



EXASCALE
COMPUTING
PROJECT

ECP-U-RPT-2020-0001

**Advancing Scientific Productivity through Better Scientific Software:
Developer Productivity and Software Sustainability Report**

IDEAS-ECP Team and Collaborators

January 28, 2020



U.S. DEPARTMENT OF
ENERGY

Office of
Science



DOCUMENT AVAILABILITY

Reports produced after January 1, 1996, are generally available free via US Department of Energy (DOE) SciTech Connect.

Website <http://www.osti.gov/scitech/>

Reports produced before January 1, 1996, may be purchased by members of the public from the following source:

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
Telephone 703-605-6000 (1-800-553-6847)
TDD 703-487-4639
Fax 703-605-6900
E-mail info@ntis.gov
Website <http://www.ntis.gov/help/ordermethods.aspx>

Reports are available to DOE employees, DOE contractors, Energy Technology Data Exchange representatives, and International Nuclear Information System representatives from the following source:

Office of Scientific and Technical Information
PO Box 62
Oak Ridge, TN 37831
Telephone 865-576-8401
Fax 865-576-5728
E-mail reports@osti.gov
Website <http://www.osti.gov/contact.html>

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ECP-U-RPT-2020-0001

**Advancing Scientific Productivity through Better Scientific Software:
Developer Productivity and Software Sustainability Report**

Office of Advanced Scientific Computing Research
Office of Science
US Department of Energy

Office of Advanced Simulation and Computing
National Nuclear Security Administration
US Department of Energy

January 28, 2020

Members of the IDEAS-ECP Project, 2017–2019

Michael A. Heroux (SNL)
 Lois Curfman McInnes (ANL)
 David E. Bernholdt (ORNL)
 Anshu Dubey (ANL)
 Elsa Gonsiorowski (LLNL)
 Osni Marques (LBNL)
 J. David Moulton (LANL)
 Boyana Norris (University of Oregon)
 Elaine M. Raybourn (SNL)
 Satish Balay (ANL)¹
 Ross Bartlett (SNL)
 Lisa Childers (ANL)¹
 Todd Gamblin (LLNL)¹
 Patricia Grubel (LANL)
 Rinku Gupta (ANL)
 Rebecca Hartman-Baker (LBNL)
 Judy Hill (ORNL)
 Stephen Hudson (ANL)¹
 Christoph Junghans (LANL)¹
 Alicia Klinvex (SNL)¹
 Reed Milewicz (SNL)
 Mark C. Miller (LLNL)
 Hai Ah Nam (LANL)
 Jared O’Neal (ANL)¹
 Katherine Riley (ANL)
 Ben Sims (LANL)
 Jean Shuler (LLNL)
 Barry Smith (ANL)¹
 Louis Vernon (LANL)¹
 Greg Watson (ORNL)
 Jim Willenbring (SNL)
 Paul Wolfenbarger (SNL)

in collaboration with the ECP Community

¹indicates IDEAS-ECP project alumni at the time of this publication, January 2020

REVISION LOG

Version	Creation Date	Description
1.0	January 28, 2020	<i>Advancing Scientific Productivity through Better Scientific Software: Developer Productivity and Software Sustainability Report</i>

EXECUTIVE SUMMARY

The Exascale Computing Project (ECP) provides a unique opportunity to advance computational science and engineering (CSE) through an accelerated growth phase in extreme-scale computing. Central to the project is the development of next-generation applications and software technologies that can exploit emerging architectures for optimal performance and provide high-fidelity, multiphysics, multiscale capabilities. However, disruptive changes in computer architectures and the complexities of tackling new frontiers in extreme-scale modeling, simulation, and analysis present daunting challenges to the productivity of software developers and the sustainability of software artifacts. Members of the CSE community—especially at extreme scales but more broadly at all scales of computing—face an urgent need to improve *developer productivity*, positively impacting product quality, development time, and staffing resources, and *software sustainability*, reducing the cost of maintaining, sustaining, and evolving software capabilities.

This report summarizes technical and cultural challenges in scientific software productivity and sustainability. We introduce work by the IDEAS project within ECP (called IDEAS-ECP, <https://ideas-productivity.org>) to foster and advance software productivity and sustainability for extreme-scale CSE, including partnerships with complementary groups. IDEAS goals are to qualitatively change the culture of extreme-scale computational science and to provide a foundation (through software productivity methodologies and an extreme-scale software ecosystem) that enables transformative and reliable next-generation predictive science and decision support. Work spans four synergistic strategies: (1) curating methodologies to improve software practices of individuals and teams, (2) incrementally and iteratively upgrading software practices, (3) establishing software communities, and (4) engaging in community outreach. Because these issues are relevant throughout all scales of scientific computing, we aim for broad readership—and we hope that these experiences and resources may be useful in other contexts, as individuals and teams work within their own projects, institutions, and communities to advance software practices and overall productivity.

Members of the IDEAS-ECP project serve as catalysts to address the challenges facing ECP teams by focusing on improving how teams conduct software efforts. A central activity is *productivity and sustainability improvement planning (PSIP)*—a lightweight, iterative workflow where teams identify their most urgent software bottlenecks and work to overcome them. We explain how teams are more productively tackling science goals through PSIP advances in areas such as software builds, testing, refactoring, and onboarding. As the ECP community works toward an extreme-scale scientific software ecosystem composed of high-quality, reusable software components and libraries, we are advancing methodologies to support Software Development Kits and to improve transparency and reproducibility of computational results. The report discusses IDEAS outreach in the high-performance computing (HPC) community: (a) establishing the *Better Scientific Software (BSSw)* website (<https://bssw.io>) as a hub for sharing information on practices to improve software productivity and sustainability; (b) launching the *BSSw Fellowship Program* to give recognition and funding to leaders and advocates of high-quality scientific software; (c) producing a webinar series on *Best Practices for HPC Software Developers*; (d) providing tutorials on topics such as software testing, verification, and refactoring; (e) organizing events to foster discussion of issues in scientific software development; and (f) working toward a scientific software community culture that invests in and benefits from an explicit focus on developer productivity and software sustainability, adapting and adopting best practices from the broader software community and establishing our own contributions to these pursuits.

These synergistic strategies and achievements provide a strong foundation for forthcoming work to advance the quality of extreme-scale CSE as needed to achieve ECP goals and to enable transformative and reliable next-generation predictive science, engineering, and decision support. Work thus far is the first phase of much longer-range vision. Members of the IDEAS-ECP project are collaborating with complementary international groups to nurture communities of practice and work toward long-term changes in the culture, funding, and reward structure of CSE—with a goal of increasing overall scientific productivity, while enabling software to fully realize its role as a cornerstone of long-term collaboration and scientific progress.

TABLE OF CONTENTS

EXECUTIVE SUMMARY	vi
1 Introduction	1
2 Context for Advances in Software Productivity and Sustainability	3
2.1 Challenges	3
2.2 Growing a community of practice	4
2.3 Characterizing the development and use of software for scientific research	5
3 Key Synergistic Strategies and Impact	5
3.1 Prerequisite: Determining crosscutting urgent needs	7
3.2 Curating methodologies to improve software practices of individuals and teams	8
3.3 Incrementally and iteratively upgrading software practices	9
3.4 Establishing software communities	11
3.5 Engaging in community outreach	12
3.5.1 Community-driven resources: BSSw.io website	12
3.5.2 Recognition: BSSw Fellowship Program	14
3.5.3 Webinar series: Best Practices for HPC Software Developers	15
3.5.4 Tutorials on better scientific software	17
3.5.5 Technical meetings on HPC software development topics	18
3.5.6 Working toward culture change	19
4 Conclusion	21
A References	22
B IDEAS Projects	25
B.1 IDEAS-ECP team	25
B.2 IDEAS-Watersheds: Advancing a software ecosystem for watershed modeling	26
C Advances by Science Teams through Productivity and Sustainability Improvement Planning	27
C.1 Advances in building and testing: A case study with EXAALT	28
C.2 Advances in code refactoring: A case study with ExaStar	29
C.3 Advances in onboarding: A case study with Exascale MPI	30
D Outreach Details	32
D.1 HPC Best Practices Webinar Series	32
D.2 Blog articles published on BSSw.io	33

1. INTRODUCTION

Unprecedented complexity and continual change. Researchers in computational science and engineering (CSE) face unprecedented disruptive changes in extreme-scale computer architectures. The transition to hosted accelerated architectures, specifically nodes with multicore CPU and multiple GPUs, requires developing new algorithmic approaches, porting code to new compiling and runtime environments, and realizing massive concurrency that is possible only by overcoming large latency bottlenecks. Furthermore, these new CPU/GPU platforms represent only the beginning of the system heterogeneity expected in the future.

At the same time, opportunities for next-generation simulation, analysis, and design present enormous challenges. Teams are working toward predictive science and engineering through multiphysics, multiscale simulations and analytics and are addressing requirements for greater scientific reproducibility. Moreover, recent community reports have expressed the imperative to firmly embrace the fundamental role of open-source CSE software as a valuable research product and cornerstone of collaboration—and thus to increase direct investment in the software itself, not just as a byproduct of other research [3, 9, 14, 16, 17, 19, 20, 35]. This situation brings with it an opportunity to fundamentally change how scientific software is designed, developed, and sustained.

IDEAS overview and history. Members of the IDEAS project (Interoperable Design of Extreme-scale Application Software, <https://www.ideas-productivity.org>) are partnering with the CSE and high-performance computing (HPC) communities to improve *developer productivity* (positively impacting software quality, development time, and staffing resources) and *software sustainability* (reducing the cost of sustaining and evolving software over its intended lifetime)—thereby helping improve scientific productivity while ensuring continued scientific success. We are defining and taking steps toward new scientific software ecosystems that emphasize the use and development of high-quality tools, the adaptation and adoption of modern software engineering methodologies, and the development and use of high-quality software components in the composition of next-generation applications.

The IDEAS project began in 2014, sponsored by the U.S. Department of Energy Office of Science as a partnership between the Offices of Advanced Scientific Computing Research (ASCR) and Biological and Environmental Research (BER), to address challenges in software productivity and sustainability, with an emphasis on terrestrial ecosystem modeling; we now refer to this original phase of the project as IDEAS-Classic. The project expanded in 2017 in the DOE’s Exascale Computing Project (ECP, <https://www.exascaleproject.org>), which requires intensive development of applications and software technologies while anticipating and adapting to continuous advances in computing architectures. This ECP-funded effort, which we call IDEAS-ECP, addresses the specific needs of ECP application and software efforts, focusing on the challenges of software development for emerging extreme-scale architectures and ensuring that investment in the exascale software ecosystem is as productive and sustainable as possible. In spring 2019, the BER-funded IDEAS-Watersheds project grew out of the original IDEAS project, with emphasis on accelerating watershed science through a community-driven software ecosystem. Throughout this report, we use the term IDEAS to refer to the overarching IDEAS initiatives, and we use the terms IDEAS-Classic, IDEAS-ECP, and IDEAS-Watersheds when a funding source or team distinction is needed.

Advancing scientific productivity through better scientific software. Scientific productivity is one of the top ten exascale research challenges [30], and software productivity (the effort, time, and cost for software development, maintenance, and support) is one critical aspect of scientific productivity [17]. A significant challenge is the need for high-quality, high-performance, reusable, sustainable scientific software, so that we can more effectively collaborate across teams on work toward predictive science. Thus, although general and actionable metrics have proven difficult to define, an overarching focus on productivity and sustainability is essential to making clear decisions as we face not only highly disruptive architectural changes but also demands for greater interaction across distinct teams and reliability of results.

At the same time, there is general awareness that the broader CSE community has similar urgent needs, even if it does not have the mandate to prepare for exascale computing platforms or the same resources to address these challenges. The Computational Science & Engineering Software Sustainability and Productivity Challenges workshop was sponsored by the U.S. Networking and Information Technology Research and Development program [29] on behalf of DOE, DOD, NSF, NIST, NASA, and U.S. industrial organizations. This workshop and its subsequent report [14] highlight the significant challenges that all CSE software projects face. The IDEAS-ECP project has substantial overlap with the concerns of these U.S. federal and industry organizations; and our efforts build upon relationships and collaborations that the IDEAS-ECP team has already established with this broader community, including an NSF-funded conceptualization project for a U.S. Research Software Sustainability Institute [43]. We are also partnering with international organizations that are addressing challenges in software quality and sustainability [18], including Software Carpentry [39] and the U.K. Software Sustainability Institute [40]. As discussed in Section 2.2, this work contributes to exciting advances in research software communities.

Better Scientific Software: community, website, fellowships. While the IDEAS-ECP project focuses explicitly on software issues for extreme-scale science, we are serving and partnering with the broader CSE community (computing at all scales) through a key initiative: establishing the *Better Scientific Software (BSSw)* website (<https://bssw.io>) as a central hub for sharing information on practices, techniques, experiences, and tools to improve developer productivity and software sustainability. Also, the *BSSw Fellowship Program* gives recognition and funding to leaders and advocates of high-quality scientific software. The long-term vision for BSSw is to serve as an international community-driven and community-managed resource, with content and editorial processes provided by volunteers, initially nucleated by the IDEAS team but over time expanding to much broader participation.

Document structure and target audience. The remainder of this document explains more about the drivers of this work, the approach and achievements of the IDEAS project, and their relationships to complementary international efforts. Section 2.1 explains technical and cultural challenges of scientific software, while Section 2.2 introduces a variety of international groups that are fostering “communities of practice” to advance the culture of scientific and research computing with explicit emphasis on high-quality software. Section 2.3 characterizes the development and use of research software for science, emphasizing unique needs relative to mainstream environments in industry and business. Section 3 explains the IDEAS approach, outcomes, and impact of advancing software productivity and sustainability through four synergistic strategies that complement and reinforce one another: (1) curating methodologies to improve software practices of individuals and teams, (2) incrementally and iteratively upgrading software practices, (3) establishing software communities, and (4) engaging in community outreach. Section 4 provides concluding comments. Appendices provide more details about the IDEAS family of projects (Appendix B), advances in scientific productivity through improving software practices (Appendix C), and outreach activities (Appendix D).

Our target readers are all those who care about the quality and integrity of scientific discoveries based on simulation and analysis. While the difficulties of extreme-scale computing and large, multidisciplinary research projects intensify software challenges, many issues are relevant across all computing scales and project sizes, given universal increases in complexity and change in scientific computing coupled with the need to ensure the trustworthiness of computational results. Each member of the scientific software community—from students and early-career professionals through mid-career and senior staff, stakeholders, members of funding agencies, and other leaders—can play important roles in addressing these technical and social challenges in scientific software by catalyzing change in our own projects, institutions, and communities to improve *how* we conduct our software efforts. While various reports consider diverse perspectives and opportunities [3, 9, 14, 16, 17, 19, 20, 35] for work toward long-term changes in the culture, funding, and reward structure of CSE, this discussion focuses on the point of view of individuals working within their own spheres of influence to promote advances in software practices.

2. CONTEXT FOR ADVANCES IN SOFTWARE PRODUCTIVITY AND SUSTAINABILITY

Before considering strategies for advances in scientific software productivity and sustainability, we must first consider the context for this work. Section 2.1 summarizes technical and cultural challenges in scientific software productivity and sustainability, while Section 2.2 discusses community groups that are working to increase emphasis on high-quality software in scientific and research computing. Section 2.3 characterizes the development and use of software for scientific research.

