in the historical database. OPC-HDA client applications retrieve historical data through the OPC-HDA server to display variable trends (Figure 1-13). The server filters and aggregates data based on the client request such that only required data is transmitted with minimum overhead. OPC-HDA goes a step further than ODBC, OLE_DB, ADO and others by defining an exact format for the data so that no custom programming is required by the user to access data from the historian. The OPC-HDA specification is plug-and-play.

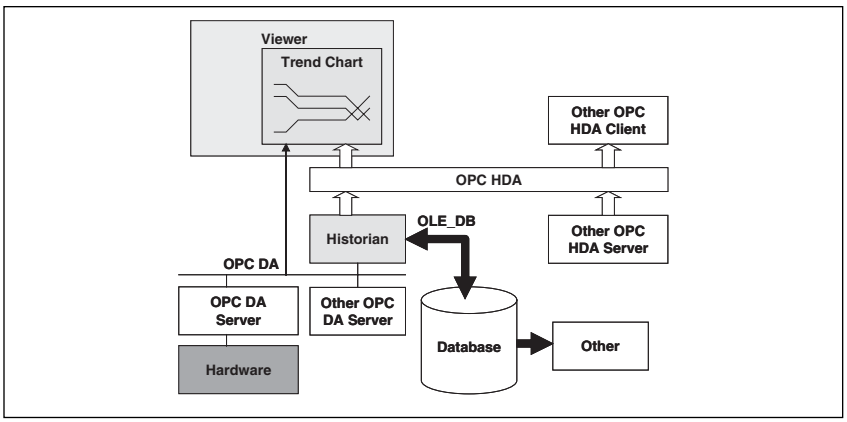

Figure 1-13. Example of OPC-HDA Architecture

Multiple OPC-HDA clients can simultaneously access the historical data from one or more OPC-HDA servers in the same computer or machines distributed on the network. An OPC-HDA trend viewer client plays back the data from many servers on the screen. Unlike proprietary systems where third-party applications cannot access the data or custom API programming is required, an OPC-HDA solution is completely open.

Organization of the internal database is different for every system. Nevertheless, what really matters is that the data can be easily retrieved. ODBC and OLE_DB require programming knowledge as well as documentation of the database format for access. OPC-HDA makes the retrieval easy, as it is possible to browse the information without knowledge of the databases or programming. OPC-HDA is highly specialized for the automation software domain and is not supported in execution and business-type applications.

Therefore, it is a good idea to choose a system that supports both OPC-HDA as well as ODBC and OLE_DB.

Live Data Gatewaying (OPC-DX)

Many different protocols exist on Ethernet, RS-232, RS-485, and other media. Application protocols above Ethernet and TCP/IP are incompatible and unaware of each other and therefore cannot exchange information. A plant that purchases different subsystems from diverse manufacturers may end up with islands of automation. Gateway hardware to convert between the different protocols is difficult if not impossible to come by, and requires a tremendous configuration effort if the amount of data is large. However, using the OPC-DX (Data eXchange) technology the plant will be able to link these subsystems together (Figure 1-14). OPC-DX thus works like a soft gateway similar to OPC-DA servers joined by a bridge application.

