Design issues for Science Tools

Design issues for Science Tools

I'm posting the message from the GSSC with regard to the GUI design discussed on Feb 22.

This is a follow up to the presentation by Marco Frailis of a unifying GUI for the Science Tools proposed by the DataMind group. We at the GSSC find the approach interesting and having promise, and we are appreciative of the efforts to date. We feel though that there are still a number of issues to be addressed.

The GSSC position is that this GUI interface presented may have great potential for simplifying the use of the science tools, but after the meeting on Thursday (Feb 22) there was not a clear implementation strategy. If this is to be a value added product internal to the LAT collaboration then the issues raised below are less relevant. If it is advocated that this GUI should be (or might be) distributed with the science tools then it would be good for the GSSC to be involved in determining the approach.

The first point is that the GUI should be just that, a GUI. In the presentation, it sounded like the DataMind group intends the GUI to link together the Science Tools, but not to include any direct scientific functionality. If the pipeline components start to take on some of the aspects of the science tools, e.g. any additional processing, then it seems to us that this might be leading to a split between the science tools released by the GSSC and those available to the collaboration. I think we all agree that this would not be a desirable outcome and we should be careful to avoid this. We think that having the ballistic science tool as the software modules in the GUI would achieve this in a natural manner.

A corollary to this is that the Science Tools should drive the GUI development, and not the other way around. The team should not be required to devote substantial time and/or resources to add features to the ballistic tools in order to support features in the GUI.

Another issue to consider carefully is the use of third-party components, especially if the GUI may be distributed with the Science Tools. For example, the DataMind group wants to use wxwidgets. That may be OK, but is it really necessary or could standard Python Tkinter widgets be used? Another example might be Java and CMT, both of which are used in the ProC software. The Science Tools will be distributed CMT-free, and currently do not require Java.

The point is not to forbid the use of any particular component, but rather that the GSSC (and even more so the HEASARC) will find it harder to support and distribute a system with dependencies on many components. It would be good for the collaboration to agree on a complete and well-articulated list of specific dependencies and requirements of the proposed GUI system before it is implemented.

Although Marco did not mention this in the presentation, there has been discussion of a connection between the proposed GUI and the build system. Again, ties between the new GUI and MRvcmt/CMT/ReleaseManager etc. would be a minus with respect to the GSSC distributing the GUI. The tools that will be released by the GSSC must build under the HEASARC build system and run under all the supported HEASARC platforms. The GSSC has made good progress on building the science tools in the HEASARC system using the current science tools environment. If there are any changes to the build system we should proceed carefully to preserve the work that has already occurred. In addition if this GUI system is to be distributed with the science tools it must also build under the HEASARC system.

We would also like to understand more completely how the concept of the "Pipeline" relates to the concept of an "Analysis thread" in the workbook. It was not clear from the meeting if this GUI would be able to use python scripts or would in "encapsulate" the workbook and lead users through the analysis.

It has also be stated several times that this GUI would address several windows usability problems. It would be helpful to know what the usability issues are so that we could evaluate how the GUI would help to alleviate these problems or if we should address these problems outside the GUI environment.

During Checkout 3, a number of problems were encountered in simulating and analyzing data owing to inconsistent assumptions about how time is represented. I summarized the issues here during the VRVS meeting of the Science Tools Working Group on October 5, 2005.

I'd like to make some specific proposals here that if adopted will result in **changes to the sources, tools, and data products**. These are important enough that they should be implemented before DC2. We did not reach consensus during the VRVS meeting, but I hope that we can here.

Please add notes and comments. Notes placed inline with the text like this may be easier to follow than comments at the bottom of the page. *Seth Digel, 5 Oct 2005*

1. MJDREF checking

The times that will come out of Level 1 processing for the events and the pointing/livetime history will be Mission Elapsed Time, seconds since a reference epoch. This reference time has been specified mission wide as midnight, January 1, 2001. In FITS files the conventional designation of a reference epoch is MJDREF, the Modified Julian Date for the reference. For GLAST this is 51910. It will not change.

