Industrial Automation Using OLE

Dr. Bruce E. Segee, Kevin S. LeBlanc
University of Maine

Abstract

Often, industrial automation software is a single monolithic program that must handle all aspects of control, data gathering, architecture, and reporting. Design of such software is time consuming and error prone. Furthermore, maintenance or modifications to the code is difficult and can "break" other functions. A more powerful approach is to use the multiprocessing capabilities of Windows95™ along with the interprocess communication provided by Object-Linking and Embedding (OLE). Using OLE, the functionality of other Windows applications can be used as building blocks for your own applications without the overhead. For example, a container application and its user interface can be developed in Visual Basic. Microsoft Excel is ideal for organization and analysis of data, thus it could perform any "number crunching". Similarly, Microsoft Word provides the mechanisms necessary to edit, format, and print documents, so it would be ideal for report generation and printing. Often, in industrial automation, software must interface to hardware to monitor and control machinery. OLE lends itself nicely to this requirement through the use of OLE controls (OCXs) developed in Visual C++. Similar to device drivers, OLE controls can be easily customized to external hardware and utilized in software. In this fashion, hardware control is easily integrated into a Windows application. Furthermore, OLE is a great educational tool because it modularizes the application at hand. This allows a number of students of differing skill levels to take part in program development.

With using OLE as a software building tool, development, management, and enhancement of software is drastically simplified. Likewise, software is less prone to errors and less time is spent in development.

1. Introduction

This paper is organized into six sections. Section 2 briefly explains OLE, why one would use OLE over other Windows interprocess communication standards, and how to program using OLE. Specifically, linking and embedding, OLE automation and OLE controls are discussed. Section 3 gives an example using OLE in a generic automation project. Examples of using linking and embedding, as well as OLE automation, are related to Microsoft Word and Excel in an automation application. An example of where OLE controls can be used in development is also discussed. The second half of Section 3 gives a specific example using OLE in a simple intelligent system. Section 4 gives concluding remarks on industrial automation using OLE.

2. Object Linking and Embedding (OLE)

2.1 Overview

OLE is a mechanism that allows two software modules to connect and communicate with each other on the same machine. Specifically, OLE allows you to create complex documents by utilizing the resources of other Windows applications.

2.2 Why OLE?

The Windows operating system has many unrelated standards for interprocess communication. These include the Dynamic Linked Library (DLL), Dynamic Data Exchange (DDE), the Windows clipboard, the Windows API, and Visual Basic controls (VBX). Unlike these standards, OLE provides a unified, expandable, object-oriented communication protocol for the Windows 32-bit platform. [Kruglinski, 1996]

2.3 OLE Programming

The integration of OLE in a Windows application is easily achieved using Microsoft Visual C++ or Visual Basic. With these development packages there are two ways to take advantage of OLE: Linking and Embedding and OLE Automation.

2.3.1 Linking and Embedding

Linking and embedding is a common means of inserting objects into Word documents, Excel worksheets, and Power Point slides. Likewise, it can be used to insert objects into your own Windows application. In this manner, your Windows application becomes the host document.

An object is embedded in a host document if the document contains the object. That is, the object can be activated, edited, and saved within the document containing the object. If an object is linked in a host document, the document contains a reference to the data and image of the linked object but does not store the data. [Hergert, 1995]. Thus the document contains a view of the object's data which is stored in an external file.

2.3.2 OLE Automation

In OLE automation, an OLE server provides objects, properties, and methods for OLE clients to use at run time. By exposing its objects in this manner, OLE servers allow OLE clients to take complete control of the objects. Since these exposed objects are not in the clients code space, any number of clients can employ the servers objects at the same time. Both Visual Basic and Visual C++ are suited well for OLE automation programming. While both allow developers to browse OLE server objects for available properties, methods, and events, Visual Basic makes it relatively quick and easy to access and control exposed objects. An example of using OLE automation is one in which we want to access a cell in an Excel worksheet in Visual Basic:

Temp = SomeWorkbook.Worksheets("sheet1").Cells(0,0).Value

SomeWorkbook references an Excel workbook set by the developer and Temp is equal to the value in cell 0,0 of sheet 1 in SomeWorkbook. In this example, Excel is the OLE server exposing it's Workbook, Worksheets, and Cells objects as well as the Value property for the Cells object.

2.4 OLE Controls

Industrial automation often involves interfacing hardware with software for data acquisition and control but not all programming environments provide the mechanism to communicate with hardware. For instance, in Visual Basic there is no means of communicating with the hardware I/O port. This imbalance can easily be solved by developing an OLE control whose sole purpose

is to handle the communication with the hardware. This OLE control can then be seamlessly integrated not only into the current Visual Basic application, but also into future applications.

