig. 9)*:

– message servers;
– message browsers.

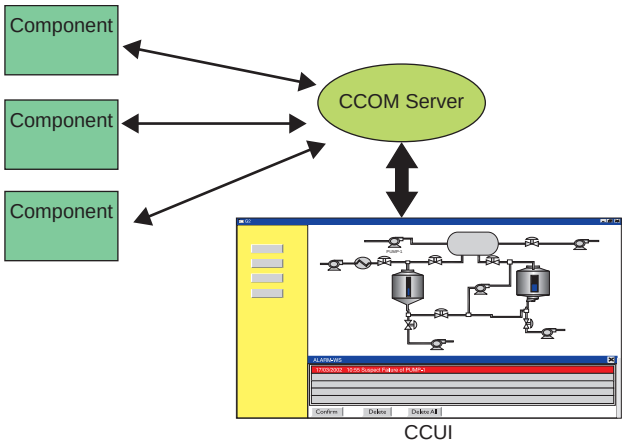

Figure 8

CHEM common user interface.

Message servers are objects that manage the message structures themselves. Application tools display text messages by sending them to message servers who store them and who manage the message status, etc.

Message browsers are windows that display messages managed by message servers. This structure is designed to separate the graphical UI management from that of the data management.

CCUI message browsers *(Fig. 10)* allow application tools to interact with users via text messages such as alarms or advisories that are displayed in scrollable lists. Users interact with the application tools through the messages by selecting and acknowledging them.

Message browsers are in fact clients of message servers and a browser can be associated with more than one server.

Applications communicate with message servers and users interact with message browsers.

When a user selects a message and then clicks on one of the buttons on the message browser window, the message server alerts the application that displayed the message so that it can take the appropriate actions.

Process Displays

This feature is a simplified version of the displays provided by DCS operator screens. It allows users to create graphical representation of the process *(Fig. 11)* by cloning process equipment icons, numerical displays for process variables etc from palettes. Application tools can animate the icons by sending messages to them.

Each process equipment icon is a graphical object made up of layers. Application tools can change the colors of these layers by sending XML messages to the process equipment objects. The message would typically contain the unique name of the icon, the layer to be changed and the new color.

In this way it is possible to indicate, for instance, that particular equipment has an alarm associated with it.

The other type of graphical objects is the Process Variable Displays (PVD). These objects allow the values of process variables stored in Data Manager to be displayed and updated automatically.

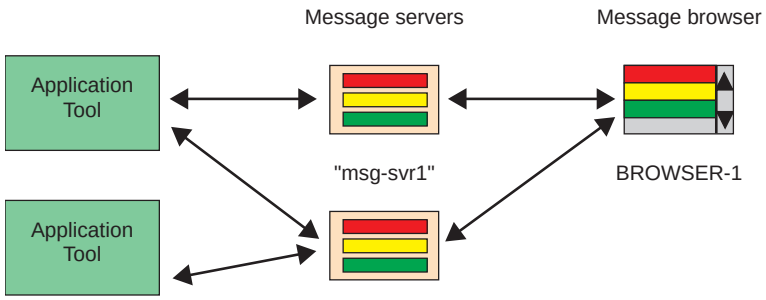

Figure 9

Example of message servers and browsers.

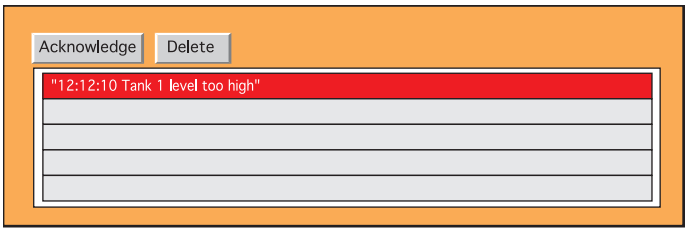

Figure 10

Message browser display.

