Submission in Response to NSF CI 2030 Request for Information

DATE AND TIME: 2017-04-03 20:36:51
REFERENCE NO: 209

Author Names & Affiliations

- Daniel S. Katz - University of Illinois at Urbana-Champaign
- Robert Haines - University of Manchester, UK
- Kathryn Huff - University of Illinois at Urbana-Champaign
- Frank Löffler - Louisiana State University
- Marlon Pierce - Indiana University
- Michael Heroux - St. John's University
- Kyle Niemeyer - Oregon State University
- Sandra Gesing - University of Notre Dame
- Jeffrey Carver - University of Alabama
- Greg Wilson - Rangle.io

Contact Email Address (for NSF use only)

(Hidden)

Research Domain, discipline, and sub-discipline

Computational and Data Science, as applied to all science and engineering fields

Title of Submission

Sustaining Software as a Key Element of Cyberinfrastructure

Abstract (maximum ~200 words).

Research across all fields is increasingly digital, and increasingly dependent on software. However, software must be continually maintained for it to continue to function. This is part of the meaning of sustainability: software is sustainable if it will continue to be available in the future, on new platforms, and meet new needs. But sustainability doesn't happen automatically; it must be encouraged, enabled, planned, and enacted. And for both individual projects and the overall software environment, NSF can and should help make this happen.

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.

Modern research is inescapably digital, with data and publications most often created, analyzed, and stored electronically. The processes by which this happens rely on software of ever-increasing complexity. While some of this software is general-purpose office software (e.g., for email, text, presentations, spreadsheet), a great deal of it is developed specifically for research, often by researchers themselves. This

type of research software is essential to progress in science, engineering, humanities, and all other fields. It has led to Nobel prizes (e.g. Karplus, Levitt and Warshel in Chemistry in 2013, Englert and Higgs in Physics in 2013, Perlmutter, Schmidt and Riess in Physics in 2011) and likely will again, such as for the LIGO discovery of gravitational waves, which relied on both data analysis and forward models to match a particular model to an observed set of data. Software also enables tens to hundreds of thousands of individual researchers to work on distinct aspects of grand challenges, and to have their work build into a set of knowledge, for example in bioinformatics, where computer scientists develop workflow tools, and bioinformaticians develop components, which then are combined together to solve problems. Despite this ubiquity of research software, few research environments offer researchers the necessary training, experience, or institutional support to robustly maintain the software they author. In particular, researchers who support sustainability of software rarely have external incentives in the existing academic landscape, and tenure and promotion committees seldom recognize the importance of software in research. All those involved in the research environment, from the researcher to the archivist to the university administrator to the funder to the publisher, must strive to make choices that contribute to improving that environment where research software is created and sustained.

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).

Research software is a key element of the cyberinfrastructure in many fields. Most research software is produced within academia, by academics, ranging in experience and status from students to postdocs to staff members to faculty. The academic environment in which this software is developed, maintained, and used, varies widely with regards to the software processes. While the environment and culture of science have developed over hundreds of years, software has become important only more recently, in some fields over the last 60+ years, but in many others, just in the last 20 or fewer years. Furthermore, much of the software is produced by academics who are not permanent employees of their institutions, and thus, many of these contributions have a transient nature: the developers may not continue to maintain the software after they graduate or change jobs. Even when software is produced by permanent employees (faculty and staff on university funding lines), these employees are often not hired, rewarded, or promoted based on their software work.

Given this, the key need is not the software element of the cyberinfrastructure, but rather, the environment that allows it to be most effectively created, maintained, used, and sustained over time. The existing environment has been organically created by a variety of stakeholders, including funders, university administrators, research leaders, publishers, librarians, etc. For this environment to be improved so that better software is created and continues to exist, those stakeholders must make conscious decisions to improve the environment, not just to attempt to create specific software tools.

