Session 1559

Software Application Design Using Visual C++ and OLE

Bruce Segee, Scott Dunning, Vincent Allen
University of Maine

Abstract

Our goal was to develop an OLE container application using Visual C++ that could provide access to multiple OLE servers. Each of the OLE servers is a separate application such as a report for processing data or an application that controls some process. The OLE container is responsible for combining all the servers in a presentable form to the user. The strength in this type of architecture is that new servers can be developed and made available to the user through the container without any modification to the container code or that servers can be mixed and matched in order to provide different functionality within the same container. The development of servers also provides valuable educational experience for students in terms of programming, software engineering, and also energy efficiency in the industrial and business communities.

1. Introduction

Since 1978, the U.S. Department of Energy has sponsored the Industrial Assessment Center program (IAC). This program provides “no-cost” industrial assessments to small and medium-sized manufacturers around the United States. The program utilizes 30 Universities which each perform 30 assessments annually at firms within 150 miles of their respective campuses. It has been highly successful in training students in energy efficiency and waste minimization techniques. The program has also been very successful helping the manufacturers that are served by the program.

In our desire to provide further assistance to smaller manufacturers, we began investigating ways to share technical expertise with these small manufacturers. At the national level, the U.S. Department of Energy has developed publications in conjunction with Rutgers University that document typical energy conservation opportunities [1]. They also have developed a self-assessment workbook to assist manufacturers in improving energy efficiency [2]. Unfortunately, most facility managers of small manufacturing firms have little time to sort through large manuals educating themselves in energy efficiency.

This led us to the conclusion that we needed to develop a software package to assist these manufacturers. Furthermore, this software package needed to be easy to use, easily extensible, customizable for a given industry, and be such that adding or removing recommendations does not “break” any existing recommendations.

2. Generalized Software

Our goal was to develop a software application that would encapsulate a wide variety of recommendations that could apply to a wide variety of companies. The software needed to be

modular so that new recommendations could be seamlessly integrated into the existing framework.

The software consists of a container that provides a general user interface and a framework for presenting recommendations to the user. The recommendations are developed using a procedure that guarantees that a recommendation is compatible with existing (and all future) recommendations. This software design methodology allows programmers with various levels of expertise to usefully contribute to the development of new recommendations. Furthermore, the length of time necessary for a programmer to come up to speed is small. Finally, once debugged, the recommendations never need to be re-compiled, thus eliminating many versioning difficulties, i.e., once a recommendation is completed, it never needs to be built again.

The following subsections discuss the implementation of the software framework.

2.1 The OLE Container

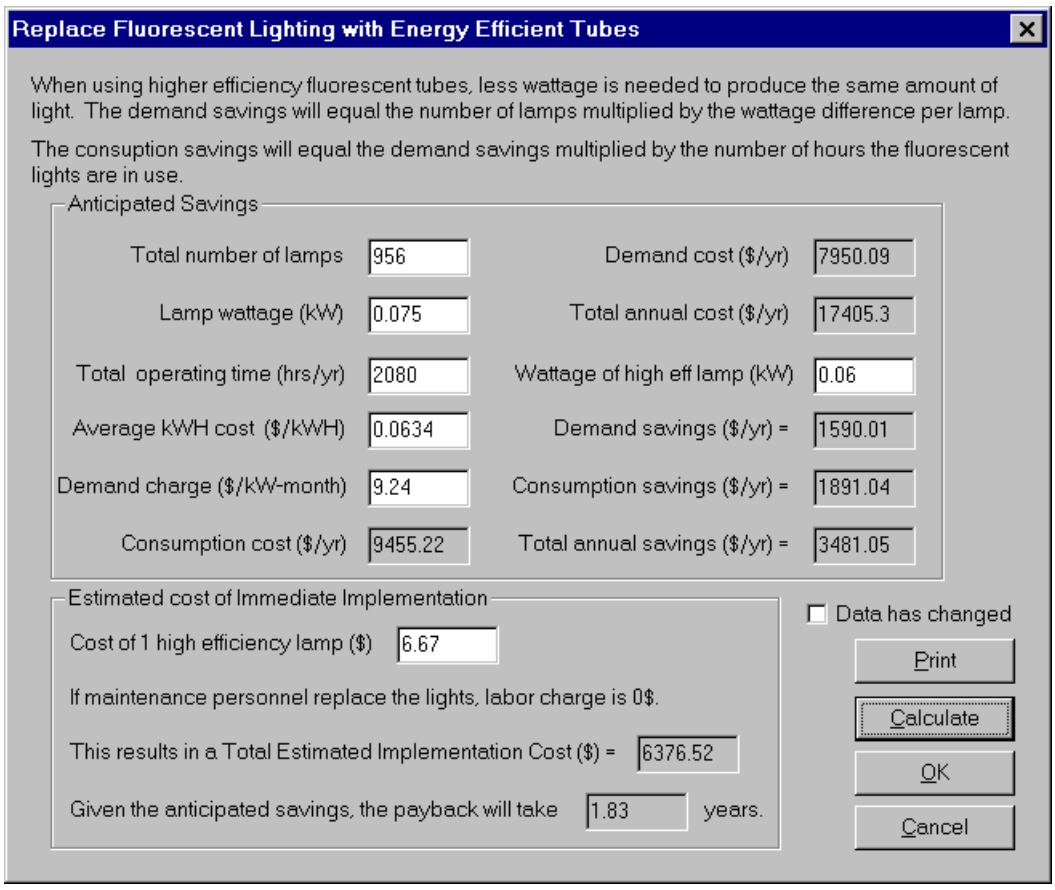
The OLE container was developed using Microsoft Visual C++. The purpose of the OLE container is an application that consolidates all of the recommendations in one package making them easily accessible by the user. The recommendations are separate applications implemented as OLE servers. When the OLE container application is launched, it searches the system registry for a list of Globally Unique ID's (GUID's) located in a static array within the container. The static array of GUID's contains all of the GUID's for the current OLE recommendation servers

