Workshop Report

2015 Software Infrastructure for Sustained Innovation (SI^2) Principal Investigators Workshop

April 14, 2015

Report from the National Science Foundation-funded workshop held February 17-18, 2015, at the Westin Arlington Gateway in Arlington, Virginia for Software Infrastructure for Sustained Innovation (SI^2) Principal Investigators.

Written by:
Frank Timmes, Arizona State University
Stan Ahalt, RENCI, University of North Carolina at Chapel Hill
Matthew Turk, NCSA, University of Illinois at Urbana-Champaign
Ray Idaszak, RENCI, University of North Carolina at Chapel Hill
Mark Schildhauer, NCEAS, University of California, Santa Barbara
Richard Brower, Boston University
Chris Lenhardt, RENCI, University of North Carolina at Chapel Hill
Karl Gustafson, RENCI, University of North Carolina at Chapel Hill

This material is based on work supported by the National Science Foundation under grant number 1521388. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Table of Contents

1. Executive Summary

Software is a crucial enabler of computation, experiment, and theory, and a primary modality for realizing the National Science Foundation's (NSF) Cyberinfrastructure Framework for 21st Century Science and Engineering (CIF21) vision. Within this vision, the Software Infrastructure for Sustained Innovation (SI^2) program has the overarching goal of transforming innovations in research and education into sustained software resources that are an integral part of the cyberinfrastructure. SI^2 is a cross-foundation program that generates and nurtures the interdisciplinary processes required to support the entire software lifecycle and promote the successful integration of software development and support with innovation and research (see Figure 1). SI^2 supports vibrant partnerships among academia, government laboratories, and industry for the development and stewardship of a sustainable software infrastructure that can enhance productivity and accelerate innovation in science and engineering.

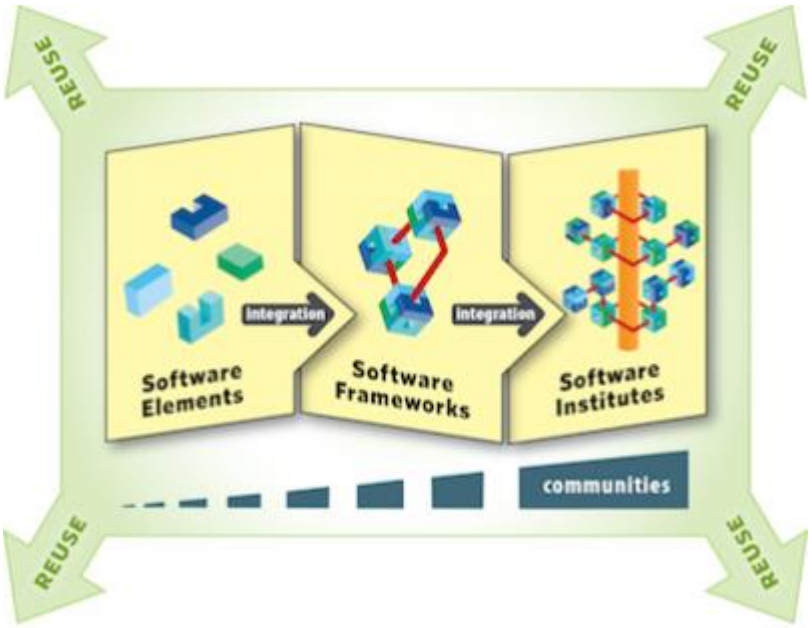

Figure 1 - SI2 software infrastructure

To sustain and enhance the growing community of SI^2 projects and researchers, the third annual SI^2 Principal Investigators (PI) workshop for researchers and NSF Program Officers was proposed, awarded, and executed February 17-18, 2015, at the Westin Arlington Gateway in Arlington, Virginia.

The 2015 SI^2 PI workshop had five strategic objectives that aligned with the broader SI^2 program goals:

- Serve as a focused forum for PIs to share technical information with each other and with NSF Program Officers.

- Explore innovative topics emerging within software communities.
- Discuss emerging best practices across the supported software projects.
- Stimulate thinking regarding new ways of achieving software sustainability.
- Gather shared experiences in an online web portal.

The meeting achieved these five objectives. In addition, it expanded the collective domain of knowledge on software infrastructure, enhanced the capability for sustained innovation within software communities, and contributed to the broader CIF21 vision.

1.1 Organizing Committee

The organizing committee consisted of eight members:

Frank Timmes (PI)
Professor in the School of Earth and Space Exploration
Director of ASU Advanced Computing Center
Arizona State University

Stan Ahalt (Co-PI)
Professor of Computer Science
Director of RENCI
Director of the Biomedical Informatics Service
University of North Carolina at Chapel Hill

Matthew Turk (Co-PI)
Research Scientist, National Center for Supercomputing Applications (NCSA)
Research Professor in Astronomy
University of Illinois at Urbana-Champaign

Ray Idaszak (Senior Collaborator)
Director of Collaborative Environments
RENCI

Mark Schildhauer (Senior Collaborator)
Director of Computing, National Center for Ecological Analysis and Synthesis (NCEAS)
University of California, Santa Barbara

Richard Brower (Senior Collaborator)
Professor of Physics and Engineering
Boston University

Chris Lenhardt (Senior Collaborator)
Scientist in Environmental Data Science and Systems
RENCI

Karl Gustafson (Senior Coordinator)
Project Manager
RENCI

The organizing committee held one-hour WebEx meetings every two weeks between October 30, 2014, and January 5, 2015, and then weekly WebEx meetings between January 12, 2015, and February 10, 2015. The workshop website first went online in early November 2014.

1.2 Workshop Style

The organizing committee decided in early November 2014 that the 2015 SI^2 PI workshop would feature an innovative workshop style that blended a traditional, top-down-driven agenda with a contemporary, on-the-fly, participant-driven agenda. See the workshop website for details on the agenda and guidance given to participants to help ensure success with this workshop style.

