Astronomical Data Analysis Software and Systems VI
ASP Conference Series, Vol. 125, 1997
Gareth Hunt and H. E. Payne, eds.

CICADA, CCD and Instrument Control Software

Peter J. Young, Mick Brooks, Stephen J. Meatheringham, and William H. Roberts

Mount Stromlo and Siding Spring Observatories, Australian National University, Canberra, ACT, Australia 2611,
E-mail: pjy@mso.anu.edu.au

Abstract. Computerised Instrument Control and Data Acquisition (CICADA) is a software system for control of telescope instruments in a distributed computing environment. It is designed using object-oriented techniques and built with standard computing tools such as RPC, SysV IPC, Posix threads, Tcl, and GUI builders. The system is readily extensible to new instruments and currently supports the Astromed 3200 CCD controller and MSSSO's new tip-tilt system. Work is currently underway to provide support for the SDSU CCD controller and MSSSO's Double Beam Spectrograph. A core set of processes handle common communication and control tasks, while specific instruments are "bolted" on using C++ inheritance techniques.

1. Introduction

Computerised Instrument Control and Data Acquisition (CICADA) is a software system developed to provide access to, control of, and retrieval of data from, instruments mounted on observatory telescopes. User access is provided through a graphical user interface running on a workstation similar to those used for data reduction. Requirement and functional specifications can be found at MSSSO's Web site.[1]

2. Instrument Control Model

As shown in Figure 1, CICADA consists of a process group running on the observer's workstation and potentially more than one process group running on perhaps more than one instrument computer (which may also be the observer's workstation). Each process group consists of a parent control process and a set of subordinate processes, each responsible for the functioning of a subset of the system. The document *CICADA Design*[2] has more detailed information of the CICADA model.

Each instrument is described by a C++ class definition. This class definition is required to provide a set of generic instrument methods as well as specific

[1] http://msowww.mso.edu.au/computing/cicada

[2] http://msowww.mso.edu.au/computing/cicada/cicada_design

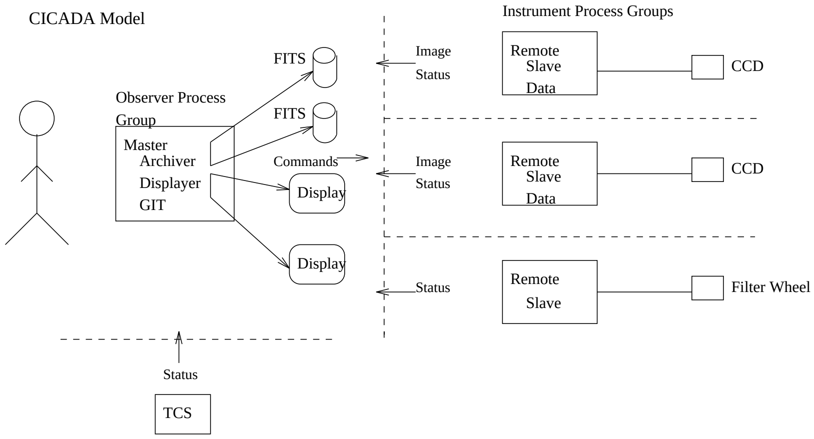

Figure 1. CICADA Process Model.

methods peculiar to the instrument. The slave process for an instrument will instantiate an object for the instrument it is talking to using polymorphism (see §3) and therefore be unconcerned with the actual type of instrument being used.

2.1. Action Requests

CICADA responds to action requests made by the observer. The user interface creates an instance of a control class object that starts the underlying processes and readies the system for action. Requests are then delivered to the `Master` process running on the local workstation which interprets the actions needed before setting the relevant slave process operating. For example, a `readout` request will result in a message being sent to an `Archiver` process to ready a FITS file and another message sent to a `Slave` process to begin the readout. The `Master` process monitors the progress of the request and delivers status information to the observer when required. The user interface process is responsible for initiating all status gathering requests.

Requests are formed using free-form text strings consisting of a verb followed by a number of optional parameter/value pairs. Requests can also be submitted in the form of a Tcl script (see §3), so that sequences of commands can be performed.

