A JAVA GUI CLIENT FOR REMOTE INTERACTIVE OBSERVING

Dione Smith, Adam Czezowski, Anthony Green, Gary Hovey, Mark Jarnyk, Jon Nielsen,
Bill Roberts, Kim Sebo, Annino Vaccarella, Greg Wilson, Peter Young.

THE AUSTRALIAN NATIONAL UNIVERSITY

RESEARCH SCHOOL OF ASTRONOMY & ASTROPHYSICS
Mount Stromlo Observatory

The Research School of Astronomy and Astrophysics (RSAA) at ANU is upgrading and extending its software support at Siding Spring Observatory (SSO) in Australia to allow its telescopes to be operated automatically or interactively with authenticated control via the Internet. In particular, the observatory's existing 2.3m telescope will be supported for interactive use, with selected new and old instrumentation, and its new SkyMapper telescope - designed to carry out an automated wide field imaging survey of the southern sky - will be supported for automated operation. TAROS (Telescope Automation and Remote Observing System) is the software being engineered to provide the support.

In this poster - the third in the TAROS series - the TAROS client application for remote interactive observing and its underlying framework are described.

Background

As part of the TAROS [1] initiative, a Java GUI client application is being developed for astronomers to conduct their observing runs from remote locations on the Internet. The application relies on two key Java technologies employed in TAROS:

- Java Message Service (JMS), for communicating between the remote observer and the TAROS server; and,
- Java Native Interface (JNI), for integrating - both newly developed Java classes and extensions to Java GUI components from the JSky [2] collection - with existing data acquisition software writtten in C++ (CICADA) [3].

TAROS posters at previous ADASS conferences presented:

- the TAROS architecture [4]; and,
- its communications infrastructure for coordinating the operation of distributed components and interactions with remote (and local) client applications [5].

TAROS client/server communication

A TAROS client communicates with the TAROS back-end via the system's key API: *ClientConnection*. *ClientConnection* is a high-level interface designed to cater for all user-related control and monitoring of the telescope, instrumentation, operating environment and data acquisition. Note that, although JMS is the message transport mechanism used for all TAROS client/server communication, the abstraction provided by *ClientConnection* means that a TAROS client can be implemented without any reference to JMS code.

ClientConnection allows a TAROS client to:

- make an authenticated connection to the TAROS back-end, with a username, password and observing run identifier;
- disconnect from the TAROS back-end;
- send a command to the TAROS back-end and get an acknowledgement in return (which specifies whether or not the TAROS back-end accepted the client's command);
- subscribe to status information updates regarding the telescope, instrumentation, operating environment and data acquisition.

Key application windows

Figure 1 shows some of the windows from the TAROS client application for remote interactive observing; this software is a work-in-progress and the screen grabs basically reflect its status as at October 2006.

Upon starting the Java GUI client for remote interactive observing, the user is presented with the *MainWindow* (smallest window at the centre of Figure 1). It serves as the application hub: from here all the other primary windows in the application are accessible. In particular, those representing key "objects" that the observer would want to control and/or monitor during an observing run can be shown via menu items in the *Observing* menu and via toolbar buttons in the toolbar.

The other key application windows include (or will include):

- a window for displaying live meteorological data from the telescope site (*WeatherWindow*, top left of Figure 1);
- a window for controlling and monitoring the telescope's operating environment (*DomeWindow*);
- a window for controlling and monitoring the telescope (*TelescopeWindow*);
- a window for controlling and monitoring the science instrument (*InstrumentWindow*, bottom two screen-grabs in Figure 1);
- a window for performing object acquisition and guiding (*GuideWindow*);
- a window for performing science data acquisition (*DataWindow*, top right of Figure 1);
- a window for displaying quick-look images of acquired science data (*DisplayerWindow*, top centre of Figure 1).