2. Agenda

The NSF awarded funding for the workshop on January 15, 2015. The organizing committee, led by workshop Principal Investigator Frank Timmes, created the workshop website to organize event materials prior to and during the workshop. Emails announced the workshop website, location, and dates. The emails were sent to the primary Principal Investigators of SSE, SSI, S2I2, EAGER, and other related SI^2 awards as described on the NSF SI2 website (http://nsf.gov/si2). Because only one PI or designate per award was allowed to attend, participants were required to register in advance. In total, 111 participants registered (see Appendix A for attendee list).

Participants had great flexibility in proposing discussion topics prior to and during the event. Before the event, participants proposed 25 topics. The format for the event was a modified unconference meeting. The objective was to reduce the amount of time passively listening and maximize cross-project interactions that would further the SI^2 program and individual project goals. Attendees were expected to contribute to the agenda and therefore the success of the workshop. Participants were continually encouraged to create, lead, and be an active participant in the breakout sessions. The organizers and the participants worked together to ensure constructive outcomes. Sharing information before and during the event was critical.

The first day of the workshop, February 17, began with a plenary session in the morning, followed by breakout sessions in the later morning and afternoon. Stan Ahalt, workshop Co-Principal Investigator and host, welcomed the attendees and recognized the NSF as workshop sponsor. He then described the workshop goals, participant-driven format, and desired outcomes. Daniel Katz and Rajiv Ramnath of the NSF next hosted an open question-and-answer period for topics relating to the SI^2 program. The morning plenary session concluded with a keynote address by Co-Principal Investigator Matthew Turk titled “Scaling a Code in the Human Dimension.”

The organizing committee then coordinated the first of several “Agenda Scrums,” in which participants refined and selected the breakout topics they wanted to focus on. The organizing committee distilled the 25 topics that had been suggested on the meeting website into a series of proposed breakout session topics. These topics were written on flipcharts and hung on the walls or easels. Participants then “voted with their feet” by standing next to the topic that interested them most. A separate area or room was assigned to each topic. Each group chose a moderator and note taker, and recorded group attendance. The organizing committee assigned a Google Document to each group so they could document issues, challenges, solutions, and actions or needs for their topic.

Breakout sessions lasted approximately two hours. After a break, groups then returned to plenary for approximately an hour to give their reports. The pattern of Agenda Scrum – Voting – Breakout Sessions – Session Reporting was used throughout the event. The breakout session topics were as follows:

Sessions in the Morning of February 17

1. Enhancing Interactions Between Domain Scientists and Computer Scientists
2. Enhancing Interactions Between Industry and the Research Community
3. Career Paths
4. Prioritizing and Predicting Software Needs
5. Enhancing Interactions: Translation of End-User Software
6. Software Needs and Requirements: Sustainability (note: this was a very large group that was consequently split into two concurrent breakout sessions)

Sessions in the Afternoon of February 17

- Licensing and Open Source
- Collaborative Environments
- Algorithms, Architecture, and Applied Math
- Metrics for Evaluation and Success
- Science and Social Ethos

Sessions in the Morning of February 18

1. Framing Success Metrics
2. Linear Algebra
3. Middleware and Software
4. Molecular Simulations
5. User Communities

The evening of the first day concluded with a two-hour poster session. Attendees were required to submit digital copies of their posters in advance and bring a printout to display. The poster session enabled interaction and collaboration.

The workshop concluded with a Prioritization Questionnaire, created and administered by Stan Ahalt. Subsequent to the workshop, participants were asked to complete an evaluation. Results of these activities are discussed in Section 4.

The workshop concluded at 11:30 a.m. Wednesday, February 18, with closing comments from Stan Ahalt and Daniel Katz. An initial draft of the workshop report was created March 30, 2015, announced, and made available to workshop attendees as an editable Google Document for open review and comment. The open review period lasted through April 13, 2015, with new versions that integrated community input. The final draft was then reviewed for typographical errors and grammar, submitted to the NSF, and posted to the workshop website for public dissemination.

3. Breakout Session Outcomes

Below are summaries of the discussions and outcomes of the 16 breakout sessions held during the workshop.

3.1 Sessions in the Morning of February 17

1. Enhancing Interactions Between Domain Scientists and Computer Scientists
2. Enhancing Interactions Between Industry and the Research Community
3. Career Paths
4. Prioritizing and Predicting Software Needs
5. Enhancing Interactions: Translation of End-User Software
6. Software Needs and Requirements: Sustainability

3.1.1 Enhancing Interactions Between Domain Scientists and Computer Scientists

Participants acknowledged the benefits of creating closer interactions between computer and domain scientists, both to develop software and algorithms that better serve scientific research and to enhance the delivery and application of those tools within the scientific research process. The practicalities of doing so, however, require resolving a number of issues, including differences in motivations, rewards, and interests between computer scientists and domain scientists; differences in language and work patterns; availability of workforce; and lack of professional options in scientific computing. Research software engineers were identified as key professionals who might best bridge the chasm.

The group discussed many challenges with collaboration between computer and domain scientists, including their separate skills sets, separate professional cultures, different perceived career paths, and very different funding sources. Many attendees favored cross-training

opportunities for students and researchers as one solution. Other ideas included closer interaction between the groups, a resolution of the perceived differences, and encouragement of collaboration at all levels.

The group agreed that domain scientists needed to enlist and fund computer scientist-based support for their projects. Bringing programmers into domain science projects, supporting interdisciplinary environments, expanding consulting roles, and helping software engineers transition from one application to the next would all go a long way toward increasing understanding and collaboration between these critical players in research.