Figure 1: University of Maine Assessment Software application.

as well as a number of unused GUID's to account for the future addition of new OLE recommendation servers. Upon finding any of the GIUD's in the system registry, the OLE container embeds the respective OLE recommendation server active in-place. In this way the OLE container application can function with only one recommendation server installed or a number of recommendation servers installed and no modification is needed when adding new recommendation servers to the OLE container. In developing a new recommendation server the programmer need only use one of the unused GUID's located in the static array within the container. This allows the container application to find the newly installed recommendation server without any modification to the application. Figure 1 shows the OLE container with two recommendation servers embedded.

2.2 The OLE Recommendation Servers

The OLE recommendation servers were also developed using Microsoft Visual C++. The purpose of a recommendation server is to provide a dialog box for gathering necessary data as well as printing the recommendation report. When an OLE recommendation server application is launched, a modeless dialog box is made visible so that the user may enter data necessary for the recommendation. Once all data is entered, clicking on the calculate button within the dialog will compute the savings that would result if the recommendation were implemented. The OLE recommendation server also allows the user to print the recommendation report as well as provides a help file explaining the significance of all questions and results within the recommendation dialog. Figure 2 (below) shows the recommendation server's dialog box for the Energy Efficient Tubes recommendation.



2.3 Printing Recommendations

The OLE recommendation server uses an OLE automation object provided in a Visual Basic Dynamic Linked Library (DLL) to print the recommendation report. The DLL uses OLE to create an instance of Microsoft Word. After setting a number of member variables, representing the data and calculated results, in the OLE automation object, the object loads up a Microsoft Word file specific to the recommendation. Word then searches for identifiers in the document representing placeholders and replaces them with the value of the corresponding member variable. The placeholders were chosen to be easy to identify and be very highly unlikely to appear in an actual document. We use ~Px~ where x is the number of the parameter that is to be inserted in place of the placeholder. An example Microsoft Word file for a recommendation is contained in the Appendix. Calling the print member function of the OLE automation object saves the template as an output file and sends the file to a printer. This approach to printing a recommendation report allows for effortless modification to the report format. Changing the recommendation report format simply involves editing a Microsoft Word file.

2.4 The Wizard

The software offers a “wizard” for displaying the recommendations. The “wizard” is a Microsoft Foundations Classes (MFC) CPropertySheet object. After all the “wizard” dialog boxes are set in the CPropertySheet object, calling the objects DoModal member function, when the object is set to run in “wizard” mode, will display the dialog boxes one at a time in a “wizard” fashion. The “wizard” asks specific questions about the company. Depending on the answers, the “wizard” will embed only the recommendation servers pertaining to that company. If the “wizard” is not used, the application will embed all servers installed on the PC. The user will then be able to choose from any of the recommendations.

2.5 Modularity

The architecture of the software is a completely modular. The container application, once developed, works with one recommendation or a number of recommendations. The development of the recommendation servers happens independently of other recommendation servers and does not affect the OLE container. If new recommendation servers are developed after a company has received the software, the new recommendation can simply be shipped to the company and installed on the PC. The new recommendation servers will be fully compatible with the container application. The software architecture also allows a custom application to be produced by sending only specific recommendation servers to a company.

3. Example Recommendation

Some of the more common recommendations are those for lighting. One of these is Replace Standard Fluorescent Lighting with Energy Efficient Tubes. This section will show an example of the application while investigating the recommendation mentioned above.

