Software in Science: a Report of Outcomes of the 2014 National Science Foundation Software Infrastructure for Sustained Innovation (SI^2) Meeting

Beth Plale
School of Informatics and Computing
Indiana University

Matt Jones
Director of Informatics Research and Development
National Center for Ecological Analysis and Synthesis

Douglas Thain
Department of Computer Science
University of Notre Dame University

Abstract

The second annual NSF Software Infrastructure for Sustained Innovation (SI^2) PI meeting took place in Arlington, VA February 24-25, 2014. It was hosted by Beth Plale, Indiana University; Douglas Thain, University of Notre Dame; and Matt Jones, National Center for Ecological Analysis and Synthesis.

This report captures the challenges and outcomes emerging from the meeting over the four topic areas discussed i) Attribution and Citation, ii) Reproducibility, Reusability, and Preservation, iii) Project/Software Sustainability, and iv) Career Paths. The report is an academic synthesis with credit to all the participants and to the notetakers who took prodigious notes and synthesized the results upon which the conclusions of this report are derived.

I. Introduction

NSF's vision of a Cyberinfrastructure Framework for 21st Century Science and Engineering (CIF21) identifies advancing new computational infrastructure as a priority for driving innovation in science and engineering. Innovation occurs through advances in computing facilities, scientific instruments, software environments, advanced networks, data storage capabilities, and the critically important human capital and expertise. Software is thus an integral enabler of computation, experiment and theory and a central component of the new computational infrastructure. The Software Infrastructure for Sustained Innovation (SI^2) program of CIF21 supports sustainable software in support of innovation in science and engineering.

The second annual NSF Software Infrastructure for Sustained Innovation (SI^2) PI meeting took place in Arlington, VA February 24-25, 2014. It was hosted by Prof. Beth Plale, Indiana University Pervasive Technology Institute (PTI), Douglas Thain, University of Notre Dame, and Matt Jones, University of California Santa Barbara with additional organization provided by Robert Ping, Indiana University PTI.

The workshop, with an attendance of 85, consisted of series of sessions, each of which began with a presentation followed by small-group discussions guided by questions suggested by the meeting organizers. Each group had their own online document where they took notes and recorded short summaries of their conversations. Additionally, there was a poster session that allowed the meeting attendees to share their research, exchange ideas, and find collaborators.

The meeting began with a welcome by Professor Beth Plale, who emphasized the importance of addressing societal pressures in meetings that have software infrastructure as their main theme. Societal pressures can be addressed in a variety of ways. The agenda of this meeting focused on the following topics: attribution and reproducibility, software sustainability, and community and workforce development. By discussing these issues, the meeting attendees were encouraged to identify the most difficult challenges and brainstorm possible solutions, with the goal that these sessions would both stimulate ideas and nurture collaborations among the various SI^2 projects.

Dan Katz, Program Director of the NSF Division of Advanced Cyberinfrastructure provided an introduction to the meeting. He discussed infrastructure in the context of both big science and long-tail science and, by contrasting and comparing them, identified the main challenges in infrastructure development and research that have motivated NSF's vision of software as part of cyberinfrastructure.

The NSF vision is grounded in understanding that software is a crucial component of the scientific research ecosystem, because it offers capabilities to advance and accelerate scientific inquiry, enable collaborations, transform practices, and stimulate education and workforce development. The NSF intends to take a leadership role in enabling cyberinfrastructure for science and engineering research and education as part of its "Cyberinfrastructure Framework for 21st Century Science and Engineering" (CIF21) vision statement[1]. A unifying theme across the CIF-21 vision is reducing the complexity of software and promoting the integration of scientific software across academic disciplines, education, and industry. Katz encouraged participants to watch or join the Working towards Sustainable Software for Science: Practice and Experiences (WSSSPE) community[2] - a community that began with a workshop and that continues to explore similar issues of software innovation and reproducibility, workforce development, and community engagement.