2.1 Challenges

CSE software efforts face both technical and cultural challenges [9, 14, 17, 35],² a situation that is particularly true for ECP software efforts. The transition of applications to exploit massive on-node concurrency, the requirement to couple physics and scales, along with analytics and learning, and the continued disruption in the underlying hardware, system software, and programming environments together create the most challenging environment for developing CSE applications in at least two decades.

System architectural changes. Massive on-node concurrency is required in order to achieve exascale performance levels. We will have node counts on the order of 10^5 and clock speeds at 10^9 . Both of these values are expected to remain constant or grow only modestly over the next few years. The substantial performance growth curves are on the compute node. To succeed with exascale, we will need to realize concurrency levels of order 10^4 . While some of this performance can be achieved by populating nodes with multiple MPI ranks, much of the performance must come from vectorization (or, more generally, single instruction, multiple thread execution on GPUs) and lightweight tasking. Given the current state of our codes, addressing these requirements will be highly disruptive, displacing a huge proportion of executable code by the time work is complete. Communitywide, the cost of this transformation depends greatly on how quickly and effectively know-how and best practices emerge and then are disseminated across the community, in turn fostering greater programmer productivity and software sustainability.

New science frontiers, expanding aggregate teams. The dramatic increase in computing capability enabled by exascale systems opens new opportunities for work toward predictive science—coupling physics and scales, coupling simulations and data analytics, and incorporating outer-loop optimization, uncertainty quantification, and learning. As scientific applications increase in sophistication, no single person or team possesses the expertise and resources to address all aspects of a simulation. Interdisciplinary collaboration using software developed by distinct groups becomes essential, requiring coupling of code bases that were independently developed and necessitating increased coordination across teams [20].

The value of better software methodologies increases dramatically in these settings. It is the nature of software projects that the poor practices of one subteam will have a disproportionate negative impact on the whole project. For example, frequent regressions in one component impact the efforts of all components in the project. Similarly, incompatible practices and poor communication across subteams impact all aspects of a project. Improved software methodologies, tools, and communication, as well as compatible practices, are essential as we bring codes and teams together into composite efforts.

Thus, we face an urgent need to improve software productivity and sustainability in order to fully support the quality and integrity of computational research results, while promoting collaboration via software. A presentation in December 2018 by Paul Messina, founding director of ECP during 2016–2017, to the advisory committee of ASCR strongly supported this need for improving software practices. Messina stated:

² The Computational Science and Engineering Software Sustainability and Productivity (CSESSP) Challenges Workshop report [14] states that “CSE software as an enterprise has yet to emerge as a creative discipline in its own right. Both model and hardware complexity are growing simultaneously, and they both make the other more difficult to manage. The time is upon us to address the growing challenge of software productivity, quality, and sustainability that imperils the whole endeavor of computation-enabled science and engineering.”

A significant challenge going forward for ASCR (and all of international computational science) centers on more explicit emphasis on high-quality, high-performance, reusable, sustainable software itself, in order to encapsulate research advances in math, computer science and applications and enable next-generation advances toward predictive science.... Increased efforts should be supported to change the culture of computational science to fully acknowledge the important role that software plays as a foundation for CSE collaboration and scientific discovery. [26]

2.2 Growing a community of practice

Responding to these challenges, various grass-roots community groups have arisen in recent years to nurture “communities of practice” [45] in their respective spheres of influence, where like-minded people share information and experiences on effective approaches for creating, sustaining, and collaborating via scientific research software. These groups articulate key issues and needs to stakeholders, agencies, and the broader research community to effect changes in policies, funding, and reward structure, while advancing understanding of the importance of high-quality software to effective collaboration and the integrity of computational research. Groups such as CIG (<https://geodynamics.org>, geosciences) and MolSSI (<https://molssi.org>, molecular sciences) focus on the needs of a particular application area, while the following organizations address broader concerns [18, 25]:

The Software Sustainability Institute (SSI). <https://www.software.ac.uk>. Cultivating better and more sustainable research software to enable world-class research for the U.K. research community and collaborators.

Conceptualization of a U.S. Research Software Sustainability Institute (URSSI). <http://urssi.us>. Making the case for and planning a possible institute to improve science and engineering research by supporting the development and sustainability of research software in the U.S.

Apache Software Foundation (ASF). <https://apache.org>: Fostering the growth of open source software communities and providing technical infrastructure and support mechanisms needed by these communities.

Software Carpentry. <https://software-carpentry.org>. Teaching foundational coding skills to researchers, empowering them to develop research software, automate research tasks and workflows, and perform reproducible science.

Working Towards Sustainable Software for Science: Practice and Experiences (WSSSPE). <http://wssspe.researchcomputing.org.uk>: Promoting sustainable research software by addressing challenges related to the full lifecycle of research software through shared learning and community action.

NumFOCUS. <https://numfocus.org>: Promoting open code for better science, with emphasis on sustainable high-level programming languages, open code development, and reproducible scientific research.

rOpenSci. <https://ropensci.org>: Enabling open and reproducible research by creating technical and social infrastructure and advocating for a culture of data sharing and reusable software.

Research Software Alliance (ReSA). <http://www.researchsoft.org>: Bringing research software communities together to collaborate on the advancement of research software.

Research Software Engineering (RSE) movement. <https://us-rse.org>, <https://rse.ac.uk>: Advancing community, advocacy, and resources for those who regularly use expertise in programming to advance research.

The IDEAS project and collaborators are growing a community of practice to make software productivity and sustainability first-class concepts in (extreme-scale) scientific computing. Work focuses on disseminating best practices for scientific research software and establishing software ecosystems, while highlighting the critical roles of high-quality scientific software and the people who design and develop it.

2.3 Characterizing the development and use of software for scientific research

While the CSE and HPC software communities clearly need a specific focus on developer productivity and software sustainability, the broader software community has created important new platforms, tools, and methodologies that are reducing impediments. Software platforms such as GitHub.com, Atlassian, and Docker have dramatically reduced barriers to collaborative software development. Agile methodologies such as Scrum and Kanban provide lightweight structured processes that help software teams deliver valuable functionality to their users and clients (including themselves), while still managing risk and adapting to changing requirements and dependencies.

To address the technical and social challenges in CSE software, however, we must consider the unique needs and environments of the scientific research community. While we have much to learn from the mainstream software engineering community (more than we may at first realize), our needs and environments are in combination sufficiently unique so as to require fundamental research and strategies specifically for scientific software.

Extensive education and experience needed to develop scientific software. One of the strongest differences is the amount of education and experience a scientific software developer needs in a problem domain. A master’s degree or Ph.D. plus several years of experience is a common minimum. In contrast, developers in business systems software can start making meaningful contributions much sooner in their careers. Another difference is that the results from a scientific simulation can dramatically change the requirements for the next version of the software. In fact, it is the nature of scientific discovery that software requirements emerge regularly and frequently, such that long-term detailed plans are always subject to change.

Unique knowledge and skills of team members. Even when we can use mainstream approaches, rules of thumb commonly used in industry may not be appropriate for scientific computing. Many agile methodologies rely on the implicit assumption that a software team has several developers who could be assigned a given project task, making dynamic assignment straightforward. This assumption is typically not true for scientific software, where most tasks require a specialized scientific background that at most one or two team members possess. This fact makes planning much more challenging and forces significant adaptation of mainstream approaches.

Historically, we have seen that rigid application of mainstream software engineering approaches applied to scientific computing projects leads to strong negative outcomes. Without significant science-specific adaptations and new approaches developed within the scientific community, we run the risk that introducing more formal approaches to improve software productivity will actually reduce it. This circumstance is why research in software productivity and sustainability specifically for science, and in particular extreme-scale scientific computing, is so important [11].

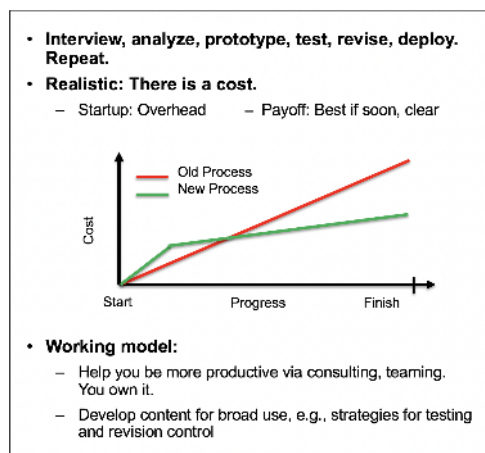
3. KEY SYNERGISTIC STRATEGIES AND IMPACT

The Exascale Computing Project’s aggressive goals require intensive development activities for both scientific applications and the supporting software tools and technologies. Developers must adapt to and anticipate new computer architectures and scale their codes to levels not previously possible, often also requiring new algorithms and approaches within the software.

The role of IDEAS within the ECP is to help ease the challenges of software development in this environment and to help the development teams ensure that DOE investment in the exascale software ecosystem is as productive and sustainable as possible. The IDEAS-ECP project brings together experts throughout the DOE community (see Appendix B.1) who have both experience in producing widely used scientific software products *and* passion for work on software productivity and sustainability issues. To address the challenges introduced in Section 2.1 in the context of extreme-scale science, we are partnering throughout all ECP

thrust areas: Application Development (science application projects and co-design centers), Software Technologies (projects that build reusable packages and tools for the ECP software ecosystem), and Hardware and Integration (including collaboration with DOE computing facilities and ECP training).

Critical to our approach is first respecting the requirements of teams to make progress on scientific and software goals and then helping them identify and deploy improved practices toward their goals. This strategy is especially important for ECP, which has an ambitious schedule, where teams must both deliver capabilities and improve software practices simultaneously. To assure providing true value to teams, we must carefully understand their requirements, including requirements for programmer productivity and software sustainability, and then facilitate improvements that deliver measurable positive impact soon after adoption (Figure 1). This approach promotes *continuous technology refreshment* [27], so that teams can improve software practices to reduce technical debt while ensuring continued scientific success.



Synergistic strategies to improve scientific software productivity and sustainability. We work in four complementary and interconnected areas, as shown in Figure 2. Advances in each area reinforce and extend the impact of the others.

Figure 1: Approach for improving software productivity and sustainability.

- **Curating methodologies to improve software practices of individuals and teams:** Providing information to improve software practices and processes—for better planning, development, performance, reliability, collaboration, and skills. A workflow for best practices content development (Section 3.2) promotes community collaboration and ensures that information is tailored to address the needs of high-performance CSE.
- **Incrementally and iteratively upgrading software practices:** Providing a lightweight iterative workflow known as *Productivity and Sustainability Improvement Planning (PSIP)*, where teams identify their most urgent software bottlenecks and work to overcome them (Section 3.3). Through improvements in software practices such as building, testing, refactoring, and onboarding (Appendix C), PSIP is helping teams to mitigate technical risk and advance overall scientific productivity.

Goal: Improve developer productivity and software sustainability while ensuring continued scientific success.

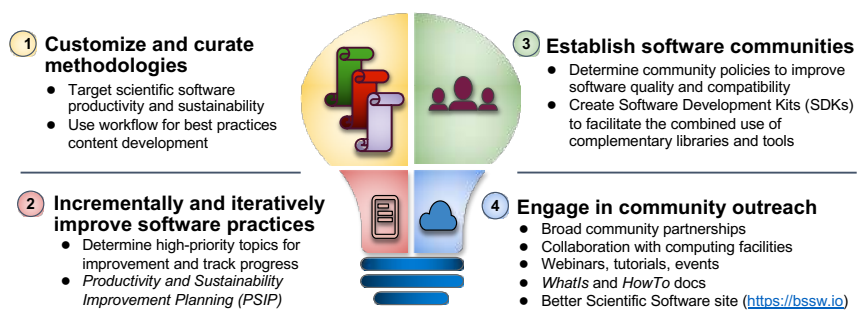


Figure 2: Community collaboration throughout four complementary areas improves scientific software productivity and sustainability, as a key aspect of advancing overall scientific productivity.

- **Establishing software communities:** Partnering with software teams to foster the growth of topical software communities (Section 3.4), whose members are working together to advance the quality, usability and interoperability of related software technologies. Key strategies are determining *community software policies* and creating topical *Software Development Kits (SDKs)*, which enable applications teams to more easily use complementary software products in combination.
- **Engaging in community outreach:** Encouraging emphasis on high-quality scientific software and sharing information about approaches to improve software productivity and sustainability (Section 3.5). For example, the *Better Scientific Software site* (<https://bssw.io>, Section 3.5.1) is a central hub for sharing information on practices, processes, tools, and experiences, while the *BSSw Fellowship Program* (Section 3.5.2) provides recognition and funding for leaders and advocates of high-quality scientific software. Additional outreach initiatives include producing the webinar series *Best Practices for HPC Software Developers* (Section 3.5.3), as well as organizing tutorials and other events (Sections 3.5.4–3.5.5) that provide training and nurture communities of practice that are working toward culture change in the broader scientific software community (Section 3.5.6).

Mitigating technical risks by building a firmer foundation for reproducible, sustainable science. We are guiding ECP teams to focus on practice improvements that not only help them work more effectively but also enable others to use and contribute to their software, thereby catalyzing collaboration and integration. For example, as discussed in Appendix C, we are assisting ECP teams to introduce process improvements such as expanded testing and more effective distributed revision control. This work helps to mitigate technical risk as ECP teams continually advance software capabilities—incorporating new algorithms, data structures, science functionality, and cross-team collaboration—in order to achieve science and performance goals on emerging extreme-scale architectures.

The IDEAS-ECP project is a focal point for resources that all teams can leverage. We work closely with any team that needs direct engagement, and we identify and disseminate to others better practices that we observe with one team, increasing the rate of improvement for all teams. Collaborations already are beginning to demonstrate that software productivity and sustainability improvements are possible. Our focus on best practices in scientific software, as well as development and use of software ecosystems to generate trusted computational results, enables scientists to engage effectively in their areas of expertise. At the same time, because many teams outside of ECP face similar challenges in next-generation CSE software and because project participants pursue ongoing work with diverse science groups, our approach also serves the broader HPC community.

3.1 Prerequisite: Determining crosscutting urgent needs

A prerequisite for this work is understanding current software practices, productivity challenges, and preferred approaches for collaboration and then identifying crosscutting, high-priority needs for training and outreach (see Figure 3). We employ a blend of community surveys for broad input and team interviews, which enable deeper conversations.

Team interviews. An interview protocol (approved by the Central DOE Institutional Review Board) and interview questions are available to the community in the [PSIP Tools repository](#). The interviews not only provide insights on high-priority needs, but also are the first stage of engagement with teams that may result in implementing the PSIP process (see Section 3.3).

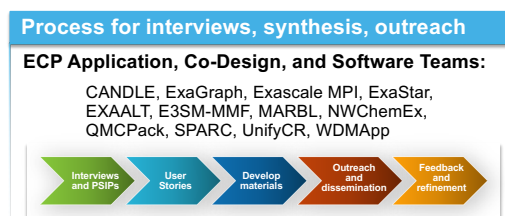


Figure 3: Interviews and surveys help convey the needs of science teams.

Insights into high-priority needs of CSE teams. Team interviews and surveys of the ECP community have revealed the common circumstance of projects being *aggregate teams* (see Figure 4), composed of multiple successful previously existing teams, where software is a primary vehicle of collaboration. Consequently, a priority is developing training and outreach materials on practices and tools that help foster more productive and sustainable collaboration (through software) for aggregate science teams [28, 33]. Additional topics of strong need include the following:

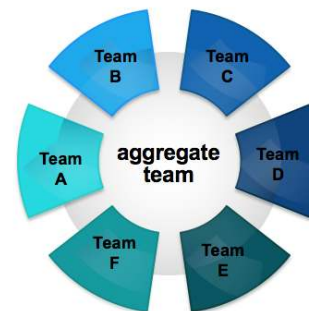


Figure 4: An urgent priority is helping aggregate teams collaborate more effectively through software, which encapsulates expertise across disciplines.

