

LA-UR-04-4368

Approved for public release;
distribution is unlimited.

Title: **Unified User Interface for Multiple Independently
Developed Instruments**

Author(s): Tom Marks
Jose March-Leuba
Joseph Glaser

Submitted to: Presented at the 45th Annual Institute of Nuclear Materials
Management Annual Meeting,
Orlando, FL, July 18-22, 2004



Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the University of California for the U.S. Department of Energy under contract W-7405-ENG-36. By acceptance of this article, the publisher recognizes that the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

Unified User Interface for Multiple Independently Developed Instruments

Tom Marks

Los Alamos National Laboratory
P.O. Box 1663, MS E-572
Los Alamos, NM 87545 (USA)

Jose March-Leuba

Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, TN-37831-6010 (USA)

Joseph Glaser, Program Manager, HEU-TIP
DOE/NNSA/NA-232
Department of Energy
19901 Germantown Road
Germantown, MD 20874 (USA)

Abstract

Los Alamos National Laboratory (LANL) and Oak Ridge National Laboratory (ORNL) have collaborated to develop an integrated second-generation non-destructive assay (NDA) instrument for continuous, unattended measurement of UF_6 gas flowing in process pipes of a blend-down facility. LANL was responsible for measuring the U-235 enrichment, and ORNL was responsible for measuring the fissile mass flow.

The first generation of these instruments utilized completely independent hardware and software systems with separate and disparate user interfaces. For the second-generation system, a very high priority was placed on having a unified graphical user interface (GUI). Having a unified GUI facilitates the infrequent and short-duration inspector visits dictated by the host facility. This presentation describes the design and implementation of a software architecture that facilitates the development of a fully-unified GUI while preserving the advantages of independent instrument development. The unified GUI allows the user to perform all functions including setup, calibration, reporting and diagnostics on either instrument from either computer.

The largely-independent development described herein accommodated well the challenges presented by: 1) having a tight schedule, 2) having the expertise for each of the two instruments residing at separate locations, and 3) needing to thoroughly test each subsystem before integration and yet assure proper operation of the integrated system with minimal time and effort at the end of the project. Item three can be especially important if there is insufficient NDA hardware to mock up the whole system at each location.

The software architecture described here could be used for remote monitoring and control and could easily be expanded to integrate more than the two systems described here.

Introduction

In 1993, the United States and Russia signed a government-to-government agreement to dilute 500 metric tons of highly enriched uranium (HEU) from Russian nuclear warheads into low-enriched uranium (LEU) fuel for U.S. commercial nuclear power plants. The HEU Transparency Implementation Program (HEU TIP) was established within the Department of Energy to provide assurance that the HEU being purchased from Russia is from dismantled weapons, and that the same HEU is converted, processed, and blended to LEU. As part of the transparency initiative, Los Alamos National Laboratory and Oak Ridge National Laboratory have produced a second-generation NDA instrument to provide the assurance needed. The first-generation instrument is currently installed in two Russian facilities. The new instrument is scheduled to be installed in a third Russian facility in October 2004 and may be retrofitted as an upgrade to the other facilities. These instruments are generically known as Blend-Down Monitoring Systems (BDMS).

The BDMS system is comprised of two major sub-systems. The Enrichment Monitoring (EM) subsystem measures and records the enrichment in U-235 in each of the three legs at the blend point. A Fissile Mass Flow Monitoring (FM) sub-system measures and records the mass of fissile material passing through each of the legs. The data acquisition and control software and the NDA instrumentation comprising the two subsystems is fully independent for the two measurements (flow and enrichment). However, the software architecture described in this paper presents a highly-integrated view of what appears to be a single instrument to the user and host facility, as required by the sponsor.

Screen Layout

Along the left edge of the application window, there is a hierarchical tree menu presenting to the user all possible functions that the instruments can perform. The items in the tree menu are a composite / merging of items provided by both instruments. For development convenience and to make the tree menu more consistent for the various instruments, the developers agreed on several of the major headings at the beginning of the project. Below the tree menu are two quick-navigation push buttons and a clock showing time elapsed. This left-hand portion of the screen is visible throughout the life of the program. The remainder of the application window is dynamic and will change depending on the function that the user has selected from the tree menu.

Home Screen

When the Common GUI program starts up, it presents a home screen consisting of several distinct functional areas. (See figure 1.)

The left-hand edge of the window is the always-displayed tree menu. The remainder of the window (the dynamic portion) of the home screen is broken into three functional areas.

There are two push buttons in the upper right portion of the window. These allow the user, even if inexperienced, to quickly produce reports showing a summary of the data for some user-selected period – usually by month or quarter year.