3.1.2 Enhancing Interactions Between Industry and the Research Community

Participants felt that there is insufficient investment and focus on software issues as compared to hardware issues. They acknowledged the difficulties of engaging with industry, due in part to the short-term timelines that businesses operate within to stay viable, as well as to the disconnects between industry and academia in general. The histories of Sage Commons and ADIOS provide valuable lessons on industry-academia collaboration.

Further discussion centered on how to find non-industry funding. The NSF offers money, but is it enough to fund large-scale development, and how would that impact be measured? There are a few small NSF programs (I-Corps and I/UCRC) that link research to industry that could perhaps be used as a matchmaking model or conduit for more substantial relationships to develop.

The group brainstormed solutions that could be win-win opportunities for industry and academia, including inviting industry partners to future workshops for collaboration. They urged that greater attention be paid to existing efforts before proposing new software, that there be a greater focus on software sustainability, and that there be stricter due diligence for NSF submissions.

3.1.3 Career Paths

This group had a wide-ranging discussion about the need for more straightforward and rewarding career paths for individuals interested in scientific computing and domain research beyond tenure-track faculty or industry positions.

The group recognized that there is a need for more—and more qualified—scientific computing professionals in research, and that current opportunities for these people are limited and problematic (due to factors such as job instability, short job duration, and lower salaries). Such professionals, who offer important contributions toward overall improved software development and sustainability, need permanent, albeit perhaps non-tenured, positions. They might also require alternative metrics (beyond publications) for assessing their contributions. National labs, supercomputing centers, and universities are perhaps too narrowly defining their job descriptions when it comes to scientific computing (i.e., only hiring researchers and not developers).

Participants also suggested that early career researchers (e.g., postdocs) need better opportunities to grow into the role of professional scientific software engineers, for example by being able to win their own awards or move onto and across projects more readily than they can now. Encouraging cross-disciplinary collaboration, finding qualified individuals, and increased mentoring and training could also open up new career paths.

The group brainstormed solutions. Ideas included: creating a better venue for finding qualified developers, NSF-enforced hiring of developers, revising CV styles to highlight software skills, developing a multi-institutional group of software employees, and encouraging non-tenure career paths within universities. All of these solutions require recognizing and rewarding "non-research" work as not just valuable, but essential.

3.1.4 Prioritizing and Predicting Software Needs

This group had a free-form discussion with a strong emphasis on how some existing high-performance computing platforms (e.g., TACC, XSEDE) might better prioritize and serve their domain user communities. Participants agreed that assessing software needs is very difficult, and that current use may not be representative of general scientific needs or priorities. They also pointed out recent changes in the high-performance computing arena, including a trend toward supporting the types of jobs that were not deemed appropriate in the past (e.g., running "R").

In addition, participants discussed how younger scientists are using computers differently than in the past, for example by using many different software solutions, which may necessitate changes in planning for community-wide computational services. Participants recognized the tradeoffs between software innovation and stability and discussed the need for scientific computing to be on the "leading edge," but not necessarily the "bleeding edge," of computer sciences.

Once again the suggestion arose that NSF could support the building of a matchmaking service for computer science research interests with domain science computational needs. Similarly, a marketplace where domain scientists could advertise their computational needs to a broad audience of computer scientists and software engineers would be helpful. A modified version of NSF's I-Corps could also help, as would better data mining of NSF-funded infrastructure and workshops in order to maximize learning.

3.1.5 Enhancing Interactions: Translation of End-User Software

This group discussed the difficulties of effectively reaching the most end-users. There are different models for success—e.g., when a few developers create software for a large user-base; or when software is developed *by* a large user-base (in this case the developers are also users and vice-versa). Discussion centered on defining an "end-user," assessing customer

needs, and measuring the success of software. Participants agreed that understanding these constantly evolving topics is critical to producing successful software.

The group proposed several solutions to achieve strong end-user translation, including the creation of a sustainability plan, garnering committed institutional support and community support, improving the mechanisms for transitioning to external funding, and ensuring ongoing interaction with industry. Finding a true model of software sustainability (perhaps akin to ICERM) would also go a long way toward advancing end-user success.

3.1.6 Software Needs and Requirements: Sustainability

This group, which was large initially, split into two smaller groups to facilitate discussion. Both groups identified a number of factors that currently impede software sustainability, including a lack of sound design due to researcher naiveté (for example, being unaware of best practices, lack of interaction with properly trained software engineers, lack of awareness of existing code-versioning, and lack of collaborative environments), the often solo nature of software creation, and the lack of rewards and attribution for creating software or for debugging or extending code. GitHub is an example of a software creation system with these positive and negative attributes.

The first group discussed ownership issues regarding free, open-source solutions versus various “tiered” commercial solutions. Who is responsible for long-term code sustainability? How can the research and software community, including the NSF, encourage software sustainability? It is also often challenging to communicate the need for software fixes to researchers who might not understand their value.

A number of potential solutions were proposed, including better education and training in computational skills for domain scientists; better defining authorship and ownership of code; establishing a “cadre” of scientific computing specialists who can work with domain scientists; creating a “marketplace” of software tasks for which domain scientists need solutions (with rewards); encouraging greater recognition of the importance of software creation among the scientific community; recognizing documentation, debugging, and enhancement as an integral part of today’s scientific ventures; and creating Software Management Plans similar to the newly instituted Data Management Plans required by NSF.

The second group focused on sustainability not just for software, but also for data, users, communities, and talent. The group identified funding, standardization, and documentation as pertinent sustainability challenges. They then brainstormed multiple solutions, including leveraging the resources of funding sources, consortia, universities, and the greater community; creating structures to assist PIs and universities in sustainability efforts; and creating an archive, repository, or wiki for software sustainability best practices.