- Testing/verification of scientific software
- Team onboarding and team member transitions
- Intermediate and advanced Git (especially for aggregate teams)
- Code reviews for identifying defects
- Agile team management
- Agile workflows for scientific software
- Use of (interoperable) scientific libraries

We are addressing these topics in training and outreach (see Sections 3.5). The initial target teams are directly contributing to the development of first-of-a-kind learning material for scientific computation projects, with concrete use cases, motivation, feedback, and refinement of methodologies.

3.2 Curating methodologies to improve software practices of individuals and teams

Distilling collective experience in scientific software best practices. The multifaceted approach introduced in Figure 2 focuses on developing, customizing, curating, and deploying best practices as the fundamental way to improve software sustainability and programmer productivity for extreme-scale science. This work requires distilling the collective understanding of team and community members with many years of valuable experience in designing and producing high-quality, reusable HPC scientific software.

This experience, when combined with knowledge obtained from the broader software engineering community, has provided a foundation for focused discussion, distillation, and development of a large and growing collection of resources for CSE software teams, as shown by the best practices content lifecycle model in Figure 5. For example, some content is produced as part of our “What Is” and “How To” documents, which provide concise characterizations and best practices for important topics in CSE software projects (such as software configuration, documentation, testing, revision control, and agile methodologies), thus enabling teams to consider improvements at a small but impactful scale.

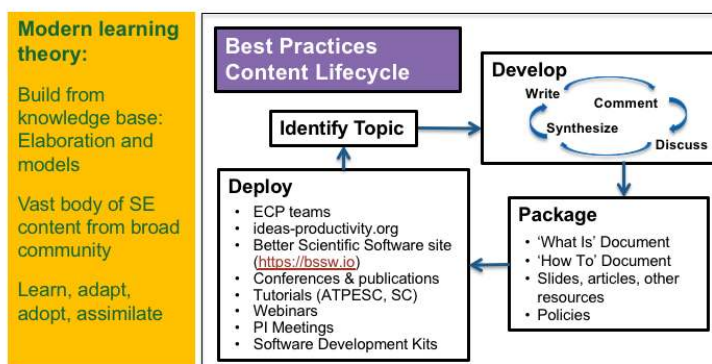


Figure 5: Workflow diagram describing best practices content development and deployment. As a team, we identify topics that we believe will be of interest to the high-performance CSE community and for which we have some expertise or we can enlist the expertise of others. Once a topic is chosen, we iterate to develop the content in a typically lively and stimulating exchange. As the content solidifies, we produce “What Is” and “How To” documents, slides, and policy lists that can be deployed in a variety of settings, including the BSSw.io site.

3.3 Incrementally and iteratively upgrading software practices

Scientific teams typically are funded to generate scientific results, not software. This is a competitive process, and teams cannot usually expend much time or effort outside of writing the software features that support the generation of new results. Therefore, any productivity or sustainability improvements must be incremental and integrated into the primary feature development process.

PSIP: a lightweight workflow for iteratively and incrementally upgrading software practices. *Productivity and sustainability improvement planning (PSIP)* (see <https://bssw.io/psip> [15, 32]) enables CSE teams to identify their most urgent software bottlenecks and work to overcome them, as a key aspect of increasing overall scientific productivity. The objectives of the PSIP process are to capture and convey the practices, processes, policies, and tools of a given software project. The PSIP workflow is intended to be lightweight and fit within a project’s planning and development process. It is not meant to be an assessment or evaluation tool. Instead PSIP captures the tacit, more subjective aspects of team collaboration, workflow planning, and progress tracking. In the potential absence of planning and development processes, and as scientific software teams scale to larger, more diverse aggregate teams, unforeseen disruptions or inefficiencies can often impede productivity and innovation [34]. PSIP is designed to bootstrap aggregate team capabilities into best practices, introduce the application of appropriate resources, and encourage teams to adopt a culture of process improvement.

PSIP workflow. As shown in Figure 6, the PSIP framework is an iterative, incremental, repeatable, cyclic process for improvement planning. The cyclic nature of the PSIP process enables software development teams to improve overall project quality and achieve science goals by encouraging frequent iteration and reflection. Software teams may work through these steps on their own or with the assistance of a PSIP facilitator, who may augment PSIP by bringing process experience and objectivity to the effort, coaching the team on improving effectiveness and efficiency.



Figure 6: PSIP workflow.

- **Summarize current project practices:** The first phase includes briefly documenting current project practices. Recording the original state of the project is important both to provide a baseline for measuring progress and to help identify areas that are ready for improvement.
- **Set goals:** Completing this step typically brings to light project practices that can benefit from a focused improvement effort. Although any number of goals may be identified in this step, a limited set is selected at any given time that can benefit the project and can be achieved within a predictable span of time (a few weeks to a few months). Goals not chosen at this time may be tabled for future iterations.
- **Construct a progress tracking card (PTC):** A PTC is a brief document containing the target, or goal, of the planning activity and a step-by-step list of activities or outcomes (range of 0–5 is suggested) that

incrementally lead to improvements in team effectiveness and efficiency. Each practice will have its own PTC. Teams may select PTCs from a PTC catalog, define their own PTC, or modify PTCs found in the catalog. The purpose of the PTC is to help teams set and achieve improvement goals. The PTC is not a tool for external assessment or comparison with other projects. In fact, since PTCs are custom designed for each project, comparisons are typically not possible.

- **Record current PTC values:** To establish baseline capabilities and track progress, teams record the initial values for each PTC.
- **Create a practice improvement plan:** To increase the values in a PTC (corresponding to improvements in software productivity and sustainability), teams develop a plan to reach a higher value for each PTC.
- **Execute the plan:** Team efforts are focused on improving the selected practices described in the PTC. At first, teams may see a slowdown, as they work to start or improve a given practice. The slowdown in most cases is proportional to the amount of change, but ideally teams should see steady progress on a weekly basis after the initial phase and be able to complete execution of a particular practice improvement within a few months.
- **Assess progress:** During execution, teams assess and determine the rate of progress. They adjust their strategy for success if needed. If progress is delayed too long, teams usually start the next PSIP iteration.
- **Repeat:** The PSIP process is iterative. Continual process improvement is a valuable attribute for any software project. The PSIP process may be used to guide improvement planning within software projects and across aggregate projects.

During PSIP or at its conclusion, teams may elect to share their PTCs, best practices, and/or lessons learned with the community in a variety of ways, including contributing blog posts on PSIP progress to the BSSw website (Section 3.5.1) and modifying, curating, or creating tools such as new PSIP PTCs or resources for inclusion in the PSIP catalog.³

PSIP outcomes. As demonstrated by ECP teams’ productivity advances using the PSIP process for improving software builds, testing, refactoring, and onboarding (Appendix C), PSIP helps to mitigate technical risk as ECP teams continually advance software capabilities to achieve ECP science and performance goals. PSIP provides a mechanism to set goals collaboratively, get team buy-in, and enable periodic status checks to ensure that the goals and execution are aligned. PSIP is easy to learn, especially for scientists who cannot dedicate time and resources to more formal or heavyweight approaches.

PSIP next steps. We are in the midst of growing the PTC catalog and conducting more research on the use of PSIP with software teams. We note that PSIP does not address the issue of teams not being rewarded for efforts to improve developer productivity and software sustainability; for PSIP to be broadly effective, the CSE community must prioritize the value of these improvements, something that we observe is happening slowly.

³See <https://bssw.io/psip> for further PSIP details, including the PTC catalog.

3.4 Establishing software communities

Needs for software ecosystems and increasing transparency of computational results. The ECP software ecosystem will comprise a wide array of tools, libraries, programming models, and performance portability frameworks, all of which are expected to be used by scientific applications. If application developers are to build simulations using this software, it must be interoperable (see Figure 7). For the interoperability to be sustainable, it must be tested regularly; and the development teams will need to work together to define and support common interfaces, as well as to use best practices in software engineering to make codes more maintainable, extensible, and modular. The software must also be portable and reliably deployed at leadership computing facilities. Without effective software deployment infrastructure and processes, application scientists will not benefit from the exascale software stack, and scientific productivity will be sacrificed as scientists struggle to deploy and scale their codes on new platforms. Moreover, to assure confidence in computational science discoveries, teams must improve transparency and reproducibility of computational results.

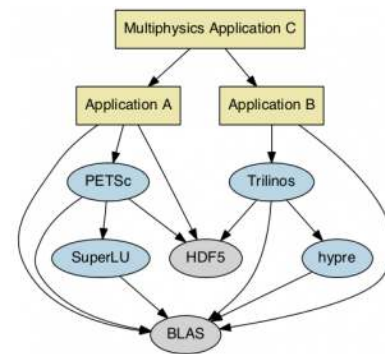


Figure 7: Simple xSDK example demonstrating the combined use of several xSDK numerical libraries, as needed for a variety of multiphysics simulations that combine application components developed by different groups.

SDKs and community policies: unity in essentials, otherwise diversity. Self-organizing *software communities*, including developers and users of related technologies, who deeply understand their own requirements and priorities, are well positioned to determine effective strategies for collaboration and coordination. Our approach centers on (1) establishing *community software policies* to advance the quality, usability, and interoperability of related software technologies, while supporting autonomy of diverse teams that naturally have different drivers and constraints, and (2) creating *Software Development Kits* to facilitate the combined use of independent software packages by application teams.

A Software Development Kit is a collection of related software products (packages), where coordination across package teams improves usability and practices, while fostering community growth among teams that develop similar and complementary capabilities. SDKs are a key delivery vehicle for ECP Software Technologies (see Figure 8), where similar products are grouped into an SDK to promote collaboration and usability: programming models and runtimes core, tools and technologies, compilers and support, math libraries (xSDK), visualization analysis and reduction, data management and I/O.

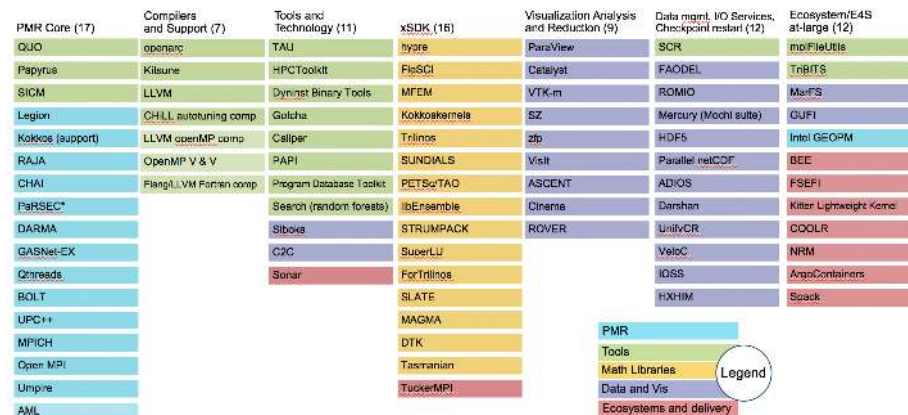


Figure 8: The initial breakdown of ECP software products into SDKs chosen in part because of the effectiveness of the horizontal coupling of the xSDK.

The SDK approach in ECP builds on experiences with the Extreme-scale Scientific Software Development Kit (xSDK, <https://xsdk.info>), which started as part of the IDEAS-Classic project and later transitioned to the Software Technologies element of ECP. The xSDK has successfully resolved a variety of difficulties in

package compatibility and combined a number of popular math libraries into a single release using Spack (<https://spack.io>)—a flexible package manager that supports multiple versions, configurations, platforms, and compilers. To help address challenges in interoperability and sustainability of software written by diverse groups at different institutions, the developers of xSDK packages have adopted a set of community policies (see Figure 9), as explained in a BSSw blog post by Piotr Luszczek and Ulrike Meier Yang [22].

The startup of the SDK project benefited greatly from IDEAS experiences. Specifically, the horizontal coupling approach for defining SDKs was chosen in part and with greater confidence because of its effectiveness in the xSDK. A horizontal rather than vertical coupling means that the members of an SDK share a similar function and purpose, for example compiler frameworks, or, as in the case of

- M1. Support xSDK community GNU Autoconf or CMake options.
- M2. Provide a comprehensive test suite.
- M3. Employ user-provided MPI communicator.
- M4. Give best effort at portability to key architectures.
- M5. Provide a documented, reliable way to contact the development team.

the xSDK, math libraries. The initial SDK community policy efforts started with the xSDK community policies, and discussions about generalization of policies have benefited from xSDK experience. Likewise, IDEAS experience has been critical in an SDK effort to better define sustainability and associated requirements in the context of the ECP software ecosystem. For example, see whitepapers and presentations at the 2019 Colleagueville Workshop on Sustainable Scientific Software [10]. IDEAS experience also is contributing to planning by ECP and DOE computing facilities for continuous integration testing, where challenges include the varied level of software engineering maturity among ECP software products, maintenance of interoperability, and unknown resource allocations for future testing.

Figure 9: Sample of mandatory xSDK community policies, which have provided a starting point for SDK policy discussions. The complete list of 16 mandatory and 7 recommended xSDK package policies is available via <https://xsdk.info/policies>.

Methodologies to support SDKs and trusted computational results. Leveraging involvement in ECP software ecosystem activities as well as experience in defining and deploying best practices in software engineering for computational science, members of the IDEAS-ECP project are devising resources to help teams prepare for and participate in SDKs and also to increase trust in computational results. We have established two repositories [37, 41]; forthcoming work will focus on expanding methodologies materials in these repositories and corresponding outreach to computational science teams. Examples of early impact on the software packages SuperLU and Ginkgo, which are part of the xSDK, are discussed in BSSw blog articles by Sherry Li [21] and Hartwig Anzt [2].

3.5 Engaging in community outreach

Central to the IDEAS project are multipronged outreach efforts [5] that help grow and mobilize a dynamic community to improve software quality, productivity, and sustainability. Key initiatives are launching the BSSw.io website (Section 3.5.1), initiating the BSSw Fellowship Program (Section 3.5.2), producing the webinar series *Best Practices for HPC Software Developers* (Section 3.5.3), and teaching tutorials on strategies for various aspects of better scientific software (Section 3.5.4). We also organize technical meetings on HPC software development topics (Section 3.5.5), along with complementary efforts to encourage emphasis on high-quality software in the broader community (Section 3.5.6). Our goal is to establish a “virtuous cycle” in which widespread awareness of the importance of software quality and related issues within the HPC computational science and engineering community, including the Exascale Computing Project, in turn promotes sharing, discussion, and refinement of practices and resources for producing better scientific software for the benefit of the ECP as well as the broader community.