2.2. Data Delivery—Data Structures

Each instrument class generates its own request data structures using its own `generate_req` method, which is called by the control class. These data structures are designed using the `rpcgen` programming language, to facilitate the use of `RPC` for inter-machine communication.

An instrument is designed to deliver status information and, for imaging instruments, image data. When an instrument process group is running on the same machine as the observers process group, these data are delivered directly to the control class' mailbox or the waiting FITS file. Otherwise, when the instrument process group is on a remote machine, data are sent via RPC to an RPC server running on the observer's machine.

2.3. Status and Message Passing

The instrument process group maintains a shared memory area (see §3) for each process in the group to store status information. A memory area is also maintained for time-stamped, information and exception messages and is operated as a FIFO queue. Since these messages are only delivered when asked for, the queue can overflow with older messages dropping off.

3. Implementation and Techniques Used

CICADA has been implemented on Sun workstations running the Solaris operating system, Intel x86 machines also running the Solaris OS and partly on VAX/VMS systems. A main objective of the project has been to build the system in a platform independent way, some of the elements of this approach are described below:

- Standards. It is important to adhere to accepted standards so as to achieve platform independence.
- Object Oriented Programming—C++. CICADA makes good use of the object oriented techniques `encapsulation`, `inheritance`, `polymorphism`, and `exception handling`.
- RPC for network communications. We have used `rpcgen` for painlessly building inter-machine communication links.
- System V Inter Process Communications Facilities. CICADA is a system consisting of groups of cooperating processes. Much use is made of UNIX System V message passing, semaphores, and shared memory.
- Posix Threads. CICADA makes little use of the Posix thread model as yet but this is planned for improved performance and utility. At present multi-threading is used in the `Archiver` and `Slave` processes to achieve coprocessing during I/O wait situations.
- Tcl for scripting. Tcl is used to provide a command scripting facility. Standard CICADA free-form text requests can be embedded in a `Tcl` script for processing sequences of commands.
- Standard Image Display programs—Ximtool, SAOtng.
- Exception Handling—ANSI C++ standard. Exception handling is an important aspect of CICADA. Throwing an exception at the level it occurs and then catching it at a convenient location is a very elegant solution in a multi-processing, distributed application.
- Real-time scheduling. Sun Solaris provides a real-time scheduling class, use is made of this for time-critical functions in the instrument process group.

4. Current Applications

CICADA has currently been built to support the following instrument types at MSSSO:

- Astromed 3200 CCD Controller.
- MSSSO's New Tip-tilt Secondary.
- San Diego CCD Controller.

More detail on the implementation of the above can be found in the full text of this paper.[3]

MSSSO plan to use CICADA to drive its double-beam spectrograph and a planned 8k $\times$ 8k CCD mosaic, so hardware classes for these instruments are being designed.

5. Adding New Instrument Support

New instrument support from within CICADA requires the programmer to provide a C++ class definition of the hardware, a set of command definitions that are supported and a GUI or CLI module to interface with the defined class. Also important, of course, is the provision of tools for directly testing the operation of the hardware. These tools should be available for running on the instrument computer without any of the CICADA infrastructure, but by using the defined hardware class.

5.1. Creating the User Interface

CICADA has been structured so that the user interface is separated from the underlying machine-specific and hardware-specific code. At the moment, the only interface is a Motif GUI developed using Sun's Sparcworks Visual. All an interface needs to do is generate the command strings expected and understood by the particular hardware class and create an instance of the specific control class to pass these commands to.

6. Graphical Image Tool

A separate stand-alone program designed to provide quick-look image analysis functions has been provided to run with CICADA. This program has an X11 GUI and allows the observer to load images as they are readout from the CCD. It can perform functions such as line plots, statistics and image arithmetic. See *The GIT Users Guide*[4] for a full description.

---

[3] http://msowww.mso.edu.au/computing/cicada/cicada_adass96

[4] http://msowww.mso.edu.au/computing/cicada/git