Improvements that may make a research environment more suitable for sustaining software will vary according to many parameters (e.g. software purpose, scientific domain, institution type). Training programs and curriculum may improve the software skills of scientists at all levels, and these are being discussed in another response to this RFI, titled “Research Software Training Initiative: Identifying and addressing challenges in scientific software development” and led by Frank Löffler. In many cases, such training may only be feasible when coupled with institutional incentives so that permanent staff have the opportunity to be rewarded for their software products. In addition to training and incentives, expert support in the form of permanent scientific software staff may be appropriate for sustaining an institution’s research software investments. Beyond this, improved symbiosis between institutions and open source communities will be beneficial in many cases. When research software products have the potential to engage a sufficiently large open source user/developer community to sustain the research software, researchers may need support and training to nurture, nourish, maintain, and lead that community.

The NSF has invested in campus cyberinfrastructure through the CC-* program and its predecessors. These investments have been in the form of science DMZs, computing resources, innovative storage, and “cyber teams” of applied scientific computing experts. We suggest a similar need to seed university-centered teams of cyberinfrastructure software architects and developers, with university-level commitments to sustainability. These university-centered software cyberinfrastructure centers of expertise can be networked into regional and domain-specific federated teams, with an ultimate goal of a national level federation. The key element is to provide incentives at the university level, such as within the Office of the Vice Presidents/Provosts for Research, to make long term commitments to supporting cyberinfrastructure software and the architects and designers who produce it.

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.

Cyberinfrastructure includes software, and the software must be strengthened as it is a key but currently under-supported element. Because of software collapse (dependencies on underlying software which itself changes over time, see http://blog.khinsen.net/posts/2017/01/13/sustainable-software-and-reproducible-research-dealing-with-software-collapse/), software stops working relatively quickly if not continuously maintained. But NSF funding for software only lasts a short period, equivalent to the construction phase of an MREFC. In some cases, this maintenance after construction can be handled by an open source community, if there is a community of sufficient size, if funding is available, and if the community has the skills to do this maintenance. However, this combination is rare: either the community isn't large enough, or doesn't have the right skills, or doesn't have the funding needed. Members of the community might be cyberinfrastructure software professionals, supported by their universities. In other cases, if there is not a continuing investment, the software will stop working, and the initial construction effort will have been wasted. NSF first needs to recognize that this is a problem, and second needs to recognize that for some software, sustainability requires NSF to continue to invest in maintaining the software over time (equivalent to the operating period on an MREFC). If NSF is not willing to provide this longer term funding, and the developers don’t have a good alternative method to acquire the needed resources, NSF should carefully consider its choice to fund the initial development against the short term payoff during the life of the award.

Funding to sustain software can be provided at multiple levels, which might range from continued funding of the project itself, to keep the development team together, to centralized funding of teams of research software developers at institutions, to take on projects as they move out of the build phase. Different projects will need different approaches. (Projects across NSF can also be incentivized to develop sustainable software from the start by providing funding in the initial award to cover the extra costs associated with performing these tasks during the life of the award, as the SI2 program tries to do.)

To call attention to this need, and to make the best award decisions, NSF should also require a Software Management Plan for all proposals. Software Management Plans should provide a justification for the development of new software, having considered the existence of similar software already available, and should detail how (and for how long) any such new software is to be sustained. This might include plans to build a community around it, or a request for further support from NSF. Both Data and Software Management Plans must be public and should be machine-readable for two reasons:

1. The science community (reviewers and program officers) needs to thoughtfully consider, at all stages of proposal review and project management, the public interest and NSF's requirement that investigators and organizations have the responsibility as members of the scientific and engineering community to make results, data, and collections available to other researchers.

2. Software and data management plans should be public documents, stored as archival documents, similar to how papers, software, and datasets are stored, so the record of what a project planned can be later verified by other scientists, including peer-reviewers of future projects by the same investigators. As shown by Van Tuyl and Whitmire (http://dx.doi.org/10.1371/journal.pone.0147942), proposers compliance with their own data management plans have been somewhat problematic.

The Software Management Plan might be combined with the Data Management Plan, but if so, it should be a true combination with a new name, not a slightly expanded Data Management Plan.