3.5.1 Community-driven resources: BSSw.io website

The BSSw site (<https://bssw.io>) is a central hub for sharing information on practices, techniques, experiences, and tools to improve developer productivity and software sustainability for CSE and related technical

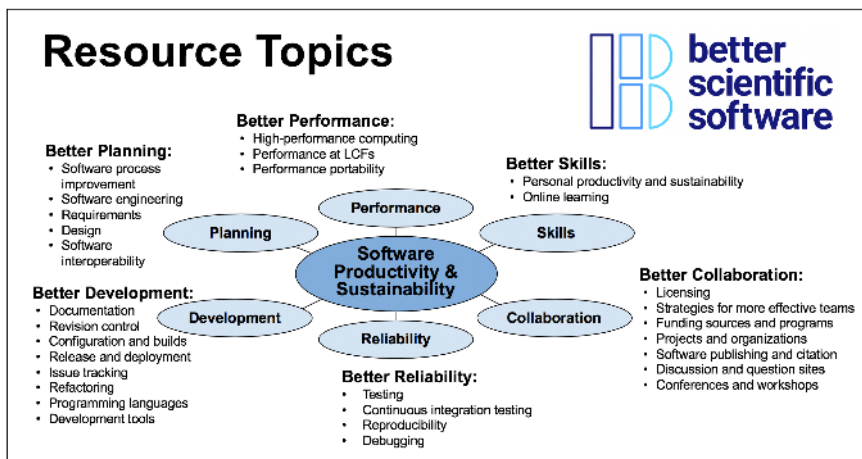


Figure 11: Resources on BSSw.io span throughout a range of topics, including scientific software planning, development, performance, reliability, collaboration, and skills.

computing areas. The site features curated content, experiences, and insights provided by the international community, including researchers, practitioners, and stakeholders from national laboratories, academia, and industry who are dedicated to curating, creating, and disseminating information that leads to improved CSE software. Historically, opportunities for CSE software developers to exchange information and experiences have been limited; BSSw provides a space to support this kind of sharing.

Community-driven resources for advancing software productivity and sustainability.

Launched in November 2017 as an IDEAS initiative, the BSSw platform offers easy access to resources and training materials provided by a growing community of HPC and CSE contributors. BSSw content spans a range of topics shown in Figure 11, including introductory *What Is* and *How To* information covering basic steps for improving software productivity and sustainability. The site includes original articles, information about events, and *curated content*—brief articles that highlight existing materials (papers, books, videos, web content, etc.), describing why the scientific software community might find them of value.

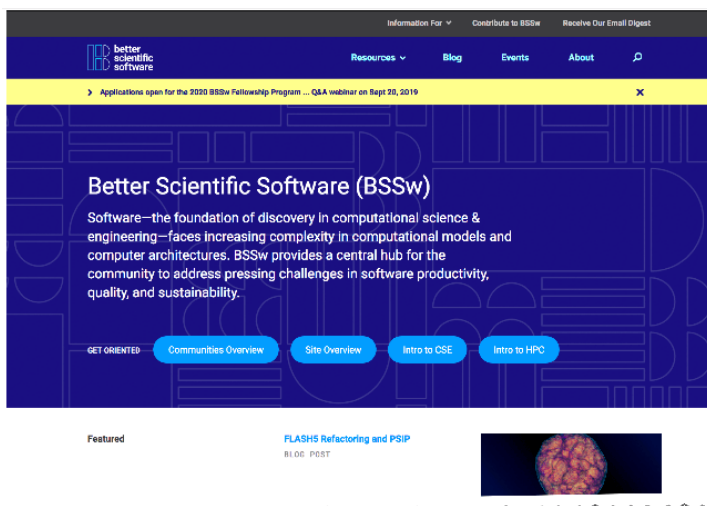


Figure 10: The BSSw site hosts a growing community-driven collection of resources on software productivity and sustainability.

Some resources on the topic of *software testing* are shown in Figure 12; a few other recent resources are the following:

- *Working Effectively with Legacy Code*, Ross Bartlett (SNL)
- *Unit Testing C++ with Catch*, Mark Dewing (ANL)
- *A Collection of Resources for Sustaining Open Source Software*, Todd Gamblin (LLNL)
- *Exploring Best Practices for Scientific Computing*, Patricia Grubel (LANL)
- *Python for HPC*, Steve Hudson (ANL)
- *An Introduction to Integrated Development Environments (IDEs) for Scientific Software Development*, Greg Watson (ORNL)

BSSw collaborative content creation through GitHub. The BSSw site is the starting point for any new user. A GitHub backend enables content development using a collaborative, open workflow. Content can also be contributed with an easy-to-use Google form. Anyone with experience or expertise who can help other scientific software teams is encouraged to contribute an article or pointer to relevant work (see <https://bssw.io/contribute>). Our approach employs a custom Ruby-on-Rails content management system that automatically imports, updates, and formats content from GitHub. The [BSSw Editorial Board](#) has established a workflow for engaging content [contributors](#) and regularly updating resources to ensure that content remains fresh. We are extending site functionality based on feedback from the community.

BSSw blog. The BSSw site features a growing collection of original [blog articles](#), addressing topics such as science teams' experiences with productivity-related software issues and strategies for collaborative computational science. Blog posts thus far are listed in [Table 3](#), [Appendix D](#).

BSSw community landing pages. BSSw encompasses a rich variety of communities who are working to advance the methods and practices of CSE software. [BSSw community landing pages](#) provide custom starting points for using the site and promote shared understanding of scientific software issues. Curators of a community landing page can customize content to serve the needs of community members. BSSw communities include a growing set of science-focused areas as well as crosscutting areas (scientific libraries, software engineering, supercomputing facilities and their users, and exascale computing).

BSSw future directions. By providing a venue to share information and experiences on software issues, BSSw is raising awareness of the importance of good software practices to scientific productivity and enabling readers to discover potential connections to their own needs and workflows. Future plans center on establishing broader community leadership and growth in order to help CSE researchers (regardless of nationality and funding sources) to increase software productivity, quality, and sustainability while changing CSE culture to fully support software's essential role.

3.5.2 Recognition: BSSw Fellowship Program

Growing a community of leaders in scientific software productivity and sustainability. Addressing the scientific software challenges introduced in [Section 2.1](#) requires broad community collaboration to change the culture of computational science, increasing emphasis on software itself and the people who create it.

The BSSw Fellowship Program (<https://bssw.io/fellowship>) gives recognition and funding to leaders and advocates of high-quality scientific software. The main goal is to foster and promote practices, processes, and tools to improve developer productivity and software sustainability of scientific codes. BSSw Fellows are selected annually based on an application process that includes the proposal of a funded activity that pro-

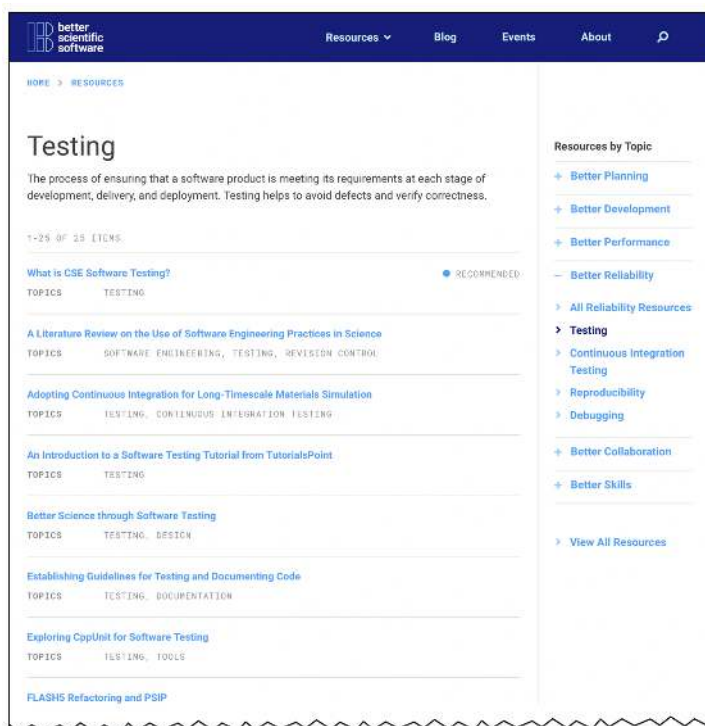


Figure 12: Selected BSSw resources on software testing.

notes better scientific software. We encourage diverse applicants at all career stages, ranging from students through early-career, mid-career, and senior professionals. Applicants must be affiliated with a U.S.-based institution that is able to receive funding from DOE.

2019 BSSw Fellows. Each member of the 2019 class, pictured in Figure 13, received an award of \$25,000 for an activity that promotes better scientific software. Modes of outreach include webinars, tutorials, workshops, online training materials, and blog articles. For example, presentations in the HPC Best Practices Webinar Series (Section 3.5.3) include *Discovering and Addressing Social Challenges in the Evolution of Scientific Software Projects* (R. Gassmoeller, Sept. 2019) and *Tools and Techniques for Floating-Point Analysis* (I. Laguna, Oct. 2019). More information on projects and perspectives of BSSw Fellows (classes 2018, 2019, 2020) is available via <https://bssw.io/pages/meet-our-fellows>, including pointers to resources for on-line learning, such as slides and recordings of tutorials.

DOE community engagement.

The BSSw Fellowship Program provides a mechanism to engage members of the broader CSE community and connect them with the DOE community—tapping their interest and experience in issues related to software productivity and sustainability. BSSw Fellows and Honorable Mentions are given the opportunity to attend the ECP Annual Meeting and to visit DOE laboratories to present their work, enable additional collaborations, and generally expose them to the DOE environment and work.

Future directions for the BSSw Fellowship Program.

We are growing a community of BSSw Fellowship alumni who serve as leaders, mentors, and consultants to increase the visibility of those involved in scientific software production and sustainability in the pursuit of scientific discovery. The long-term vision for the BSSw organization is to serve as an international community-driven and community-managed resource for scientific software improvement, with content and processes provided by volunteers. Future plans center on working toward broad community leadership, growth, and diversification of the BSSw organization and BSSw Fellowship program—that is, expanding collaboration with complementary groups, stakeholders, sponsors, and individual contributors.

2019 Class









Fellows			
			
Rene Gassmoeller University of California, Davis	Ignacio Laguna Lawrence Livermore National Laboratory	Tanu Malik DePaul University	Kyle Niemeyer Oregon State University
Guiding your scientific software project from inception to long-term sustainability	Improving the reliability of scientific applications by analyzing and debugging floating-point software	Reducing technical debt in scientific software through reproducible containers	Educating scientists on best practices for developing research software
Honorable Mentions			
			
Stephen Andrews Los Alamos National Laboratory	Nasir Eisty University of Alabama	Benjamin Pritchard Virginia Tech	Vanessa Sochat Stanford University
Staff Scientist, XCP-8: Verification and Analysis	Ph.D. Student, Computer Science	Software Scientist, Molecular Sciences Software Institute	Research Software Engineer, Stanford Research Computing Center

Figure 13: The BSSw fellows program fosters and promotes practices, processes, and tools to improve developer productivity and software sustainability of scientific codes.

3.5.3 Webinar series: Best Practices for HPC Software Developers

Webinars are, by now, a well-established approach to reaching a broader community with seminar-like content. The so-called “HPC Best Practices” (HPC-BP) webinar series, launched in 2016, provides a venue for many topics in the development of software for high-performance scientific computing from practitioners mostly outside of the IDEAS project.

The webinar series was initially developed in conjunction with the three ASCR computing facilities (ALCF, NERSC, and OLCF), as a response to a growing need among the facility’s users for a better understanding and use of software engineering best practices. The first seven webinars were presented in the summer of 2016 (roughly every two weeks), alternating software best practices and other HPC topics. In the summer of 2017, the series was restarted, in collaboration with the ECP Training program, with a monthly cadence and more emphasis on topics related to productive and sustainable HPC software.

Webinar series strategy and successes.

We have drawn on our experience and engagement in the HPC software development community to identify and prioritize topics for webinars. At the same time, we look to our colleagues who are working to raise the visibility of and improve software development practices to present a wide variety of topics that we believe are relevant to the developers of HPC scientific software. A total of 35 webinars have been presented in the period 2016–2019 as detailed in Table 1 in Appendix D.

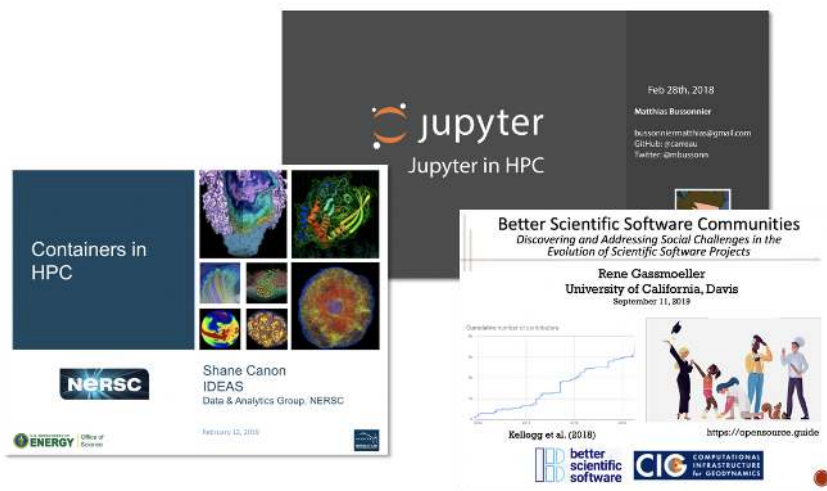


Figure 14: Sample of webinars in the HPC Best Practices series.

The reception for the webinar series has been overwhelmingly positive. A total of 2181 people have participated in the 28 webinars organized to date by the IDEAS-ECP project (2017–2019), which is an average of 78 participants at each webinar. Registrations consistently run about twice the number of those who participate, and ECP members constitute roughly 30% of registrations. (For further details, see Table 2 in Appendix D.) Our list of webinar registrants and participants for the entire series comprises more than 1900 unique email addresses, which is used to announce upcoming HPC-BP webinars as well as other IDEAS-organized events. The webinars are also announced to ECP members, DOE computing facility users, and other venues.

The methodology we use for delivering the webinars is described in the *HowTo* short paper [24], presented in the Fifth SC Workshop on Best Practices for HPC Training and Education at SC18. Based on our experience, we have created a repository [23] with information and documents related to a process for delivering webinars, including a generic checklist that can be adapted to different types of organizations.

Webinar sessions, recordings, slides, and curated Q&As are archived after each webinar event at <https://ideas-productivity.org/events/hpc-best-practices-webinars>. This lasting repository of content on software development practices is useful for those who could not attend the live presentation.

Impact through DOE computing facilities partnership. The level of interest in the HPC-BP webinar series, as measured by the numbers of registrants and attendees, greatly exceeds the typical experience of ASCR facilities with HPC-oriented training. The response demonstrates that many software developers recognize that they need more than just HPC skills to be productive. This interest in the HPC-BP webinar series also demonstrates that there is a receptive audience for information to be provided through high-quality webinars. A unique feature of HPC-BP is that its agenda is dynamic and implemented in concert with ASCR’s computer facilities, the webinars *Python in HPC* and *Jupyter and HPC* being examples. For

future topics, we will continue paying close attention to the community, for example, through interactions at events and through an annual survey conducted by the ECP training component.

3.5.4 Tutorials on better scientific software

Strategy for training on topics in software productivity.

Tutorials are an important opportunity for the IDEAS team to present what we have identified as important considerations and practices in scientific software development, typically based on “best practices” identified in the broader software engineering community, tailored and adapted based on the experiences of team members others in the HPC CSE software community. We have designed these tutorials to target learners at primarily beginner and intermediate levels of experience, and we cover a range of topics including software design and refactoring, testing, and reproducibility, as well as development and management practices from both individual and team perspectives. During our tutorials, we encourage interactions with the audience that bring their own experiences and questions into the discussion.



Figure 15: Four IDEAS team members presented a well-attended and highly rated tutorial at SC19.

