Open Source Software as the Default for Federally Funded Software

Travis E. Oliphant, PhD
Founder and Director of Anaconda
Founder and CEO of Quansight

Summary: An open source license should be the default for all NASA-funded software with exceptions requiring specific justification for such reasons of national security, "grandfathering" previous agreements, or adhering to dependency licensing constraints. Furthermore, all NASA vendors and/or grant recipients who depend on other open-source projects should be required to show that a portion of their proceeds are sent to individuals or organizations supporting those open-source dependencies. While not a specific part of this proposal, it is instructive to note that the tracking and verification of this financial flow could be facilitated by the creation and management of one or more digital tokens that specifically benefit open-source communities.

One of the most satisfying and stimulating experiences of my life has been to observe and participate in the growth of open-source at the foundation of nearly every meaningful software stack in wide use today. Browsers, databases, operating systems, web-applications and machine-learning frameworks, are all either open-source or largely based on open-source software. In fact, nearly all proprietary software begun in the last 10 years either includes large swaths of open-source software or relies completely on open-source software for its use.

As a result, not only does NASA today depend deeply on open-source, our entire society now depends critically on open-source software. This dependency also creates an existential vulnerability as our collective ability to adequately maintain this software infrastructure rests on too few people with not enough sustainable and institutional support.

Federally funded programs can produce powerful incentive structures in society and small actions can result in powerful changes in how software development progresses. If NASA were to institute two policies in how it spends funds to support its missions, it could make a big difference in how open-source software is supported. First, all software it funds should be open-source by default. Closed-source software should be permitted as a justified exception, but the default should be open-source. Second, all software it funds should make a regular list of its open-source dependencies and regularly show that it is helping to maintain those dependencies by sending a portion of the NASA funding it receives to the people or organizations that are supporting those dependent open-source projects.

My journey with open-source started as a graduate student in Biomedical Engineering (Imaging) at the Mayo Clinic in 1997 where I learned Python and started releasing to the world extension modules to Python that eventually became SciPy and parts of f2py. While a professor of Electrical & Computer Engineering at BYU, I wrote NumPy and contributed an extended buffer protocol to Python core development while continuing to develop SciPy.

After leaving academia and working as a consultant across dozens of companies and government programs, I have also been involved in writing and growing the ecosystems around Chaco, and Traits as well as Numba, Dask, Conda, Bokeh, Holoviews, Datashader, Param, GeoViews, and

JuptyerLab. These open-source projects have been funded in numerous ways including as part of the DARPA-led XDATA program, direct company investment, venture-capital investment, and as a side-effect of consulting projects. While leading the team at Continuum Analytics producing much of this software, we built a profitable business producing open-source software that was funded directly from other companies, indirectly as part of projects for other companies, and government grants.

It is instructive to look at the growth of the NumPy community as an example of the impact that open-source can have – particularly when it comes to infrastructure software. NumPy was based on its predecessor Numeric released by Jim Hugunin in 1995 and on Python released by Guido Van Rossum in 1991. I began using Python because Numeric existed. I wrote SciPy in 1998-2010 and then NumPy in 2005-2013 only because Jim first released Numeric as open-source. Jim first released Numeric only because Guido released Python as open source in 1991. The popular Pandas, and Scikit-learn libraries were both released with dependencies on NumPy and SciPy respectively only because they existed as open-source. Hundreds of people have joined in the development of NumPy and SciPy over the past decade. Tens of thousands of additional Python extensions have been written since 2009 by a distributed collection of hundreds of thousands of people all over the world because NumPy and SciPy existed and were released as open-source. Millions of people have used this stack of software, and it has become one of the foundations of data-science and machine-learning. This non-linear ability for open-source to coordinate intelligent human development at a fraction of the cost of proprietary development is inspiring and should be promoted by NASA and all of the federal government.

The XDATA program from DARPA is another case in point. This program directed by Christopher White required participants to open-source their funded software for large scale data analytics. This led to a great many new open-source projects being released or improved as listed in a published catalog: https://opencatalog.darpa.mil/XDATA.html. The initiative created a significant amount of innovation and development that is still seeding the growth of communities world-wide around advanced analytics and data analysis. Unlike other federal programs, the software that XDATA produced or amplified such as Spark, Julia, Dask, Bokeh, and Numba is still being developed and relied upon by a large community.

Innovative and beneficial software was efficiently produced from XDATA and shows the power of "open source as default" in organizing societies around building better software together. However, without additional policy incentives, the XDATA program also just contributed to the problem of maintenance of useful open-source software by not providing any facility or incentive for maintenance. This is a growing problem for open-source. The excitement of creating new software and sharing it with the world is eventually replaced with the more important but often less popular challenge of maintaining that software for an often-impatient user community. Unless a long-term sustainability program is created, open-source communities and developers can become disenchanted and cynical as the ability to support themselves by contributing to open-source software disappears. I have witnessed this personally and in the lives of dozens of other key developers.