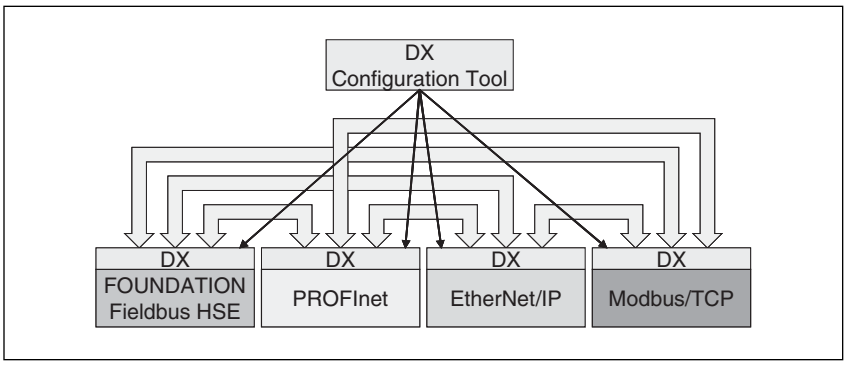

Figure 1-14. OPC-DX Software Gateway

OPC-DX is a direct server-to-server communication mechanism. Devices, networks or subsystems connect to OPC-DA servers. OPC-DA servers can in turn be connected to each other directly using OPC-DX interfaces communicating peer-to-peer, eliminating the need for intermediate bridge applications to transfer the data. The connections are established using a configuration tool that uses the universal OPC browser mechanism to locate the data source and sink in devices. For the most part, linking data in one device to another will now be a simple drag-and-drop operation. Safety and availability concerns for bridging data through a PC must be considered.

Since different protocols and devices represent data in different forms, it is often necessary to transform the data as it is transferred from one device to another.

Script and Macro Language (VBA)

Some supervisory control tasks require complex interaction of data input from the user and manipulation of data to and from the devices and databases, such as Excel, for user interface or even as part of the control strategy. VBA is a textual, programmatic scripting language that requires a little bit of programming knowledge, but not a "black belt." For simple conversion, only an equation or scaling should suffice and VBA should not have to be used for this. Similarly, a simple function such as opening another page or writing a value should not require VBA. For a complex task, a script or macro function is required. VBA is a macro script closely related to the Visual Basic (VB) programming language, but they are not the same. While VB is used to create applications and components, VBA is embedded in an application and used to customize the behavior of that application. OPC and VBA have lowered the barrier of software expertise required. Automation system engineers, as opposed to software programmers, can now build solutions. VB is beyond the scope of this book.

VBA is helpful for automation of complex functions that cannot be described through a single simple equation or action. This includes, for example, collecting data and displaying to the user a sorted list of options to chose from, transforming data into another format, putting data as a file in a folder where it can be accessed by other applications, and parsing files from other applications to extract data. VBA is thus a way to share data among applications that don't support OPC or OLE_DB, or when data, such as a report, is unsuitable for OPC transfer. This may be used for conversion between the different data formats and semantics different bus protocols use, such as the many flavors of Ethernet. Another example is when certain actions require verification and confirmation from the user. A sophisticated example may be for VBA to make all the necessary changes to change the control objective from "max throughput" to "minimum energy consumption." VBA may also be used in some batch control applications to read recipes from Excel, etc. You may use VBA for complex computations and looking up in tables (e.g., for product properties for use in compensation, and so on). Such functions can be date/time scheduled or triggered by the operator, OPC events and alarms, or other applications. Control strategy languages such as ladder diagram and function block diagram are not suitable for this. For some system integration requirements scripts are the only feasible solution. Apart from MS-Office

applications like Excel, Word, and Access, VBA is used in hundreds of other applications including most operator visualization and popular plant information systems. Applications from different manufacturers that host VBA can be integrated with each other using a single language, but it does require some programming. However, advanced system integrators will be able to do it. Applications that host VBA can be customized providing tailored solutions to very specific user needs for applications supporting OLE automation. By configuring properties, methods, and events, the applications can be made to respond to user actions such as opening or closing an application.

It is therefore a good idea to use a system host that has embedded VBA in the graphics application, not in a separate add-on application.

If VBA is not embedded in the application itself then it may become necessary to interface to other applications that do. This interface adds complexity, partly nullifying the benefits of a single language. VBA script is very much easier to use than full-fledged programming languages like C/C++/C# etc., and even VB itself, albeit a bit limited. In fact, even non-experts can write simple scripts using VBA.

For this reason, it may be a good idea to use applications that have VBA embedded, not as an add-on.

VBA is currently the most popular scripting language in automation software. Executing VBA is not supported on thin-client solutions for Web environment. Instead, Web pages use VBScript and JScript. VBScript and JScript are less powerful than VBA but can perform many tasks. If Web visualization will be done in the system, make sure it supports VBScript or JScript.