Broad impact through both in-person attendance and online self-study. Since 2016, initially under the IDEAS-Classic project and then IDEAS-ECP, we have given more than a dozen tutorials in a variety of venues, including ECP events (the ECP Annual meetings, and the Argonne Training Program for Extreme-Scale Computing) and major conferences (Supercomputing, ISC, SIAM Computational Science and Engineering). Most of these venues involve either invitations from the organizers or a competitive peer review-based selection process, where repeated offerings are indicative of the value of these tutorials to the participants and organizers. Tutorials vary in attendance but are typically in the range of a few tens of participants to many tens of participants. Collectively, we conservatively estimate that more than 500 participants have been directly impacted through these tutorials, including many early-career researchers and members of ECP teams. We make a practice of making the slides from each tutorial presentation available in an archival location (usually FigShare.com); for example, see SC19 tutorial slides [6]. Additionally, in some venues, the organizers have recorded the presentations and made the videos available. Through such archives, the tutorials content can also be reviewed later by the participants and used for self-study by others.

Future directions for expanding our reach and our content. Since its inception, our tutorial material has undergone continual evolution, with at least minor changes to nearly every module from one presentation to the next. There have also been more significant changes, introducing new modules and substantially rethinking our presentation of some topics. We expect such advances to continue, as the tutorial reflects our evolving understanding of key elements that lead to better scientific software, and how to explain it to others. We also plan to broaden our selection of topics, based on participant feedback and experiences with our ECP and other colleagues through both formal (e.g., PSIP activities) and casual interactions. Additionally, we plan to explore approaches that will allow live, interactive tutorials to scale to even larger audiences. For example, we are working toward multi-site offerings in which instructors at different locations share lecture responsibilities via videoconferencing and locally support students in hands-on exercises.

3.5.5 Technical meetings on HPC software development topics

Promoting the discussion of software development practices and experiences. A key reason that software development practices promoting productivity and sustainability are not more widely known and used is that, historically, there have been few opportunities and little encouragement for people to share experiences. The CSE community is mostly organized around the sharing of new *scientific* advances, as well as methodological and algorithmic improvements. Much less common is discussion of *how* the software that gave rise to the advancements has been created and sustained.

Consequently, along with tutorials and our webinar series, IDEAS also includes a strong focus on providing venues at technical meetings to encourage the discussion of *software* on par with the other aspects of our work. A growing number of scientific meetings “crowd source” portions of their content, allowing meeting participants to propose and organize sessions of various kinds. In some cases, such proposals are peer reviewed and can be competitive. We typically organize such sessions together with like-minded community members and invite a broad selection of speakers. We advertise these events widely within our community, and create archives to capture the events for future reference. In some cases, the meeting organizers featured these sessions, recording them and making them available available online after the event. We also sometimes organize standalone workshops, which offer the opportunity for more in-depth interactions, often over an extended period of time.

Recent events. The most prevalent events we have organized have been "minisymposia", which are thematic sessions within a larger conference. SIAM meetings, such as Parallel Processing (PP) and Computational Science and Engineering (CSE), as well as the PASC conferences in Switzerland offer this format. Since 2015, we have co-organized a total of 9 minisymposia at six different meetings. SIAM meetings have also begun offering opportunities for thematic poster sessions, and we have organized two at recent SIAM-CSE meetings, attracting 28 posters for each.

Standalone workshops have been less common, because of the significantly larger effort to organize, and the challenges of getting participants to commit to “extra” time and travel in their busy schedules. However, the recent 2019 Collegeville Workshop on Sustainable Scientific Software [10] provided an opportunity for forty participants to gather for two full days of discussions on sustainability of scientific software. In addition to these types of events, we have taken advantage of opportunities to organize less formal sessions, such as Birds of a Feather (BoF) at Supercomputing and ISC conferences (6 events since 2015) and breakout sessions on various topics at ECP Annual Meetings (8 events 2018–2020).

Next steps in raising the profile of software topics. These events (e.g., [4]) have played a significant role in promoting the discussion of software development practices in the CSE community, not only raising the profile of this topic in the minds of both participants and observers (i.e., those seeing the events in the meeting program, or listings of contributions on participant CVs), but also, very practically, facilitating the exchange of information that is a critical component of the virtuous cycle toward which we are working. We have received universally positive feedback on these events from speakers and audience members, who appreciate the opportunity to discuss their software-related experiences on par with their scientific and methodological work. Given the effectiveness and popularity of these events, we plan to continue and, where possible, expand them. IDEAS is not the only group thinking along these lines, and we have taken numerous opportunities to engage with others domestically and internationally to expand the reach of these activities.



Figure 16: Thematic poster group at SIAM CSE19, archived on Figshare.

3.5.6 Working toward culture change

Various community groups (see Section 2.2) are addressing the urgent challenges of scientific software (see Section 2.1) by promoting changes in the culture of scientific computing. This section discusses initiatives in transparency and reproducibility that are increasing the integrity of computational results, actions by funding agencies that are promoting attention to high-quality scientific software, and changes in career paths and institutional structure that support the development of high-quality software. In addition, we articulate the urgent need for explicit funding for work on software productivity and sustainability.

Transparency and reproducibility initiatives. Scientists generally seek to produce trusted scientific results. At the same time, scientific research is competitive, and the level of assurance a scientific team can obtain when producing results is at least partly a function of the practices in its research community. If one team seek to improve the quality of their results, they will typically require more effort, investment in new skills, and some reduction in their output rate. At present, there is concern in the scientific community about transparency and reproducibility of scientific results. While much of the high-profile focus has been on social sciences [7], the computational science community is not immune. One recent case, discussed in the article “The war over supercooled water” [38], the condensed matter research community did not require transparency or reproducibility in how results were obtain. Two research teams obtained very different results for the behavior of pure water at low temperature and the differences were resolved years later only when one team finally obtained the software environment used by the other. This example and others illustrate that improving trustworthiness requires elevating community expectations for transparency and reproducibility. One compelling approach is to introduce transparency and reproducibility initiatives as part of the research publication process.

Numerous journal and conference publications have developed reproducibility incentives and requirements for publications. Examples include the Association for Computing Machinery (ACM) Transactions on Mathematical Software reproducible computational results initiative [12] and the Supercomputing Conference series reproducibility initiative [36]. In both cases, authors are given badges as rewards for improving the transparency and reproducibility of their results [1]. Over the past four years of the Supercomputing Conference (SC16–19), expectations for making computational artifacts available have gone from being optional to being mandatory for best paper consideration to being mandatory for all accepted papers. This kind of gradual increase in rigor has enabled the SC community to adapt their workflows over the span of several years.

Reproducibility expectations raise the value of improved developer productivity and software sustainability. Any reproducibility process will require software and data versioning, opening up code and data to people beyond a particular research team, and similar transparency changes, creating incentives to invest in productivity and sustainability. Improved developer productivity and software sustainability are critical enablers of effective and efficient transparency and reproducibility.

Career paths and institutional structure. An important cultural advance is the expansion of career paths in which researchers focus explicitly on high-quality scientific software, along with expansion of corresponding community and organizational structures. For example, *research software engineers* [42, 44] combine an intimate understanding of research with experience in software engineering, while *research software scientists* study and improve the development and use of research software [11]. Such people are employed worldwide by institutions that develop scientific software, typically integrated within traditional organizational structures and also in newly established groups that focus on research software, such as the Research Software Engineering Group in the Computer Science and Mathematics Division of Oak Ridge National Laboratory and the Software Engineering & Research Group in the Center for Computing Research at Sandia National Laboratories.

Influence of funding agencies. Funding agencies value the impact of software products whose development they sponsor. As scientific software ecosystems mature, the number of software products that are used in research increases (see, e.g., [18]), while the complexity of the ecosystems into which new software is introduced also grows. Funding agencies can provide natural incentives for elevating software quality by explicitly requesting software quality plans in funding opportunity announcements and by incorporating quality assessment in funded project review cycles. These incentives will naturally lead to an increase interest in improving developer productivity and software sustainability and result in benefits from productivity and sustainability investments.

We have seen some activity from funding agencies in this area. For example, in the funding opportunity announcement SC_FOA_0001724, the DOE Office of Science Subsurface Biogeochemical Research office requested a Software Productivity and Sustainability Plan as part of all proposals. The DOE Exascale Computing Project represents an even larger and more comprehensive expectation on software quality. ECP emphasizes software planning, development, and delivery with a level of rigor that is much higher than historical levels. Project funding is tied to successful delivery of capable and sustainable software.

Urgent need for explicit funding for software productivity and sustainability. Scientific software, which supports scientific research in much the same way that a light source or telescope does, requires a substantial investment of human and capital resources, as well as basic research on software productivity and sustainability so that the resulting software products are fully up to the task of predictive simulations and decision support. As discussed in Section 2.3, efforts to improve scientific software must be more than just straightforward adoption of mainstream approaches. The needs and environments of the scientific software community are in combination sufficiently unique so as to require fundamental research specifically for scientific software.

To create high-quality scientific applications, libraries, and tools (generally, but especially in support of DOE’s missions in high-performance computational science), we advocate a three-pronged approach illustrated in Figure 17.

- Long-term investment in *research software science*, that is, basic research in the science of research software [11]: A persistent focus on exploring, adapting, and adopting new productivity enhancements is essential for continued innovation in meeting the challenges of high-performance computational science.
- Incorporation of basic research into reusable software and best practices.
 - Development of interoperable scientific software ecosystems: Careful design and implementation of a complementary system of scientific software components can provide tremendous value, enabling rapid application development via component composition, performance portability, and access to the latest algorithms and software tools. Effective ecosystems will require well-conceived software architectures and interfaces—demanding advanced design skills from lead developers,

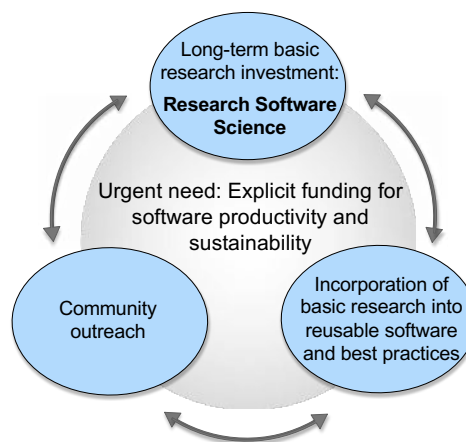


Figure 17: Long-term investment in research software science (i.e., research on scientific software productivity and sustainability) is essential in order to determine best practices that can then be readily incorporated into software products and disseminated to the broader community. New insights about urgent needs then feed back to inform high-priority areas of new research. Improving the quality of software developed by scientific teams will also require investment in new skills and infrastructure by the teams themselves. Funding agencies, publishers, and other stakeholders will play a key role in spurring team innovations.

which will come from nurturing software expertise in key staff.

- Collaboration with science application teams: Effective teaming with application partners will require techniques for assessing and analyzing requirements, definition and tracking of relevant metrics, and strategies for engaging application developers to incorporate productivity improvements that are sustainable.
- Community outreach: Broad deployment of new ideas, tools and techniques will require careful packaging of all productivity improvement capabilities, and deployment in collaboration with leadership computing facilities to the HPC science community.

All three elements are critical and feed into one another. In particular, without funding for basic research on scientific software productivity and sustainability, the pipeline of new improvements will stagnate, thereby weakening the resulting research software and the scientific investigations it supports.

Improving the quality of software developed by scientific teams will also require investment in new skills and infrastructure by the teams themselves. We believe that uniformly raising expectations on these teams will be important. Funding agencies, publishers, and other stakeholders play an essential role in elevating quality expectations. Expecting funded projects to produce higher quality, more sustainable software and increasing expectations for reproducible published results are two means to spur innovation without inherently creating a disadvantage for any particular team. All teams are faced with the same challenges to adapt their team skills, processes, practices, and tools to meet the increased demand for quality software and thus to realize the resulting improvements in trustworthy results [13].

4. CONCLUSION

The IDEAS-ECP project is partnering with the ECP and broader computational science community to increase software productivity and sustainability, as a key aspect of improving overall scientific productivity while tackling new challenges in simulation and analysis on extreme-scale computing platforms. Our community is on the frontier of discovering, synthesizing, and adopting new approaches to improving scientific discovery; we are succeeding, step by step, by transforming our own efforts and working toward long-term changes in the culture of high-performance scientific computing. This document, *Advancing Scientific Productivity through Better Scientific Software: Developer Productivity and Software Sustainability Report*, and subsequent versions will provide a periodic summary of advances, plans, and challenges as the Exascale Computing Project proceeds.

ACKNOWLEDGMENTS

This work was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations—the Office of Science and the National Nuclear Security Administration—responsible for the planning and preparation of a capable exascale ecosystem—including software, applications, hardware, advanced system engineering, and early testbed platforms—to support the nation’s exascale computing imperative.

We gratefully acknowledge the vision and support of Thomas Ndousse-Fetter (ASCR), Paul Bayer (BER), and David Lesmes (BER), who served during 2014–2017 as program managers of the IDEAS-Classic project.

A. REFERENCES

- [1] ACM PUBLICATIONS BOARD, *Artifact review and badging*. Association for Computing Machinery, April 2018, <https://www.acm.org/publications/policies/artifact-review-badging>.
- [2] H. ANZT, *The art of writing scientific software in an academic environment*. BSSw blog article, February 11, 2019, https://bssw.io/blog_posts/the-art-of-writing-scientific-software-in-an-academic-environment.
- [3] M. R. BENIOFF AND E. D. LAZOWSKI, PITAC CO-CHAIRS, *Computational science: Ensuring America's competitiveness: President's Information Technology Advisory Committee*. https://www.nitrd.gov/pitac/reports/20050609_computational/computational.pdf, 2005.
- [4] D. BERNHOLDT, A. DUBEY, C. JONES, D. S. KATZ, L. C. MCINNES, AND J. WILLENBRING, *Talking about software development at CSE19*. BSSw blog article, May 29, 2019, https://bssw.io/blog_posts/talking-about-software-development-at-siam-cse19.
- [5] D. E. BERNHOLDT, *Think locally, act globally: Outreach for better scientific software*. BSSw blog article, July 18, 2018, https://bssw.io/blog_posts/think-locally-act-globally-outreach-for-better-scientific-software.
- [6] D. E. BERNHOLDT, A. DUBEY, M. HEROUX, AND J. O'NEAL, *SC19 Tutorial: Better Scientific Software*. <https://doi.org/10.6084/m9.figshare.10114880>, 2019.
- [7] B. CAREY, *Many psychology findings not as strong as claimed, study says*. New York Times, August 27, 2015, <https://www.nytimes.com/2015/08/28/science/many-social-science-findings-not-as-strong-as-claimed-study-says.html>.
- [8] A. DUBEY AND J. O'NEAL, *FLASH5 Refactoring and PSIP*. BSSw blog article, August 27, 2019, https://bssw.io/blog_posts/flash5-refactoring-and-psip.
- [9] W. GROPP, R. HARRISON, ET AL., *Future directions for NSF advanced computing infrastructure to support U.S. science and engineering in 2017-2020*. National Academies Press, 2016. <http://www.nap.edu/catalog/21886/>.
- [10] M. HEROUX, E. BAVIER, D. BERNHOLDT, R. GUPTA, J. HEMSTAD, D. KATZ, S. KNEPPER, E. RAYBOURN, AND J. WILLENBRING (ORGANIZERS), *2019 Collegeville Workshop on Sustainable Scientific Software*. <https://collegeville.github.io/CW3S19>.
- [11] M. A. HEROUX, *Research software science: A scientific approach to understanding and improving how we develop and use software for research*. BSSw blog article, September 25, 2019, https://bssw.io/blog_posts/research-software-science-a-scientific-approach-to-understanding-and-improving-how-we-develop-and-use-software-for-research.
- [12] M. A. HEROUX, *Editorial: ACM TOMS replicated computational results initiative*, ACM Trans. Math. Softw., 41 (2015).
- [13] ———, *Trust Me. QED.*, SIAM News, 52 (2019). <https://sinews.siam.org/Details-Page/trust-me-qed>.
- [14] M. A. HEROUX, G. ALLEN, ET AL., *Computational Science and Engineering Software Sustainability and Productivity (CSESSP) Challenges Workshop Report*, September 2016. available via <https://www.nitrd.gov/PUBS/CSESSPWorkshopReport.pdf>.
- [15] M. A. HEROUX, E. GONSIOROWSKI, R. GUPTA, R. MILEWICZ, J. D. MOULTON, G. R. WATSON, J. WILLENBRING, R. J. ZAMORA, AND E. M. RAYBOURN, *Lightweight software process improvement using Productivity and Sustainability Improvement Planning (PSIP)*, 2019. 2019 International Workshop on Software Engineering for HPC-Enabled Research, held in conjunction with SC19, <https://www.osti.gov/biblio/1574620>.