OLE controls (OCXs) are software modules that plug into the host document. Unlike ordinary Windows controls, OCXs have properties, C++ member functions called methods, and events, that replace the Windows notification messages. OCXs can easily be created using the Visual C++ OLE ControlWizard; the developer need only specify the OCXs properties, methods and events. Once developed, the OCX can be employed in a large number of programming environments. The end result is a software module capable of accessing and controlling hardware that can be easily integrated into a Windows application.

3. Examples

Automation in the industrial environment requires considerable planning. First the necessary data must be gathered. This may involve various hardware to provide the data such as pressure sensors or proximity sensors. Second, the data must be presented to the user in a meaningful way. Third, the data may have to be stored for future reference or report generation. In all, this may require a large software package involving several computers to accomplish the task.

3.1 General Example

An example of this arrangement could involve one computer controlling material flow, another collecting and archiving data, and yet another presenting the data for monitoring. Written in C, this could become a very large undertaking. With 32-bit Windows and OLE, such a task is greatly simplified.

Referring to Figure 1 below, data collection and material flow involving external hardware can easily be accessed and controlled using the modular OLE control while data storage and representation could be handled in Excel. If it is desired to view the information in an Excel worksheet, the worksheet can be linked or embedded in the application. Data in the worksheet can be edited using OLE automation or by activating the worksheet within the application. OLE automation also plays a role when information in the host document depends on data in an Excel worksheet. Alarms for example could depend on certain data points in the worksheet. When a

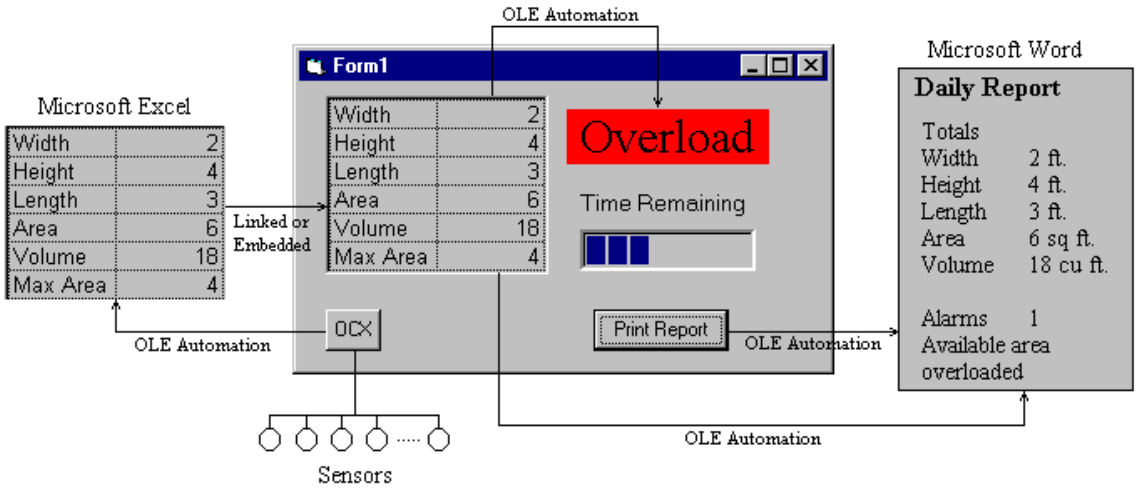

Figure 1.

threshold is exceeded in the worksheet, an alarm goes off in the host document. Report generation and printing could be handled in Microsoft Word. Word Basic, the macro language for Microsoft Word, provides a set of methods accessible through OLE automation to format paragraphs, change fonts, insert objects, find and replace, and print documents. The host document can communicate with Word in this way to fill out forms, tags, or reports with data stored and collected in Excel and send them to the printer. The user interface of the host document can be developed in Visual Basic or Visual C++. Visual Basic allows for quick and easy development of powerful Windows applications and thus is preferred. Although software development packages like Visual Basic are used to develop the host document, OLE is the underlying glue of the entire application.

OLE also allows people with differing levels of expertise to participate in the development of these Windows applications. For instance, a person familiar with Word or Excel but no programming experience, could create a report file to their preference, a beginner Windows programmer could fill in the report file with pertinent data and develop a user interface, and an experienced Windows programmer could write an OLE control to communicate with an I/O card. In this way, a non-programmer can change the functionality of the application by changing a Word document and the programmers can develop the application without knowing the format of the Word document.

