STScI Response to STUC – IRAF and PyRAF

19 October 2006

From Summary of STUC Recommendations, April 2006

6. *We also would like to hear how the Institute will respond to the dropping of IRAF support by NOAO. What are the costs to STScI, and is pyraf a complete substitute?*

Summary:

STScI still has a number of operational dependencies on IRAF. We do not believe there is any immediate crisis regarding IRAF support, though there is some worry about IRAF's viability over a several year span, due mainly to worries about future operating system changes breaking IRAF and requiring major updates to IRAF's architecture and code. STScI has analyzed its current dependencies on IRAF, and we have a rough plan of action to address removing them in the event IRAF can no longer be used. This work can be performed in about 6 months, so we do not currently plan removing existing dependencies unless we see impending problems with IRAF on an approximately one year time frame. All new development (with one exception that is detailed), including work done in the past two years, does not require IRAF, thus IRAF dependencies are no longer being added to operational software.

Details:

First of all, there may be less than meets the eye regarding the dropping of IRAF support at NOAO. All indications are that the people that used to provide the support are still doing so (perhaps at higher levels than they previously were). How this is possible will be left to others to explain or speculate about.

On the other hand it is likely that support for major initiatives regarding IRAF may be in jeopardy. One expects that it isn't likely that major changes to IRAF will be supported. Some of these changes may be needed to keep IRAF working on certain platforms. Mike Fitzpatrick is very concerned that the GCC compiler will eventually break IRAF and that major changes are needed to IRAF memory management to prevent this from happening. Still, NOAO has significant internal dependencies on IRAF so even such an instance may result in the needed changes being made.

Nevertheless, there are concerns for IRAF's future. It now appears nearly certain that it will not be enhanced in any major way regarding its features and capabilities, nor will a great deal of new applications be written for IRAF. How long it will be maintained on popular platforms is harder to judge. We suspect that this depends on whether NOAO is able to remove operational dependencies on IRAF, at least for unsupported platforms.

Taking the question "how will the Institute respond to the dropping of IRAF support by NOAO" in its more narrow sense, and taking into considerations the apparent real situation as opposed to the stated situation regarding helpdesk support, the Institute response has been primarily to:

1) Survey the current dependencies on IRAF for its operations to understand how much work would be involved in removing such dependencies should IRAF stop working on needed platforms. (A brief summary can be found below).

2) Further commit to the fact that it's future software development will have no IRAF dependencies (as is already the case, with one exception).

PyRAF does not solve the problem because it simply uses IRAF tasks from Python, but if IRAF stops working on a platform, PyRAF does nothing to remedy that. PyRAF does allow us to remove dependencies from IRAF in a more gradual way since we develop non-IRAF software that runs under PyRAF, that can look much like IRAF tasks and thus be used in an environment that is similar to the IRAF CL. It gives us a way of transitioning to non-IRAF software that doesn't requires a drastic change in the user's approach to running such software, or require them to choose between IRAF and PyRAF (much like IDL and other systems do) since they can use both together.

The only new software development that depends on IRAF is the WFC3 pipeline. That reuses elements of the ACS and NICMOS pipelines since the detectors are so similar. All other new developments have no dependencies on IRAF (aside possibly from some on-the-fly calibration file generation tasks in OPUS)

In the following paragraphs we review the current operational dependencies on IRAF and the work needed to remove them.

The most difficult to remove is CALWP2 since it is written in SPP. It is likely that IRAF will outlive WFPC2 so we are not very concerned about having to replace CALWP2.

CALSTIS, CALACS, CALNICX, and CALWF3 all use IRAF libraries, but it would not be a great amount of work to remove the IRAF dependencies. These tasks are written in C. Their use of image I/O goes through a layer (HSTIO), which can be rewritten to use a different FITS library (CFITSIO) without a great amount of work. More work would be needed to remove the use of the STSDAS *tables* library. But we would likely write a similarly layered library that uses CFITSIO instead. There are minor uses of a couple other libraries.

The ETCs depend on *synphot*. However, *synphot* functionality has been re-implemented in Python so this dependency is nearly gone. Some calibration pipelines use *synphot* as well. They could be changed to use reference files (as do some of the other pipelines) instead.

The pipeline dependencies on IRAF (which includes OPUS, CDBS, and the SI team reference file production pipelines) are of a similar nature to those already described above:

- The pipelines make use of various IRAF/STSDAS conversion tools for converting between FITS, GEIS, and wFITS data formats. If these tools became unavailable, it is assumed that Python versions could be implemented in their place.

- CDBS reference file delivery tools (e.g. *certify, mkload,* etc.) link to IRAF libraries for image I/O and other similar features. Replacement of the HSTIO layer with an interface to CFITSIO would likely cover these dependencies.

- SI-team reference file pipeline scripts use PyRAF to call underlying IRAF and STSDAS tasks. Some could likely be reworked to call existing Python-package tasks, and others might instead be reworked to invoke IDL functions instead (image statistics functions, etc.)

- The interactive target-acquisition package is still part of the pipeline code base, though it is very rarely used, e.g., for NICMOS coronagraphic target acquisitions. This package is the one that is most heavily dependent on IRAF, with many tasks actually written in the IRAF programming language SPP. That said, it is not a particularly sophisticated package and corollaries for its functionality could likely be found in other codes that show HST aperture layouts on the sky. In light of the post-SM4 capabilities we will decide if this operational capability is still needed and if so, an investigation of alternatives will have to be undertaken.

Not taking into account the interactive target-acquisition package (last paragraph), we estimate that all this work could be done on a time-scale of about 6 months, thus have not seen any need to start on any of this work at the current time since we do not see any impending IRAF crisis on anything approaching this timescale. Should we see impending problems with the viability of IRAF on an approximately one year time horizon, we will of course reconsider whether work should start on these items.