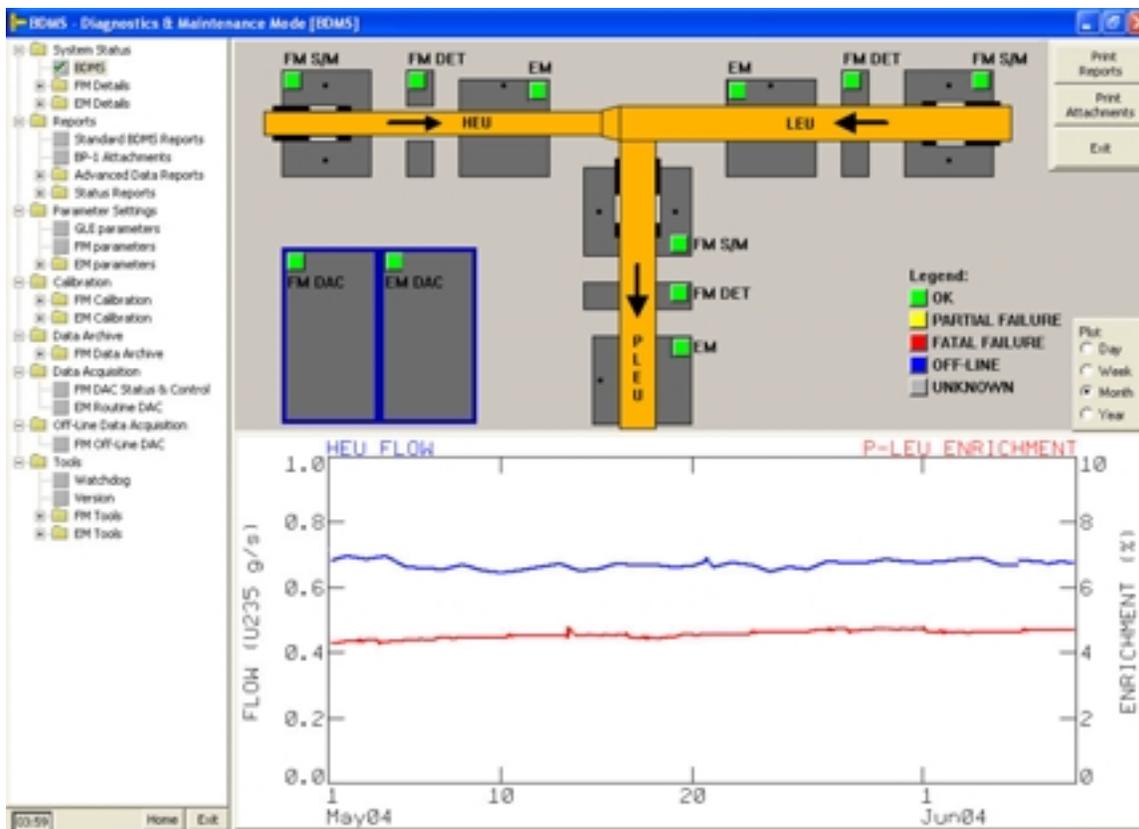


Figure 1. Common GUI Home Screen

The bottom portion of the window is a graph using two colored lines and colored vertical axes to display both the measured enrichment and measured mass flow as a function of time. The user may elect to show the most recent day, week, month or year of data.

A graphical representation of the blending area is presented in the upper center portion of the window. Each of the three legs has identical instrumentation. For each leg there are three colored buttons that show the overall status of the components. The flow monitor (FM) has two major components: 1) a source modulator and 2) a detector. The enrichment monitor has a single button to represent the detector and its associated multi-channel analyzer (MCA). The user may click on any of these colored buttons and a new status screen will appear, replacing all but the tree menu portion of the screen. The new screen displays a more-detailed overview of the status of the particular subsystem component that the user selected (clicked-on). Figure 2 illustrates how each subsystem may display the same type of information (in this case a summary of status details) in a manner that is appropriate to each specific subsystem but distinctly different for the two subsystems.

There are two additional colored buttons in the lower left portion of the graphical blending area window. These buttons show the status of the respective data acquisition and control (DAC) software modules that are running as Windows NT system services.

