Submission in Response to NSF CI 2030 Request for Information

DATE AND TIME: 2017-04-05 16:56:33
REFERENCE NO: 298

Author Names & Affiliations

- Christie Koehler - NumFOCUS
- Matthew Turk - National Center for Supercomputing Applications, NumFOCUS
- Gina Helfrich - NumFOCUS
- Andy Terrel - NumFOCUS

Contact Email Address (for NSF use only)

(Hidden)

Research Domain, discipline, and sub-discipline

Computer & Information Science & Engineering, Advanced Cyberinfrastructure, Free and Open Source Software Sustainability

Title of Submission

Building Castles on Sand: Risks to Scientific Research Posed by a Lack of Sustainable Open Source Scientific Software

Abstract (maximum ~200 words).

Sustainable scientific software is a fundamental component of modern research and engineering. We identify three primary challenges with respect to the sustainability of software: the funding mechanisms to support the ongoing maintenance and improvement of existing software, the development of software as a research endeavor, and the furtherance of the careers of those who develop and support research software. Current cyber infrastructure inadequately addresses these needs and we offer suggestions for improvement. We also highlight the particular sustainability challenges open source scientific software faces, due to the nature of the community and culture that supports it.

Question 1 Research Challenge(s) (maximum ~1200 words): Describe current or emerging science or engineering research challenge(s), providing context in terms of recent research activities and standing questions in the field.

Software is a fundamental component of modern research and engineering. The support of software is in itself a grand challenge in science and engineering; with the increase in the complexity of computational architectures, the volume of data produced by simulation and experiment, and the inescapable interdependence of science and computation, software is a principle venue in which much of science is now conducted.

All of these grand challenges in science and engineering are dependent on a computational software stack that is reliable, that adjusts and adapts to changes in computational architecture, that promotes a culture of reproducibility and openness, and that is available and

accessible to researchers at all career stages. In order for scientific software to continue to meet these criteria they must be what we call sustainable. Sustainable software development requires a team of people with knowledge of software development practices and who can be relied on to contribute to the production of software on a consistent full- or part-basis for over non-trivial amount of time. As they mature, gain new features and robustness, and become more central to research projects, software tools occupy a crucial place in the research process; this maturation process takes time. Both during and after this process, the software must be supported. Open source software, in particular, faces sustainability challenges due to the nature of the community and culture that supports it. Bugs, errors, and security loopholes in open source code can have a direct and often extremely negative impact on any programs or research results that rely upon it. The need for sustainable practices in the development of open source scientific software is thus tightly linked to the need for reliable, reproducible scientific research.

We identify three primary challenges with respect to the sustainability of software: The funding mechanisms to support the ongoing maintenance and improvement of existing software, the development of software as a research endeavor, and the furtherance of the careers of those who develop and support research software.

Software requires on-going professional maintenance. It is not a collection of fixed entities; over time, requirements change, bugs are found (and hopefully eradicated), communities that use it grow and shift in their interaction with it, and the underlying scientific drivers themselves are subject to change. Sustainability, broadly defined, involves a care-taking of software projects and the ecosystem that surrounds them. This includes at a bare minimum ensuring that software continues to meet the needs of its users (for instance, fixing bugs) but can also be construed to indicate an overall need of software to respond to new challenges and computer architectures. While not all software needs to be sustained and extended, there are components of the research software stack that are critical infrastructure, and maintaining these is of extra important because of the number of other scientific software projects that rely on them. As an example, NumPy (a fiscally-sponsored project within the NumFOCUS organization) is approximately 11 years old and has never received direct funding for development or maintenance. The most active core contributor to NumPy today is a retiree. This is particularly relevant because nearly all of research software utilizing Python utilizes NumPy or NumPy-derived code; this includes everything from the LIGO gravitational waves discovery to incidental data analysis scripts utilized by undergraduates. The NumPy project is absolutely critical to research and engineering, but its sustainability is far from assured.

Producing and supporting software is a full-time professional endeavor, and yet most researchers do it on the side with minimal institutional or financial support. Building the software that makes scientific research possible is generally not considered a research endeavor in and of itself. This affects not only the quality but the longevity of critical software infrastructure. Quality is affected because working on software as a side-project and working primarily with volunteers means that critical parts of the software development process (automated testing, documentation, continuous integration, security, user experience and user interface design, contributor recruiting, on-boarding and mentoring, etc.) are often skipped or given minimal attention. Furthermore, researchers are trained in their area of expertise, not as software development professionals. Unless supported by an experienced technical staff, researchers often have to implement these critical aspects of their research without knowledge of software development best practices, or must spend needless effort rediscovering and reinventing them.