Regarding my note to scisoftlist on the reference time for the LAT, Steve Culp says that 'midnight, January 1, 2001' is meant to be interpreted as 00:00 UTC on January 1, 2001. The MJDREF corresponding to this time is 51910, in UTC. Also, Steve points out that Spectrum Astro is no longer planning to have GPS time converted to UTC onboard, so FSW will not have to handle leap seconds onboard (which was the original understanding), and we will not have to remove them in converting to TT in Level 1 processing. *Seth Digel, 11 Oct 2005*

The problems in checkout 3 arose in part because the tools did not check the value of MJDREF, and just assumed that all times were MET, i.e., with respect to MJD 51910. Unfortunately, this was broken when an FT2 file with a different MJDREF was used.

My proposal, toned down somewhat from discussions in the VRVS meeting, was that *anything that reads absolute times from a FITS file should check the value of MJDREF and issue a warning if it is not 51910, i.e., if the times are not GLAST MET.*

The majority opinion, however, seemed to be that the tools should read MJDREF and use it (whatever its value) to derive absolute times without complaint. This would make the tools potentially more multi-mission friendly and would also mean that the times in FT1 and FT2 files *would not have to be GLAST MET.*

My argument against this position is that we know that the FT1 and FT2 times WILL be MET and that the reference date for MET will not change. I think that the sources that generate time-varying sources (like SpectralTransient for blazars, PulsarSpectrum for pulsars and GRBobsmanager for GRBs) internally use MET. I believe that the analysis tools do, too. If we allow MJDREF to be different from 51910., then the tools will need to work in terms of absolute times rather than MET.

Another opinion that was held by more than one person at the meeting was that in any case, the tools should not have hard coded in them what the 'correct' MJDREF value is. I don't think this argument is particularly compelling if one believes that MET is MET.

Still, there's no doubt that we'd be preserving the most flexibility and have a more robust analysis environment by having the tools work with absolute times rather than MET internally.

Unless converting the tools to use absolute times internally is easier than I think it is, I'd still **propose that we have the tools and simulator check the value of MJDREF and issue a warning if it is not 51910.**

Our tools have or will have multimission capabilities, and other FTOOLS will be able to process our data files. Our tools and data files live in an environment larger than the SAE, and should follow the standards of that environment. Therefore we should NEVER hardwire our MET standard into our tools, nor should any of our files EVER assume our MET standard (i.e., MDJREF = 51910 should be in all our files).

That being said, we should ALWAYS use our MET standard in all of our data files, even if it is not real data. This is the purpose of a standard. The result is that times will always be enormous, but this is the reason that times are double precision.

There are ways of dealing with smaller numbers inside of a code and yet writing data files that follow the correct standard. For example, one can use TZEROn for a given FITS column, and the FITSIO package that reads the column will automatically add the TZEROn value to all values in that column. Thus a simulation package could use time relative to the beginning of the simulation, with TZEROn equal to the beginning of the simulation (relative to MJDREF). Note that any code reading the FITS file with a FITSIO package will see a large time value relative to MJDREF.

David Band, 6 Oct 2005

2. Specification of a simulation starting time

In terms of defining sources for the observation simulator, a big inconvenience of having the sources use MET is that for times during the mission, MET will be 200,000,000+. We should have a way to specify a reference date for a simulation. This is not to be a substitute for the one true MJDREF but instead a MET value with respect to which the times for a given simulation are specified.

During our discussion at the VRVS meeting, it quickly became clear that we can't just have an additional XML tag that specifies a reference time for an entire simulation because complicated simulations can be constructed as the concatenations of several XML files. A less-appealing alternative is to have the reference MET specifiable as an optional parameter for any source that requires a time specification.

Toby mentioned during the meeting that for running Gleam he has implemented a reference time like this, something like the launch date of GLAST. It isn't clear to me that we want to have just one reference time, because a few years into the mission we'd be back to dealing with time offsets in the 100,000,000 + range, but I've asked Toby to describe how his implementation works, and I (or he) will post them here.

3. Times in data products

The tools that write FT1 or FT2 files use template files that define the keywords and many of the values, the extensions, etc. The way things are arranged now, any packaged that needs one of these template files has its own copy. For FT1, this means that we have 3 copies (which are not now currently all the same) in different packages (fitsgen, observationSim, and tip) and none of which is necessarily related to the current definitions of the FT1 and FT2 files that Masa maintains on the Web. FT2 template files are in 2 packages (fitsGen and observationSim).