3.2 Sessions in the Afternoon of February 17

1. Licensing and Open Source
2. Collaborative Environments
3. Algorithms, Architecture, and Applied Math
4. Metrics for Evaluation and Success
5. Science and Social Ethos

3.2.1 Licensing and Open Source

This group discussed various licensing options and how best to engage with those licenses. For open source licenses, participants discussed copyleft and non-copyleft licenses. The Apache license, a community-oriented license similar to the BSD license, is one example that is "permissive" in that it does not restrict closing the source of derived products. The GPL, or GNU General Public License, is a copyleft license that could discourage commercialization, as derived products are required to be licensed in a compatible way. RedHat was identified as having run a successful company around GPL products.

The session attendees also discussed the role of the NSF in potentially mandating open source licenses. One suggestion was to have the review community set guidelines. However, participants recognized the possibility of conflicts arising between open source community standards and those of a university, particularly in the context of the Bayh-Doyle act, which provides universities with the right to commercialize the results of federal funding. In this context it is recognized that some universities may want to retain rights to code produced by their researchers. Patent obligations may present additional challenges. However, the group discussed several counterexamples of commercially developed software that transitioned to a community model. Some session attendees expressed concerns about the stability of licensing models, particularly those involving contributor-licensing agreements, such as the Apache license.

The group had a number of ideas to address these issues. One important step is to raise community awareness of licensing issues and project governance: when code is being written, it should be understood which license it is under and what that means for the developers. Some organizations aid in this process already, such as OSS-Watch and the Open Source Initiative. Finally, the community should strongly support developing an understanding of a sustainability model and how that relates to licensing.

3.2.2 Collaborative Environments

The first question attendees in this breakout session addressed was the meaning of a "collaborative environment." The general consensus was that a collaborative environment is one that enables individuals to collaborate around building software. Additional components of this concept may include: the ability to learn in real time, the collaborative use of infrastructure,

collaboration between domain and computer scientists, collaboration between academics and open source communities, and collaboration across time (e.g., institutional memory).

Participants recognized two types of collaboration challenges: social and technical. A few technical solutions, such as GitHub, Google Docs, and Google Hangouts, were very positively perceived. However, the tools used to collaborate around writing code are largely still in their infancy; the connection between design and coding was identified as a particularly weak element. Sphinx and Qt were noted as good tools, but participants recognized that most tools are designed not for scientists but for software engineers.

Session attendees agreed that they need a list of tools for collaboration and educational materials, and so they started shared document titled Collaborative Resources for Research Software Development (tiny.cc/si2resources), a resource collection that participants hope will be actively used and continue to grow. Additionally, participants discussed how incentivizing learning of new tools and techniques can encourage educational growth, as can the shifting cultural stereotypes of domain scientists and computer scientists.

3.2.3 Algorithms, Architecture, and Applied Math

This session focused on two issues: developing algorithms and connecting domain scientists and computer scientists for maximal computational throughput on advanced architectures.

A main issue for participants was how to define a problem well enough to gain traction with collaborators from other domains. Participants also identified issues related to identifying and training skilled multi-disciplinary collaborators.

3.2.4 Metrics for Evaluation and Success

This session focused on communicating the success of software projects. The relative ease of *collecting* information was noted in contrast to the difficulty of both communicating that information as well as communicating context; the development metrics of one community are not necessarily translatable to other communities.

The group identified a few specific questions to help drive the adoption of particular metrics. Chief among these was how to identify strategies for communicating success to funding agencies, both for progress reports and to indicate potential success of future endeavors. Other questions included: What metrics will convince my provost of the impact of software in research? What metrics will help me, as the PI, understand the impact of my software on my field? And what metrics tell me what aspects of the software are most useful to its users?

This was followed by brainstorming a framework for describing success in software projects, divided into four levels:

Level 1: Project Activity
Metrics include lines of code, test coverage, performance, publications, bug tracker count, recognition by community, size of community, industry connection, and diffusion of innovation.

Level 2: External Activity
Metrics include community size, citations, email activity, downloads, requests for new features, inclusion in educational activities, and increasing sophistication of support requests.

Level 3: Indirect Activity
Metrics include press coverage and user satisfaction.

Level 4: Activity in Context
Metrics include evidence that new science has been enabled, staff transfer to other jobs, and knowledge transfer.

Participants proposed and evaluated a potential case study to answer the question: What metrics will convince my provost of the impact of my software in research so that he/she pays for a software developer on my team? A framework for this hypothetical case study is as follows:

- Level 1: Project Activity
  - Existing software development funding (EASY to measure)
    - Reduction of ongoing costs
    - Reduction in effort for maintenance (HARD)
  - Value of software
    - Value of future sale of software (HARD)
- Level 2: External Activity
  - References to use of software
    - Citations (MEDIUM)
    - Acknowledgements (HARD)
  - Measure of user base
    - Users within the institution (EASY to find the visible ones/lower bound, HARDER to get an upper bound)
    - Diversity of user base, particularly the number of people who might bring in new money (e.g., alumni) (MEDIUM for lower bound, NEARLY IMPOSSIBLE for upper bound without specifically asking)
    - Number of students at institution using the software (EASY to find the visible ones/lower bound, harder to get an upper bound but easier if you require sign-in with an institutional user identity)
    - Number of courses relying on software
    - Number of colleagues able to use software before/after investment
  - Enabling more efficient researchers

        - How many researchers at the institution will be able to work more efficiently, and how much more can they expect to do (HARD; easier if the software produces a quantifiable aspect; testimonials will be mostly qualitative)
    - Number of visitors or collaborators that have been attracted (EASY)
- Level 3: Indirect Activity
    - Visibility of software in general media (EASY if you have a media clippings service and your software has a unique name; HARD to be comprehensive)
- Level 4: Activity in Context

The group brainstormed other ways to facilitate software tracking and to improve software in general. For example, developers can make software easier to track by developing a good name and using citations, such as a DOI. Developers should also focus on addressing user needs and building relationships with users to work together in a more collaborative way (for example, through participatory design). Agile sprints and regular releases can also improve response to evolving user needs and increase user satisfaction.