- [16] S. HETTRICK, *Research software sustainability*. Report on a Knowledge Exchange Workshop, 2016. http://repository.jisc.ac.uk/6332/1/Research_Software_Sustainability_Report_on_KE_Workshop_Feb_2016_FINAL.pdf.
- [17] H. JOHANSEN, L. C. MCINNES, D. BERNHOLDT, J. CARVER, M. HEROUX, R. HORNING, P. JONES, B. LUCAS, A. SIEGEL, AND T. NDOUSSE-FETTER, *Software productivity for extreme-scale science*, 2014. Report on DOE Workshop, January 13-14, 2014, <https://science.osti.gov/-/media/ascr/pdf/research/cs/Exascale-Workshop/SoftwareProductivityWorkshopReport2014.pdf>.
- [18] D. S. KATZ, L. C. MCINNES, D. E. BERNHOLDT, A. C. MAYES, N. P. CHUE HONG, J. DUCKLES, S. GESING, M. A. HEROUX, S. HETTRICK, R. C. JIMENEZ, M. PIERCE, B. WEAVER, AND N. WILKINS-DIEHR, *Community organizations: Changing the culture in which research software is developed and sustained*, special issue of IEEE Computing in Science and Engineering (CiSE) on *Accelerating Scientific Discovery with Reusable Software*, 21 (2019), pp. 8–24. <https://dx.doi.org/10.1109/MCSE.2018.2883051>.
- [19] D. KEYES, V. TAYLOR, ET AL., *National Science Foundation Advisory Committee on CyberInfrastructure, Task Force on Software for Science and Engineering, final report*, 2011. http://www.nsf.gov/cise/aci/taskforces/TaskForceReport_Software.pdf.
- [20] D. E. KEYES, L. C. MCINNES, C. WOODWARD, W. GROPP, E. MYRA, M. PERNICE, J. BELL, J. BROWN, A. CLO, J. CONNORS, E. CONSTANTINESCU, D. ESTEP, K. EVANS, C. FARHAT, A. HAKIM, G. HAMMOND, G. HANSEN, J. HILL, T. ISAAC, X. JIAO, K. JORDAN, D. KAUSHIK, E. KAXIRAS, A. KONIGES, K. LEE, A. LOTT, Q. LU, J. MAGERLEIN, R. MAXWELL, M. MCCOURT, M. MEHL, R. PAWLOWSKI, A. P. RANGLES, D. REYNOLDS, B. RIVIÈRE, U. RÜDE, T. SCHEIBE, J. SHADID, B. SHEEHAN, M. SHEPHARD, A. SIEGEL, B. SMITH, X. TANG, C. WILSON, AND B. WOHLMUTH, *Multiphysics simulations: Challenges and opportunities*, International Journal of High Performance Computing Applications, 27 (2013), pp. 4–83. Special issue, see <https://dx.doi.org/10.1177/1094342012468181>.
- [21] S. LI, *SuperLU: How advances in software practices are increasing sustainability and collaboration*. BSSw blog article, April 30, 2018, https://bssw.io/blog_posts/superlu-how-advances-in-software-practices-are-increasing-sustainability-and-collaboration.
- [22] P. LUSZCZEK AND U. M. YANG, *Building community through software policies*. BSSw blog article, August 12, 2019, https://bssw.io/blog_posts/building-community-through-software-policies.
- [23] O. MARQUES, *Producing a webinar series*, 2019. Process for delivering webinars: <https://bssw.io/items/producing-a-webinar-series>.
- [24] O. A. MARQUES, D. E. BERNHOLDT, E. M. RAYBOURN, A. D. BARKER, AND R. J. HARTMAN-BAKER, *The HPC Best Practices Webinar Series*. <https://figshare.com/s/01feff6c31d391886a90>, 2018. Fifth SC Workshop on Best Practices for HPC Training and Education (BPHTE18), SC18.
- [25] L. C. MCINNES, D. S. KATZ, AND S. LATHROP, *Computational research software: Challenges and community organizations working for culture change*. SIAM News, December 2019, <https://sinews.siam.org/Details-Page/computational-research-software-challenges-and-community-organizations-working-for-culture-change>.
- [26] P. MESSINA, *Update on ASCAC subcommittee documenting ASCR impacts*. Presentation at Advanced Scientific Computing Advisory Committee Meeting, December 2018, https://science.osti.gov/-/media/ascr/ascac/pdf/meetings/201812/Subcommittee_on_ASCR_Impacts_ASCAC_201812.pdf.
- [27] M. C. MILLER AND H. AUTEN, *Continuous technology refreshment: An introduction using recent tech refresh experiences on VisIt*. BSSw blog article, April 12, 2019, https://bssw.io/blog_posts/continuous-technology-refreshment-an-introduction-using-recent-tech-refresh-experiences-on-visit.

- [28] D. MOULTON AND E. M. RAYBOURN, *Enhancing productivity and innovation in ECP with a team of teams approach*, 2018. Breakout session at the 2018 ECP Annual Meeting, <https://doi.org/10.6084/m9.figshare.6151097>.
- [29] *The Networking and Information Technology Research and Development (NITRD) Program*. <https://www.nitrd.gov>.
- [30] OFFICE OF SCIENCE, U.S. DEPARTMENT OF ENERGY, *The Top Ten Exascale Research Challenges*, 2014. <https://dx.doi.org/10.2172/1222713>.
- [31] J. O'NEAL, K. WEIDE, AND A. DUBEY, *Experience report: refactoring the mesh interface in FLASH, a multiphysics software*, in 2018 IEEE 14th International Conference on e-Science, 2018.
- [32] *Productivity and Sustainability Improvement Planning (PSIP)*, 2020. <https://bssw.io/psip>.
- [33] E. M. RAYBOURN AND D. MOULTON, *Scaling small teams to a team of teams: Shared consciousness*. BSSw blog article, April 17, 2018, https://bssw.io/blog_posts/scaling-small-teams-to-a-team-of-teams-shared-consciousness.
- [34] E. M. RAYBOURN, J. D. MOULTON, AND A. HUNGERFORD, *Scaling productivity and innovation on the path to exascale with a "team of teams" approach*, 2019. in: F. F.-H. Nah and K. Siau (Eds.): HCI in Business, Government and Organizations. Information Systems and Analytics. HCII 2019. Lecture Notes in Computer Science, vol 11589, https://doi.org/10.1007/978-3-030-22338-0_33.
- [35] U. RÜEDE, K. WILLCOX, L. MCINNES, AND H. D. STERCK, *Research and education in computational science and engineering*, SIAM Review, 60 (2018), pp. 707–754. <https://dx.doi.org/10.1137/16M1096840>.
- [36] SC19 REPRODUCIBILITY COMMITTEE, *Reproducibility initiative*. SC19, <https://sc19.supercomputing.org/submit/reproducibility-initiative/>.
- [37] *SDK-Tools repository*, 2020. <https://github.com/betterscientificsoftware/SDK-Tools>.
- [38] A. G. SMART, *The war over supercooled water: How a hidden coding error fueled a seven-year dispute between two of condensed matter's top theorists*. Physics Today, August 22, 2018, <https://physicstoday.scitation.org/doi/10.1063/PT.6.1.20180822a/full/>.
- [39] *Software Carpentry*, 2020. <http://software-carpentry.org>.
- [40] *Software Sustainability Institute*, 2020. <https://www.software.ac.uk>.
- [41] *Trust-Tools repository*, 2020. <https://github.com/betterscientificsoftware/Trust-Tools>.
- [42] *UK Research Software Engineering Association*, 2020. <https://rse.ac.uk>.
- [43] *United States Research Software Sustainability Institute*, 2020. <http://urssi.us>.
- [44] *US Research Software Engineering Association*, 2020. <https://us-rse.org>.
- [45] E. WENGER, *Communities of Practice: Learning, Meaning, and Identity*, Cambridge University Press, 1998. ISBN 978-0-521-66363-2.
- [46] R. ZAMORA, *Adopting continuous integration for long timescale materials simulation*. BSSw blog article, September 25, 2018, https://bssw.io/blog_posts/adopting-continuous-integration-for-long-timescale-materials-simulation.

B. IDEAS PROJECTS

B.1 IDEAS-ECP team

The IDEAS-ECP project brings together experts throughout the DOE community who have both experience in producing widely used scientific software products *and* passion for work on software productivity and sustainability issues in collaboration with the ECP and broader computational science communities. Moreover, team members have the depth and breadth of expertise needed to communicate effectively with domain scientists, so that collectively we are working to understand the software capabilities and needs of ECP teams, to consider opportunities for improvement, and to pursue targeted work toward better practices.

Mike Heroux (SNL) and Lois Curfman McInnes (ANL) co-lead the IDEAS-ECP project, extending their effective partnership in co-leading the IDEAS-Classic project. Co-PIs David Bernholdt (ORNL), Anshu Dubey (ANL), Elsa Gonsiorowski (LLNL), Osni Marques (LBNL), David Moulton (LANL), Boyana Norris (University of Oregon), and Elaine Raybourn (SNL) serve as institutional leads, with responsibility for coordinating institutional contributions to the project. The project includes leaders at computing facilities relevant for ECP: Rebecca Hartman-Baker (NERSC), Judy Hill (OLCF), Hai Ah Nam (LANL), Katherine Riley (ALCF), and Jean Shuler (LLNL). As liaisons between IDEAS-ECP and computing facilities, they help identify opportunities for and facilitate partnerships between facilities staff and the IDEAS-ECP team on topics of mutual interest.

David Bernholdt leads all outreach efforts for the project; Anshu Dubey leads strategic partnership with URSSI [43]. Rinku Gupta (ANL) is editor in chief of the Better Scientific Software (BSSw) website; Hai Ah Nam is coordinator of the BSSw Fellowship Program; Osni Marques coordinates the HPC Best Practices webinar series; Elaine Raybourn leads work on Productivity and Sustainability Improvement Planning; Jean Shuler (LLNL) coordinates our partnership with Software Carpentry; and Jim Willenbring (SNL) leads work on methodologies for Software Development Kits. All project members work synergistically in the development of methodologies materials, partnering with ECP teams on adoption of software best practices, and outreach.

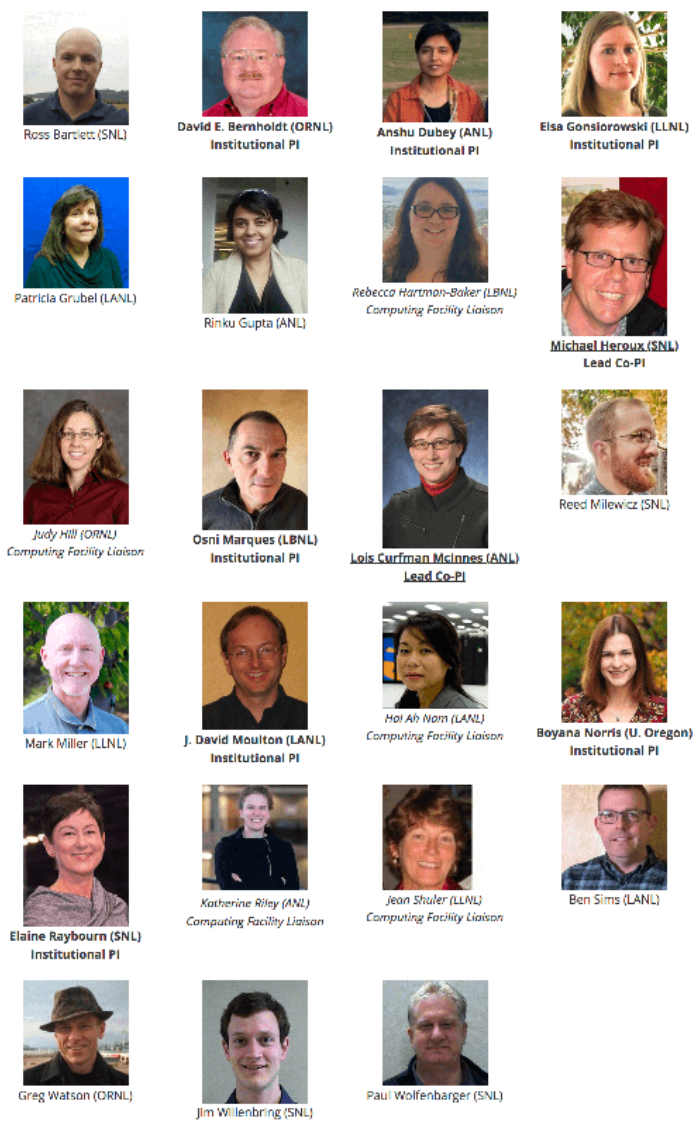


Figure 18: Members of the IDEAS-ECP team, January 2020; see <https://ideas-productivity.org/ideas-ecp/team>. Also, alumni of the project have contributed to work discussed in this report: Satish Balay, Lisa Childers, Todd Gamblin, Steve Hudson, Christoph Junghans, Alicia Klimvex, Jared O’Neal, Barry Smith, and Louis Vernon.

B.2 IDEAS-Watersheds: Advancing a software ecosystem for watershed modeling

Simulations of watershed systems. Water resources that are critically important for energy production, human use, agriculture, and ecosystem health are under increasing pressure from growing demand, land-use change, and Earth system change. Those stresses on the water supply are transmitted largely through watersheds. Through its Science Focus Area (SFA) projects, the DOE’s Subsurface and Biogeochemical Research (SBR) program is tightly integrating observations, experiments, and modeling to advance a systems-level understanding of how watersheds function and to translate that understanding into advanced science-based models of watershed systems.

Science-driven advances in software practices.

Significant progress was made during the first phase of the project (IDEAS-Classic), in which watershed use cases balanced advances in software engineering and scientifically driven modeling. To broaden SFA impact, the IDEAS-Watersheds project,⁴ established in spring 2019, builds on these successes to improve watershed modeling capacity through an agile approach to creating a sustainable, reliable, parallel software ecosystem with interoperable components. The project’s structure addresses many challenges in software development with distributed interdisciplinary teams. Organized around six research activities (see Figure 19) that develop use cases to drive advances in the software ecosystem, the project integrates improvements in software practices for aggregate teams with the model and algorithmic development needed to address scientific challenges. The project is exploring a co-funding model in order to create a team of early-career researchers who will be trained in modern software engineering, while acting as liaisons who contribute to shared deliverables.

