Java Based Run Control for CMS Small DAQ Systems

M. Bellato[2], *L. Berti*[1], *D. Ceccato*[1], *M. Gulmini*[1], *G. Maron*[1], *N. Toniolo*[1], *G. Vedovato*[1], *S. Ventura*[2], *X.Q. Yang*[1]

[1] INFN, Legnaro, Italy
[2] INFN, Padova, Italy

Abstract

Modern Data Acquisition Systems (DAQs) are composed by several phisically distributed cooperating devices which have to be controlled during the data taking. A Run Control System has to provide a flexible, efficient, and user-friendly environment where the final user can monitor and control all the components (Readout Units, Builder Units, Trigger, Event Manager, etc.) through a Graphical User Interface (GUI). The different characteristics (hardware and software) of the components, their geographical distribution, and the necessity to provide a possibly Web based GUI make Java an interesting candidate, thanks to its features of platform independence, native C/C++ code interface, embedded multithreading, high level communication protocols (CORBA, RMI, TCP), and Web support (applets, swing). This contribution will focus on the Run Control System developed for CMS (Compact Muon Solenoid) small DAQ systems. Thought as an easy to customize framework for small DAQs in general, it uses Java 2 and its CORBA implementation as communication backbone. Each component is controlled by a CORBA server, which behaviour is modelled through Finite State Machines. A number of services for configuration and setup purposes, error detection and recovery, job control, and status information management are included in the system. Finally, a Web GUI developed using HTML, Javascript and Java applets is presented.

Keywords: Java, Corba, run control, Web

1 Introduction

CMS (Compact Muon Solenoid - one of the four LHC experiments) data acquisition system will connect in 2005 about 500 readout units (RU) to about 500 builder units (BU). RUs, BUs, and other components (Trigger, Event Manager, etc.) have to be controlled and monitored during the data taking. Before 2005 a number of small DAQs have being implemented for testbeam purposes. The Java Run Control System (JRCS) we are going to present is thought as a general framework for such DAQs, and as a prototype for gaining experience with new software technologies (Java, CORBA, Web). Java and CORBA have been used exstensively to achieve platform (SPARC, Pentium, PowerPC) and operating system (Solaris, Linux, vxWorks, NT) independence. CORBA, proposed by the Object Management Group (OMG), is a distributed object framework, based on the client-server model, that allows transparent invocation of object methods on a network; it supports several programming languages including C++ and Java. Interoperability between applications developed with different ORB implementations is achieved by the OMG CORBA IIOP communication protocol (CORBA2). Java provides platform independence running applications on a Virtual Machine (JVM) free available for most operating systems; since the 1.2 version the Java Development Kit (JDK) provides a free ORB CORBA2 compliant implementation. Interoperability between this and other CORBA2 implementations (Visibroker for Java and C++, Orbix, OrbixWeb, TAO) has been investigated providing quite satisfactory results. The JVM embedded in modern browsers allows to run java code (applets). Running CORBA applets makes the browser fully integrated in the control system, so that Web based techniques can be used to provide a flexible and customizable GUI. Several approaches are suitable to the graphical development. Java and its swing classes are a good tool for the implementation of graphical applets. Dynamic HTML represents an interesting option provided by modern browers, allowing to modify a web page without any interaction with a Web Server through the use of a script language (Javascript). Script languages represent then an interesting tool for writing control command scripts.

A JRCS architectural overview will be presented in the section 2; section 3 will focus on the JRCS components that need to be customized for a specific DAQ.

