Title: Software engineers perspective on open source projects at NASA/GSFC

Authors: Chiu Wiegand, Rick Mullinix, Justin Boblitt

Date: 12 January, 2017

As software engineers at NASA, we are fully supportive of making any software libraries and/or tools open source; so that it can benefit the community with the goal of advancing science as well as engineering research. We ourselves have benefited greatly from various open source libraries and tools when developing our own software applications at the Community Coordinated Modeling Center (CCMC). As users of open source code/packages, we find that some software is more useful to the general public than others. A simplified chart is provided below with various software types that we can think of along with their impact, if they are made publicly available to the community:

Software Type	**Description**	**Impact to the community**	**Concerns/Observation:**
Software Toolkits/library	A set of software modules or library code that can be plugged into a larger program (example: python libraries/packages, CCMC Kameleon software suite)	High: Many software modules have been written to solve specific problems. If a solution already exists to solve the same problem, it makes sense to make it publicly available for everyone. We should not waste resources on solving the same problem repeatedly.	This is not a concern but an observation: many of the software modules and libraries are language specific. Also, as users, we should be cautious that the open source library/module does not contain malicious code that can harm our system.
Stand-alone software tool with or without a GUI (graphical user interface)	Any software tool that can be installed and run on the user computer (example: many Linux/Unix command line tools like 'tail', 'head', etc., CCMC Space Weather Explorer tool, visualization tool)	Medium: If there is a software application/tool that many are interested in using, it makes sense to provide the tool to the public. An example would be the CCMC Space Weather Explorer or other visualization	Documentation on how to install or use the application is very important. A set of use cases or examples are helpful to users. Without proper documentation and an expert who can answer questions for users, the tool/application will not be much use for the general public.

		tool that might be helpful to scientists.	
Web application	Software web application that runs on the browsers (example: CCMC iSWA, DONKI, Scoreboards)	Low: If it is a web application, it is already publicly accessible to everyone as long as they have internet access. Making it open source does not provide much impact to the user community and could potentially be a security concern.	Security is a major concern, and it is uncertain how useful it will be to open source a web app. In addition, documenting how to set up and install a complex web application is not a trivial task (i.e. configurations of webservers, databases, file share etc.)
Application Programming Interface (API)	Communication interface that applications can 'talk' to each other (example: Amazon API or web services AWS)	Medium to low: The source code of a certain API is not as important to users, as long as they know how to use (or call) the API. If an API is popular, it might make sense to make it open source; so others can contribute to the implementation of the API.	Resources to maintain an open source API needs to be provided to the project.

NASA has defined a set of procedures to follow in order to properly release any software whether it is open source or not. This 'software release process' requires a number of forms to be filled out and approved by the proper authorities. The time and effort spent in getting any NASA funded software to be open source is not insignificant. Unless NASA is willing to simplify the process for everyone or provide additional incentives/funding/support to help in this effort, it is really up to the developers/owners of the software project to make their software open source or not. Logically, for our team, unless we strongly believe that a piece of software has high impact to the community, we would not take on the software release process lightly due to our limited resources.

Another issue that we have noticed which is worth pointing out is the lack of documentation on many open source software. Having source code to a certain tool is not as useful unless we also have documentation on how to install and use that tool. If the mandate is to make our

software open source, the agency should also provide guidelines and resources to properly document software including installation procedures and user's guide. In addition, a successful open source project should provide a channel for users to ask questions, which means that the project should provide technical support that can assist the user community. Support and resources are also needed by the project to effectively take in source code contributed by the public. An open source project needs to set up a robust test environment (e.g. unit tests, regression tests, automated tests etc.). So source code contributed back to the project is thoroughly tested before it can be integrated and released to the wider user community. Code management and version controls are essential for all software projects, and they are even more important for any open source project.

In conclusion, given the time it takes to set up and maintain an open source software, it is not logical to mandate all software to be open source except for high reward/high impact software packages/tools. With that said, it would be useful to the code owner if the agency can provide guideline in determining whether a piece of software should be made open source or not. Having a software system to be public accessible whether open source or not is beneficial to the community. However, it is not a 'free' for all situation like some might believe. A lot of effort is required to make an open source project successful and useful for the community.