Looking forward, this group articulated the following goals: to create a metric for software success metrics (a metametric?), to persuade funders to describe potential success metrics in grant guidelines, and to create a dashboard of metrics built by SI^2 grant holders. The group agreed to produce a white paper as a first step toward these goals.

3.2.5 Science and Social Ethos

This panel focused on open source issues, metrics for success, and intellectual property. Software affordability was one important challenge participants noted. MATLAB, ENVI, and IDL were identified as illustrative examples. Participants discussed the different ways industry responds to open source. Although participants cited recent examples of industry being very favorable toward such licenses, it was noted that the copyleft method is occasionally found to be problematic.

Participants discussed the pitfalls of software patents, which can be particularly challenging because intellectual property is much trickier for patents than for copyright. It was also noted that open source is not necessarily free (which can help when attempting to work with industry).

Session attendees recognized the social problems that can arise when employers disapprove of their employees contributing to open source repositories. In one extreme example, a student who spent a summer interning in industry was not allowed to participate in an open source project later.

Participants identified a few other problems including how to best cite software and how to realistically track the uses and impact of software (given that it is the “invisible enabler”). Visualization software, for instance, often is not cited, but is quite important within software.

Looking forward, attendees identified several goals to help overcome these challenges. Two "Dummies" guides would be helpful, for example: one for open source software licensing and one for intellectual property issues for scientists and researchers. Advocating for free and open source software for educational, access, and reproducibility reasons would also help, as would a set of guidelines for writing letters of support for software.

3.3 Sessions in the Morning of February 18

1. Framing Success Metrics
2. Linear Algebra
3. Middleware and Software
4. Molecular Simulations
5. User Communities

3.3.1 Framing Success Metrics

Participants in this breakout group expressed a desire to create a methodology and framework for success metrics, with one downstream goal being to increase funding for worthy and successful projects. The group endeavored to scope requirements and create a framework for scientific software measures of success (for example, case snips, analysis of existing implementations, and others).

The group began the discussion by spelling out its needs, including a need to integrate or map a funding agency's success metrics, a need for goals showing that the metrics have been met, and a need to say why a particular metric is useful for one's software. The group also wants to encourage the use of "free but mandatory" licenses to track software.

The group also discussed the needs of the wider community, posing many questions about how others might react to tracking software usage, why metrics are useful, and how scientific software can best be evaluated. Participants reviewed resources pertinent to these topics including previous work on data guidelines, peer reviewing software, and the peer review process. In an attempt to understand the users, the group discussed why people create scientific software, identifying reasons such as problem solving, the need for useful tools, to facilitate collaboration, and for fun.

The group then split into two subgroups: one did a deeper dive into the topic of a framework for the evaluation of software, and the other came up with examples of metrics in use.

The first group discussed many facets around software evaluation, including criticality, usability, performance, functionality, availability, and scientific impact. An objective set of review criteria, applicable to different types of software, including regular reviewing of said criteria, was determined to be the best place to start, and group members agreed that they should submit their own software to this process. By volunteering their own software, they will be able to

answer questions such as: Can a review be objective? What does a review seek to measure? And who makes an “expert” reviewer?

The second group discussed metrics in depth. If metrics could follow a prescribed template, including a description of the metric, what it measures, what type of software it is used in, any qualifications or constraints, and why the researcher chose that metric, then success would be much easier to measure. Example metrics include: downloads, citations, user groups, publicity, automatic reporting techniques, and all the complexities that come from defining and measuring that information. Another important aspect of using metrics is convincing software developers that measuring metrics is a worthwhile endeavor. Do they need an incentive to do so, and if they do, what would work best?

At the end of the session, the group reunited and agreed on two next actions. First, a “review club” will attempt to set out a framework of success metrics for scientific software. Participants noted that it would be useful to incorporate a pilot project to help define and refine the criteria under consideration for such a framework. Second, the group will create a set of guidance-giving examples, and eventually a website and white paper, for best practices for software metrics. After being circulated to the community, the white paper will be published with authorship from this meeting.

3.3.2 Linear Algebra

A large contingent of NSF SI^2 software uses linear algebra packages in some form. Participants in this breakout session identified example applications, including some of the more established approaches such as sparse matrices and dense matrices, and some newer approaches including low rank matrices (data sparse), tensors, grids, and a matrix of complex objects, such as matrices and nonlinear objects. The group raised several issues, including facets affecting linear algebra developers and users, and the need for more functional algorithms, testing, and interoperability. Participants identified the need to address the disconnects between linear algebra developers and users.

The group identified a variety of challenges in linear algebra, including data transfer, tensor decomposition, domain-specific compilers, and data decomposition. A few suggestions to help the linear algebra community find the right solutions included distributing work, reviewing code the way we review papers, and encouraging other fields to adopt general-use linear algebra software.

In particular, the group discussed the need to expand BLAS (Basic Linear Algebra Subprograms), either through BLIS (a BLAS-like Library Instantiation Software) or through machine learning-based auto-tuning.

The group recommended that the linear algebra community should think more generally about software and also build a general language, perhaps using Connection Machine Scientific Software Library (CMSSL) as inspiration. They also suggested establishing a linear algebra

community wiki or forum to focus on sustainability. Such a forum could improve upon the software list available at http://www.netlib.org/utk/people/JackDongarra/la-sw.html and include participation from the National Institute of Standards and Technology.