Components and Controls (OLE Automation, ActiveX)

Component-based applications make it possible to reuse software controls and components from other suppliers. Sometimes third-party controls and components are required to meet specific project needs. ActiveX is currently the most popular component technology in automation software, but is only supported in MS-Windows.

Components are modules of software that perform some function. A component can consist of many smaller components. Even large

software such as an OPC server or HMI are also components, albeit built from many small components.

In essence, there are three levels of sophistication for ActiveX objects:

- ActiveX code components
- ActiveX controls
- ActiveX OLE controls

ActiveX code components are the most basic components and thus the most difficult to use. They must be created through code and therefore require extensive programming knowledge. ActiveX components are more sophisticated, having an additional layer making them easier to use; they can (e.g., appear graphically and be dragged-and-dropped from the software's toolbox). ActiveX components do require some programming skills, but no more than what many system integrators can handle. ActiveX OLE controls are the most sophisticated components and therefore easiest to use, requiring little or no programming knowledge. Components supporting OLE (Object Linking and Embedding) can be linked or embedded without writing code. Thousands of ActiveX controls and components are available. For example, there are ActiveX controls such as document viewers and code components such as Modbus drivers and so on.

It is therefore a good idea to use system software that supports ActiveX code components, ActiveX controls, and ActiveX OLE controls.

ActiveX Code Components

ActiveX code components are hidden functions such as communication drivers for proprietary protocols that the operators do not see. ActiveX components exist for many common functions. Advanced system integrators can develop ActiveX components for specialized functions.

ActiveX Controls

Many suppliers provide general purpose controls such as slider, spin button, scrollbar, calendar, clock, hierarchical browser tree, toolbar, progress bar, status bar, list box, combo box, grid (spreadsheet/table), tabs, etc. Hundreds of third parties sell ActiveX controls tailored for process control, manufacturing automation, and building automation, such as pumps, conveyor

belts, motors, fans, tanks, compressors, condensers, heat exchangers, and stirrers, etc. (Figure 1-15).

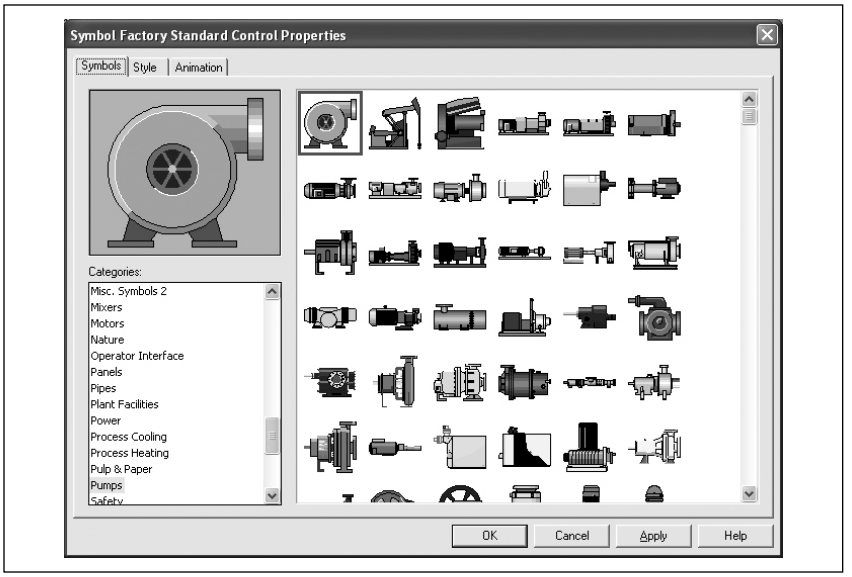

Figure 1-15. Third-Party Library of ActiveX Components (Software Toolbox Symbol Factory)

These days most operator visualization software products come with many ActiveX controls included. Others can be purchased from third parties, or even developed by advanced system implementers using provided tools. It is possible at design time to simply drag-and-drop ActiveX controls from a toolbox into container applications. Some VBA scripting is required to connect to the data source.