II. What Makes Scientific Software Unique?

[1] See http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=504730
[2] http://wssspe.researchcomputing.org.uk/

Software is fundamentally computer code. It can be delivered to end users in multiple formats, ranging from an archive that a user downloads and builds to an executable or a service running on a remote system to which a user connects. Especially at large scale, software is generally difficult to design, implement and then maintain, and the software needed by the science, engineering, and education communities is particularly complex. Software must be reliable robust, and secure; able to produce trustable and reproducible scientific results; yet its architecture must be flexible enough to easily incorporate new scientific algorithms, new capabilities, and new opportunities provided by emerging technologies. Software also must be supported, maintained, developed and eventually replaced in part or in entirety, over its lifecycle.[3]

III. Attribution and Citation

As Neil Chue Hong, Director of the Software Sustainability Institute, UK stated in this introductory talk in the session on attribution and citation, current research practices do not reward the research programmers who create and maintain well-engineered software. What are the forms of credit by which a research programmer who works in the academic or research lab setting receive the credit they need for promotion and career advancement for the critical work that they do in advancing science and scholarship? Dr. Hong argued that current research practices do not reward the research programmer who creates and maintains well-engineered software. For instance, the final publication about a science outcome that utilizes software often glosses over the contribution that the software made to the result, if it is mentioned at all.

The several key observations emerging from this session are summarized as:

- Software papers are a way of giving the research programmer publication credit for the software itself. It also establishes a single point of reference (citation) to the definitive description of the software, so that subsequent science and software papers can reliably cite the definitive reference. This observation is made with the recognition that software papers are an imperfect solution because they are traditionally a 'flat' system for attribution. With digital objects of research a hierarchical system that would allow attribution for prior contributions is needed. With dependencies in software and multiple software objects, subcomponents need to be 'counted' in impact metrics, allowing for reward to reach as many contributors as possible.

- How to measure impact and the intellectual contribution of software developers? The number of users or downloads is not a good measure. Citation is not the only form of credit, so alternative metrics of contribution and attribution are needed.

- Standardization is important; it encompasses standards of citing software (e.g., citation styles; norms of crediting large groups of contributors; indexing; provenance), but also

[3] http://www.nsf.gov/pubs/2012/nsf12113/nsf12113.pdf

standards of software descriptions (e.g., the why, what, and how aspects of scientific software; metadata), standards of evaluation (e.g., expert feedback, RunMyCode, software comparisons, and persistent identification standards (i.e., going beyond DOIs and repositories;challenges with old software that won't work).

- An open challenge to the community is to create an economy of credit so that appropriate investments are made, and participants are rewarded. The beneficiaries of the economy of credit are primarily the developers, the data science research programmers who develop software in a scientific setting.

IV. Reproducibility, Reusability, and Preservation

Software plays a central role in the long term repeatability and reproducibility of computationally-based science. But the mechanisms and best practices for handling the long term availability of software are still in the early stages of development. Sites such as RunMyCode provide a means to create a citable "snapshot" of code that is a companion to a standard research publication. Code repositories serve a somewhat overlapping purpose, but are generally used for more complex and dynamically developing software. Software publications serve a different purpose as they provide the definitive description of the software but do not serve as a record of the code itself.

Services exist for sharing citable code snippets, serving as a means for code sharing. These snippets can be citable. Services include GitHub Gists, https://help.github.com/articles/creating-gists, Python Notebook Viewer http://nbviewer.ipython.org/, or Wakari https://wakari.io/.

The several key observations emerging from this session are summarized as:

- The existence of long term software repositories promotes reproducibility and reusability of scientific software. They could be made more easily useable by the development of depositing standards of how to prepare software for reuse and provide a searchable list of what they have funded, which would be stable and well-documented. In addition, there could be a software showcase space. This could be modeled after repositories such as GitHub, though participants argued that services like GitHub have done a fine job in the role of long-term repository for software.

- Assistance from a third party organization can be helpful. Participants acknowledged that it might be useful, especially, in finding and reusing software. Software incubators or institutes were mentioned as an additional help with technical resources (e.g., CPU cycles) as well as in business model development.

- Software papers were discussed as a possible contribution to software reuse and reproducibility. While software paper is considered a good idea in general, there were many arguments of why this may not be an efficient approach to citations and reproducibility. One issue remains whether there is scientific value and venue for software papers. Another issue is at its core, the need for accurate indexing and citation of software papers is important. For instance, how to recreate the exact version and the environment to be able to reproduce the claims? The onus is on reviewers to ensure quality and proper attribution.

- Reproducibility, reusability, and preservation challenges are related to the previous challenge, i.e. attribution and citation. The observations important in attribution and citation, such as versioning and standards, have implications on the challenges in this section. The approach to software versioning and standards will affect reproducibility, reusability, and preservation.