Double clicking on the Replace Standard Fluorescent Lighting with Energy Efficient Tubes recommendation bitmap (see Figure 1) will launch the OLE server active in-place and display the dialog box shown in Figure 2. Data is entered into the white edit boxes in the dialog. Hitting return or clicking the Calculate button after all data is entered, will update the dialog and display the savings and implementation costs in the gray edit boxes. Once this is done a recommendation report can be printed by clicking the Print button seen on the dialog in Figure 2.

The appendix shows the Microsoft Word file used to generate the recommendation report. Notice the identifiers ~P1~ through ~P20~. These are the identifiers that get replaced by the data in the dialog. It is clear from looking at the file that it would be very easy to modify the format of the report.

4. Conclusion

The result of this project is a software application that provides the following features: user-friendly interface, modularity, and customizability. With this software package, companies that do not qualify for the IAC evaluations will now be able to benefit from an IAC-like energy assessment. The development of servers also provides valuable educational experience for students in terms of programming, software engineering, and also energy efficiency in the industrial and business communities.

5. References

[1] Muller, M.R., Simek, M., Mak, J., "Modern Industrial Assessments: A Training Manual", Rutgers University Press, 1996

[2] Muller, M.R., "A Self-Assessment Workbook for Small Manufacturers", Rutgers University Press, 1996

[4] Brockschmidt, K., "Inside OLE", Microsoft Press, 1995

[5] Kruglinski, D., "Inside Visual C++", Microsoft Press, 1995

[6] Matteson, B., ed., "Microsoft Word Developer's Kit", Microsoft Press, 1995

Appendix-
Microsoft Word File for a Specific Recommendation

REPLACE STANDARD FLOURESCENT LIGHTING WITH ENERGY EFFICIENT TUBES

Recommended Action

Replace all standard fluorescent tubes with high efficiency tubes.

Anticipated Savings

The present lighting in the office and the plant consists of ~P1~ of the ~P2~ kWatt-8 ft. standard lamps. Assuming a ballast factor of 1.1 and noting that the lighting is on an average of ~P3~ hrs/yr, the power consumption (PC) and energy consumption (EC) by the 8 ft. tubes is:

PC = (~P1~ lamps x ~P2~ kW/lamp) x 1.1 = ~P4~ kW

EC = ~P4~ kW x ~P3~ hrs/yr = ~P5~ kWH/yr

With an average kWH cost of $~P6~/kWH and a demand charge of $~P7~/kW-month. This amounts to a yearly cost of:

Consumption Cost = ($~P6~/kWH) x (~P5~ kWH/yr) = $~P8~/yr

Demand Cost = ~P4~ kW x ($~P7~/kW-month) x 12 months/yr = $~P9~/yr

Total Annual Cost = $~P10~/yr

Using a higher efficiency tube, less wattage is needed to provide essentially the same amount of light. If a ~P11~ kWatt high efficiency lamp is `used there will be a savings of ~P12~ watts per lamp.

Demand Saving = ~P1~ x ~P12~ kW/lamp = ~P13~ kW

Consumption Saving = ~P13~ kW x ~P3~ hrs/yr = ~P14~ kWH/yr

Implementation of the efficient tubes would then provide a cost savings of:

Consumption Cost Savings = ~P14~ kWH/yr x $~P6~/kWH = $~P15~/yr

Demand Cost Savings = ~P13~ kW x ($~P7~/kW-month) x 12 months/yr = $~P16~/yr

Total Annual Savings = $~P17~/yr

Immediate Implementation

The estimated cost of implementation of this recommendation is:

Materials:

New Tubes: (~P1~ x $~P18~) = $~P19~

Labor:

Can be replaced by maintenance personnel = $0

Total Estimated Implementation Cost = $~P19~

Based on the above implementation cost and energy cost savings, the simple payback period for this recommendation is:

($~P19~ implementation cost) / ($~P17~/yr) = ~P20~ year payback

Biography

Dr. BRUCE SEGEE received a PhD in 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 Research 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, automation, and networking. Work also includes the use of fuzzy logic and artificial neural networks.

SCOTT C. DUNNING is an Assistant Professor of Electrical Engineering Technology at the University of Maine, Orono, Maine. He teaches undergraduate courses in electrical machinery and power systems. He received the BSEE and MSEE from the University of Maine. He is a licensed professional engineer in the state of Maine. He is currently Chairman for the Executive Committee of the Institute of Electrical and Electronics Engineers in Maine (IEEE) and a Member of the American Society for Engineering Education(ASEE).

VINCENT M. ALLEN has recently received a Bachelor of Science degree in Electrical Engineering at the University of Maine. Currently he is working in the Instrumentation Lab at the University of Maine toward a Master of Science degree in Computer Engineering.