ActiveX OLE Components

Generally, operator visualization software is an "OLE container," meaning it can serve as an execution environment for OLE controls, which cannot execute on their own. OLE technology is widely supported by many software suppliers, making it relatively easy to put together components from many suppliers, in some cases without any programming knowledge at all. OLE automation is the ability to change properties of OLE components, such as their shape and color. OLE objects can be linked or embedded.

Embedding means the container application hosts an OLE component making the embedded object appear within the hosting

application, displaying the menu system and tools of the embedded application to the user. A classic example is editing an Excel spreadsheet from within a Word document or PowerPoint presentation. At the end of the editing all the data from the embedded application is stored in the container application. ActiveX OLE objects can be dragged or opened in any container application and edited just as in the native application. For example, a Word document or Excel spreadsheet can be opened within the operator visualization software, without having to start another application, and the menus and toolbars from the native application will be incorporated in the operator visualization application. This is ideal for laboratory data entry, batch submissions, and batch recipes (Figure 1-16).

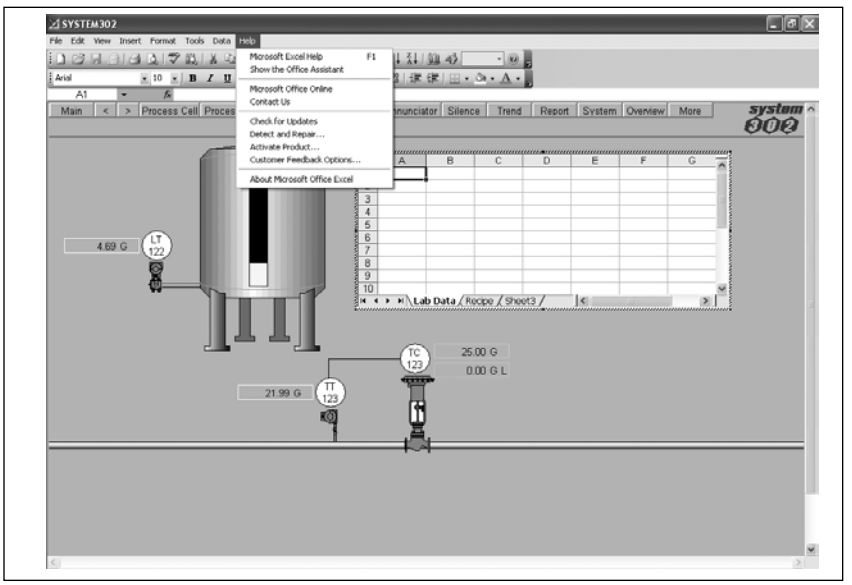

Figure 1-16. Embedded OLE Object Edited in Operator Visualization Displaying Native Menus (SMAR SYSTEM302)

Linking means that the applications are independent and the data resides in the source application, only transferring data from one to the other. Linking thus makes it possible to display the data in several applications at the same time. ActiveX in general, and OLE automation in particular, permits powerful integration of applications and customization of look and feel.

An OLE control with built-in OPC access, such as an animated gauge or a live trend viewer, can be dropped into a container and then get data from OPC without any programming at all (Figure 1-17).

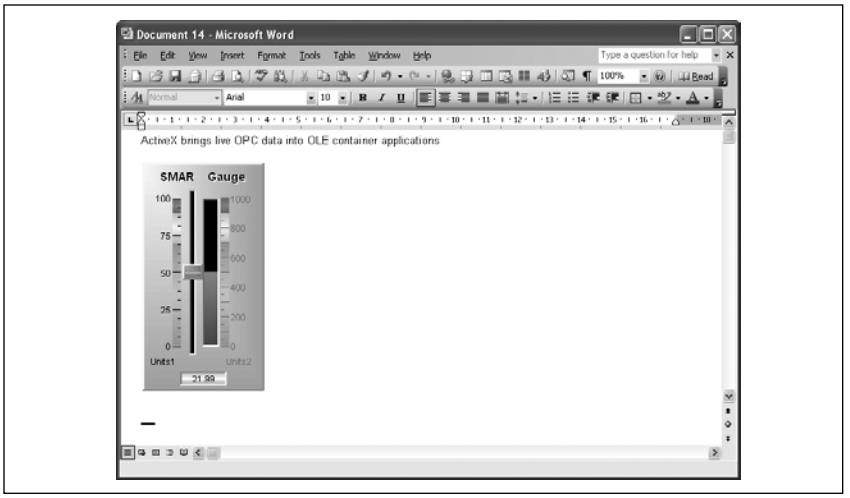

Figure 1-17. ActiveX Control Executes with Live Data in Any Container (Microsoft Word)