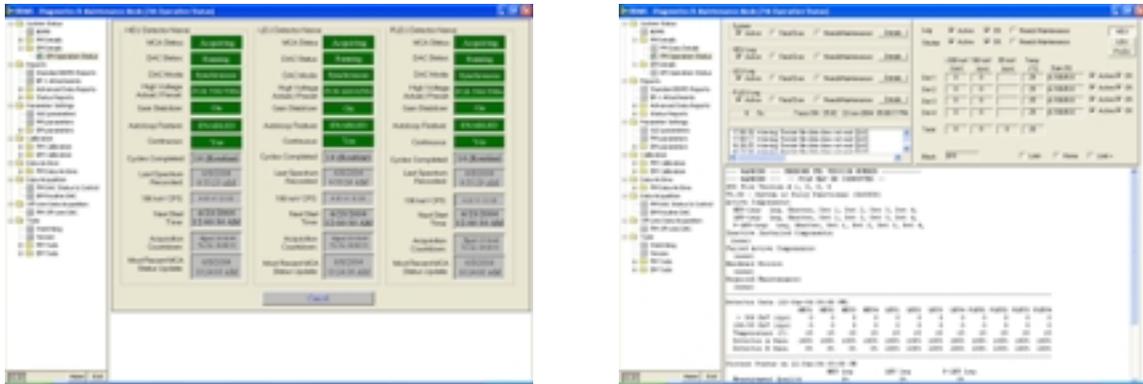


Figure 2. Instrument-specific status screens. Screen details are not important here – notice the tree-menu is the same and otherwise the screens are very different for EM and FM.

Hardware Configuration

There are two independent computers; each is connected to the NDA instrumentation that it will control and monitor, **either EM or FM**. These otherwise independent computers are connected through Ethernet and to a single printer as illustrated in figure 3.

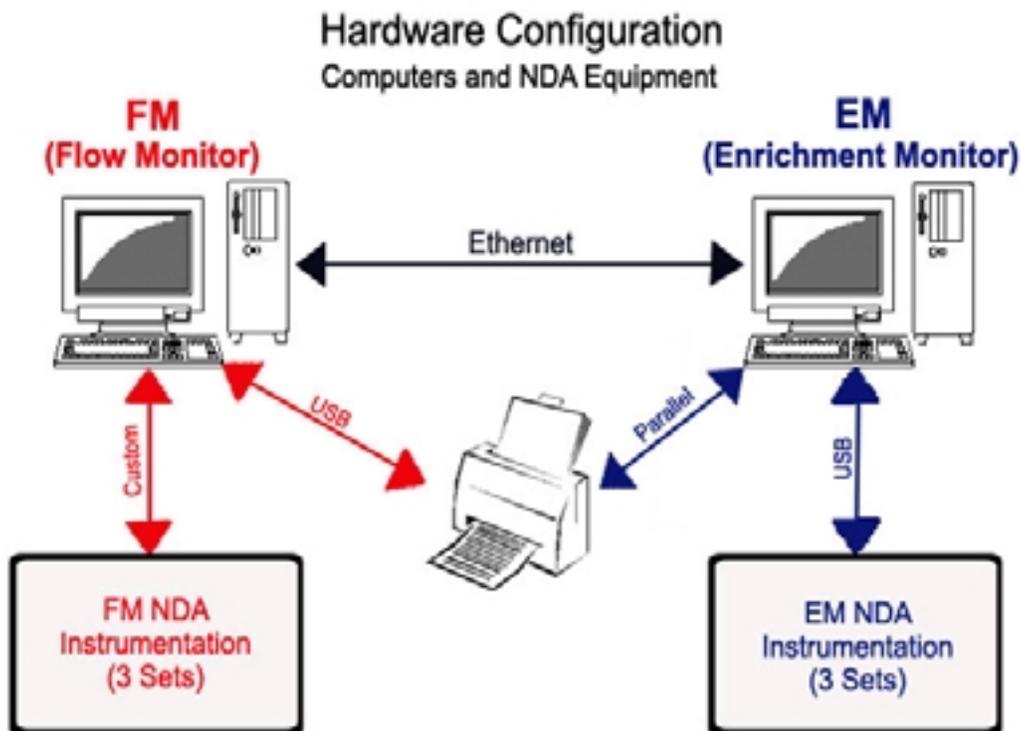


Figure 3. Hardware Configuration

Software Components

Both computers are loaded with the main program, Common GUI, and the two instrument-specific GUIs, (**EMGUI** and **FMGUI**). These software modules are shown above the dashed line in figure 4. Up to this point, both computers have the same software configuration.

Note that each computer is physically connected to its own specific instrumentation as shown in figure 2. The data acquisition and control software, either **EMDAC** or **FMDAC** is installed on only the computer that is physically connected to the specific instrumentation.

One more item distinguishes the two computers. The data from each instrument is stored only on the computer connected to that instrument.