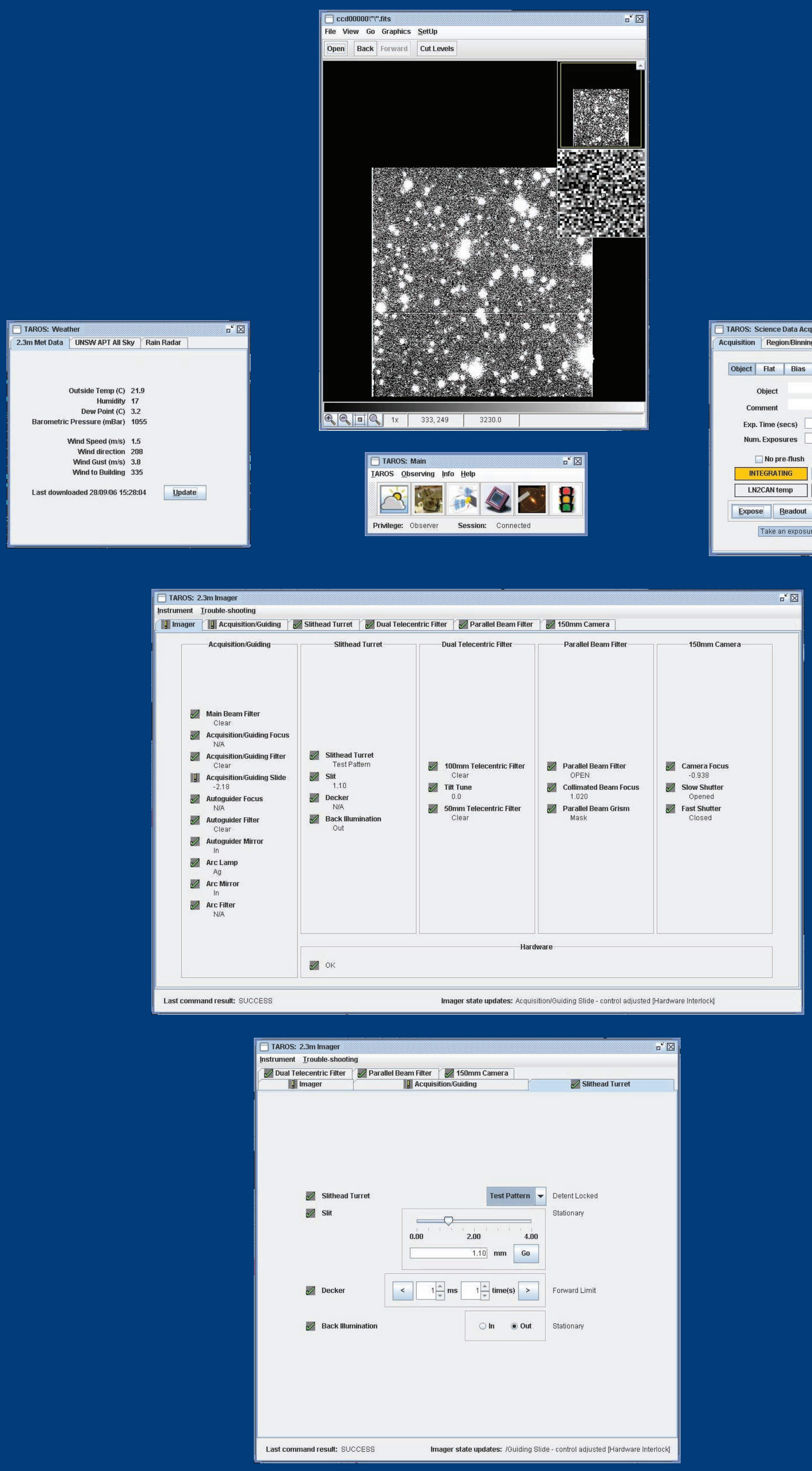

Figure 1. Windows from TAROS' remote interactive observing client

TAROS GUI framework

In tandem with developing the Java GUI client for remote interactive observing, a conceptual GUI framework has been evolved – and is evolving – from generalising specific GUI window needs. The TAROS GUI framework is based on Java's Swing GUI components, which form part of the Java Foundation Classes (JFC), and relies on Java 1.5 language features, most notably enums and generics.

