The Process Manager GUI for CLEO III

Natalie Griffith

Department of Materials Science and Engineering

Wayne State University, Detroit, MI, 48202

Abstract

Because of the numerous hardware and software systems associated with the operation of the CLEO III Particle Detector, it is necessary to develop a user friendly GUI to serve as an error monitoring and control system for its various software components. Utilizing the Java Programming Language and CORBA, the Process Manager GUI is developed to interface with all of the Process Managers, or software programs that monitor and control all of the processes, throughout the CLEO III system.

Introduction

The CLEO III Particle Detector is controlled by many different smaller software and hardware units, built one upon another, in layers. Due to the large number of hardware and software components associated with its operation, it is necessary to develop a number of GUI's, to serve as monitoring and controlling devices, for the software components of the system. One of the major components in this plan is to develop a Process Manager GUI, to monitor and control all of the Process Managers running throughout CLEO III.

One area of concern is which of the many programming languages to use. Because of the ease in GUI development, the Java Programming Language is used to develop the actual GUI, and CORBA (Common Object Request Broker Architecture) is used as the communicator between the Process Manager and the Process Manager GUI.

An understanding of the operation of CLEO III's Online System and its functionality, Java, CORBA and IDL files, the concepts of a Process Manager, and generic GUI's are all important components in developing the Process Manager GUI.

Online System

The Online System essentially controls and monitors all of the hardware and software components that are necessary for the successful operation of the CLEO III particle detector.

The CLEO III detector is directly linked to the various hardware systems such as the power supplies, the gas controls, the high voltage control systems, the cooling systems, etc. These many hardware systems are monitored and controlled by C++ software, essentially software that controls hardware, and is linked to that hardware by system specific connections. C++ is utilized for this layer because of its the high speed and good interface capabilities to the hardware components. The C++ software is then connected to the Process Managers, or the software that controls other software, through generic operating systems such as NT Windows, Unix, and VxWorks. The Process Manager GUI interacts directly with the user and communicates with the Process Managers via CORBA, as represented by Figure 1.

