A FRAMEWORK FOR JAVA APPLICATION PROGRAMS IN THE CERN PS CONTROL SYSTEM

Michel Arruat, Jan Cuperus, Marine Gourber-Pace, Roger Hoh, Eric Roux, CERN, 1211 Geneve 23, Switzerland

Abstract

The user interface for a system controlling 5 interconnected accelerators is composed of a large number of windows organized in a tree structure of application programs with a console manager at the top. All programs run in a single Java Virtual Machine (JVM) without interfering with each other. The windows show accelerator data and interaction widgets. A framework has been built to design these windows and make them interact and cooperate with a minimum of effort from the programmers. A project manager controls the life cycle of the programs, including use of templates and CVS [3]. Program development and execution can be done on both the Microsoft Windows and Linux platforms through files that are visible to both systems.

1 INTRODUCTION

The CERN PS accelerator complex consists of 5 accelerators that deliver protons, antiprotons and ions for local experiments or injection into the SPS accelerator and later the LHC. Some can change completely their mode of operation from cycle to cycle. To control this complex we need a generic control system that is data-driven so that it can work with any accelerator. We are moving from an application layer written in C/C++ to one written in Java.

2 APPLICS AND FRAMES

The basic program unit is what we will call an applic which extends the AbstractApplic class. An applic can have an associated window or frame. An application consists of a tree of applics. Application trees can also form trees and so on but formally there is only one tree of applics (see Fig.1).

Any applic can be started as a dependent of a manager applic and then executes in the same JVM. It can also be started independently in its own JVM, in which case it becomes a top manager with special privileges.

A console manager is a top manager with a frame that is essentially a large menu to start applications and to hide or iconify entire application trees. AbstractApplic contains code for managing these trees and the communication between applics.

![Diagram of applic frame](image)

Figure 2: Class diagram for an applic frame.

A frame can be internal or external (see Fig.2) and if the programmer uses only the common AbstractFrame interface, he can transform one into the other by changing just one word in the code.

![Diagram of default empty frame](image)

Figure 3: The default empty frame.

The frame can receive a configurable TopZone and a BottomZone with a message area, expandable to a full frame, for looking at the latest messages. In between is the UserZone that can receive Java Beans and, more specifically, components for reading and setting accelerator parameters (Fig.3). The components of the default TopZone are (from left to right):
3 ERROR REPORTING
Class Display has static methods for reporting:
- showMessage(message, source)
- showWarning(message, source)
- showFault(message, exception, source)

Any component or service can call these methods. The framework tries to display the message in a relevant place and faults are guaranteed to be displayed. If object source is in any way associated with a frame, the message appears in the BottomZone of that frame, else in the BottomZone of the top manager, else as a message pane or, if there is no graphical interface, on the output stream. In addition, faults are logged on a relational database with JDBC (Java DataBase Connectivity).

4 APPLICATION ENVIRONMENT

Figure 4: The environment for applics and components

467
8 THE PROJECT MANAGER

We need to create new projects, safeguard them in CVS [3], install them and generate Javadoc. Also, we wish to work from Linux and Microsoft Windows interchangeably which produces long file paths for common visibility through the SAMBA [4] file system. The Project Manager automates all this.

Figure 5: Select Project page of the project manager.

First, one selects a project category and a project name (see Fig.5). Note that the project manager uses the framework and that it is used to maintain itself! When the Goto Project button is pushed, we get the design page shown in Fig.6. Many file operations are supported but not editing, for which you need a separate tool such as JBuilder.

Figure 6: File Operations page (here on Linux).

The CVS operation page is shown in Fig.7. CVS is implemented on a server that can be addressed both from Linux and Windows. There is also a Test+Install page that looks similar to the CVS page and allows compilation, testing, Javadoc documenting, and installation as a .jar file.

At any time, one can expand the message zone, on the bottom of the frame, to view a complete log of one’s actions since the beginning of the session.

9 CONCLUSIONS

With the framework, it is possible to write useful programs in a short time without necessarily being a professional programmer. The programmer, using a large library of proven components, can concentrate on his or her goals without being bothered by too many details.

Application programs using an early version of this framework have been in use for over a year. They perform satisfactorily as stand-alone Java programs called from a console manager in C++. This limits their number to 5 because of the large resources taken by each JVM. Also, a start-up time of 15s is rather long.

When enough Java programs exist, they should be started from a console manager in Java, all in the same JVM. Such a console manager must be able to run for several days without restarting. Also, the speed must be sufficient to update several hundred parameters every second. After recent improvements in the interfaces, the middleware, and the component library, we have confidence that we can reach these goals. A console manager in Java can also start (and to some extent control) C/C++ programs, but the ultimate goal should be an homogeneous Java control system.

REFERENCES