I saw this problem of sustainability early on in 1998 while I was delaying my PhD by writing SciPy with a wife and 3 children at home and making $18,000 a year as a graduate-student. I

wondered how I was going to provide for my family in the future while giving away software. In response, I studied economics literature in my free time and learned how economies worked and many hypotheses for why societies prospered. The philosophy of economics is a subject still debated but there are clear logical connections between trade, freedom, capital structure, and entrepreneurial activity in creating opportunities and prosperity. There are also clear activities that centralized action can do to promote prosperity. There are still a lot of unanswered questions and opportunities for economic innovation in society. There is also a lack of education and basic understanding of economics which plagues today's society including open-source communities. My studies have led me to experiment with several approaches to sustaining open-source in the world.

For the past 20 years, I have experimented with several modes for producing more open-source software while also providing for my family. These include working in academia, direct-public donations, writing books, organizing consulting companies, organizing foundations, raising venture-capital, and writing government grants. A full description of all of these activities is out-of-scope of this paper. However, it is useful to describe a few activities to underscore the importance of **incentive alignment** to increase revenue.

I have experience raising money from donors. There is a small flow of revenue that can come from the goodwill of people reminded of the need. I have seen this first hand with money raised for my academic lab via direct marketing donations, money raised for John Hunter (of Matplotlib) when he passed away, and money raised for NumFOCUS projects via donations. The money raised is hopeful and useful but woefully inadequate for any of the intended purposes. It cannot be the only way open-source is sustained.

My experience is that 2 to 3 orders of magnitude more money is available by consistent incentive alignment (and sales/marketing programs). I have learned that you have to start by requiring people to give to the project in order to get something else they want. In the companies I have built, I ensure that we sell something that is desired but otherwise unavailable and then direct a portion of the proceeds to open-source projects. This enabled me to produce and maintain sustainably a growing catalog of open-source software at Trelgol, Enthought, Continuum Analytics (now Anaconda), and now at Quansight. With very few initial resources at my disposal, I have directed tens millions of dollars towards open-source development as a result of this basic and repeatable activity.

As a non-profit organization of the US government, NASA does not sell things, however it can still powerfully affect incentives in the people it interacts with. NASA buys things from other vendors and provides grants. NASA spends several billion dollars a year. If all NASA vendors were required to send a portion of their revenues to individuals and organizations who were supporting the open-source they relied on in order to do business with NASA, this would have a powerful effect. If this same concept were extended to every publically-funded organization it would solve the open-source software maintenance problem.

There are many ways that the tracking and verification of these required financial flows could be facilitated. One way that takes advantage of the recent rise in popularity of digital cryptocurrencies is that these financial flows could be accomplished via the creation of digital

tokens that specifically benefit the open-source communities. There are several approaches for establishing this process. The essential activity would amount to vendors and grant recipients purchasing an independent audit of their software usage to track dependencies. This audit would also come with the creation of tokens that correlate with the maintenance support the vendor is required to make. These tokens would then be provided to software community participants according to their respective governance and fiscal models. If these digital tokens become popular, they could be used directly by open-source participants to purchase services and products. These tokens could always be exchanged for the currency that was specifically given to the non-profit entity which represents a floor to their market value.

Prior to the availability and existence of simple market mechanism, the requirement to spend some of the proceeds a vendor receives on the open-source software the recipient depends on could be restricted to just software grant recipients. In this way, the general idea could be applied fairly straight-forwardly to everyone to direct hundreds of millions of dollars towards productive open source that will exponentially benefit the lives of billions.

What percentage of the proceeds should be made available to support open-source? This is a good question and should be adjusted based on iterative experience with the market. In my consulting practice, 15% to 20% of the total cost of software is charged annually to maintain that software. Another number is the notion of a "tithe" or 10% that has been used by religious organizations to support charitable works for millenia. As a CEO of Anaconda, I felt that 10% to 15% of our product revenues should go directly to support our open-source work. As a result, I believe the percentage of the total amount charged that should go to support the maintenance of open-source that a project depends on should be somewhere between 10% and 20%.

NASA has a unique opportunity because of the public fascination with space and NASA's visibility to the world. A bold initiative that requires that software funded by NASA would default to open-source and that all NASA vendors who use open-source would be required to demonstrate that 20% of the revenue they receive from NASA directly supports the maintenance of the open-source they depend on would have the power to extend the transformation of the world's economy that open source is enabling.