Astronomical Data Analysis Software and Systems IX
ASP Conference Series, Vol. 216, 2000
N. Manset, C. Veillet, and D. Crabtree, eds.

Three Instruments in Four Years: The UCO/Lick Data-Driven Toolkit

S. L. Allen, D. A. Clarke

UCO/Lick Observatory, University of California, Santa Cruz, CA 95064

Abstract. A panoply of new software tools has accompanied the rapid instrument construction schedule at UCO/Lick Observatory (PFCAM to Lick in 1997, ESI to Keck in 1999, DEIMOS to Keck in 2000). These Tcl/Tk-based tools maintain and interact with a relational database of instrument characteristics. They assist software design, generate thousands of lines of code, documentation, and configuration files, facilitate motor-control calibration, permit fully-automated remote testing and evaluation of performance, aid user-interface design, produce diagrams, and serve as a communications nexus between software and hardware teams. The toolkit has greatly reduced the amount of effort required from many UCO/Lick technicians, and has set new standards for quality and performance of the resulting instruments. We present here an overview of these tools and their outputs.

1. Introduction

At ADASS VI we reported initial results of employing a relational database as an integral component of the development of instrument control software (Clarke & Allen 1997). This effort was an extension of the philosophy in the existing instrument control software at Keck Observatory. At Keck, FITS keywords are used as a metaphor which is isomorphic with the complete state of the telescopes and instruments. The current state is determined via keyword reads, and commands for control are issued via keyword writes (Lupton 2000). This same metaphor is also being applied to instruments built for Lick Observatory.

UCO/Lick had already delivered the HIRES spectrograph to Keck with a keyword-based control system, but the code contained numerous tedious, repetitive, and fragile sections. Our goal was to reduce the effort required for subsequent instruments, and at the same time to increase their reliability.

As we analyzed the usage of the control system keywords we recognized patterns which led us to enlarge the scope of the effort far beyond the initial FITS-inspired goal. It became possible to model how the smallest bits of information participated in the many data flows inherent in the construction and operation of a large, automated optical instrument. Drawing from linguistics and genetics the work became known as the Memes Project[1].

[1] `http://www.ucolick.org/cgi-bin/Tcl/memes.cgi`

In a companion paper at this conference we discussed the implications of this Memes Project on the productivity of software development for instrument control (Clarke & Allen 2000). In this poster paper we presented snapshots of various software tools which evolved as we implemented the control systems for the PFCAM and ESI instruments. The full content of these snapshots cannot be meaningfully presented within these proceedings; we direct the reader to the web pages[2] which contain all of the images.

2. Interactive Database Input

Operation of the memes database requires input of all of the units of information which flow through the system. The relationships between various keywords are also modeled, and the agents which act upon the flows of memes must be defined. Several different interactive forms are used to input all these data. These forms are created by the `wisql` and `fosql` programs [3], which are generic database I/O tools that predate the memes project.

The flow of information through the instrument hardware and software is sufficiently complex that a purely text-based representation is beyond human comprehension. Figure 1 shows a pictorial database entry tool called `etcha` which makes it easy to create inventorial lists of agents; at the same time it permits documentation their dependencies on each other. The final output material from this database of meme flows aids debugging because it is possible to perform queries about the entire life history of a particular meme from creation (e.g., at a sensor) to destination (e.g., archival storage).

3. Graphical User Interfaces

Every measurable aspect of the instrument operation is available via keywords, and the memes database contains all their attributes. The `Dashboard` program is a GUI constructor which uses the database to provide a set of Tcl/Tk widgets and procedures suitable for display and control. Small custom GUIs can be constructed in as little as one hour. Figure 2 shows the main observer GUI for ESI; there are many other detail panels for keywords of less import.

4. Automated Regression Testing

The memes database contains the operating limits and nominal positions of every mechanical stage on the instrument. The `ktest` utility draws on the database of keyword characteristics to provide a simple, declarative language for constructing tests. Each stage can be run through an exercise of all its positions, including commands that attempt to send the stage into hardware and software error limits. All of the results are recorded for posterity, and related tools permit easy intercomparison of test runs.