We believe that the practice of creating the scientific software that research depends upon should be recognized as an endeavor in its own right—one that is integral to and inseparable from core research activities. Creating and maintaining scientific software is a fundamental research endeavor, and those who participate in creating and maintaining this software need to be funded and given academic and professional credit commensurate with conducting and publishing core research. This requires mechanisms to help advance the careers of researchers who contribute to scientific computing software.

When researchers are able to secure funding for the development of software integral to their work, it is often for starting a project rather than maintaining it. Very rarely do funding streams fund full-time developers for any significant length of time. Thus, researchers depend inordinately on unpaid volunteers from the wider open source community as well as graduate and PhD students.

Those who contribute most to computational research software need available time and incentive to do so. For this reason many projects rely on a rotating cohort of graduate students for core maintainer-ship and new feature development. However, the more mature a project is, the harder it is for researchers early in their careers to meaningfully participate. The issues appropriate for their experience level by and large have already been addressed, and as such the opportunity for credit (via publishing and citation) is minimal. Furthering the mechanisms by which contributions to scientific computing software are recognized and cited is critical to making visible this important work and providing needed career advanced opportunities to those who make it possible.

Relying almost entirely on unpaid volunteer and student labor creates instability over time for critical scientific software projects. Project

leads can't plan and execute a roadmap along a known timeline with any certainty. Core projects such as NumPy, on which hundreds of other scientific computing projects rely, can go months or years without releases, new features and bug fixes languishing on development branches. Development can grind to a halt if core maintainers switch jobs, have a change in their family demands, get sick, or simply burn out.

Question 2 Cyberinfrastructure Needed to Address the Research Challenge(s) (maximum ~1200 words): Describe any limitations or absence of existing cyberinfrastructure, and/or specific technical advancements in cyberinfrastructure (e.g. advanced computing, data infrastructure, software infrastructure, applications, networking, cybersecurity), that must be addressed to accomplish the identified research challenge(s).

Current structures are not designed to provide long-term resources of the kind needed to support sustainable software.

Current funding and support structures are not well adapted to support sustainable software. Much of the labor and expertise required to create and maintain usable scientific computing software is invisible to current cyber infrastructure. Because this critical work and expertise is not visible, it is undervalued and underfunded. Finding ways to foreground, recognize, and fund this expert-level labor is critical to ensuring high-quality, sustainable scientific computing software—and thus to ensuring reproducible, useable scientific research.

Any recognition system we create must take into account the entire software production life cycle. Current infrastructure allows for partial insight into the work of producing software, but because only one narrow aspect is measured, these mechanisms do not provide an accurate, complete understanding of the ecosystem's resource needs. For most projects commits (direct contributions of code) are the only aspects of the software production cycle that are tracked and reported on. However, code commits are but one activity of many required to create software. Invisible from our accounting of the work that goes into producing software includes: documentation, project management, community building and mentorship, testing, marketing and outreach, user support, technical product management, user experience, and design. These are not trivial tasks. They require material resources as well as time and expertise. Furthermore, most tracking systems do not differentiate paid (funded) contributions from unpaid (unfunded) contributions, so it is very difficult to determine how much unpaid volunteer labor our critical scientific software relies upon. Critical projects that on the surface appear to be robust and productive can in fact be extremely fragile, because they are dependent on an unpredictable, optionally accountable volunteer labor force.

We need to build infrastructure and processes that recognize and value the totality of work that goes into creating scientific software so that we can in turn develop explicit funding mechanisms to support this work. Such support would eliminate the fragility and uncertainty of relying on unpaid volunteer labor for critical infrastructure.

Current infrastructure does not approach scientific software development as a legitimate research endeavor in and of itself.

Despite the fact that all of the grand challenges in science and engineering are dependent on scientific software and that software engineering and development is recognized as a discipline in its own right, the development of scientific software is not generally recognized, supported or rewarded as its own research endeavor. This negatively impacts not just the funding available for scientific software endeavors and the recognition given for making advances within scientific software, but also the quality, capacity and pace of science and engineering research overall.

Science and engineering research of the type the NSF supports absolutely requires quality, reliable, highly-performant, adaptable, and innovative scientific computing software. Computer science and software engineering are recognized as their own research areas and disciplines outside of science and engineering research. It should be no different when applied within science and engineering research. Building a robust scientific software engineering practice within research teams requires building and improving knowledge-sharing infrastructure as well as providing infrastructure fundamental to the production of software itself.

Because most who contribute to scientific software do not have a computer science or software development background, knowledge-sharing infrastructure needs to support the building of scientific software engineering expertise within research units. It should specifically enable and encourage sharing software engineering advances and best practices across and within disciplines.

Furthermore, we need to build, implement, or otherwise provide infrastructure that supports basic, fundamental software engineering tasks: source code control and sharing, issue tracking and triage, project management, user experience design, user support, continuous

integration and testing, and build and deployment systems.

