

KOSMOS System Design Note 2.02

Title: KOSMOS User Software Considerations
Author: Jay Elias
Revision: 3
Date: March 24, 2010

Introduction

This document discusses some of the considerations and requirements for the KOSMOS User Software, from the perspective of science users. It is not intended to be a complete requirements document, but rather a summary of the things that the typical observer or support staff member will care about.

Detailed observing scenarios are laid out in the Operations Concept Document (OCD). The latest version of the OCD is posted on the KOSMOS web page.

General Principles

Four general principles apply:

1. The software should allow for efficient preparation of observations.
2. The software should allow for efficient execution of observations.
3. The software should allow for efficient reduction of data.
4. The software should allow for efficient instrument set-up and support

Since telescope time is over-subscribed, clearly principle (2) must have precedence where there is a conflict. Since it's also true that people generally spend more time reducing data than preparing to observe (unless they have bad luck in getting data), principle (3) should have precedence over (1), though it's hard to imagine significant conflicts.

In applying these principles, it will be assumed that observing on the 4-m will always take place with an investigator present, either in person or remotely. In particular, there is no requirement to set up "queue-ready" observations or programs; observer intervention is always possible.

These requirements include efficient support of the instrument by mountain support staff. At one level, this implies access to engineering/diagnostic screens that the normal observer doesn't need. These should *not* be needed for routine set-ups, however. At another, it implies an interface that is intuitive enough (and certainly documented enough) that the support staff can use it productively despite the fact that they are responsible more many more instruments than just KOSMOS.

Requirements

Efficiency at the Telescope

There are several potential sources of inefficiency at the telescope related to software. Of these, three predominate:

- Commands executed serially that could be done in parallel
- Commands that require unnecessary user input
- Commands or interfaces that are hard to use/poorly documented/hard to remember

For example, if the observer can pre-define a configuration that includes science target information, desired instrument configuration and rotator/guider configuration, which can then be loaded and executed, considerable time at the telescope is potentially saved. At the same time, the software should allow the observer to modify such configurations directly (i.e., not require creating, editing, and up-loading a new file just to make a small change in a planned observation).

Similarly, normal operation is best done using buttons on a GUI or simple pull-down menus. There may be situations where a command line typed in somewhere is the best way to do something, but these situations should be infrequent (for example, loading observing sequences [”scripts”]) and should involve simple syntax.

It should be possible to stop or abort an observation and recover without significant lost time. In this context, most observations need to end in a way that allows the data taken up until that point to be salvaged (e.g., default to always read out the array; end a nod-and-shuffle sequence after matching on/off integrations, etc.)

While KOSMOS will not do the equivalent of NEWFIRM mapping sequences, some control of telescope offsetting is needed for certain kinds of observations, and these sequences should be programmable/interactive as well. The five basic sequences are:

- Acquisition in imaging mode (requires motion of the telescope/guider and little else)
- Acquisition in long-slit mode (requires motion of the telescope/guider and potentially slit wheel)
- Acquisition in multi-object mode (requires motion of the telescope/guider/rotator and potentially slit wheel)
- Nod and shuffle (requires motion of the telescope/guider synchronized with detector control)
- Dither (requires motion of the telescope/guider synchronized with detector control, somewhat different than nod-and-shuffle however)

“Guider control” may or may not require motion of the guide probe or may just involve centering the guide star at a different point in the guider field.

Note that the offsets related to acquisition are unknown *a priori*, so the software needs to allow the offsets to be determined efficiently (some form of simple interactive tool) and then applied, and to skip unneeded positioning iterations.

In addition, calibration requirements *may* lead to a need to program sequences involving calibration lamps and the selection mirror. Similarly, it would be desirable to be able to set up afternoon calibration sequences as a single observation that does not require observer intervention every few minutes. These potential requirements will be evaluated as the design progresses.

It may be easier to break down a full observation into a small number of sequences – calibration, acquisition, science observation (for example) rather than trying to build a more complex sequence with additional internal logic tests. It is worth exploring whether the variety of observations is small enough to simply access them via a menu or buttons on a GUI (or a combination)¹.

It is important to include the ability to just do something manually. For example, an observer might carry out a pre-set observation of an object, and then at the end of the observation might want to do one more exposure. It should be possible to do this directly without redefining an observation file, and the relevant observing parameters (object name, for example) should carry over into the additional exposure.

Efficiency Reducing Data

Data handling can be thought of as comprising up to four successive levels of processing:

- DHS (data handling system) provides an “archive ready” data frame assembled using information from the CCD controller, instrument, and telescope.
- RTD (real-time display) displays the raw image data in some format; image manipulation may be possible but the processed image isn’t saved.
- QLP (quick-look processing) allows for some routine processing of the data but doesn’t result in a science-ready output. Final science reductions don’t necessarily start by using the this output either, so it can take short-cuts in the interests of speed.
- SP (science pipeline) provides a science-ready output. In the absence of a science pipeline, investigators would go through the same steps with more manual interaction. This would also be the procedure for non-standard observations – that is, observers would not be limited by the input requirements for the pipeline, but

¹ KOSMOS configurations involve only three parameters – slit wheel, filter, and disperser. (Although there are also focus adjustments, these are normally set at the start of a run and then left alone, and in any case they are not really “user options”.) All of these involve only discrete positions, and in general the disperser implies a filter (not true for direct imaging, also not true if the disperser can be used in more than one order). So the time saved by pre-programming observations may not be as much as for a more complex instrument. Overall, we need to look carefully where the additional programming effort outweighs the time saved at the telescope.

will need to understand that they would then be much more on their own for the reductions.