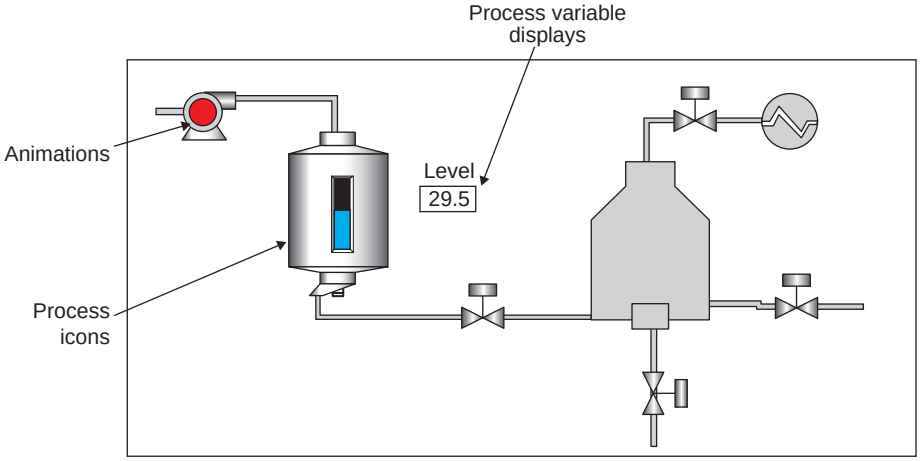

Figure 11

Process display.

A PVD would be configured with the name of the process variable. It then periodically requests the value of the variable from Data Manager.

These process displays are indeed much less complex than DCS screens but illustrate how object oriented development and messaging enable applications to provide simple animated displays. The advantage of this technique is that new graphical objects can be developed without requiring changes to existing applications.

Message Dialogs

Application tools can interact with the user through message dialogs *(Fig. 12)* that they can display in order to provide notifications or request user input. To do this, a tool would simply send an XML message to CCUI specifying which type of dialog to display and the text to be shown in the dialog.

When the end-user clicks on a button of the dialog the application that posted the dialog receives an XML message indicating the text that was entered and the name of the button (Yes, No OK, Cancel..) that was pressed.

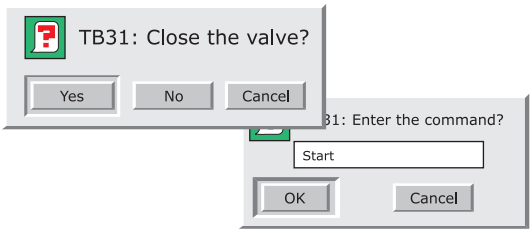

Figure 12

Message dialogs.

It is important to note that the different sub-components of CCUI (message management, Process Displays, Message Dialogs), in the same way as Data Manager, all operate by sending and receiving XML messages via the CCOM messaging middleware. Application tools only need to know the formats of the various messages to be able to use these components.

This is an example of how interoperability can be achieved simply and without requiring that the applications develop specific software interfaces for each type of component with which they communicate.

2.3.3 Process Interface

The role of the process interface is to connect the integrated application to the source of real-time process data, which in most cases will be some distributed control system. The process interface may be one of the standard software bridges commercialised by *Gensym* such as OPC or some process specific interface software. Most modern control systems offer an OPC server as the standard interface to external systems, thereby avoiding the need for custom proprietary interfaces. The process interface will obtain process data from the external system and send it via the Communications Manager, to the central data repository provided by Data Manager (DTM). In this way the application tools do not need to be individually interfaced to the process.

2.3.4 Software Tools

The application tools of the integrated application provide all the specialized data processing, diagnostics and alarm management. They communicate with each other and with the other components through the message services provided by the Communications Manager.

3 THE NEED FOR STANDARDS IN APPLICATION INTEROPERABILITY

To face the pressure of the competition and comply with the new environmental and safety regulations, chemical, oils and gas companies are increasingly investing in adding advanced technology from the instrumentation, automation and control levels, up to the enterprise resource planning (ERP) level. Unfortunately, it is uncommon for companies to have a very long-term vision of their future plant-wide configuration and comply with it. One reason for this is that technology advances and new regulations cannot be predicted and these have become some of the main factors influencing manufacturer's requirements for applications. As a result, applications, systems and devices are being added on ad-hoc or as needed basis.

The first question an end-user becomes concerned with every time a new application, system or devices is to be added, is whether it can integrate/interface with his existing configuration. Historically, most manufacturers of hardware and software solutions used a proprietary communication protocols or interfaces. As a result, in a many plants where a hybrid environment exists with many heterogeneous systems and devices, end-users were faced with the maintenance of multiple hardware and specific software bridges, with a very complex communication topology. Such configurations were also the reason for a high number point of failures, increased costs from the need to purchase and maintain the interface devices, and the challenges of the scalability of their environment. End-users became reticent to add new applications.

