Impacts, Consequences, and Perspectives on a Future Open Code Policy for NASA Space Sciences

Ross A. Beyer
Sagan Center at the SETI Institute
NASA Ames Research Center

January 17, 2018

Summary

I am in favor of an Open Code Policy for NASA Space Sciences. This white paper details my thoughts on the impacts of such a policy, what consequences it might have for NASA Space Sciences, how such a policy should consider repositories over archives, on lessons from NASA's open data policies and proprietary code, and how NASA might both require and encourage open source code via a flexible implementation of such an Open Code Policy.

1 Personal Background

I am a Planetary Scientist. I have been funded by various components of NASA, primarily Planetary Science Division ROSES and mission programs. Although I work at a non-profit research institute, I also hold a Cooperative Agreement with NASA for work I do at NASA Ames Research Center. I perform data analysis from spacecraft and airborne instruments to perform fundamental research. I work with instrument teams to plan spacecraft missions, planetary landings, and design spacecraft mission plans and observation sequences.

I work with a lot of software.

I use software, and I write software. I also manage a development team that produces some NASA software that we have distributed under an open source license (Apache 2) since 2009 (Beyer, Alexandrov, and McMichael, 2017).

2 Impacts

If all or most NASA-funded science code were required to be open-sourced, I think the positive impacts would outweigh any negative ones. In science, we rely on the written

record of past research to allow us to move forward without having to start at first principles with every investigation.

In the modern era, so much of the science and engineering we do relies on software to process and manipulate data, and also to help us interpret that data to arrive at meaningful results. As more and more of the 'meat' of science and engineering work moves into exactly how scientific software operates, there is an increasing need to have transparency into that code. Often scientists attempt to represent the details of software they have written as high-level equations and descriptions in their manuscripts, but this frequently fails to describe the process in sufficient detail to allow a 3rd party to really 'reproduce' that work according to the scientific method.

A NASA open-source policy would facilitate that transparency, and would allow greater scientific progress. Especially in an organization that is funded by taxpayers, such a policy is well-aligned (and perhaps overdue) with respect to scientific and national goals.

3 Consequences of making existing NASA-developed code Open Source

One of the suggested topics for consideration were the positive or negative impacts of requiring existing codes, previously developed under NASA funding, to be made open source.

This is potentially a very large amount of work. Just making this a blanket requirement would probably be a bad idea, but such an initiative with a more targeted implementation could have lots of positive consequences. The benefit of making such existing codes open source would be to broaden the ability of the community to leverage that software. However, just slapping an open source license on every last source code file ever written under NASA funding and releasing it probably isn't going to be very useful for most pieces of such software. That's not to say that there aren't gems that are hidden away, or very useful pieces of code that should be shared, but the approach is important.

I would suggest two approaches, pursued in parallel: (1) offer funding via ROSES calls to re-release previously developed code under the new policy, and (2) target specific software for re-release.

So under the first approach, this would allow volunteers to make the case to NASA and their community, via a proposal, that their software is worth retooling and re-releasing under an open source license. If they can make that case to a review board, then they can get the money to pay for that effort. This would be similar to cleaning up and properly archiving old data, which is fundable under the PDART program. Properly archiving all old NASA data would be prohibitively expensive, but the competitive process allows the community to identify which historic data sets are most important to archive.

The second approach is for individuals that choose not to volunteer their code for release (on the selfish assumption that keeping that code private gives them some sort of competitive advantage) but that NASA deems important software for being shared. This

process could hurt people's feelings especially if individuals consider it 'their' software, even though it was developed under NASA funds. However, if some software that was previously developed with NASA funding would have a greater impact if it were open-sourced and shared, then a process should be devised to enable it.

4 Archives vs Repositories

We have a pretty good handle on what it takes to archive data in a way that it will be preserved and usable decades from now. However, attempting to archive software in such a way that it will still be usable for future generations is not a problem that has a good solution. Any piece of software that gets written depends on other components of the language it is written in, and the computer operating system that it runs on, if not other entire libraries.

There is some hope of 'archiving' software by not just saving a copy of the source code, but by using a virtualized environment in which to preserve the code and the web of dependencies in which its author developed it. However, one still has to figure out how to archive this virtualized environment, which is just a different iteration on the problem of 'archiving' the original software, and I do not think that there are any internationally-accepted procedures for archiving software in this way. It isn't that they couldn't be developed, but it might be a substantial effort to do so.