2 Java Run Control System Overview

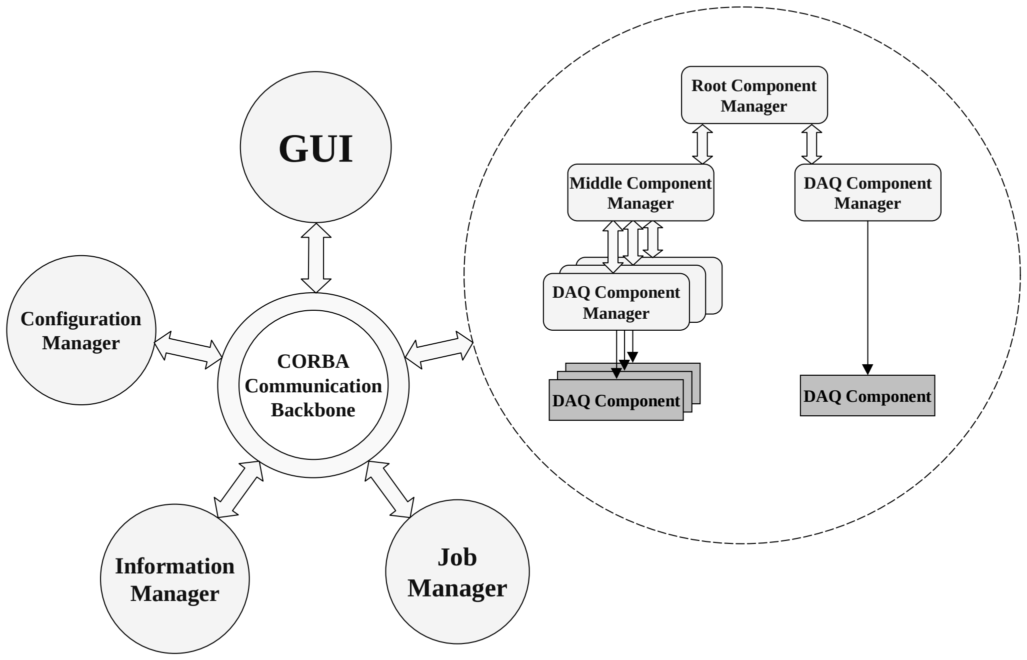

Figure 1: Run Control System Architecture

JRCS is based on a CORBA communication backbone, and consists of several cooperating Java-CORBA servers. Some of them are dedicated to the control of DAQ Components, others provide auxiliary services necessary in any control system.

As shown in Figure 1, each DAQ Component is controlled by a CORBA server called Component Manager (CM). A CM acts as a command filter able to maintain the logical sequence of commands sent to the DAQ Component; it also manages the communications with the other Component Managers organized as a tree where only the leaves are directly connected to a DAQ Component. The root (Root Component Manager - RCM) and the intermediate nodes allow to filter the commands related to all the CMs at a lower level in the tree. A user interacts with the system through a Web based GUI, sending control commands to the RCM, which dispatchs them to all the CMs at a lower level in the tree (broadcast), or to a subset of them (multicast). The user GUI and all the JRCS components make use of three auxiliary services:

- The Configuration Manager provides information related to the structure of the tree of CMs in the system, and, for each of them, stores a description of the features of the DAQ component which has being controlled. The information is actually stored on flat files, but the integration with a DBMS, possibly supporting JDBC (Java DataBase Connectivity) interface is a nice feature we plan to implement.

- The Job Manager allows to launch, kill, and monitor Component Managers; the task is done using the information provided by the Configuration Manager;
- The Information Manager provides the status of the JRCS components, and handles error and log messages.

Error detection and recovery procedures are provided. Both synchronous errors generated by a control command, and asynchronous errors (a server or a DAQ component that does not work properly or is crashed) are detected and managed by the Information Manager.

3 Customisation issues

Since JRCS main goal is to provide a framework to develop control systems for small DAQs, this section will focus on the components that need to be customized for a specific DAQ. They are mainly the Component Manager and the Graphical User Interface.