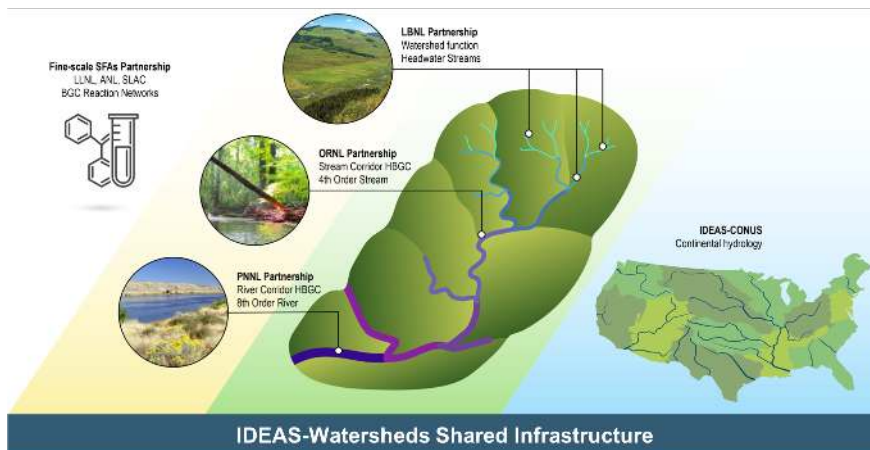


Figure 19: IDEAS-Watersheds comprises six research activities. This schematic shows the relationship of these activities with the three SBR cornerstones. The Shared Infrastructure activity works on common needs with shared solutions and moves capability across the scales associated with each cornerstone.

The project’s structure addresses many challenges in software development with distributed interdisciplinary teams. Organized around six research activities (see Figure 19) that develop use cases to drive advances in the software ecosystem, the project integrates improvements in software practices for aggregate teams with the model and algorithmic development needed to address scientific challenges. The project is exploring a co-funding model in order to create a team of early-career researchers who will be trained in modern software engineering, while acting as liaisons who contribute to shared deliverables.

Software ecosystem as a model for collaboration. Foundational advances begun during the IDEAS-Classic project, such as completing the open-source release of all major codes in this ecosystem, and broad adoption of agile methodologies have led the community to embrace the software ecosystem (see Figure 20) as an effective collaboration model. Furthermore, the team initiated a pilot study of the important role of a *code maintainer* to increase both the sustainability of the code and the health of the community it supports. The team reinstated a software engineer as the code maintainer of ParFlow, and he took on the difficult challenge of repairing the neglected tests; migrating multiple, largely disconnected branches from distant research groups; and implementing a git workflow with pull requests and code review. Despite initial skepticism of the team, a transformation of the community as well as the code resulted, with increases in both software and scientific productivity.

Changes in project structure to promote multidisciplinary collaboration and training. The successes in watershed modeling in the IDEAS-Classic project, along with many technical and cultural challenges, led to the development of the expanded and restructured IDEAS-Watersheds project. First, the project expanded to include all six of the SBR SFAs and uses a human-centered design approach to develop balanced and

⁴<https://ideas-productivity.org/ideas-watersheds>

impactful use cases. In addition, several team members felt their time was too fractured during the first phase and that early-career staff needed more training and integration. The IDEAS-Watersheds project thus developed a co-funding model that creates a team of fully funded early-career staff, dubbed the Integrated Computational and Domain Sciences Team. The project casts its relationship with the SFAs as a partnership, with joint funding of early-career staff to avoid time fragmentation while promoting their key role as liaisons. This approach also provides an ideal venue for training in software engineering and best practices, as well as modeling and numerical methods.

Increasing transparency and collaboration. The impact of the IDEAS synergistic family of projects on the watershed modeling community has been profound. The community is embracing the software ecosystem concept and the collaboration model that goes with it. Overall, a significant increase in the transparency of the SFA projects has occurred, along with an increasing willingness to collaborate and share capability development across the SFA projects through the software ecosystem.

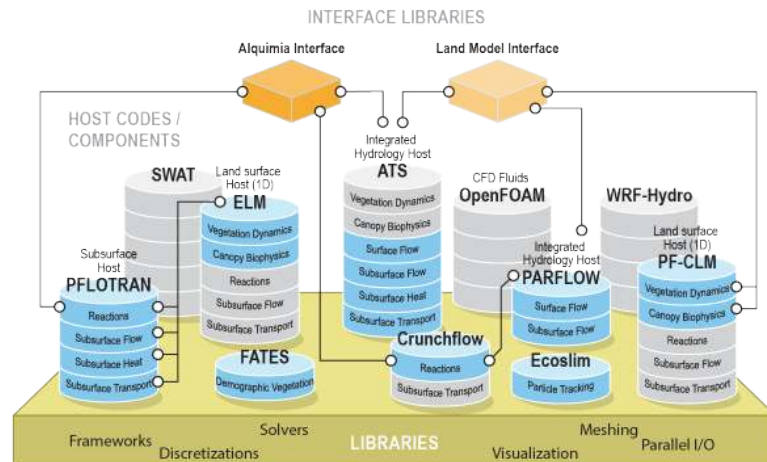


Figure 20: IDEAS-Watersheds software ecosystem: The project is advancing a scientific software ecosystem comprising BER application codes that can increasingly share components (blue disks) either directly or through common interface libraries (orange boxes). Less mature components and those using simplified representations are shown in gray. The xSDK currently includes numerical libraries, the Alquimia biogeochemistry interface library, and PFLOTRAN (subsurface processes, biogeochemistry engine). The land model interface library is targeted for development and later inclusion [figure courtesy of PNNL].

The project will continue to adopt new and improved practices emerging from the IDEAS-ECP project and the broader community. For example, the team explored the use of PSIP in its early stages of development but had trouble engaging teams with the process. Now that the PSIP process has matured, the project members are excited about using it to further improve software development practices, as well as to improve the efficiency with which the team completes some capability development. In addition, the project members are excited about training a cohort of early-career scientists (postdocs and staff) not only to appreciate the critical need for best practices in software engineering and design, but also to be able to apply these best practices in interdisciplinary aggregate teams.

C. ADVANCES BY SCIENCE TEAMS THROUGH PRODUCTIVITY AND SUSTAINABILITY IMPROVEMENT PLANNING

As introduced in Section 3.3, the PSIP process provides a lightweight workflow for iteratively and incrementally improving software practices. This section presents three case studies where application and software teams have advanced practices using the PSIP process, leading to improvements in overall effectiveness: building and testing (EXAALT, Appendix C.1), code refactoring (ExaStar, Appendix C.2), and onboarding (Exascale MPI, Appendix C.3)

C.1 Advances in building and testing: A case study with EXAALT

Combining software components for materials modeling.

The Exascale Atomistics for Accuracy, Length and Time (EXAALT) project⁵ in ECP is developing a materials modeling framework that leverages extreme-scale parallelism to produce accelerated molecular dynamics simulations for fusion and fission energy materials challenges. The official team comprises approximately 10 researchers at Los Alamos National Laboratory and Sandia National Laboratories working on four subprojects. While two of these subprojects are driven by a handful of people, the others are larger open source efforts with many external contributors. The EXAALT team members were keen to partner with the IDEAS-ECP team because they recognized the potential to improve their software engineering practices, particularly in the area of continuous integration (CI) testing. Since the project integrates several distinct software packages (see Figure 21), each with its own list of dependencies, the team frequently struggled with build regressions in the early days of development.

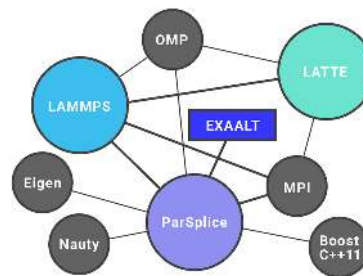


Figure 21: Illustration of the EXAALT framework. The three main software components (LAMMPS, LATTE, and ParSplice) are represented as colored circles, while other libraries are represented as grey circles. Lines (graph edges) depict dependencies between the various software components.

Identifying most urgent needs for improving software practices: builds and testing. After a few informal discussions with IDEAS-ECP members, the team agreed on the need to (1) improve their end-to-end build system, (2) implement a CI pipeline to automatically detect build regressions, and (3) add unit/regression testing to the CI pipeline. Although the team had not committed to an explicit project-management process at the early stages of the collaboration, the steps taken during these discussions correspond to the first two steps of the PSIP cycle shown in Figure 22. In order to prioritize their efforts, it was critical to clarify the current project practices and specify both near- and long-term goals.

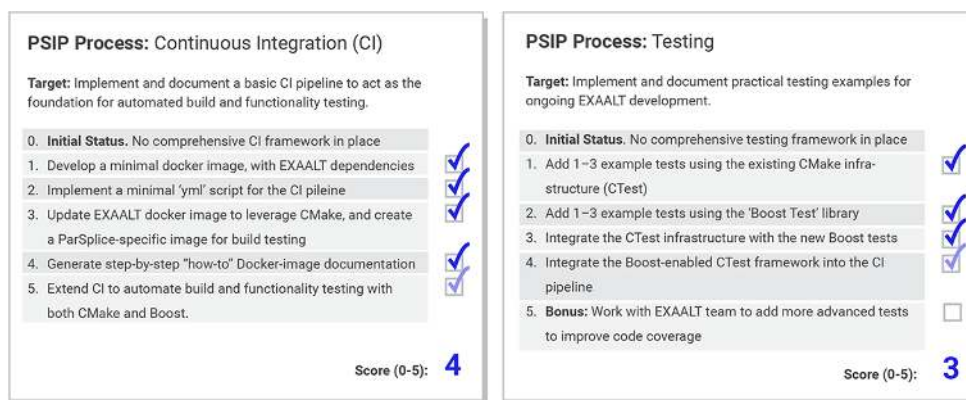


Figure 22: Summarized versions of PSIP project tracking cards used for the EXAALT-IDEAS collaboration. The specific scores in the figure correspond to the state of the project. Note that some details about dependencies and timeline are excluded from the PSIP cards for clarity.

Making progress in incremental steps. For the initial stage of the implementation of an automated end-to-end build system, the PSIP process was used only implicitly for project planning and execution. For the two remaining goals, however, PSIP was followed explicitly using the project tracking cards (PTCs) shown in Figure 22 (in summarized form). During steps 3-4 of the PSIP cycle, these PTCs were both fully annotated but reflected a “score” of zero. For step 5 of the PSIP cycle, each PTC step was resolved in both Jira and GitLab as distinct stories and issues, respectively. The actual implementation of these Jira/GitLab issues

⁵ <https://exascaleproject.org/research-group/chemistry-and-materials>

corresponded to step 6 of the PSIP cycle, and the assessment of the completed work was the final step. A BSSw blog article by Richard Zamora discusses more details about this work [46].

Next steps. The completion of these cards does not mean that the EXAALT team members are finished improving their CI and/or testing infrastructure. Like most aspects of software engineering, PSIP is an iterative process, and the initial plan may need to change if unexpected roadblocks emerge. Whether or not a progress tracking card can be followed to completion, documenting, revising, and repeating the process make sense when a natural finishing point is reached. The PTC used in this effort (see Figure 22) is available in the PSIP PTC catalog.

C.2 Advances in code refactoring: A case study with ExaStar

Refactoring to prepare for heterogeneous computer architectures.

FLASH—a large and complex multiphysics, multiscale code for simulating plasma physics and astrophysics (see Figure 23)—has been in public release since 2000 and has undergone two major revisions to build an extensible and flexible infrastructural framework with the goal of achieving robustness and longevity. The onset of platform heterogeneity requires another major refactoring because several components of the infrastructure are inadequate to meet the challenges posed by these platforms. The code restructuring, begun two years ago, is being carried out under the ECP’s AMReX Co-design Center,⁶ with the intent of enabling exascale simulations to be done under the ExaStar application project,⁷ which models stellar explosions.

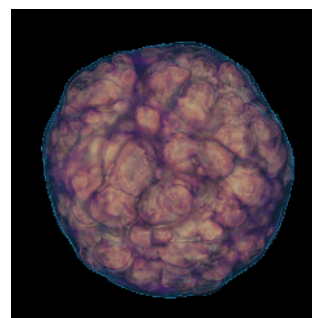


Figure 23: Simulation of a core-collapse supernova using the FLASH application.

The primary objective of the refactoring is to modify the interfaces of the so-called Grid code unit so that (1) looping over domain sections (blocks) can be done using smart iterators and (2) an alternative adaptive mesh refinement library, AMReX, can be used with the code. The smart iterators enable out-of-order execution of blocks, and therefore asynchronization, so that a block can be handed to an operator as soon as its dependencies are met. AMReX was adopted because it supports hierarchical parallelism and asynchronous operations, which the old adaptive mesh refinement (AMR) library, Paramesh, does not. Details of this refactoring effort are described in [31].

Using the PSIP process to advance practices in testing, verification, and revision control. The PSIP process aligns well with that of FLASH: take stock of where the project is, and make changes in small, well-planned, and manageable steps. Using PSIP retroactively mapped the approach for improving two development processes to improve productivity and the quality of the work.

The first improvement addressed the need to grow the test suite and improve techniques for documenting how the test suite evolves in response to changes in the software. This work was retroactively represented by a PTC for verification coverage and test-suite management, shown in the left-hand side of Figure 24. The work was linked to the effort to refactor the mesh management component of FLASH to work with AMReX, so that perceived barriers could be addressed.

The refactoring strategy relied on two team members carrying out simultaneous, incremental refactoring efforts with similar goals. One person added in AMReX from the bottom up, while the other person undertook a top-down refactoring. In the latter approach, the data structures for storing solution data were constructed with AMReX; but the original library, Paramesh, was used to drive the mesh refinement. The first task in the PTC indicated that the FLASH test suite had enough tests in place to verify the top-down modifications through simulation-level regression testing but was inadequate for the bottom-up part. Test-driven development was therefore used to design and implement integration-level regression testing, where

⁶<https://exascaleproject.org/research-group/co-design-centers>

⁷<https://exascaleproject.org/research-group/earth-and-space-science>

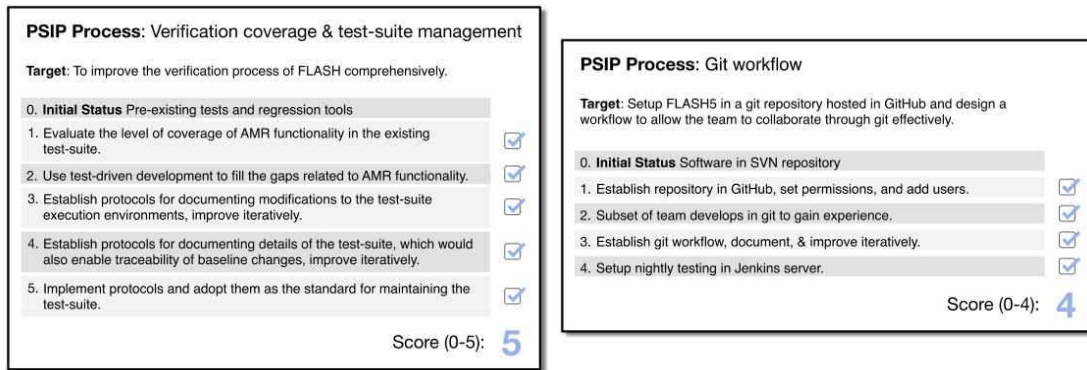


Figure 24: Summarized versions of PSIP project tracking cards used for the ExaStar project.

each new test covered a single aspect of the internal AMR functionality.