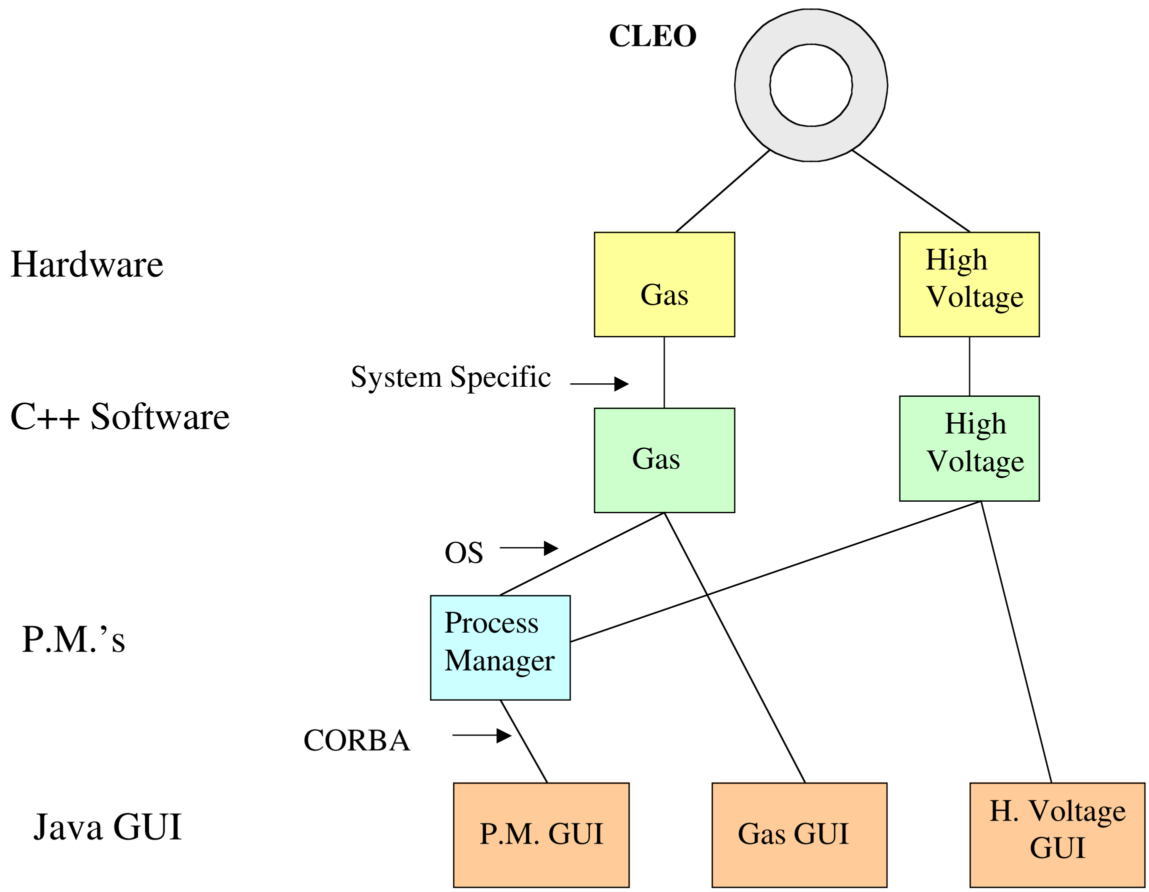

FIGURE 1. A Generic Example of the CLEO III Online System.

Java

Java is a general purpose, object-oriented programming language that supports the development of GUI applications, as well as, the development of client server applications. It is chosen as the ideal programming language in which to develop the Process Manager GUI because of the availability of the tools necessary for the design process. These tools are not available with the C++ programming language. Although Java is slower than C++ it is not necessary to utilize the faster language, as the GUI only needs to operate as quickly as the user accessing it. Also, Java GUI's are portable and can be used as applets in web pages. They do not have to be recompiled when used on different operating systems, increasing the ease of its use.

CORBA and the IDL

CORBA, or Common Object Request Broker Architecture, is a client-server communication package that supports both Java and C++. Its purpose is to serve as a connection between the client, in our case the Java GUI, and the server, written in C++. The IDL file defines the client server communication. Within its file, the IDL has a list of functions that both Java and C++ can use to talk with one another. Essentially, CORBA locates the object, routes the request, and returns the results. A CORBA object, or variable, contains the connection to a specific server, which provides a service. In our case, it gets the list of the processes.

Process Manager

The Process Manager is one of the software programs that monitors and controls the processes throughout the detector, providing the functionality to start, stop and restart programs on remote hosts. When retrieving code from the database, the Process Manager interacts with the programs database server, translating the version select input parameter. Finally, the Process Manager serves as a log for each of the processes, and outputs a list of the processes that occur.

When one of the many Process Managers running throughout the CLEO III system is started, it is registered with Igor, a C++ process that keeps lists of registered servers. Once the Process Manager GUI is initialized, it uses CORBA to request the list of the Process Managers that are registered.

GUI

Graphical User Interfaces, or GUI's, are user friendly programs that are based on the processing of events such as the click of a mouse or the stroke of a key. The Process Manager GUI processes events such as List Selection events and Action events. List Selection events occur when the user selects a specific line of information from a list of data. Action events occur when the user clicks on a button within the GUI panel.

Again, when the Process Manager GUI is started, it uses CORBA to request the list of Process Managers that are registered, and displays them in the "Process Manager" list, as shown in Figure 2.

Upon selecting a Process Manager from the Process Manager list, a "Processes" table appears, displaying all the processes that are connected to that specific Program Manager, as in Figure 3. The table displays information such as the name of program that is running, the programs' log identification number, the status of the program, and the programs' starting and ending dates.