We also suggest creating discovery systems to foreground the value of open source computing software and enable and encourage researchers across different institutions and domains to use and contribute to the open source scientific computing ecosystem. Having a centralized, easy-to-use platform for sharing and collaboration on open source code (GitHub) has been revolutionary for the commercial software ecosystem. Imagine what would be possible if we had something similar for government-funded scientific computing software.

Current infrastructure lacks sufficient mechanisms to support the career advancement of those who contribute to scientific software.

Because much of the labor and expertise required to create and maintain scientific software is invisible and because it is not approached as a research endeavor in and of itself, the career advancement of those who contribute to it is limited by current infrastructure.

One of the main ways researchers obtain the kind of recognition that advances their career is through publishing and citations. As such, the greatest career “return on investment” (ROI) for contributing to scientific software under the current infrastructure occurs at a very narrow intersection of circumstances: When the contributor has enough domain knowledge, programming skill, and time to meaningfully contribute and when the project has increasing visibility and relevance to the scientific community and there are still novel, sizeable problems to solve. This means that graduate and PhD students and staff who kick off or contribute in the earlier stages of a scientific software project get most of the career credit. Those who come later and contribute to maintenance, incremental improvements, or undervalued activities like the ones described earlier receive little if any recognition of the kind that advances careers. This incentivizes starting but not maintaining scientific computing projects.

The impact of this is greatly amplified for mature projects with widespread adoption. Everybody uses them, but very few are motivated to maintain or improve them. One way we need to improve our cyber infrastructure is to encourage researchers whose work is absolutely dependent on these software projects to make that known, and to do so in ways that provide career-advancing credit to the individuals who continue to maintain said software.

Question 3 Other considerations (maximum ~1200 words, optional): Any other relevant aspects, such as organization, process, learning and workforce development, access, and sustainability, that need to be addressed; or any other issues that NSF should consider.

Sustaining an open source scientific software project requires much more than code contribution. The management of contributions, outreach to the community, and supporting users are all vital to the project's success. These activities require people of many different skills and backgrounds as well as different types of organizations. While universities and government labs have traditionally been the largest contributors to scientific software, corporations and volunteers are becoming critical to the development process. To this end, NumFOCUS has built an organization focusing on helping sustain the community in ways that have not been available in most academic or government environments.

NumFOCUS is a 501(c)(3) nonprofit whose mission is to supports and promotes world-class, innovative, open source scientific computing and reproducible scientific research. NumFOCUS accomplishes this mission through educational programs and events as well as through fiscal sponsorship of open source software projects. The original cohort of NumFOCUS fiscally sponsored projects—NumPy, Matplotlib, IPython, and AstroPy—are now considered foundational software tools for a wide variety of scientific research agendas. NumFOCUS tools undergird research discoveries like the LIGO gravitational waves announcement. The organization has grown to provide fiscal sponsorship for 19 open source scientific computing projects using multiple language ecosystems (R, Python, Stan, Julia, C/C++), developed and maintained by a community of approximately 4500 contributors. The conferences, events, and meetups NumFOCUS organizes for the PyData community connect thousands of individuals across five continents with learning and professional development opportunities: the PyData Youtube channel hosts nearly 800 talks with over 1 million views; almost 30,000 individuals participate in 53 different PyData meetups; and the annual PyData conference series brings together hundreds of users and developers of scientific computing tools in approximately a dozen cities each year.

The NumFOCUS sustainability project aims to connect the NumFOCUS Projects to each other to jointly develop and share information on sustainability strategies, connect Project leads with people with relevant expertise and networks (including a Projects Director and a

Sustainability Advisory Board), provide training on business and financial planning, marketing strategies and effective communication, develop and disseminate an Open Source Projects Guide to fundraising and project management and support infrastructure that would help the Projects more effectively manage finances and client and business relationships.

From this project, NumFOCUS is developing the Open Source Nurturing and Actionable Practices Kit (OSNAP Kit). This collection of practices, guidance, and materials for fostering sustainability will be accessible in online form, downloadable form, and will be maintained through a version controlled repository. This set of resources will be designed as a “kickstart” for developing or building sustainability, and will be accessible to both projects that are directly engaged with through this grant as well as publicly available. This will be a living, evolving resource for OSPs to foster better practices, including both socially-relevant and technically-actionable materials (such as templates for project governance documents, user and developer engagement, and devops recipes and scripts).

Consent Statement

- “I hereby agree to give the National Science Foundation (NSF) the right to use this information for the purposes stated above and to display it on a publically available website, consistent with the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (https://creativecommons.org/licenses/by-nc-nd/4.0/legalcode).”