Instead of talking about archiving software in a way that a program could be reliably used generations from now, we should instead just talk about 'depositing' software source code in repositories where the source code can be preserved as a document of its functionality. Software source code can be saved as a plain text document and that document could be archived. Future generations may not be able to just take it and run it, but if they can see the source code, it can be used as a foundation upon which to build similar or improved algorithms in the future. Such an approach strikes a good balance between preservation for the future, and the effort needed in the present to perform that preservation.

Something like the above would serve future generations, but today's users are also an important audience for software, and may need a different solution. Source code control systems keep a record of the deposited source code, and also allow easy mechanisms to record and track changes and modifications to software. Online systems like GitHub are an example of such depositories. From the perspective of today's user of software, systems like this can be more useful than a preserved text document in a long-lived archive.

Both long-term preservation and present-day utility should be kept in mind, and may need different, parallel, solutions.

5 Lessons from Open Data Policies

Open data policies in Planetary Sciences ROSES solicitations are still relatively new, but I think that the main lesson is that the 'community' won't make this shift without both carrots

(prospect for extra funding to cover the time needed to properly document and release code) and sticks (requirements to release code as a stipulation of NASA funding).

Also, I would say that any future policy be flexible so that its implementation can be adaptable. For example, the open data policy in Planetary Sciences ROSES solicitations is a blanket policy, but flexible in that review panels judge whether the proposal is making open the 'right' data for that proposal. Not every last datum, lab note, or spreadsheet needs to be archived for an open data policy, and neither should every last scrap of shell script or list of copy commands be needed for an open code policy. The communities of scientists and researchers know where that boundary is between the important and not important and that expertise should be leveraged. A policy that is too draconian will not serve NASA well.

Additionally, 'data management plans' are required for all Planetary Sciences ROSES programs, but only some programs count the evaluation of those plans towards the final score of the proposal. Presumably, this is temporary, and it will be required for all programs eventually. I assume that this is to get the community (proposers, reviewers, and NASA officials) more comfortable with the whole process before it becomes integral to funding decisions, which is a pragmatic approach, and should be followed with any open code policy.

6 Proprietary or Regulated Code

If you were implementing a policy like this at very the beginning of NASA, you might approach this differently, but the simple fact is that there is NASA-developed code that is either itself regulated or proprietary or is built on or depends on regulated or proprietary code.

Having a on open code policy that is flexible in its implementation, as I describe above, will allow the implementation of that policy to adapt to the presence of this kind of code, and hopefully evolve away from that model where it doesn't make sense. Of course, there can be very good reasons for why such code is regulated or proprietary, and there must be allowance for that.

However, if the policy is crafted in such a way that there is positive pressure to open-source code (funding, prestige in the community, etc.), then those authors which develop code that currently is proprietary or regulated would be incentivized either to open-source their code (if they can) or break their code into modules, some of which may be open-sourced, even if not all of it can be.

7 NASA Encouragement

How could NASA encourage open source licenses for NASA-funded code? The suggested topic talks about non-policy approaches, but then concludes talking about 'enforcement' which would seem to require a policy of some kind.

A policy requiring open source code whenever NASA funds code (and having lots of paperwork to justify closed code–but allowing it) would certainly change the "default" mode that NASA-funded code is created in.

Policy aside, the best encouragement is financial: providing funding mechanisms that encourage and allow people to open-source their code. Certainly a bounty-program to open-source code previously closed is similar to the PDART effort to fund proper archiving of data that would not otherwise be archived. Allowing (and encouraging) proposals for new code to ask for funded time to properly open source the code they are creating would also create the right kind of environment.

The other way to normalize open source code is for NASA itself to be explicit about its preference for that kind of code, both when seeking solutions, and when celebrating victories. Mention it by name in proposal calls. When projects are successful, and they contain open-source code: highlight that as one of the positive aspects in press releases and announcements to the community. If word gets out that 'NASA prefers open source code' then that will factor in to decisions that people make when they use and create software in a NASA context.

In addition to providing funds and making open code a priority, NASA must also provide tools and education to make the process easy and streamlined. At the moment, as attested to in a whitepaper submitted to your committee by Dr. Rigby at Goddard, the process of open-sourcing code at NASA is fraught with difficulty and complexity. NASA must have a simple, straightforward process in place to help creators select the right license to use and education about how to properly create code under an open source license and how to properly use and attribute other code with that kind of licensing.

All of these things: policies, perceived attitude of the agency to open source code, and ease of implementation would be ways for NASA to encourage the practice of open-sourcing the software which it pays to develop.

References:

Beyer, Ross A., Oleg Alexandrov, and Scott McMichael (2017). *NeoGeography-Toolkit/StereoPipeline: Ames Stereo Pipeline 2.6.0*. DOI: 10.5281/zenodo.581187. URL: https://doi.org/10.5281/zenodo.581187.