The following analogy of today's manufacturing systems can be made:

– Imagine a company with 100 employees where each one spoke a different language, and to be able to communicate with a colleague, relied on using a translator. Imagine how easy it would be if there was one common language, which was also a requirement for any new hires. Everyone's life would be made easier and the company would be much more effective.

In the above example of CHEM, it has been shown how industry standards and techniques such as XML and message oriented middleware were successfully applied to resolve the problem of application interoperability. *Microsoft* has been addressing the problem of simplifying communication between applications ever since the release of Windows with Dynamic Data Exchange (DDE) and subsequently Object Linking and Embedding (OLE).

In the early to mid 90s when Windows was becoming an acceptable alternative to UNIX for industrial applications, process control system and equipment vendors saw the chance to standardize interfaces between products.

A number of leading system and equipment manufacturers collaborated as a task force to produce the specification for a data acquisition standard based on Microsoft's OLE. This was the origin of the OLE for Process Control (OPC) standard which is continually evolving and is now accepted as the new standard for process data communication.

3.1 What is OPC?

OPC is a technology that enables interoperability between applications and provides a set of standards for developing enterprise-wide systems. OPC reduces the costs associated with the development and maintenance of hardware and software systems.

The OPC foundation was created from the original task force and has was created by the leading over 300 process controls hardware and software manufacturers including *Microsoft.* Their objective is to create and maintain the set of OPC specifications to facilitate ease of interoperability between systems, applications and devices. Almost all process control vendors in the world have adopted OPC. This has allowed them to not only eliminate their development cost of custom bridges, but also reduce the integration life cycle. The end-user now enjoys a much more maintainable and scalable environment, a wider range of vendor's choices. The OPC foundation's objective was simply to create out-of-the box plug-and-play connectible and interoperable applications without having to worry about complexity of each individual manufacturer's communication protocol.

The OPC standards are based on *Microsoft* OLE Component Object Model (COM) and Distributed Component Object Model (/DCOM) technologies and provide the foundation for developing OPC compliant software.

In order to address the specific needs of the process industry, the original OPC standards for process data acquisition has beem extended through the following sets of specifications:

– OPC-DA (data access for real-time data), OPC-HDA (historical data access), OPC-AE (alarms and events), OPC-XML, and OPC Batch. Further information can be found at www.opcfoundation.org

Figure 13 shows an example of an industrial software configuration where three different applications (Display, Trend and Reporting) need to communicate with several different devices or systems. Each device or system has its own specific protocol which means that the applications have to be able to "speak" several different languages.

Figure 14 shows how OPC provides a common language though which applications can communicate. Each device or application makes its data available by providing an OPC server. The Trend Application, for instance, only needs to make a connection to the appropriate OPC servers. If a new device or system is added, the applications at the top would no longer need to be modified in order to cater for the newcomer as they would just connect to the new server. The common language is OPC and the only requirement that the end-user needs to specify when purchasing additional hardware or software is that it must be OPC compliant.

Below are the most common OPC specifications adopted by software and hardware vendors:

– **OPC Data Access** (OPC DA) was the first OPC specification. It defines access to real time plant data such as current temperatures, flows, levels etc from the control system
– **OPC Historical Data Access** (OPC HDA) allows applications to read historical process data from data historian.
– **OPC Alarms and Events**, (OPC AE) is used for alarm and event notifications such as process alarms, operator actions, information messages, and tracking/auditing messages.
– **OPC Data eXchange** OPC DX, defines how OPC servers exchange data between each other.
– **OPC Extensible Markup Language** (OPC XML-DA) defines rules for representing process data in XML form.

3.2 OPC - A Continually Growing Number of Commercial Solutions

With the widespread adoption of OPC, companies such as *Gensym's* partner Integration Objects (www.integ-objects.com), have developed a number of commercial products to facilitate the ease of integration including:

– **Gensym G2-OPCLink** facilitates access real-time data to all process controls data. In the past, users had to purchase a bridge for every DCS vendors and every major PLC vendor such as Honeywell TDC 3000, Foxboro I/A, Allen Bradley, Modicon, GE, etc.