3.3.3 Middleware and Software

Following brief introductions of their projects, the participants discussed many challenges for middleware and software. These included middleware developers' issues; lack of uniformity in middleware; interoperability, standardization, and metadata challenges; and the need for long-term sustainability measures.

Participants reached a consensus on action items that encompass education (promoting awareness that middleware is vital to research yet is frequently underserved), standardization and provenance of middleware, and supporting the middleware community. Support can be offered through increasing awareness of these issues, identifying current NSF infrastructure or middleware that needs assistance, and enabling various middleware aspects (Skeletons, mini-apps, sub-workflows, analytical templates) to become more discoverable.

3.3.4 Molecular Simulations

Participants in this breakout group identified several challenges in advancing software for molecular simulations. Achieving effective collaboration can be particularly challenging: PIs may be restricted from formal collaboration, organizing several teams of researchers is difficult and time-consuming, and teams may be using different software or file formats. There is also a lack of knowledge about what other projects are doing. A matching site for funding and projects could facilitate and improve collaboration.

Participants offered diverse solutions to meet these challenges. Meeting more often, formally or informally, at workshops, conferences, symposia; establishing a software institute or website (described as a "science match.com"), and establishing best practices would all encourage interdisciplinary work. Increasing the interest in and knowledge of molecular simulation among computer science researchers would also go a long way toward bridging the collaboration gap.

3.3.5 User Communities

This session covered the process of developing, sustaining, and growing user communities. Do communities grow around software, or do you build software around a community? And how do you bootstrap a user community, particularly with a new project?

Discussion focused on how to engage and then support communities. Ideas included encouraging individuals to take ownership in the community, answering community questions, and managing the overwhelming number of outlets and locations that could be queried—social, e-mail, web-based, and so on. Perhaps incentivizing sharing software usage stories could be one way to gain buy-in, establish trust, and encourage community growth. Determining the level

of contribution—and the ability of the community to accept the contribution—can pose challenges, particularly for tenure-track individuals who have to focus on publication rather than software productivity.

A few ideas for making a project successful included building high team morale, providing transparency in your project, defining your community, and offering prestige to contributors. One person suggested that it was best to focus on software users rather than developers (while recognizing the importance of volunteer developers for project sustainability). The group agreed that learning from successful communities, creating a best practices guide, and sustaining community-developed content were essential to online community-building.

Group members mentioned many resources that could be used as training or for learning community-building lessons, including Sagemath, Dataverse, MESA, and VAPOR. The Software Sustainability Institute has many resources worth looking at. The RSE community and Richard Millington's work are also places for researchers to learn about online communities.

4. Prioritization Questionnaire and Evaluation Results

In addition to the notes and direct outcomes of the workshop, there were two additional opportunities for participant feedback. The first was a prioritization exercise based on information culled from the breakout session notes. The second was an end-of-workshop set of evaluation topics.

4.1 Prioritization Questionnaire

During the meeting, each breakout session was asked to suggest one to three actions that, if implemented, would have a substantial impact on the state of software development in that area. These were then consolidated into 57 actions. Attendees each cast a total of 15 votes, with no more than seven votes assigned to any one action, for the items they prioritized as most important.

As the results below show, there is a clear need for training opportunities of varying types and duration. Additionally, there is a need for modest funding for specific software development efforts.

Action	**# of votes**
Support education and training in scientific software engineering skills.	**57**
Create workshops to teach basics in software development (e.g., software carpentry). As an example, a summer institute for molecular biologists (that also includes computer scientists) could help address the software issues of that community.	**54**

Create alternative funding mechanisms for modest software development activities.	**53**
Develop best practice guidelines for “scientific” computing: Tools should be designed so others can also use them (with documentation, versioning, bug reporting, and so forth).	**49**
Create an inter-institutional matchmaking system to help software engineers to transition from one application project to another.	**41**
Create mechanisms to recognize and reward valuable “non-research” work (e.g., community building efforts).	**41**
Create a set of guidance-giving examples of specific metrics for the success of scientific software in use, why they were chosen, what they are useful to measure, and any challenges or pitfalls.	**41**
Develop “Dummies’ Guide to Open Source Software, Software Licensing, and Intellectual Property Issues for (Academic) Scientists/Researchers.” (Wikipedia might have usable descriptions for software licensing.)	**39**
Organize a proposal to apply to the NSF for funding to create a methodology and framework for recommending success metrics that projects should capture.	**36**
Consider funding or proposal call changes.	**35**
Create a wiki or other forum for sustainability.	**35**
Develop publication model/altmetrics for code development or bug fixes.	**34**
Establish standard ways to enhance discoverability and description of scientific processes and software applications (e.g., metadata/ontologies, documentation standards, community-curated catalogs).	**34**
Hold an inexpensive workshop and hackathon that encourages molecular and computer sciences to interact around specific software topics (akin to http://www.cecam.org/workshop-1214.html); encourage close involvement from students and postdocs.	**34**
Create certificate programs and/or minors that encourage professional development in interdisciplinary training, software engineering, or related topics and skills.	**28**
Create a multi-institutional matchmaking system to expose academic software to industry.	**25**

Require projects that require long term support to comply with strong standards regarding compelling sustainability models.	**24**
Set up a “review club” among SI^2 grant holders. The initial aims would be for self-improvement, but the ultimate goal would be to identify a process to peer-evaluate scientific software.	**23**
Collect and disseminate (e.g., through a paper) information about the variety of software shapes and sizes software can take (examples and related topics include Sagemath, Dataverse, MESA, VAPOR, and the goal of targeting end users rather than only developers).	**23**
Establish technical and social methods for sustaining user-generated content, including guidance on how to reference, license, store, and share.	**20**
Develop an improved website for linear algebra software packages.	**17**
Bolster and improve middleware, which is particularly underserved, leading to major cost inefficiencies in operations of existing high-performance computing due to executions of non-optimized codes.	**17**
Organize a molecular simulation subgroup for this NSF workshop (coordination needed, especially in recruiting interested computer science participants).	**17**
Identify libraries other than BLAS that can be handed off to the computer scientist and tuned as a building block for different architectures.	**12**
Identify current NSF infrastructure/middleware that needs to be supported (e.g., MPI, workflows, others).	**12**
Consider same for other levels of data in particular, IO (beyond file exchange between apps).	**11**
“Producing SI^2.”	**9**
Encourage the community to think more generally about software, build general “languages” (e.g., connection machine CMSL as inspiration.)	**5**