The framework provides a structured basis for the development of individual GUI windows. Wherever and as far as possible, an attempt has been made:

- to separate the "application logic" from actual GUI elements;
- to facilitate customisation of GUI text;
- to systematically categorise and manage the application's user-initiated tasks; and,
- to coordinate interactions between the application's top-level windows.

Current features of the TAROS GUI framework include:

Window class abstraction

To allow GUI windows to be coded independently of the window-scheme chosen for running the application, an abstraction over *javax.swing.JInternalFrame* and *javax.swing.JFrame* is provided.

GUI building blocks to support user-initiated tasks

The framework extends a *javax.swing.AbstractAction* action to include an associated functor (an object representing a single, generic function) which knows how to execute the action. Action/functor pairs are the GUI building blocks which underpin the functionality attached to a window's menu items, toolbar buttons and the like. Representing user-initiated tasks as action/functor pairs allows them to be handled in a homogeneous fashion.

Codified notions of primary and secondary windows

The framework provides base classes for primary and secondary windows to promote the streamlined, structured development of all application windows.

GUI controls based on JComponent's client properties

The framework supports both single and multi-action GUI controls; using *JComponent.getClientProperty()* and *JComponent.putClientProperty()*, the framework can treat a number of distinct GUI elements as a unit.

Integration of ClientConnection

The framework incorporates a *ClientConnection* object for application windows to use to send commands to the TAROS back-end and to receive updates from it regarding current information on the telescope, instrumentation, operating environment and data acquisition.

Actions categorised with respect to connectivity and privilege

The framework provides a means of categorising actions as either dependent or independent of being connected to the TAROS back-end, as well as a means of identifying a privilege level with each connection-dependent action.

Primary window hub

The framework includes a singleton "hub" class to coordinate interactions between an application's primary windows.

Customisation of GUI text without recompiling

The framework works with a *java.util.PropertyResourceBundle* to supply GUI text for each action and each window.

Handling exceptions

The framework defines a fallback exception handler which can minimally log an exception (using Apache's *log4j* logging) but can also propagate it (unchecked) or cause the application to abort - in which case, there is an opportunity for the application to attempt a graceful exit.

About the Computer Section

Staff in the RSAA Computer Section are part of a larger team of engineers and technicians who have worked together successfully over recent years in designing and building instruments for the international astronomical community - in particular, the Gemini consortium. This expertise draws upon decades of experience. RSAA has an impressive track record in designing, developing and maintaining software for its own telescopes and instruments at Mount Stromlo and Siding Spring Observatories.

The School's newly constructed Advanced Instrumentation and Technology Centre (AITC) will be officially opened while ADASS XVI is in session. The facility has been specifically planned for ELT-era technology, which the School is embarking on already, as a signatory to the Giant Magellan Telescope (GMT) consortium in April 2006.

For contact details and more information about RSAA, visit the School's website at *http://www.mso.anu.edu.au.*

References

[1] RSAA Computer Section, *TAROS Critical Design Review Document* (DC-TAROS-03-CDR001), project documentation, September 2003.
[2] *JSky - Java Components for Astronomy*, http://archive.eso.org/JSky/rootpage.html
[3] Young, Peter J., Roberts, William H., Sebo, Kim M., *CICADA - Configurable Instrument Control and Data Acquisition*, ADASS VIII 1999, ASP Conference Series, vol. 172, p. 115.
[4] Wilson, Greg et al., *Telescope Automation and Remote Observing System (TAROS)*, ADASS XIV 2004, ASP Conference Series, vol. 347, p. 563.
[5] Czezowski, Adam et al., *Java Message Service (JMS) use in the Telescope Automation and Remote Observing System (TAROS)*, ADASS XV 2005, ASP Conference Series, vol. 351, p. 208.
[6] Hovey, Gary, *A Standardized Meteorological Data System* (DC-TAROS-03-TCS009), technical memorandum, July 2006.
[7] Smith, Dione, *TAROS: GUI conceptual framework* (DC-TAROS-03-GUI002), technical memorandum, June 2006.