The first two items (DHS and RTD) are essential for pretty much any efficient operating scenario – without the DHS the observer ends up transcribing data into image headers, and without the RTD the observer will be constantly retrieving stored images to carry out acquisitions and real-time quality checks. Quick-look processing requirements need to be carefully specified – at one extreme basic processing of raw images can be done with existing software (e.g., IRAF ccdproc) so no development effort; at the other extreme the QLP can look rather like a full science pipeline, with a corresponding requirement for development resources. It is likely that resource constraints will cause the KOSMOS QLP to be closer to the first case than the second.

In the abstract, the most efficient way to reduce data is to provide all capabilities through a full science pipeline. However, the current project baseline does not include the resources to provide such a pipeline. Nonetheless, data reduction is most efficient if the user software provides essentially the same level of header information as would be needed by a pipeline (so that manual reductions can be done efficiently). In general, if there is potentially useful information available, it should be put into the image headers. Disk space is cheap, and lost information is lost forever. Specifically, header data should include:

- Instrument, detector controller, and telescope configuration
- Relevant temperature sensors, controller voltages (TBD) and other diagnostic data
- Coordinates and any information on offset patterns
- User-supplied data (target name, program and observer ID, etc.)
- Mask coordinates (if mask is used)

Keywords should conform to standards where they exist.

Efficiency Preparing Observations

There are three philosophical approaches to preparing observation sequences (“scripts”). One is to have a “standard” tool that allows observations to be prepared for any instrument on the telescope (or even, at the observatory). Such a tool must handle the full range of possible observations. A second approach is to have a tool for each instrument, which may indeed have a similar interface to the tools for other instruments, but which does no more than is necessary for the specific instrument. A third approach is to have no tool at all, and simply have users edit appropriate templates. (A fourth approach is to have no scripting capability at all – this is undesirable.)

Of these approaches to scripting, the third is unattractive because of the potential for errors – unless there is a verification tool that checks observations for mistakes and provides a useful error message. But the effort to create such a tool may well be comparable to that needed to create a preparation tool. The first approach may or may not be cost-effective, depending on the number and variety of instruments involved – but it

surely has a steeper learning curve for the individual observer, who typically uses one instrument, maybe two.

Observatory support staff will interact with more instruments, so a single tool may be more helpful to them, though on the other hand they probably don't have much need for pre-programmed observations in their support role.

A second consideration is that the observer will normally be present, so any tool needs only to produce sequences for "normal" observations. Anyone who wants to do something unorthodox may have to execute parts of the sequence manually; this is acceptable because such observations will be rare and because the observer can ensure that what he/she wants to happen actually occurs.

Details

There are some detailed requirements that need to be evaluated. For example, at present, the telescope operator selects guide stars at the time the target is being slewed to; this process is by all accounts quite efficient and does not lose telescope time. If the observer was not required to identify guide stars, the preparation tool could operate in a much more stand-alone mode. But continuing in this way assumes that very few targets do not have guide stars. Other issues involve the performance of the telescope rotator and the guide probes, the ability to define guide locations in the guider field (how many? absolute accuracy?), and doubtless others that will appear as we work through the details.

In addition, the observer/instrument does not have full control over the telescope motions, since for safety reasons large motions – slews – must be initiated by the operator and cannot be started automatically.

Additional Issues

Other CCD Uses

The KOSMOS project intends to use CCDs mounted in standard KPNO dewars. This will allow these detectors to be used elsewhere when not in use on KOSMOS. The most likely candidates for such additional use are the RC spectrograph and the echelle.

At present both instruments run using a separate interface for the detector and for the instrument itself. It would clearly be desirable to provide at least this level of functionality for the new detectors. However, this effort will not be charged to the KOSMOS project and this desire cannot be allowed to significantly increase the KOSMOS-related effort (for example, by adding to or modifying the baseline requirements).

Graphical Display

It is useful *but not necessary* to have a graphical display of the instrument configuration. That is, rather than just having a series of buttons or status displays indicating the

instrument configuration, a cartoon of the instrument shows the current configuration. This is done with MOSAIC and many instruments on larger telescopes.

If observations are scripted and/or the observer is organized, this display is not that useful, but these conditions don't always apply and a display like they may help avoid wasted telescope time.

The level of effort to provide such a display should be assessed. This particular feature should not be a primary driver in selecting the software architecture.

Versions

Version	Date	Changes
1	Dec. 22, 2009	First draft
2	Mar. 22, 2010	Minor changes, initial release
3	Mar. 24, 2010	Revisions after initial comments: additional discussion of levels of data reduction, various minor clarifications