4.2 Workshop Evaluation Questions

The workshop evaluation had four areas for participant feedback:

1. Comments or suggestions for improvements for the 2016 SI^2 PI workshop organizers?
2. How did the participant-driven style of this workshop work for you?
3. Did you get out of this workshop what you wanted? Why or why not?
4. How could future NSF SI^2 solicitations be improved?

Overall, the responses from participants were thoughtful, constructive, and positive. Software sustainability and cross-project collaboration were frequently mentioned. There were also some suggestions to move the meeting to the spring instead of the winter. The below is a summary of the feedback provided; see Appendix B for additional ideas.

1. Comments or suggestions for improvements for the 2016 SI^2 PI workshop organizers?

Comments were mixed, divided between positive comments, comments with specific suggestions for improvements, and a small number of negative comments. The positive comments were enthusiastic, finding the format engaging and worthwhile.

Participants offered specific suggestions to improve the poster session, such as allowing more time for the posters to be available or having a series of lightning talks related to the posters. Another specific suggestion was to have more plenary talks in order to delve more deeply into some of the workshop themes; these presentations could be broad, exploring themes across topics, or narrow, perhaps discussing emerging trends or best practices, or even full presentations on the process of software development at places like Google or RedHat.

Additional ideas included clarifying the intended audience for the workshop outputs and learning what the NSF views as central to the SI^2 program. There was some criticism about the length of time to decide session topics and a suggestion to do more agenda determination in advance.

2. How did the participant-driven style of this workshop work for you?

The responses to this question were overwhelmingly positive. In a number of instances, the approach exceeded the respondents' pre-workshop expectations. Most appreciated the flexibility of the format and the opportunity to delve more deeply into topics of interest. There was some, very limited, sentiment indicating that there was not enough structure, that the outcomes were somewhat vague and without specific endpoints, and that some of the sessions should have been devoted to detailed technical topics.

3. Did you get out of this workshop what you wanted? Why or why not?

The feedback in response to this question was positive. Participants indicated that the opportunities that they were looking for and received out of the workshop included networking

with peers and the NSF, discovering new ideas, and delving more deeply into specific topics. Suggestions for improvement included a more efficient system to determine the subtopics, more short presentations on key topics, and opportunities for new projects to learn lessons from the more mature projects.

4. How could future NSF SI^2 solicitations be improved?

Feedback on this question was mixed. Some attendees said the solicitations are adequate, while others suggested the NSF provide more funding, either in terms of duration or number of awards. Other suggestions were to include a software management plan requirement and opportunities to build interoperability across software projects. There was also some convergence around the idea of creating a software catalog with appropriate metadata. The purpose of such a catalog would be knowledge-sharing: to enable projects to find potentially relevant software rather than recreating that software *de novo*, and to enable clarification as to how one particular project's code differs from another's. According to one attendee, knowledge sharing could also identify software infrastructure gaps that could potentially be filled via future solicitations. Metrics were also raised in this context. A final suggestion was to figure out what sustainability means and how it can be operationalized at the project level: Should sustainability be addressed at the institutional level or at the developer level?

Acknowledgements

This workshop was funded by the National Science Foundation through grant number 1521388. The organizing committee expresses its sincere appreciation to the National Science Foundation for their support. The committee extends a special thank you to all the attendees for participating in the workshop discussions and panel sessions, leading breakout sessions, taking notes, and contributing to this report. The interest in sharing dialogue among research communities and related projects is now stronger than ever as a result of this workshop. The outcomes provided will be of great value to the NSF and our respective research communities.

Appendix A: Workshop Attendees

*Denotes attendance in-person. Many who registered were unable to attend due to severe inclement weather.

*Gagan Agrawal, The Ohio State University
*Stan Ahalt, RENCI, University of North Carolina at Chapel Hill; NCDS
*Anthony Aufdenkampe, Stroud Water Research Center
*Paul Barbone, Boston University
*Jerry Bernholc, North Carolina State University
*Bruce Berriman, California Institute of Technology
George Bosilca, University of Tennessee, Knoxville
*Richard Brower, Boston University
*Maxine Brown, University of Illinois at Chicago
*K. Selcuk Candan, Arizona State University
*Jeffrey Carver, University of Alabama
*Neil Chue Hong, Software Sustainability Institute
*Cecilia Clementi, Rice University
*John Clyne, National Center for Atmospheric Research
Daniel Crawford, Virginia Tech
*Mercè Crosas, Harvard University
Peter Cummings, Vanderbilt University
Joseph Curtis, National Institute of Standards and Technology
Damian Dechev, University of Central Florida
*Viktor K. Decyk, University of California, Los Angeles
Ewa Deelman, University of Southern California
Constantine Dovrolis, Georgia Institute of Technology
Kevin Eliceiri, University of Wisconsin - Madison
*Renato Figueiredo, University of Florida
*Allison Fish, University of California, Davis
Tony Fountain, University of California, San Diego
*Lawrence Frank, University of California, San Diego
*Brent Fultz, California Institute of Technology
*Emilio Gallicchio, CUNY Brooklyn College
*Ashish Gehani, SRI International
Yolanda Gil, University of Southern California
Ganesh Gopalakrishnan, University of Utah
*Boyce Griffith, University of North Carolina at Chapel Hill
*Karl Gustafson, RENCI, University of North Carolina at Chapel Hill
Teresa Head-Gordon, University of California, Berkeley
Richard Hennig, University of Florida
*Larry Hoyle, University of Kansas
*Ray Idaszak, RENCI, University of North Carolina at Chapel Hill
Nancy Ide, Vassar College