VI. Project/Software Sustainability

The long term sustainability of software that is in use and serving a useful purpose in scientific and scholarly research is an ongoing issue.   Dr. Craig A. Stewart, Executive Director of the Indiana University Pervasive Technology Institute, opened the session on software sustainability with his presentation "Initial Findings from a Study of Best Practices and Models for Cyberinfrastructure Software Sustainability"[4]. The study, funded by an NSF EAGER grant, surveyed members of cyberinfrastructure projects to investigate their interaction and governance models. Among the factors mentioned as contributing to the success and sustainability of cyberinfrastructure (CI) projects were capabilities and features of a software product, total cost of ownership, committed leadership, long-term availability, and reliability and maturity of the code. User engagement and tight control over the code base were two pieces of advice that Stewart offered.

The several key observations emerging from this session are summarized as:

- Participants addressed the issues of external funding, user base, sustainability, and competition in NSF-funded cyberinfrastructure projects. Participants acknowledged that they do not belong to a single community, so perhaps, a way to advance sustainability of the CI projects and to build a user base is to focus on labs as intersections of multiple interests, needs and efforts. There are examples of labs in nuclear physics that keep software and communities around it alive. Sustainability of software is a common interest that can bring SI^2 communities together.

- Another alternative for sustainability is to delegate support of software to third parties. The idea of software institutes was discussed as a way to provide assistance with software sustainability. The institutes could also help with workforce training. A participant who is a freshwater biologist brought an example of a postdoc who keeps “reinventing the wheel” by developing software for their needs because most of the software is poorly documented, poorly architected, and is hard to use. Having better

[4]https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxzaTJwaW1lZXRpbmcyMDE0fGd4OjZlODZjOWMwYTM5Yzg2MzQ

practices of sharing and supporting existing software could help to save resources.

- Software Institutes would help the SI^2 community with the issue of sustainability of software if they serve as a marketplace for technology transfer, partnerships, and education. NSF could specifically target some of their programs to address that. While the concept of Institutes hold a lot of promise, business models for software institutes are vague, scientists may not be the best candidates in the development of business models and models of sustainability, and ultimately scientific software is highly specialized and serves niche markets. Consequently, any sustainability plan for software arising from NSF must also recognize that funding for specialized scientific software generally will not arise from traditional business models in which commodification of the software for a large market segment is the main goal. While a few communities (e.g., health care) may have the critical mass for that strategy, the majority of disciplines do not. Instead, sustainable business models for scientific software will likely involve continued governmental investment, perhaps with a renewed examination of how to increase the efficiency of software development programs, reduce redundancy, and increase collaboration among investigators. This last point is of particular importance because the very nature of NSF's competitive funding model, a model in which novelty is awarded above all else in funding decisions, drives the diversity of approaches and lack of integration among software projects. Thus, one of the largest changes that could be made to promote software sustainability would be to re-align the NSF funding model to encourage collaboration over competition, and robustness over novelty.

VII. Career Paths

The data science research programmer is a software developer works in an academic or lab setting and develops software that is used in support of science and scholarship. The academic or lab setting may be oriented to software innovation, but is more likely oriented to non-IT research. The path that an individual takes to find themselves in the data science research programmer role is varied: 1) He or she may come out of a science discipline and have taken an interest in technology so acquired software development skills. These people have strong discipline knowledge, but do work that is of a software nature. 2) The person may come out of an informatics background, and have been trained in both discipline and computer science skills (e.g., the "bioinformatics" person). 3) The person may have come out of a computer science background, and have acquired enough expertise in one or a small number of science disciplines to be effective. Regardless of how they got there, the data science research programmer is characterized by being one in an academic or lab setting where he/she architects, develops software and tools in support of science and scholarship.

The career of the data science research programmer is frequently not stable over the long term. Labs are grant funded, this person is often not tenured, and may not even have the simple benefits of being a research faculty member (which may include a small commitment to providing bridge support should grant funding hit a dry spell.) Coupled with this, as indicated earlier in the section on citation, the incentives for this career path are not well structured because publications on the science or scholarship produced by the research group are focused on the primary result, and fail to acknowledge the innovation in the software (which may have research value in and of itself.)