OLE controls are primarily used in the graphics screens in the operator visualization software, but many other applications also make use of ActiveX, and controls can be dropped into office applications like Word and Excel.

Web

The technologies described above are largely based on a client-server approach and operate in a single computer, or across different computers located on the same subnet of an automation system's control-level network. For information and business network integration across corporate firewalls onto the private Intranet and beyond, across the public Internet, it is necessary to use Web technologies designed for this purpose (Figure 1-18).

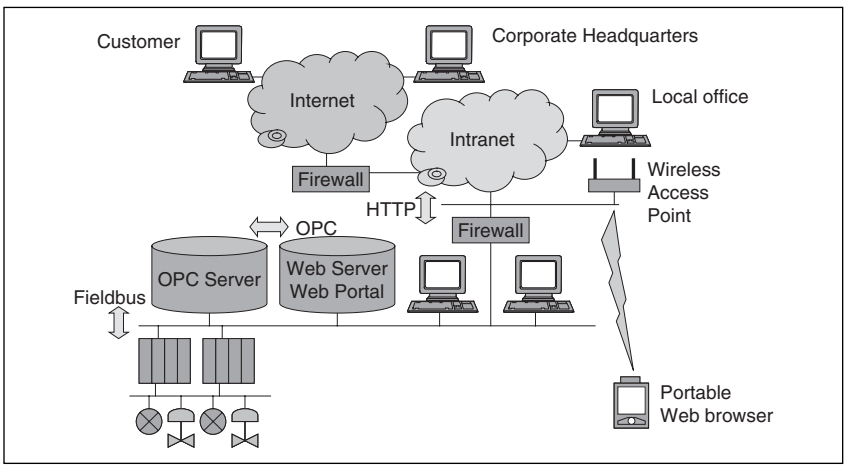

Figure 1-18. Integration of the Automation System Throughout the Enterprise

Fieldbuses, building automation network protocols and DCOM (e.g., OPC) do not communicate through a firewall unless its protection mechanisms are largely disabled. This would expose the plant to cyber intruders. Using Web technologies is one of the best ways to get data out of the automation system more securely without making it dangerously easy for others to get into the system. By presenting and transacting information as static or dynamically generated Web pages, it can be displayed and transported through routers and firewalls with lower risk, but not entirely without risk. Information can thus be disseminated throughout the enterprise into the execution and business domain using the corporate Intranet and the public Internet.

Applications with OPC and OLE_DB client interface collect selected information from the network and databases, process it, and present it in different formats for various users. Depending on the application, pages may be predefined asset management screens, customized reports or operator screens identical to the operator workstations. The information can be seen using a normal Web browser. All kinds of information can thus be delivered on a need-to-know basis.

As explained further below, it is a good idea to use a system that has an HTTP Web server and permits publishing data and reports in HTML format.

Web Transport (HTTP)

HTTP (HyperText Transfer Protocol) is currently the most common protocol for transporting Web pages. HTTP is firewall-friendly, meaning that it provides request and response through the firewall without the security of the firewall having to be completely compromised. HTTP uses a single firewall port, and because HTTP is “stateless” and easily recognizable, it is easier to protect. Thus HTTP makes it more difficult for hackers to attack both the Web server and the Web browser, but it isn’t impossible to penetrate. HTTP is a widely accepted standard supported in many products, not just for Web browsing. A Web server is software that runs on a workstation or dedicated server. A Web server called IIS comes built into Windows, but others are also available in the market.

The major advantage of HTTP is that it was originally designed for communication across the Internet. However, a slight disadvantage is that it is not particularly efficient. Therefore HTTP