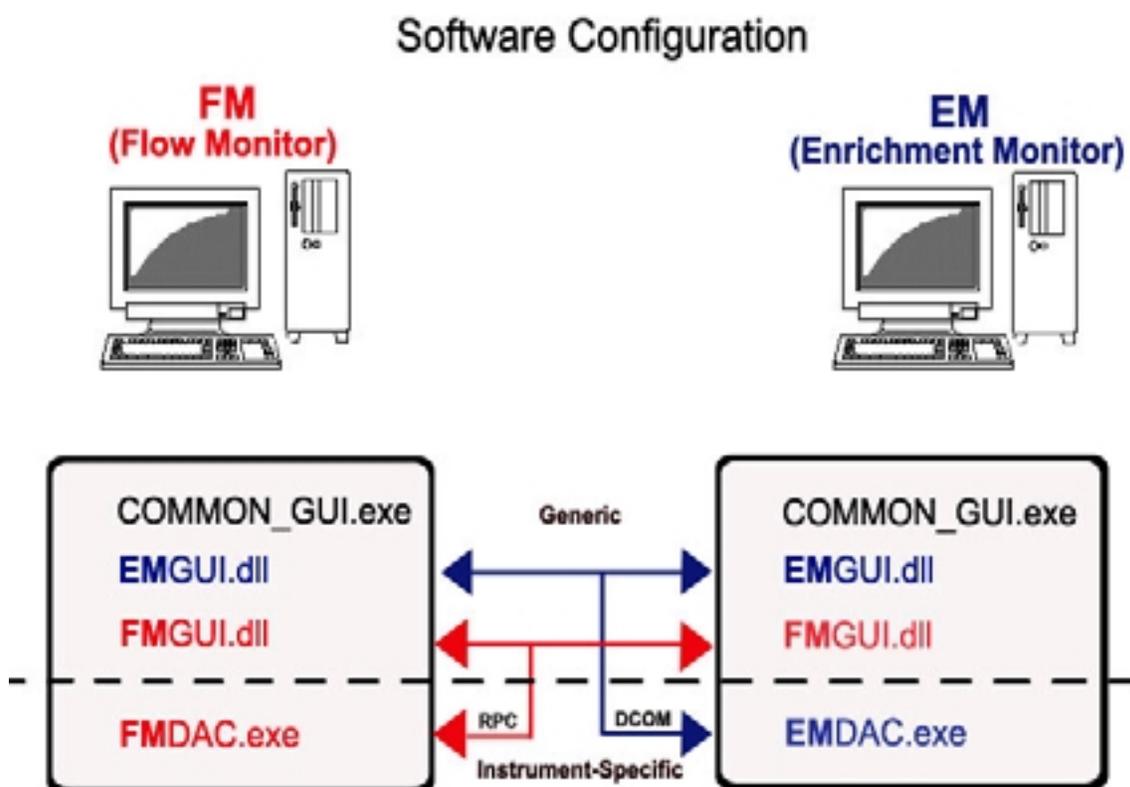


Figure 4. Software Configuration

Instrument-specific Hardware Control

Each computer communicates directly to only the instrumentation for its particular measurement. All of the software required for data acquisition and control (DAC) of the instrumentation is contained in one of two programs (**EMDAC.exe** or **FMDAC.exe**). Because these are Windows NT system services, there is no user interface provided in these modules. All the required user interface is contained in the respective GUIs.

These DAC modules must be able to run automatically, reliably control the instruments and log the acquired data because of the unattended nature of the measurements. Setting startup type to 'automatic' in the WIN-NT system services DAC modules will cause the DACs to start up automatically when the computer boots, without the need for any user intervention – not even a login.

The communication between each instrument-specific GUI and its respective DAC is private to that subsystem. The subsystems do not need to know anything about the other subsystem's internal software or hardware details. The only requirement is that the communication between the DAC and its respective GUI works across a network.

The architecture allowed each Laboratory to use the most expedient DAC-to-GUI communication protocol. In the case described here, one instrument uses Remote Procedure Calls (RPC) and the other uses the Distributed Common Object Model (DCOM) protocol for communication between the GUI and its associated DAC.

Instrument-specific GUIs

The two instruments / measurements may have very little in common. For example, the EM system acquires and analyzes gamma-ray spectra using MCAs. This requires that, for example, the EMGUI allow the user to establish Regions of Interest (ROIs) for the peaks. The FM system does not use spectral data and therefore does not need to interact with the user to establish Regions of Interest.