3.2 Specific Example

A current project at the University of Maine Instrumentation Lab uses OLE in a closed loop control system. The project itself is a levitation controller designed for teaching about intelligent systems. The idea is to float an object at a constant height by pulse width modulating an air blower. The control system developed to accomplish this is shown in Figure 2 below. The object floats between the array of infrared sensors. The infrared pair that trips, which determines the objects height, is monitored by the embedded controller and sent via RS-232 to the host document. The host document communicates with a controller implemented in Excel using OLE automation. Excel takes this height and calculates the error and corresponding controller gain. These values are passed back to the host document through OLE automation to determine how much to increase or decrease the power delivered to the blower. The host document then talks to the embedded controller through an RS-232 OCX to control the blower. Features of the control

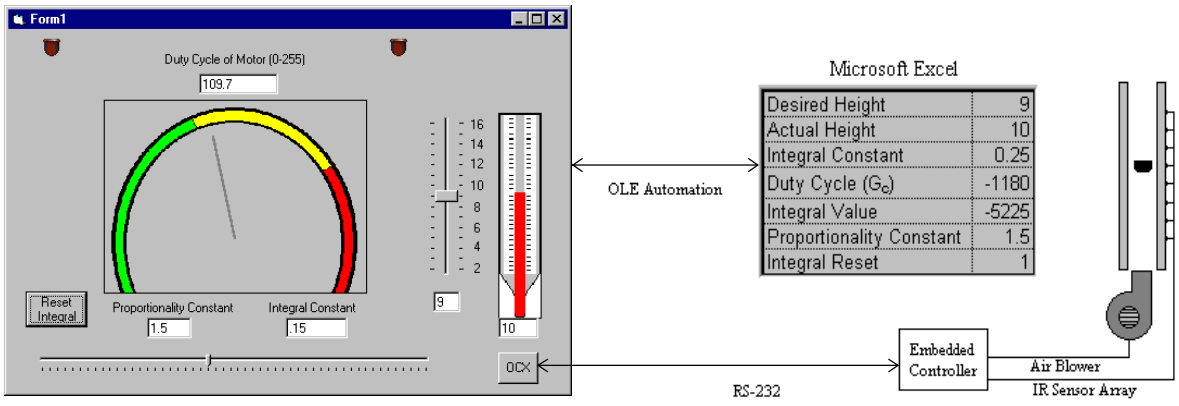

Figure 2.

system include a visual representation of the power delivered to the blower and the current height of the object. Through OLE, the user has the capability to change the desired height of the object and gain parameters. The most notable feature of this system however, is the ability to easily change the controller specified in Excel. Since Excel is designed for "number crunching", it is ideal for implementing controllers. Because of OLE, the host document can interface with these controllers in Excel with little effort. The result is a powerful yet simple control system.

4. Conclusions

A typical automation project involves data collection, data monitoring, and control. C++ code could be written to achieve this but much time would be spent in development and in debugging. Windows and Windows95 in particular, are ideal environments for data presentation and user interfaces. Development software such as Visual Basic and Visual C++ allow programmers to create effective front-end applications. Applications such as Excel are excellent for number manipulations, chart generation, and spreadsheet operations and Microsoft Word provides document formatting and printing capabilities. Also, many students and professors are familiar with these Windows applications so using them to develop there own application is straightforward. To control external hardware, OLE controls could provide software modules representative of the functionality of the hardware. The result is a powerful application capable of controlling an industrial operation with little effort.

The decrease in development man-hours is one benefit of using OLE, another is the decrease in debugging. Because the functionality of Word and Excel has been thoroughly debugged and tested, the use of their objects decreases the chance of error in your application. Development in Visual Basic also decreases the chance of errors since there is less code to write to achieve the same results in other such developers as Visual C++. In conclusion, OLE produces safe and powerful applications in a fraction of the time spent developing DOS or Windows applications not utilizing OLE.

5. References

[Kruglinski, 1996] Kruglinski, David J. (1996), Inside Visual C++, Microsoft Press, Redmond, WA.

[Hergert, 1995] Hergert, Douglas (1995), Foundations of Visual Basic 4 For Windows 95 Programming, IDG Books Worldwide, Inc., Foster City, CA.

6. Biography

Dr. BRUCE SEGEE received a Ph.D. in Electrical Engineering from the University of New Hampshire in 1992. He has been an assistant professor of Electrical and Computer Engineering at the University of Maine since that time. At the University of Maine, he heads the Instrumentation Laboratory, an organization dedicated to research and teaching involving instrumentation and automation. Work in the lab includes the use of PC's, PLC's, and embedded controllers for instrumentation, and networking. Work also includes the used of fuzzy logic and artificial networks.

KEVIN S. LEBLANC received a Bachelor of Science Degree in Electrical and Computer Engineering at the University of Maine in 1996. He is currently pursuing a Master of Science Degree in Computer Engineering at the University of Maine.