David Irwin, University of Massachusetts Amherst
*Erkan Istanbulluoglu, University of Washington
*Doug James, University of Texas at Austin
Matthew B. Jones, University of California, Santa Barbara
*Hartmut Kaiser, Center for Computation & Technology, Louisiana State University
George Karypis, University of Minnesota
*Daniel S. Katz, National Science Foundation
Mike Kirby, University of Utah
*Scott Klasky, Oak Ridge National Laboratory
*Julien Langou, University of Colorado, Denver
Chris Lenhardt, RENCI, University of North Carolina at Chapel Hill
*Philip Maechling, Southern California Earthquake Center
*Edward Maginn, University of Notre Dame
Glenn Martyna, IBM TJ Watson Research Center
*Devin Matthews, University of Texas at Austin
*Matthew Mayernik, National Center for Atmospheric Research
Clare McCabe, Vanderbilt University
*John McGregor, Clemson University
A. J. Meir, National Science Foundation
Samuel Midkiff, Purdue University
Dane Morgan, University of Wisconsin - Madison
*Paul Navratil, Texas Advanced Computing Center
*Greg Newman, Colorado State University
Michael Norman, University of California, San Diego
*Dhabaleswar Panda, The Ohio State University
Alberto Passalacqua, Iowa State University
*Abani K. Patra, University at Buffalo
Dmitry Pekurovsky, San Diego Supercomputer Center; University of California, San Diego
*Marlon Pierce, Indiana University
*Jeffrey Potoff, Wayne State University
*Elbridge Gerry Puckett, University of California, Davis
*Adrian Roitberg, University of Florida
*Christopher Roland, North Carolina State University
*P. (Saday) Sadayappan, The Ohio State University
*Mark Schildhauer, University of California, Santa Barbara
*Karsten Schwan, Georgia Institute of Technology
*Inanc Senocak, Boise State University
Nigel Sharp, National Science Foundation
*Amarda Shehu, George Mason University
*David Sherrill, Georgia Institute of Technology
*J. Ilja Siepmann, University of Minnesota
*Piotr Sliz, Harvard Medical School
*Shava Smallen, San Diego Supercomputer Center
*Andrew Sommese, University of Notre Dame

*Edgar Spalding, University of Wisconsin - Madison
*John Stanton, University of Texas at Austin
*Andreas Stathopoulos, College of William and Mary
*William Stein, University of Washington
Alejandro Strachan, Purdue University
*David Tarboton, Utah State University
*Douglas Thain, University of Notre Dame
Frank Timmes, Arizona State University
*Charles Torre, Utah State University
Steve Tuecke, University of Chicago, Globus
*Bruno Turcksin, Texas A&M University
*Matthew Turk, University of Illinois at Urbana-Champaign
*Selcuk Uluagac, Florida International University
*Eric Van Wyk, University of Minnesota
*Jan Verschelde, University of Illinois at Chicago
Rich Vuduc, Georgia Institute of Technology
*Shaowen Wang, National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign
*Xiyin Wang, University of Georgia
Von Welch, Indiana University
*Arthur Wetzel, Carnegie Mellon University
*Michael Wilde, University of Chicago
*Joseph Jay Williams, Harvard University
*Philip A. Wilsey, University of Cincinnati
*Nathan Wilson, University of California, San Diego
*Theresa Windus, Iowa State University
Jonathan Wren, Oklahoma Medical Research Foundation
*Bo Zhang, Indiana University
*Xiaodong Zhang, The Ohio State University

Appendix B: Summary of Feedback

Notes on What Went Well

- Implementing good software practices such as testing, debugging, release cycles.
- Each presentation is followed by a table discussion.
- Talks the first thing in the morning to set the stage.
- Emphasis that SI^2 is a "translational" program (pushing research to users), hence, the need for "sustainability."

Notes on What Did Not Go Well

- Need to have some baseline questions [for breakout sessions?].
- Breakout session groups were too large.
- Participant-driven format didn't ensure tangible end points.
- Too many topics: Need to focus on a limited number of topics (perhaps based on what emerged from the 2015 workshop) and come up with direct actions and commitments to carry forward.
- Outcomes of the discussions were not always clear.

Suggestions for Improvement

Workshop Format:

- Organizers should contribute a moderator and scribe to each breakout session.
- Few more thoughtful presentations would be helpful.
- Metrics drives SI^2 – both quantitative and qualitative (e.g., use cases) metrics.
- Use user boards where sub-communities engage in deeper conversations and determine an agenda prior to the workshop.
- Useful: prior to the meeting – allow participants to suggest the subjects of interests (for breakout sessions) – (poll?).
- Several subcommittees were formed during the meeting – in the future – form them prior to the meeting, engage members in creating agenda which may include short presentations by members.

Future NSF Solicitations:

- "We are down here in the trenches trying to grow the state of the practice – I cannot shake the feeling that it is asking too much to expect us to promise sustainability down the road in the absence of institutional support."
- Need for a "software management plan" similar to the "data management plan."
- Incorporate community building and usability sections.
- Need for centralized source of information about existing software projects.
- Funding could be made more efficient by funding groups that were targeting gaps in the existing software infrastructure.
- A Wikipedia like approach where a central location for information is available for NSF investigators will help optimize both production software and distribution of funds.