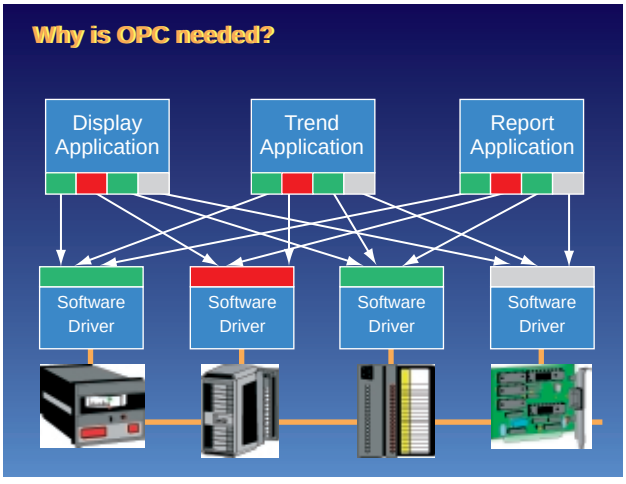

Figure 13

Communication through proprietary interface*.

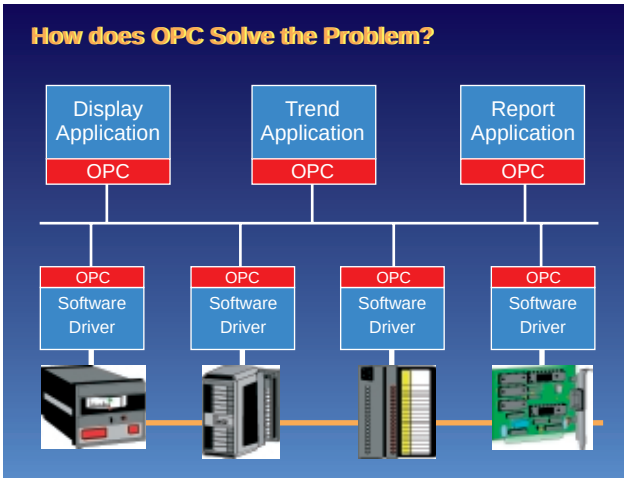

Figure 14

OPC based communication*.

* Figures 13 and 14 adapted from Miller (2003) presentation at IEEE IA Standards Workshop, Gaithersburg, MD, December 2003, http://ieeeia.org/iasw/Miller.pdf

– **Gensym G2-HDALink** provides connectivity to historians including OSIsoft PI, Honeywell PHD,..etc.
– **Integration Object's OPC Alarms and Events Archiver** allows data to be collected directly from the DCS, ESD, and PLCs and stored in a database such as SQL Server or Oracle for further analysis.

Other examples include plug and deploy OPC-ActiveX components that allow data to be gathered in any ActiveX container such as Microsoft Excel. Now, users can access process control data from their desktop without any development efforts.

The acceptance of OPC as an industry standard is no longer in doubt. Since the first OPC specification version 1.0 was released on August 29, 1996, the OPC foundation has continued to improve and increase it's scope and explosion of commercial application will secure it's future.

CONCLUSIONS

The use of standards and proven industrial technology coupled with message oriented middleware considerably eased the process of application tool integration. The inherent flexibility of the CHEM Integration Platform greatly reduced the development effort required for application tools to interface to each other. Application tool developers did not need to be concerned with issues such as network protocols, specific application interfaces, message routing, etc.

Applying the techniques of business application interoperability to process applications implies that many other issues must be addressed when considering interoperability especially those related to safety and performance.

ACKNOWLEDGEMENTS

The funding for the CHEM project (G1RD-CT-2001-00466) by the European Commission is acknowledged.

INFORMATION SOURCES

Hunter, D., Watt, A., Rafter, J., Duckett, J., Ayers, D., Chase, N., Fawcett, J., Gaven, T. and Patterson, B. (2004) *Beginning XML*, 3rd Edition, Wrox Press, ISBN: 0-7645-7077-3

OPC Foundation home page http://www.opcfoundation.org

Gensym Corporation home page http://www.gensym.com

Integration Objects home page http://www.integ-objects.com

XMLBlaster home page http://www.xmlblaster.org/

CHEM project home page http://www.chem-dss.org/

Final manuscript received in May 2005