Each Laboratory developed software and user interface screens necessary to meet its measurement and instrumentation requirements. The Common GUI need only be informed of the tree menu items and this is done dynamically during Common GUI startup by querying each of the instrument-specific GUI DLLs. Thus, each instrument developer is free to add, change or delete functionality for his/her instrument without impacting any other code. The changes are distributed by providing a new instrument-specific GUI DLL. (The instrument-specific DAC may need to be changed on the one computer that interfaces to the instrument hardware if the new or changed functionality requires more computer-instrument interfacing.)

GUI Interface Functions

There are five essential functions that each Instrument-specific GUI (InstrumentGUI; i.e., EMGUI or FMGUI) must perform at the request of the Common GUI program.

1. **InitializeDLL:** Called during Common GUI program startup. Communicates program instance handle, parent dialog window handle, network addresses and data-share name to the InstrumentGUI.
2. **GetTreeMenuInfo:** Common GUI requests a hierarchical list of items to be displayed in the tree menu. InstrumentGUI returns a structure containing the tree menu items and an ID for identifying each of the leaves on the tree. Additionally, this call may contain information to identify which menu tree is being requested. For example, this instrument

has an expert mode where all menu items are available and a casual user mode where only a few menu items are available.

3. **ProcessRequest:** Common GUI calls the appropriate InstrumentGUI with the ID of the menu item (procedure) that the user clicked on. All of the menu items and associated IDs were registered via the structure returned in GetTreeMenuInfo.
4. **EndRequest:** Common GUI calls InstrumentGUI passing in the ID of the procedure that is currently active and should be terminated before a new procedure is started. InstrumentGUI may return FALSE if the currently active procedure cannot terminate immediately. In this case, the pending ProcessRequest will not be sent.
5. **Cleanup:** Called just before Common GUI terminates to give InstrumentGUI a chance to do any program termination or resource de-allocation that is required.

In addition to the basic interface functions above, we found it convenient to add six additional interface functions specific to our combined instrument software.

1. **GetStatus:** The home screen has buttons whose color represents the status of the major components. This function is called periodically and returns a structure containing the enumerated status colors of each of the status indicators for the particular instrument. The get-status request can also be triggered by one of the InstrumentGUIs by sending a message to the Common GUI. This allows each of the instruments to assure that the displayed status immediately reflects any status change.
2. **StatusButtonPushed:** The user can drill down to get more detailed information about the status of a particular major component. This function is used to notify the InstrumentGUI that the user clicked on a specific status indicator on the main screen. Doing so will take the user from the screen presented in figure 1 to **one** of the screens presented in figure 2, depending on which status button was clicked.
3. **GetPlotData:** This is called during the Common GUI startup to get the data to plot on the home page. This process can be triggered later by one of the InstrumentGUIs by sending a message to the Common GUI.
4. **GetStandardReport:** The user can request various pre-defined reports via two push buttons on the home screen. Pushing one of these buttons causes the Common GUI to present the user with a list of standard reports. When the user selects one of the reports, Common GUI calls this function for the appropriate InstrumentGUI passing in information identifying which report is being requested, the report period and information necessary to either display or print the report, such as a printer dialog or window handle.
5. **GetServiceInfo** and 6. **GetDriverInfo:** These functions pass back to Common GUI information about the instrument-specific DACs and a list of instrument-specific drivers and their versions for display in the Common GUI About Box.

NB: The entire public interface of the software for this combined instrument is comprised of:

- these eleven functions,
- three predefined structures, and
- constants defining colors, predefined menu items etc.

Remote Control and Monitoring

The system we have developed consist of two computers located in the same cabinet with a simple Ethernet connection. The computers could be placed on a much broader network connection (e.g., the Internet), thus allowing anyone to run the Common GUI from any computer on the broader network that will run the Common GUI and the two InstrumentGUIs. With access to the Common GUI and instrument-specific GUIs, the remote user could monitor and even control the instrumentation from any remote location that has a network connection to the instrument computers.

If users are allowed to control the instruments from remote locations, one must be careful handling colliding or conflicting requests. For the implementation described in this paper, the users sit next to each other so we relied on “administrative” controls to avoid these problems.

Summary

We have produced a real-world NDA instrument comprised of two subsystems that appear as a single instrument to the user. Yet, development of the two subsystems was done with nearly total independence. This independence is well illustrated by the choice of software development tools: Microsoft Foundation Classes, Microsoft Visual Studio 6.0, C++ and DCOM for the EM instrumentation at LANL vs. native WIN32 calls, Microsoft Visual Studio .NET, C and RPC for the FM instrumentation at ORNL.