White Paper on Possible NASA SMD
Open Code Policy and Practices

Acton, Charles H.; Manager, Navigation and Ancillary Information Group, Jet Propulsion Laboratory, California Institute of Technology; (818) 354-3869; Charles.h.acton@jpl.nasa.gov

Abstract
NASA's Navigation and Ancillary Information Facility (NAIF) provides planetary scientists worldwide a large suite of software used to help plan and analyze space science observations. What might this experience inform the NASA SMD open source question?

Disclaimers
Neither the author nor his group has substantive, practical experience working in a truly open source environment. These comments represent only the opinion of the author.

Background
NASA's Navigation and Ancillary Information Facility (NAIF) has, for about twenty-five years, provided NASA and worldwide researches in the space sciences an information system named SPICE[1], used to compute a large assortment of observation geometry supporting mission design, science observation planning, science data archiving and analysis, and mission engineering. The SPICE system includes a large suite of code known as the SPICE Toolkit, consisting primarily of Application Program Interfaces (APIs, also known as "subroutines"), but also accompanied by about two-dozen executables. This code is provided in a "near open source" arrangement wherein source code is always freely provided, but contributions to the SPICE Toolkit from the user community are not accepted.

The SPICE Toolkit software is made available for multiple platform/operating system combinations, and in multiple languages–Fortran 77, C, IDL, MATLAB, and Java Native Interface. (Some SPICE users have made their own, publically offered Python instances.) The SPICE system and its SPICE Toolkit has been in use since the time of the Magellan mission to Venus (1990). The software has always been 100 percent backwards compatible. The source code is highly documented and it is accompanied with a variety of technical references, tutorials and coding lessons.

The SPICE system is in use throughout NASA, as well as at ESA, ROSCOSMOS, ISRO, JAXA, and soon in South Korea. It is used by many hundreds of space science researchers around the globe, mostly within the planetary science domain, but increasingly in space physics and earth science.

[1] **S**pacecraft, **P**lanet, **I**nstrument, **C**-matrix, **E**vents

Discussion re Open Code

I think there are four primary considerations to this question:

1. Competition for NASA funding
2. The large effort needed to produce code truly useful to others, including portability, testing and documentation
3. The large effort needed to maintain open source code
4. The level of trust users of open code will have in the available products. What would be the effects of unwarranted trust, and of general lack of trust. Here, "trust" includes not only getting correct results, but also being able to understand and correctly use another's code.

Competition for NASA Funds

The NAIF Team enjoys relatively constant NASA funding, so the first of the three considerations mentioned above–funding competition–is not an issue for NAIF. I suspect this is **NOT** the case for the majority of recipients of NASA research funds.

The struggle to survive in a competitive market place seems likely to make many code producers uninterested in openly sharing their code with others. Such a presumed lack of interest can be exacerbated when a researcher is busy trying to find the next source of funds, such as writing a new research proposal or participating in a competitive mission selection (e.g. Discovery, New Frontiers).

Cost of Developing Quality Code

Over the years NAIF has well (sometimes painfully) learned the immense effort needed to produce high quality, enduring, portable code. Our rule of thumb is that one third of development time is spent on each of: code design and writing; writing and exercising test code; and code documentation. This makes our time to release much longer than it might be, but at the same time, our feeling is that we get only one shot with users: if we give them poor code they'll discontinue using it.

I believe this circumstance does NOT pervade the research community; why waste one's precious time helping others to beat you out later on? Yes, there are some excellent counterexamples, but I believe these are, and would continue to be, more the exception than the rule.

An additional, architectural consideration exists in the NAIF environment–one that may not apply to the kinds of code being considered by this committee. A contribution from an "external contributor" might conflict with planned further development of the code by the primary author (or by a maintenance consortium that has been shepherding an open source tool).

Cost of Code Maintenance
As computing architecture, operating systems, compilers and third-party libraries evolve, maintenance of code becomes increasingly difficult. The situation can be exacerbated if the code to be maintained was not well architected, implemented and documented in the first place by the code provider–a person who may have had little motivation to do so.

One recent experience of my group is telling. We offer a mission 3D visualization tool that makes use of the 3rd party Qt GUI controls library. The last Qt version update has played havoc with our visualization tool.

Trust
The "not invented here" syndrome is well known in the space science community. Sometimes this is because another's code doesn't meet one's needs, or is hard to understand/use. Sometimes it's because one simply doesn't trust anyone else's code.

The question of trust might be further complicated in a true open source environment where persons other than the originator of the code have contributed to it. Perhaps such contributions would result in more fully vetted code, but the opposite seems an equal possibility. (Obtaining more capability when multiple contributors are involved is a given.)

All of the above suggests there could be little success with simply mandating that all research code be submitted to some kind of open source repository; there would be little motivation to do so, and no viable means for evaluation and enforcement. Further, it seems unlikely NASA would provide code maintainers the resources needed to effectively do so.

An approach that might be more successful, at least with regard to code production, would be use of rewards. Whether as part of traditional research grants, or as a separate line of award, NASA might consider rewarding code producers for provision of archival quality research codes, with reward decisions made by funded peer review panels. Awards might be in the form of supplements to future grants, but the NRC panel should contemplate yet other means as well.

What constitutes high quality open source code can be debated ad nauseam. NASA would have to provide some rather specific standards, and probably some examples. The requirements should include the need to provide comprehensive, portable test codes.

The rewards approach suggested above might lead to the delivery of some amount of reasonable open source code, but does nothing about execution of the long-term maintenance of such code. Execution of the maintenance function could well be a more difficult task, especially since it drags on forever and provides the doer little in the way of accomplishment or excitement. NASA evaluation of the success of such maintenance efforts, leading to awards, seems problematic. At this writing, I have no suggestions about how to address these aspects of a rewards-based program.

As indicated in the call for white papers, further complicating matters are ownership and copyright issues brought by the home institutions of code producers. Similarly, the issue of one providing intentionally erroneous codes must be dealt with. Use of proprietary/commercial language products such as the very popular IDL and MATLAB suites would require special consideration.

Summary opinion

This is a laudable idea but an extremely tough nut to crack. It would likely be far more difficult than conducting NASA's open data policy.

In some cases the provision of open source code can and does work, and should be thoughtfully encouraged and carefully managed. But as a general requirement tied to all NASA SMD research funding, I don't believe it would be successful.

The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.