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.

Use of Inheritance Techniques in Real-Time Systems under DRAMA

T. J. Farrell, K. Shortridge

Anglo-Australian Observatory, P.O. Box 296, Epping N.S.W. 2121 Australia

Abstract. In real-time instrument control systems such as the AAO's DRAMA system, where relatively complex individual tasks control their own parts of a system (spectrograph, camera, telescope, etc), there are often significant similarities between tasks. At AAO, all DRAMA tasks have a number of standard "actions" such as "INITIALISE", "RESET" and "EXIT", and all camera control tasks provide actions such as "WINDOW", "EXPOSE" etc. We have developed a mechanism which allows tasks to "inherit" code from a generic instrument task, a generic camera task, etc. This makes the initial coding easier and also enforces the standard behavior we require of our tasks.

Such techniques for inheriting the behavior of software objects have become popular over recent years, with most attention focusing on C++ and similar languages. DRAMA is written in C, but combines ideas from the X Windows Xt toolkit with its use of named actions in tasks to provide a successful, effective, inheritance mechanism.

1. DRAMA - A Quick Introduction

The DRAMA data acquisition environment is summarized elsewhere in these proceedings (Farrell, Bailey, & Shortridge 1995). For the purposes of this paper the important point is that DRAMA systems are built from individual tasks each responding to named "actions". These actions are invoked by name through messages sent from other tasks—either user interfaces or intermediate level control tasks.

2. Standards

DRAMA imposes no limits (other than an arbitrary 20 character length) on the names of actions. However, there are clear advantages to having standards imposed. For example, most tasks have an initialization action, but some authors will call it "INITIALIZE" and others "STARTUP". Some authors may specify "EXPOSE" to start a CCD exposure and others will use "RUN". This makes it hard to use an unfamiliar task; consistency is a virtue always emphasized by user interface guidelines for windows-based programs, and it applies here as well.

In 1988 the AAO began imposing well-defined standards on all its new tasks—which then were being written in the earlier ADAM system. All tasks were to obey a "Generic Instrumentation Task" standard. This specified, for example, the use of an action named "INITIALIZE" for initialization and "EXIT" to shut down a task. In addition, we took this technique one stage further and defined "Detector Tasks", "Data Handling Tasks", etc. Each of these standards "inherited" the "Generic Instrumentation Task" standard and imposed its own additions.

The benefits of such a scheme are greatly increased when you have to write complex control tasks—tasks which coordinate other tasks. A control task can now be designed to allow you to swap lower level tasks in and out as required—say changing the type of detector system used by changing the detector task, while leaving the data handling task unchanged. In addition, the control task's initialization and reset code is easier when all tasks behave in the same way. This scheme played a major part in the success of the AAO's "OBSERVER" CCD/Infrared System.

3. Improvements with DRAMA

Under ADAM, the major benefit to this scheme came purely from having consistent interfaces to the tasks. Coding the tasks themselves was not made significantly easier (except by some 'cutting and pasting' of existing code). The top level of an ADAM task has a fixed application routine, invoked for all actions, with a big, fixed, IF-ELSE-IF sequence that calls the appropriate action routine based on the action name passed to it. This rigid structure did little to ease the inheritance of existing code.

When we started to work on DRAMA, we imposed similar standards on the DRAMA tasks. It turned out that a few relatively simple changes in style made by DRAMA solved the inheritance problem in a very elegant way. DRAMA drops the requirement for a fixed application specific routine with a big IF-ELSE-IF sequence. Instead, the application sets up an association between action names and C routines. When an 'obey' message with a particular action name arrives, the appropriate C routine is invoked directly by a 'fixed part' of the task common to all DRAMA tasks. The association between actions and C routines is dynamic and may be changed by the task itself at any time. This gave rise to the following implementation of generic standards:

- Each defined standard is associated with a package of C routines.
- Each such package has an "Activation" routine.
- The activation routine of a package sets up the association between action names and C routines, defining default implementations of all actions defined by that standard.
- An application "inherits" standard packages by invoking the activation routine of each standard it is to meet.
- The application then sets up action name to C routine mappings for new actions specific to the application and for cases where it wishes to override the default implementation of standard actions.

For example, an AAO camera control task must obey the "Generic Instrumentation Task Specification" and the "Generic Camera Task Specification". In DRAMA, such a task can pick up default implementations of all required actions by calling the activation routines of the packages associated with the standards.

4. Implementation of Packages

The easiest way to implement a Generic Package is to provide "null" routines for each action defined in the package. This does not gain you very much. Each application would have to override almost all Generic actions. The real power of such a scheme arises when it is possible to provide standard implementations of generic actions. For some generic actions, such as "INITIALIZE" this is probably not possible. However, for other actions it is obvious. An example is our "SIMULATE_LEVEL" action. This action is used to set the simulation mode of our tasks. All it has to do is to read its argument and check that it is valid. If so, it sets a global task parameter to that value. Most of our ADAM tasks ended up with their own versions of this action, often slightly modified versions of older implementations. In our DRAMA systems, there is only one such implementation, which is available when a task inherits the generic package.

Taking this idea further, many of our DRAMA programs are also written as packages that can be inherited. Consider a DRAMA program to run a simple Video Frame Grabber (VFG). By inheriting the "Generic Instrumentation task" and "Generic Camera Task" packages, almost all the required actions are defined and our program is quite simple. By inheriting the VFG package other DRAMA programs can add all the basic actions needed to control a VFG.

Although a similar effect could be obtained more conventionally by use of a VFG subroutine library, this scheme is much simpler. A subroutine library might provide the necessary action handlers, but these would still have to be associated explicitly with the named actions, and the various parameters needed would still have to set up for them. The inheritance scheme requires one simple subroutine call, with little more than a status argument, at the start of the program to give a task that has all the standard actions built-in.

A subroutine interface can of course be used to simplify the writing of actions that override the standard actions. In the case of the "Generic Camera Specification", a WINDOW action is defined to set the detector window size. A C interface to the standard action is provided so that an inheriting package may add to the semantics of the WINDOW action. For example, the basic specification does not use 'binning' (the on-chip summing of rows or columns of data). A inheriting task can add binning to the WINDOW action without rewriting the standard argument handling in the package. It redefines the WINDOW action to call the provided interface and then handles any additional action arguments itself.

5. Inheritance Techniques

We often describe our DRAMA tasks in an object-oriented way—a DRAMA task is a software object which receives and initiates messages. The package technique gives us a way of adding inheritance to this. The style is similar in some respects to that used by the X Windows Xt toolkit to inherit widgets. A significant difference is that in DRAMA existing handler routines are always overridden instead of being added to a list of routines to be executed. This is not a fundamental restriction and such a facility could be added. However, it was felt that in the real-time systems we are building it is better to avoid the overhead associated with searching such a list. We also gain here by having the behavior of the program clearly defined, a requirement of real-time systems.

6. Results

This approach has been used with considerable success in the AAO's Two degree Field (2dF) project. Using C, it has allowed DRAMA to provide many of the benefits normally expected from a well designed object-oriented scheme without the problems sometimes associated with purer O-O implementations.

References

Farrell, T. J., Bailey J. A., & Shortridge K. 1995, this volume, p. **??**