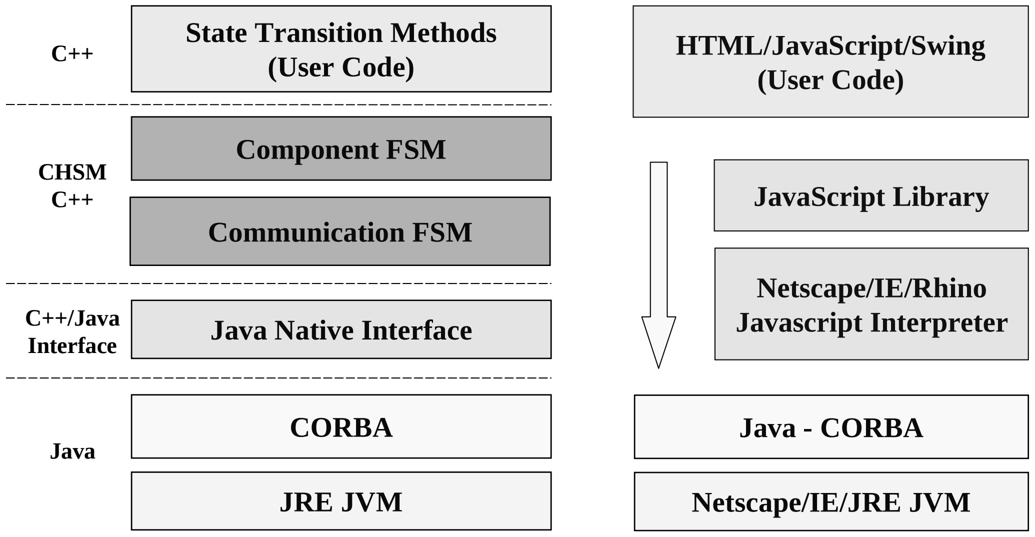

Figure 2: a) Component Manager software layers. b) GUI software layers

3.1 The Component Manager

Each DAQ Component (RU, BU, etc.) has itsown Manager. Component Managers' behaviour is modelled using a theoretically rigorous language system for implementing Concurrent, Hierarchical, finite State Machines (CHSM)[1]. CHSM package, already choosed by other experiments like Chorus and Atlas[2], provides a compiler producing C++ code. As shown in Figure 2a, CHSM C++ code has to be interfaced with the CM Java-CORBA code; this is achieved by the Java Native Interface (JNI), a native C/C++ interface provided within the JDK1.2. Two Finite State Machines (FSM) have been implemented. A 'Communication FSM' models and manages the communication between CMs, and a 'Component FSM' manages the DAQ Component behaviour. The 'Component FSM' is general enough to be suitable to most DAQ Components. When necessary, a new 'Component FSM' can be written in order to manage devices (for instance tapes) that need a different State Machine. The actions performed during a state transition in the FSM are associated to C++ methods that can be overloaded in order to write specialized code for the DAQ Component to control. Since most modern DAQs are developed in C++ this is a coherent choice. The developed FSMs are the same for all the CMs; state transition methods for the Root Component Manager and the other managers that do not control directly a DAQ Component might be used to implement common code related to a subset of Component DAQs. The integration between Component Manager and DAQ software can be achieved in several ways: the DAQ software can be a thread of execution inside the Component Manager software, or it might be a separate process. In that case the communication with the CM is achieved by using IPC (processes on the same machine), TCP or other communication protocols (processes on different machines).

3.2 The Graphical User Interface

Web based techniques have been used to produce a Graphical User Interface framework. A CORBA applets library for the communication with the JRCS servers has been developed. Figure 2b shows the GUI software layers; several approaches are feasible to the graphical development. Upon the JVM and the CORBA applets a user can implement hisown GUI by writing swing based applets. The option to use dynamic HTML (javascript language), is also provided through LiveConnect, a modern browsers feature that allows to integrate javascript with java. The implemented CORBA applets are in this way available from javascript code; moreover, the use of javascript allows a user to write hisown sequence of control commands. A javascript library providing a friendly interface to the JRCS system has been developed.

4 Conclusions

The integration between Java and CORBA is a really powerful platform independent distributed environment. The overhead introduced by these technologies makes JRCS suitable to control DAQs composed of up to few tens of components. Web technologies are suitable to implement a Web based GUI; by our experience, the graphical development through applets is still to prefer to the dynamic HTML approach. The first JRCS based DAQ control was tested on a real data taking environment at CERN on July 1999 for the CMS muon chamber testbeam [3]. The experience allowed to test the whole of the functionalities, and identify the inadeguaties of the system implementation.

References

1 P. J. Lucas, F. Riccardi, ”CHSM: A Language System Extending C++ for Implementing Reactive Systems”. http://www.best.com/ pjl/software.html.

2 ATLAS Run Control System design http://rd13doc.cern.ch/Atlas/Notes/029/Note029-1.html.

3 M. Bellato, L. Berti, M. Gulmini, G. Maron, N. Toniolo, G. Vedovato, S. Ventura, X.Q. Yang, ”Experiences using CMS Daq Column system at H2 testbeam area”, CHEP2000, Padova.