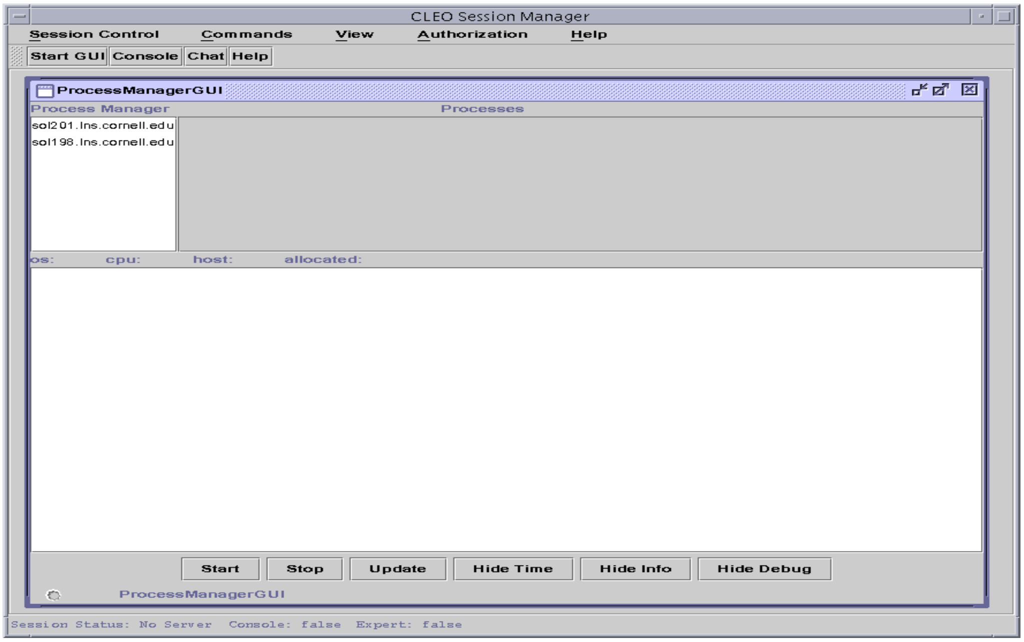

FIGURE 2. Process Manager GUI – Initial Implementation. Process Manager list filled.

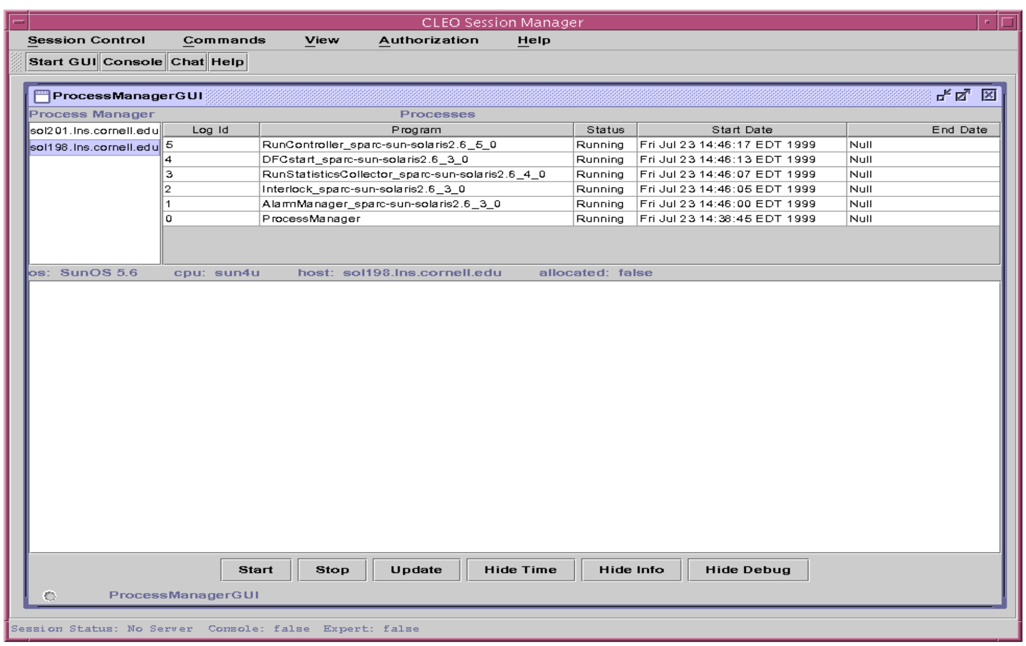

FIGURE 3. Process Manager GUI – Process Manager selected. Processes table filled.

After selecting a specific line in the Processes table, an Error Message Log is displayed. The Error Message Log outputs lines of information, as shown in Figure 4, each of which contains one of the four following keywords: Info, Debug, Warning, or **Error**.