The session on career paths began by drawing inspiration from a new and large scale effort to bring about cultural change in academia around data science. Joshua Greenberg of the Sloan Foundation and Chris Mentzel of the Moore Foundation described the Data-Driven Discovery Initiative, a recently funded initiative to support innovation through the creation of data science hubs at major research universities and through investigator awards and data science projects[5]. The hubs involve partnerships within and among stakeholders of the New York University, the University of California, Berkeley and the University of Washington. The partnerships are envisioned to be very multidisciplinary, with the main goal to effect change in university culture around data, rather than to build a data science center or provide resources. The core question in this initiative is "How can data scientists, data tools builders and researchers from natural and social sciences can all be engaged in data practices so that it stimulates discovery and has a long-lasting impact?" The funding of this initiative will go towards various staff positions and toward what speakers called "a lot of connecting tissue," i.e., communicative and administrative support that facilitates interactions and advances projects.

While the problem of career path for the research programmer is complex, an interesting idea emerged in the form of Communities of Practice (CoP). The discussion of CoP, stimulated by an introductory talk on community of practice given by Dr. Inna Kouper at Indiana University. Communities of practice (CoP) are groups of people who share a concern or a passion for something they do and learn how to do it better as they interact regularly[6]. CoPs are different from project teams and communities of interest in their focus on developing members' capabilities and learning by doing. The CoP approach may facilitate the development of better solutions by bringing together diverse perspectives, it can reduce cost and time of development by sharing practices and knowledge and improve community success and sustainability by increasing trust and confidence.

In the subsequent discussion participants considered the concept of CoP in the contexts of projects from various domains. They acknowledged that there is a potential negative cost to viewing the SI^2 community as a single community. It was noted that there is a great diversity in members' backgrounds, interests, and goals. Software developers and researchers may not consider themselves as one community. In some (sub)communities the concepts of using software are very different. On the other hand, there are benefits of viewing SI^2 as a single community - coming together and addressing common challenges and learning what works and what does not can be beneficial for everyone.

Participants agreed that this is a unique community and it must be nurtured. The emergence of

[5] http://www.moore.org/programs/science/data-driven-discovery

[6] Wenger-Trayner. E. Communities of practice, a brief introduction http://wenger-trayner.com/theory/

supercomputing and use of workflows are needs that cut across sciences and can help to form a community. The following commonality can help to build this community - work on tools -> help scientists -> contribute to scientific innovation. Benefits - a community approach can help to define good and best practices. Problem - a lack of common language or professional jargon prevents members from finding shared goals and interests.

The several key observations emerging from this session are summarized as:

- The SI^2 institutes can take the lead in establishing CoPs and maintaining connections.

- For communities of practice to be viable across the entire SI^2 community, the SI^2 community has to be better defined. Is doing that meaningful? The answer is 'yes'. It is a community of scientists who know that their path to new understandings is computation and those who are experts to provide the tools to allow this to happen. Computation is a necessary ingredient to drive this field, so in short the field is "technical expertise + domain problems". The focus on shared problems rather than interests is important. When people face similar problems, they can get together.

- How can Communities of Practice thinking be applied to create stronger support group across and between projects to give this poorly represented group (the data science research programmer) a sense of belonging?

- It was noted that there is a shift occurring in careers from individuals doing science to team-based science. Team science is different because it requires knowing how to collaborate and sometimes how to cross disciplinary boundaries. Crossing the boundaries between a discipline and computer science is another skill that is useful in teams. Those who can do that are valuable people and there should be jobs from them. Research programmers should be encouraged to develop their skills at crossing the boundaries between a discipline and computer science. This may include developing soft (people) skills as well as hard (technical/scientific) skills.

- There remain sizeable challenges of finding and hiring the right people, importance of making explicit career paths, appropriate recognition, and bridge funding.

Acknowledgements

The organizers thank the attendees of the 2014 SI^2 PI meeting for the stimulating discussions. Additional organizational support provided by Robert Ping, Indiana University Pervasive Technology Institute. Drs. Inna Kouper and Miao Chen attended the meeting where they faithfully recorded the plenary and breakout discussions. Their notes provided an excellent foundation from which conclusions in this report are drawn. The workshop web site can be found at https://sites.google.com/site/si2pimeeting2014/home. This workshop and resulting products funded by the National Science Foundation under award # 1419131.