Astronomical Data Analysis Software and Systems IV
ASP Conference Series, Vol. 77, 1995
R. A. Shaw, H. E. Payne, and J. J. E. Hayes, eds.

Interfacing the Tk Toolkit to ADAM

D. L. Terrett

Rutherford Appleton Laboratory, Chilton, Didcot, Oxon OX11 0QX, U.K.

Abstract. This paper describes how Tcl and the Tk toolkit have been extended to send and receive Adam inter-task messages, and how it has been used to build graphically-oriented user interfaces for various data reduction tasks.

1. Introduction

The Starlink software environment (Adam) is used for telescope control (on the JCMT & UKIRT), instrument control and data acquisition (at the AAO, La Palma, UK Hawaii telescopes), and data reduction (throughout the Starlink network). In the real-time environment of telescope and instrument control, an Adam system consists of several independent tasks cooperating by exchanging messages using the Adam inter-task messaging systems. Usually, one or more of these tasks has the job of providing the user interface.

When used for data reduction, there are also several tasks in the system but usually only two—the user interface and one of the data processing tasks—are active at any one time. This is because, although the system allows several data processing tasks to run simultaneously, controlling such a system with a traditional command line interface is clumsy and confusing.

2. User Interfaces

The Adam software environment was designed to support a variety of user interfaces; there are three currently in use:

ICL A command line interface widely used for telescope and instrument control and for data reduction.

ADAMCL A command line interface, now considered obsolete, but still used by one of the UKIRT instruments. It only runs on VMS.

SMS A menu interface that runs on VT type terminals. It is used by several telescope and instrument control systems. It only runs on VMS but an X windows equivalent that will run on UNIX is being considered.

Adam data reduction tasks can also be run without any separate user interface (i.e., directly from the operating system shell) as each task has its own, built in, simple user interface.

A graphical user interface is widely seen as the best way of satisfying the demand that Adam systems should be easier to use, and Tcl and the Tk toolkit have been selected as the most promising tools for writing such an interface. Other approaches considered were:

- Write a GUI in C using an X toolkit such as Xt or Motif. This was rejected because of the time and effort required to get a prototype that would work well enough to be evaluated and to be useful for learning what styles of GUI would be most appropriate.
- Adapt an existing GUI such as AVS. Pilot experiments with AVS and XLips indicated that the behavior of Adam tasks does not match that of the GUI well enough for this to be successful.

Tcl and Tk (Ousterhout 1994) were selected because of the ease with which prototypes can be generated and the freedom this gives for exploring a wide range of interface styles in a reasonable time. Its status as freely available software, and its acceptance in a range of disciplines (including astronomy), are also attractive features. An Adam user interface has to communicate with Adam tasks. This means extending Tcl with commands for sending and receiving messages via the Adam inter-task messaging system.

3. The Message System

A typical Adam system consists of a user interface process plus one or more Adam tasks (processes); each task supports one or more commands (up to several hundred in the case of some data reduction tasks). The tasks are independent, and can all be processing a command at the same time.

The user interface uses the message system to control the task. There are four types of message that can be sent:

SET Sets the value of a parameter (parameter values are stored by the task).

GET Gets the value of a parameter.

OBEY Tells the task to execute a command. A typical data reduction task supports commands which, to the user, look like separate applications.

CANCEL Cancels the command currently being obeyed. This is only used in telescope and instrument control tasks.

SET, **GET** and **CANCEL** messages are acknowledged by the task with a single message; an **OBEY** can result in the exchange of many messages between the user interface and the task. These messages are used by the task to request new values for parameters and to display information to the user.

4. New Tcl Commands

The following Tcl commands have been created to allow Tcl to control Adam tasks:

adam_init Connects the Tcl application to the Adam message system.

adam_send Sends a message to a task.

adam_reply Sends a reply to message from a task.

adam_receive Reads an incoming message from the message queue.

adam_exit Closes down the message system.

If there are no messages in the message system queue when **adam_receive** is called, it will block until a message arrives. In a Tk application this will cause the user interface to "hang". For a GUI that is tightly bound to a single application, this is acceptable (in fact, it feels quite natural—the interface freezes while the application is busy, just like a conventional GUI application with the interface and application in a single process). However, for a user interface that controls several tasks, it is unacceptable.

The solution adopted (pending possible modifications to the message system) is to use two processes: one running the user interface and not communicating directly with tasks, and another, called the relay process (also written in Tcl), that does. The user interface sends messages to the relay task using the Adam message system, which forwards them to the Adam tasks. Replies from the Adam tasks go to the relay task, which forwards them to the user interface via a pipe. The relay task can safely call **adam_receive** since it does not have to respond to X events and the user interface never has to call **adam_receive** since all incoming messages are delivered via a pipe. The I/O on the pipe can be integrated with the X event loop with *Tcl_CreateFileHandler*.

5. GUI Architectures

When a GUI is being added to an existing application, the GUI has to handle prompts and information messages from the application; the number of messages and the order in which they arrive is unpredictable (except for very simple applications). Information messages can be difficult to display appropriately; for example, they may contain instructions to the user that only make sense in the context of a command line interface. Interactive graphical applications will bypass the user interface when doing the interaction and this results in an inconsistent interface style. Consequently, existing application will usually need some modification before a satisfactory GUI can be constructed.

An Adam application can be written specifically to be run by a GUI by ensuring that it never requests parameter values from the user interface and by formatting messages to be interpreted by the GUI rather than the user.

References

Ousterhout, John K. 1994, Tcl and the Tk Toolkit (Reading, Addison-Wesley)