The FT1 and FT2 files in fitsGen and observationSim may not be "necessarily related" to the current proposed definitions on Masa's page, but they are in fact related in a very real way since I copied them from Masa's links, changed the FT2 extension name back to "Ext1" since that value was hard-coded in several places in the ScienceTools code, and I removed the incorrect and extraneous DSS keywords that appeared in Masa's examples. As of 5 weeks ago, the FT1 and FT2 templates appearing in the tagged versions of fitsGen and observationSim are the same.

JC Oct 8, 2005

Yes, "necessarily related" was intended to mean that they are not automatically related to what is posted on the Web as the current definition. Someone has to manually make the association, as you point out. And as you also point out, what is on the Web as the current definition can be effectively out of date.

In ScienceTools v6r0p5, the FT1 and FT2 templates in fitsGen and observationSim are the same; in v6r0p4 they were not. The version of the FT1 template in tip is just plain different from the copies in fitsGen and observationSim in either version. *Seth Digel, 8 Oct 2005*

Jim proposed that we keep one copy of each of the template files in the same package and that all packages that need them look for them in this central location. Also, the definitions in the template files in the repository would be THE definitions, that presumably would be mirrored to the Web.

I think that this makes sense, but **I'd appreciate it if someone could explain why each of those packages needs the FT1 and/or FT2 templates in the first place.**

> The alternative to using template files is to set the required keywords in the code using lines like
>
> ```
> header["MJDREF"].set(51910.);
> ```
>
> thereby hard-wiring all of the keyword values in the compiled code. Under the present scheme, the code setting these keywords would be duplicated in all programs that had to write out these headers.
>
> *JC Oct 8, 2005*

Right, I wasn't asking about alternatives to using the templates but instead whether each package that had the templates actually used them. It wasn't a very bright question. I can see that observationSim and fitsGen use both of them. I think that James said at the science tools VRVS meeting this week that the copy of ft1.tpl in tip was used only by a test program. I don't know whether that means tip can get along without it; probably not. *Seth Digel, 8 Oct 2005*

Also, other packages have template files: e.g., evtbin has LatEnergyBinDef.tpl and LatTimeBinDef.tpl and pulsarDb has PulsarEph.tpl. I don't think that any other package will want to use these templates.

As James pointed out, the down side of moving template files out of the packages where they currently live is that we would have to start versioning the templates (e.g., with a new header keyword) and having the tools check the versions of the template files that they use.
versioning would be necessary. (The implication is that if a template file is kept in the package that uses it, then they are kept in synch - the code of the tool is updated as needed by the package maintainer when the template file is changed.)

So, if it turns out that we really weren't using all 3 copies of the FT1 template file and both copies of the FT2 template, the question becomes whether we should put all of the template files in one package to be centrally maintained, I hope by someone like Masa. *What do you think?*

> Under the present scheme, anybody that runs applications in fitsGen (makeFT1, makeFT2, makeFT2a) really does use the template files in that package, and anybody that uses the applications in observationSim (gtobssim, gtorbsim) really does use the template files in that package. Since the output of these applications are supposed to conform to the current FT1/FT2 standards no matter what version of the code is being run, it makes sense to put these files in **one place** and maintain these template files so that they conform to the accepted (not proposed) standard.
>
> *JC Oct 8, 2005*

I agree that they should be in one place. Is there one most sensible place? I'd have thought that it would be tip. Should all of the templates go there, or only ft1.tpl and ft2.tpl? They are the ones needed by more than one package, and they also happen to be the only template files that correspond to deliverable LAT data products. *Seth Digel, 8 Oct 2005*

Is there any convenient way to have the current versions of the template files linked to Masa's Web page of the FITS definitions of the Science Data Products? I think that the answer is only maybe. ViewCVS obviously could be used, if the revision number is known. For example, here is ft1.tpl from observationSim. Masa's Web pages have HTML versions of these templates, but with additional information that does not appear in the template file, including the formats of the values expected for each header keyword. So ViewCVS views of the template files could be expected to replace only part of Masa's pages.

The current template files do not all have analogs in the FITS formats described on Masa's page, although perhaps they should. The focus for Masa's Web page (and for the definition of science data products) has been on the products that will be exchanged between the LAT team and the GSSC. But we clearly have data formats, e.g., for binned event time series, that need to be maintained for the science tools.