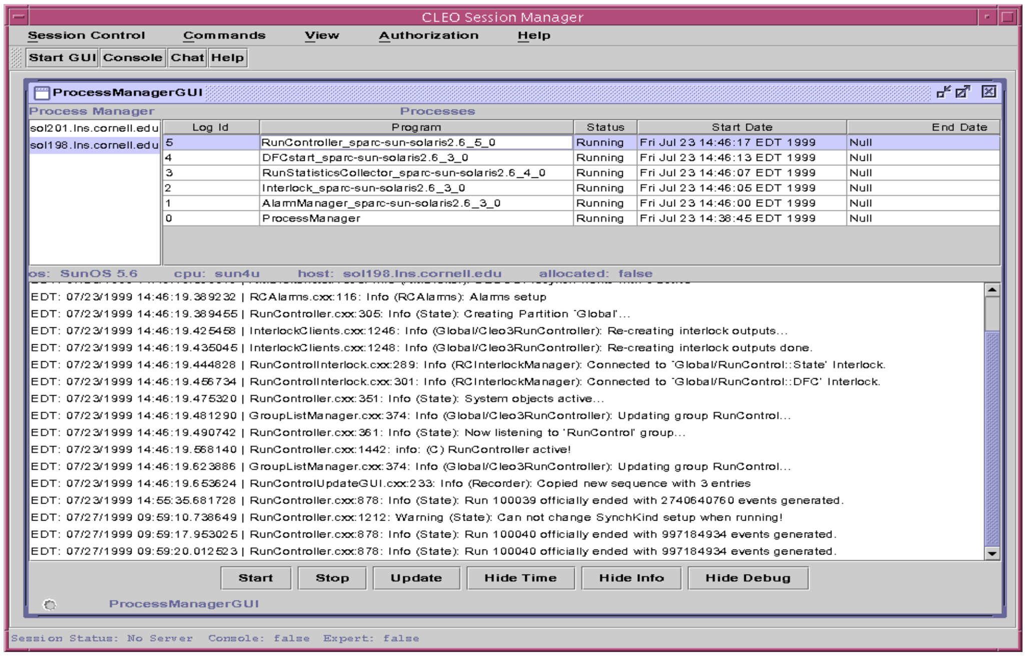

FIGURE 4. Process Manager GUI – Process Selected. Error Message Log displayed.

The Info and Debug messages show only as early warnings of impending problems, helping the GUI user to understand, more readily, the meaning of the warning and error messages that occur. Because of the importance of finding the message lines containing only the “warning” and “error” keywords more easily, two buttons, “Hide Info” and “Hide Debug” were designed. When selected, each suppresses all lines, within the Error Message Log, that contains those keywords, leaving only the problematic “Warning” and “**Error**” messages displayed. After the buttons have been engaged and the desired messages have been suppressed, the text within each of the two buttons changes to “Show Info” and “Show Debug”, as seen in Figure 5. This allows the user to re-display the suppressed information at any time. When either of the buttons are again selected, the suppressed text re-appears, and the text within the buttons reverts back to their original prompts.

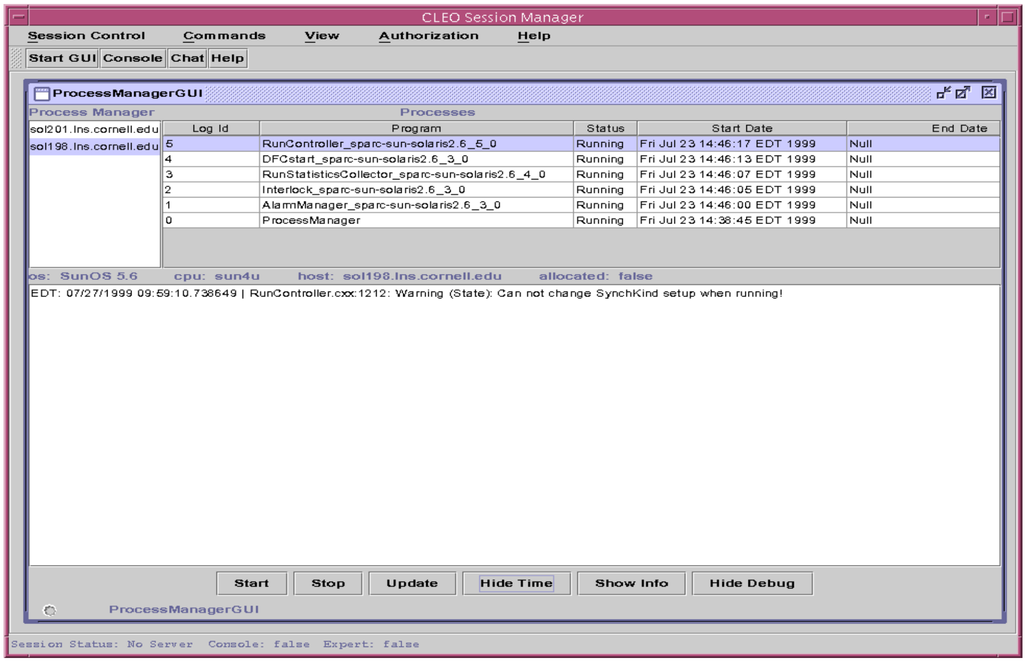

FIGURE 5. Process Manager GUI – Error Message Log. Keyword "Info" suppressed. Text within the button changed to "Show Info" allowing user the opportunity to re-display the suppressed information.