---

[2] `http://www.ucolick.org/software/adass1999/P3-04/`

[3] `http://www.ucolick.org/cgi-bin/Tcl/database.cgi`

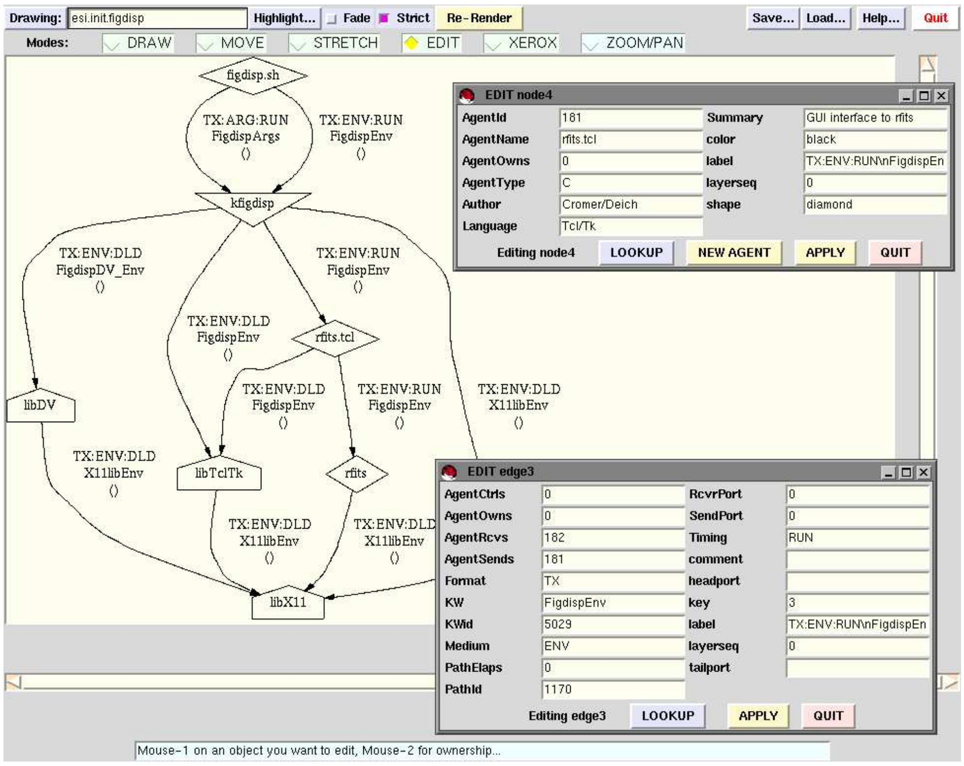

Figure 1. Meme flows between agents during real-time display startup.

The ESI test ensemble included some 9000 sub-tests and ran for over six hours. The tests found instances of unintended friction, collision of parts, and loosening screws. As a result of such testing the partially-assembled instrument was redesigned and fixed. By the time ESI reached Keck most motorized components had experienced 5000 position cycles.

5. Data Mining

The ESI instrument has over 450 control keywords, most of which indicate the state of some part of the instrument. With known attributes of each keyword it is straightforward to define a database which can serve as a log of the instrument state. This task is performed by `ktlwatch`. By watching events indicating state changes and taking regular snapshots of the instrument, `ktlwatch` collects a history of the instrument in operation and writes it to a database.

`DataMynah` provides a pseudo-natural-language interface for exploring that database. It inherits much of its vocabulary directly from the memes database; this frees the engineering user from educating `DataMynah` about underlying details. Its result is a set of highly interactive plots and other views of the instrument in operation. These plots revealed several instances of unintended correlations between ESI subsystems, leading to more pre-shipment fixes. ESI began science observations just after midnight on its first night.

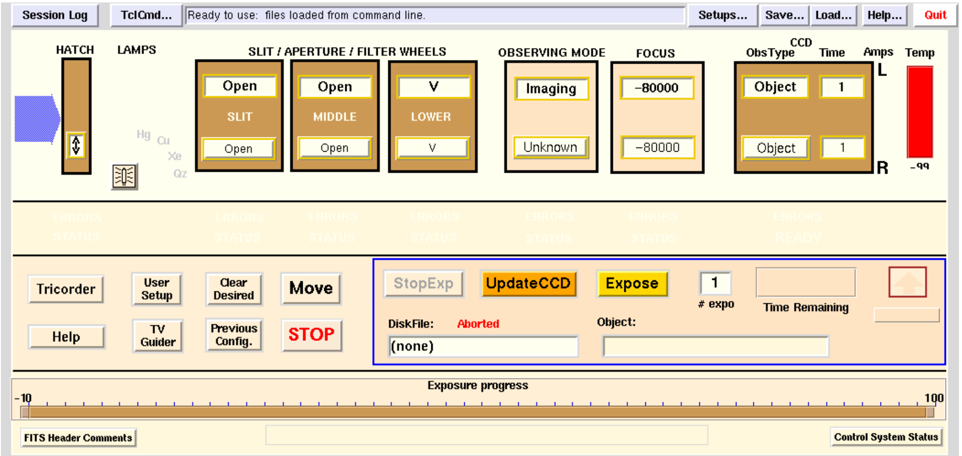

Figure 2. The basic GUI for the Keck Echellette Spectrograph and Imager (ESI).

6. Documentation

The memes database alleviates the drudgework of producing hundreds of pages of design and operational documentation, including diagrams. The on-line documentation for ESI software[4] provides a good example. All such generated documentation is kept automatically up-to-date whenever any changes are made.

7. Summary

The Memes Project reduced the amount of repetitive work required to deploy instrument control systems and user interfaces. The relational database mentality forced us to identify patterns and work out designs much more than before so that a machine could produce code. This freed the software staff to produce a wider variety of high quality output than had been possible with development techniques on previous instruments.

References

Clarke, D. A. & Allen, S. L. 1997, in ASP Conf. Ser., Vol. 125, Astronomical Data Analysis Software and Systems VI, ed. G. Hunt & H. E. Payne (San Francisco: ASP), 241

Clarke, D. A. & Allen, S. L. 2000, this volume, 16

Lupton, W. 2000, this volume, 261

[4] `http://www.ucolick.org/software/esi/`