A large amount of information is included on each message line of the Error Message Log. To cut out insignificant information such as the date and time that the message occurs, a "Hide Time" button is programmed. After selecting the button, and the date and time information from each of the lines of the Error Message Log is eliminated, the user has the option of re-displaying the date and time as needed, shown in Figure 6.

The final two buttons, the Start and Stop buttons have not been implemented because the needed functionality has not yet been implemented on the server. When implemented, the user will be able to start a program. When errors occur, they can stop the program for repairs, and then, restart the program, again represented by Figure 6.

Conclusions

The CLEO III Particle Detector is controlled by many different smaller software and hardware components necessitating the development of GUI's to control and monitor the various parts of the system. One main GUI, the Process Manager GUI is developed to allow the user to control and monitor the Process Managers running throughout the CLEO III Particle Detector. The Process Manager GUI should aid in the debugging and maintenance of the CLEO III online system.

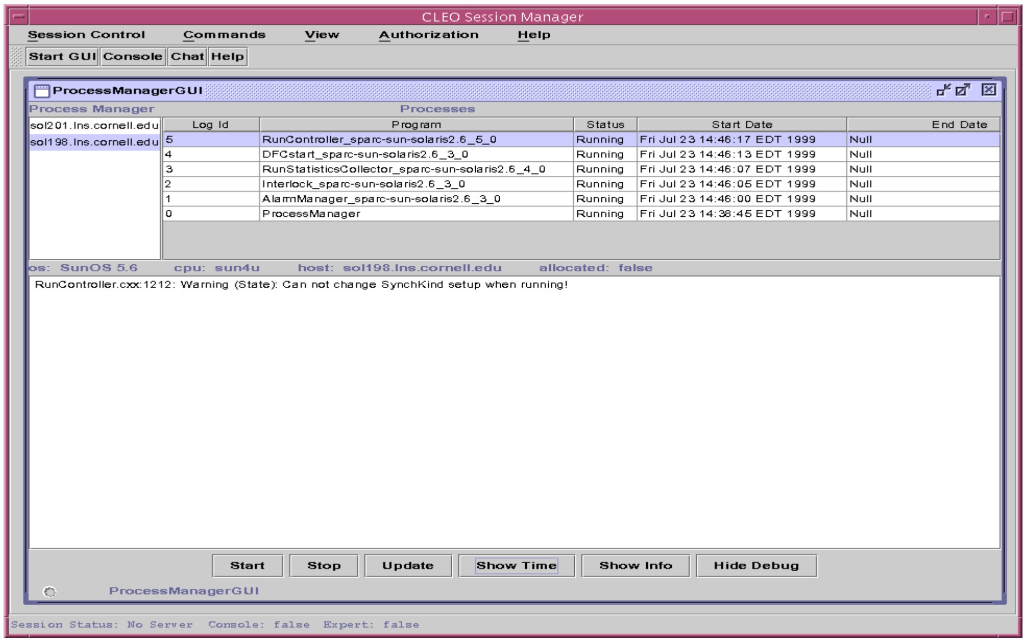

FIGURE 6. Process Manager GUI – Error Message Log. Date and Time information is suppressed. Text within the button changed to “Show Time” allowing the user to re-display the suppressed information.

Acknowledgments

I would like to acknowledge Mr. Elliot Lipeles and Professor Alan Weinstein of the California Institute of Technology. They developed and supervised this REU project and Mr. Lipeles gave his time, resources and knowledge willingly to help introduce me to, and educate me in, the world of the Java Programming Language. I would also like to thank Professors Giovanni Bonvicini and David Cinabro of Wayne State University for their assistance in preparing for this program. In addition, I would like to thank Professor David Cassel for allowing our continued participation in this program. This work was supported by the California Institute of Technology, Division of Physics, Mathematics and Astronomy, the National Science Foundation REU grant PHY-9820306, REU grant PHY-9731882, and research grant PHY-9809799.

Footnotes and References

1. http://www.lns.cornell.edu/restructed/COMP/DEC/java121/api/.
2. Horstmann, Cay S., et.al. *Core Java Fundamentals 2*, Sun Microsystems, Inc. (1999).