In addition to these changes to the test suite, the team improved the documentation of the setup of each execution environment used to run the test suite, including maintaining a history of which third-party libraries were installed, when, and why, and adding a procedure for documenting how a baseline was verified both when created and when updated.

The second improvement related to designing and evolving a test-driven git workflow in order to improve collaboration when team members integrate work developed in parallel through a revision control system. Rather than adopt a full-featured and possibly excessive workflow, the team started simple and added capabilities as needed. This incremental process is shown in the right-hand side of Figure 24. A BSSw blog article by Anshu Dubey and Jared O’Neal provides further information [8].

Toward further improvements in software practices. To date, the code in the git repository is a small subset of the production version of FLASH. Although this relatively simple workflow has worked well so far, the team members have identified an area where they would like improvement and can explicitly apply PSIP, namely, to smoke test changes made to the software in the repository with a continuous integration test server such as Travis CI. (Smoke testing is preliminary testing to reveal simple failures.) As more code components from the production version are transitioned to git and more users and contributors switch to the git version, the team will be faced with many more challenges and the need for process improvements. While the FLASH development philosophy has always mirrored PSIP, the formalization brought by PSIP makes the philosophy explicit to new team members and external contributors. The team foresees many instances of PSIP being used before the new version of FLASH is ready for production.

C.3 Advances in onboarding: A case study with Exascale MPI

Challenges in technical onboarding training for new project members. The Exascale MPI (Message Passing Interface) project⁸ in ECP is developing a production-ready, portable, high-performance MPI implementation (MPICH) that scales to the largest supercomputers in the world. One challenge for the project, based at Argonne National Laboratory, is a continuous influx of new contributors, who are expected to already have technical expertise with MPI or learn these skills on the fly as needed by the job. While the team does provide mentoring to new members, limited resources require that newcomers be fairly independent and proactive when it comes to learning the basic technical aspects of MPI.

⁸<https://exascaleproject.org/research-group/programming-models-runtimes>

Using the PSIP process to define a strategy for creating onboarding materials. The Exascale MPI team worked with an IDEAS-ECP facilitator to implement the PSIP process. Through the documentation of current practices in the project, the need to improve the project’s onboarding was identified. The Exascale MPI team and the facilitator agreed to work on a PSIP cycle focused on improving the training for contributors, by creating a single destination training resource for new team members to use during the onboarding phase. Four key aspects of a satisfactory solution were identified:

1. A central “repository” for all training material, relevant to the Exascale MPI team
2. Visually interesting, easy navigation across all topics
3. Easy administration and ability to update the “repository” sustainably
4. Open collaboration to allow external contributors to contribute new technical topics and resources

With the overall goal and desired outcome defined, the Exascale MPI team worked with the facilitator to create a timeline and PTC, shown in Figure 25. Each step in the PTC represents an important advance to move toward the desired goal. Yet, PTC cards are live entities, which may change depending on unexpected progress or bottlenecks.

Following the project’s self-defined PSIP steps to create onboarding resources. Once the PTC was created, the team focused on the execution aspect of the PTC. The PSIP process aims to engage a full team through the execution of PTC steps. Each step is approachable yet builds toward a larger goal. Throughout the PSIP process, the Exascale MPI team, with help from the facilitator, continually evaluated progress on the path to building a resource for improving onboarding and training.

PSIP Process: Onboarding		
Target: Implement a technical onboarding process to facilitate integration of new team members		
0	Initial Status: No training process in place.	
1	Understand MPICH current onboarding training practices and define training categories	<input checked="" type="checkbox"/>
2	Review and gather resources for training in at least two categories	<input checked="" type="checkbox"/>
3	Design website and integrate content for two categories	<input checked="" type="checkbox"/>
4	Solicit feedback, improve content, design processes for external contributions and updating of website	<input type="checkbox"/>
		Score (0-4) = 3

Figure 25: Summarized version of PSIP project tracking card used for the MPICH project.

For improving the training process, the team identified what categories of topics needed to be covered in the onboarding training. For each category, the team worked on and solicited resources (based on the potential expertise level of new onboarding members), reviewed the material for accuracy and applicability, and worked on the design to integrate them into the training website. The Exascale MPI team, with help from the facilitator, explored the viability of using cloud repository services (e.g., Google Drive). In the end, the team decided to design a custom stand-alone website to serve as a training portal, based on project needs and input from the team members. The training-base portal is a continual work in progress and can be a resource for the broader HPC community. At this stage, the Exascale MPI team is testing the training portal with new hires and soliciting feedback. The next step is to determine a plan to improve the training portal, which will focus on adding new content categories and establishing processes to sustain the content and its validity.

Broader applicability of onboarding resources. With the PSIP process, the Exascale MPI team learned that a PSIP topic developed for a particular team may sometimes be generalizable enough to be relevant and important to several teams in an organization or across multiple organizations. The topic of technical onboarding training for new hires may sometimes be team specific; however, most teams working in the HPC field need to train people in the common practices of the community. Thus, a PTC created for one team can be used by many other teams as a starting point. Moreover, the resulting output can end up being immensely useful across many other teams as well. The PTC used in this effort (see Figure 25) is available in the PSIP PTC catalog.

D. OUTREACH DETAILS

As discussed in Section 3.5, multipronged outreach efforts are critical aspects of IDEAS training and community engagement. This appendix provides details about webinars in the series *Best Practices for HPC Software Developers* (see Section 3.5.3) and BSSw.io blog posts (see Section 3.5.1).

D.1 HPC Best Practices Webinar Series

A major training initiative is the *HPC Best Practices Webinar Series*, a collaboration with IDEAS-ECP and the three ASCR computing facilities (ALCF, NERSC, and OLCF). Table 1 lists webinar topics, including hyperlinks to slides, recordings, and Q&A materials, while Table 2 provides information about registrations and attendees.

Table 1: Webinars of series *Best Practices for HPC Software Developers* (May 2016 – December 2019). The sessions during 2016 were offered under the IDEAS-Classic project.

Date	Title, Presenter(s), and Affiliation
2019-12-11	Building Community through xSDK Software Policies , Ulrike Meier Yang (LLNL) and Piotr Luszczek (Univ of Tennessee)
2019-10-16	Tools and Techniques for Floating-Point Analysis , Ignacio Laguna (LLNL)
2019-09-11	Discovering and Addressing Social Challenges in the Evolution of Scientific Software Projects , Rene Gassmoeller (UC Davis)
2019-08-14	Software Management Plans in Research Projects , Shoaib Ahmed Sufi (Software Sustainability Institute)
2019-07-17	When 100 Flops/Watt was a Giant Leap: The Apollo Guidance Computer Hardware, Software and Application in Moon Missions , Mark Miller (LLNL)
2019-06-12	Modern C++ for High-Performance Computing , Andrew Lumsdaine (PNNL & Univ of Washington)
2019-05-08	So You Want to be Agile? Strategies for Introducing Agility into Your Scientific Software Project , Mike Heroux (SNL)
2019-04-10	Testing Fortran Software with pFUnit , Thomas Clune (NASA Goddard)
2019-03-13	Parallel I/O with HDF5 - Overview, Tuning and New Features , Quincey Koziol (NERSC)
2019-02-13	Containers in HPC , Shane Canon (NERSC)
2019-01-23	Quantitatively Assessing Performance Portability with Roofline , John Pennycook (Intel), Charlene Yang and Jack Deslippe (NERSC)
2018-12-05	Introduction to Software Licensing , David Bernholdt (ORNL)
2018-10-17	Open Source Best Practices: From Continuous Integration to Static Linters , Daniel Smith and Ben Pritchard (Molecular Sciences Software Institute)
2018-09-19	Modern CMake , Bill Hoffman (Kitware)
2018-08-21	Software Sustainability: Lessons Learned from Different Disciplines , Neil Chue Hong (Software Sustainability Institute)
2018-07-18	How Open Source Software Supports the Largest Computers on the Planet , Ian Lee (LLNL)
2018-06-13	Popper: Creating Reproducible Computational and Data Science Experimentation Pipelines , Ivo Jimenez (UCSC)
2018-05-09	On-demand Learning for Better Scientific Software: How to Use Resources and Technology to Optimize your Productivity , Elaine Raybourn (SNL)
2018-04-18	Software Citation Today and Tomorrow , Daniel Katz (NCSA and UIUC)
2018-03-28	Scientific Software Development with Eclipse , Greg Watson (ORNL)
2018-02-28	Jupyter and HPC: Current State and Future Roadmap , Matthias Bussonnier (UC Berkeley), Suhas Somnath (ORNL) and Shreyas Cholia (NERSC)
2018-01-17	Bringing Best Practices to a Long-Lived Production Code , Charles Ferenbaugh (LANL)
2017-12-06	Better Scientific Software (https://bssw.io): So your code will see the future , Mike Heroux (SNL) and Lois Curfman McInnes (ANL)
2017-11-01	Managing Defects in HPC Software Development , Tom Evans (ORNL)
2017-09-13	Barely Sufficient Project Management: A few techniques for improving your scientific software development efforts , Mike Heroux (SNL)
2017-08-16	Using the Roofline Model and Intel Advisor , Sam Williams and Tuomas Koskela (LBNL)
2017-07-12	Intermediate Git , Roscoe Bartlett (SNL)
2017-06-07	Python in HPC , Rollin Thomas (NERSC), William Scullin (ALCF) and Matt Belhorn (OLCF)

Continued on next page

Table 1 – Continued from previous page

Date	Title, Presenter(s), and Affiliation
2016-08-09	<i>Basic Performance Analysis and Optimization - An Ant Farm Approach</i> , Jack Deslippe (NERSC)
2016-07-28	<i>An Introduction to High-Performance Parallel I/O</i> , Feiyi Wang (OLCF)
2016-07-14	<i>How the HPC Environment is Different from the Desktop (and Why)</i> , Katherine Riley (ALCF)
2016-06-15	<i>Testing and Documenting your Code</i> , Alicia Klinvex (SNL)
2016-06-02	<i>Distributed Version Control and Continuous Integration Testing</i> , Jeff Johnson (LBNL)
2016-05-18	<i>Developing, Configuring, Building, and Deploying HPC Software</i> , Barry Smith (ANL)
2016-05-04	<i>What All Codes Should Do: Overview of Best Practices in HPC Software Development</i> , Anshu Dubey (ANL)

Table 2: Registrations and attendees in webinars of the series *Best Practices for HPC Software Developers* (June 2017 – December 2019). Many additional people view archived slides, recordings, and curated Q&A’s.

	Registrations	ECP-Affiliated Registrations	Attendees
Minimum	69	22	26
Average	148	46	78
Maximum	245	71	182
Std. deviation	47	14	36
Total	4154	1281	2181

D.2 Blog articles published on BSSw.io

As introduced in Section 3.5.1, the BSSw site (<https://bssw.io>) is a central hub for sharing information on practices, techniques, experiences, and tools to improve developer productivity and software. The site features a growing collection of original **blog articles**, addressing topics such as science teams’ experiences with productivity-related software issues and strategies for collaborative computational science. Blog posts thus far are listed in Table 3.

Table 3: Blog articles published on the BSSw site (from launch in 2017 through January 2020).

Date	Title, Author(s), and Affiliation
2020-01-15	<i>US Research Software Engineer (US-RSE) Association</i> , Ian Cosden (Princeton Univ), Chris Hill (MIT), Sandra Gesing (Univ of Notre Dame), and Charles Ferenbaugh (LANL)
2020-01-02	<i>Better Scientific Software: 2019 Highlights</i> , Rinku Gupta (ANL)
2019-12-13	<i>Introducing the 2020 BSSw Fellows</i> , Hai Ah Nam (LANL)
2019-12-05	<i>Hello CSE World</i> , Heather M. Switzer (College of William and Mary), Elsa Gonsiorowski (LLNL), and Mark C. Miller (LLNL)
2019-11-25	<i>Give Thanks!</i> , Angela Herring (LANL)
2019-11-14	<i>Software Sustainability in the Molecular Sciences</i> , Theresa L. Windus (Iowa State Univ and Ames Lab) and T. Daniel Crawford (Virginia Tech)
2019-10-31	<i>Bloodsuckers, Banshees and Brains: A Bestiary of Scary Software Projects and How to Banish Them</i> , Neil Chue Hong (Software Sustainability Institute) and Benjamin Cowan (Tech-X Corporation)
2019-10-15	<i>Accepting High-Quality Software Contributions as Scientific Publications</i> , Hartwig Anzt (Karlsruhe Institute of Technology)
2019-09-25	<i>Research Software Science: A Scientific Approach to Understanding and Improving How We Develop and Use Software for Research</i> , Michael A. Heroux (SNL)
2019-09-18	<i>Data-driven Software Sustainability</i> , Daniel S. Katz (UIUC)
2019-09-12	<i>Making Open Source Research Software Visible: A Path to Better Sustainability?</i> , Neil Chue Hong (Software Sustainability Institute)
2019-09-04	<i>Applications Open for 2020 BSSw Fellowship Program. Q&A Webinar on Sept 20, 2019</i> , Hai Ah Nam (LANL)
2019-08-27	<i>FLASH5 Refactoring and PSIP</i> , Anshu Dubey and Jared O’Neal (ANL)

Continued on next page

Table 3 – Continued from previous page

Date	Title, Author(s), and Affiliation
2019-08-12	<i>Building Community through Software Policies</i> , Piotr Luszczek (Univ of Tennessee) and Ulrike Yang (LLNL)
2019-07-29	<i>When NOT to Write Automated Tests</i> , Roscoe A. Bartlett (SNL)
2019-07-15	<i>Celebrating Apollo's 50th Anniversary: Users' Stories from Space</i> , Mark C. Miller (LLNL)
2019-06-27	<i>Leading a Scientific Software Project: It's All Personal</i> , Wolfgang Bangerth (Colorado State University)
2019-06-17	<i>Celebrating Apollo's 50th Anniversary: The Oldest Code on GitHub</i> , Mark C. Miller (LLNL)
2019-05-29	<i>Talking about Software Development at SIAM CSE19</i> , David E. Bernholdt (ORNL), Anshu Dubey (ANL), Michael A. Heroux (SNL), Catherine Jones (Science and Technology Facilities Council), Daniel S. Katz (UIUC), Lois Curfman McInnes (ANL), and James Willenbring (SNL)
2019-05-15	<i>Celebrating Apollo's 50th Anniversary: When 100 FLOPS/Watt Was a Giant Leap</i> , Mark C. Miller (LLNL)
2019-04-26	<i>Streamlining Software Development through Continuous Integration</i> , Glenn Hammond (SNL)
2019-04-12	<i>Continuous Technology Refreshment: An Introduction Using Recent Tech Refresh Experiences on VisIt</i> , Mark C. Miller and Holly Auten (LLNL)
2019-03-28	<i>2018 BSSw Fellows Tackle Scientific Productivity Challenges</i> , Hai Ah Nam (LANL)
2019-03-19	<i>Accelerating Scientific Discovery with Reusable Software: Special issue of IEEE CiSE</i> , Scott Lathrop (NCSA)
2019-02-25	<i>Software As Craft</i> , Paul Wolfenbarger (SNL)
2019-02-21	<i>The Art of Writing Scientific Software in an Academic Environment</i> , Hartwig Anzt (Karlsruhe Institute of Technology)
2019-01-29	<i>Preparing the Next Generation of Supercomputer Users</i> , Marta García Martínez (ANL)
2019-01-04	<i>Better Scientific Software: 2018 Highlights</i> , Lois Curfman McInnes (ANL)
2018-12-11	<i>Introducing the 2019 BSSw Fellows</i> , David E. Bernholdt (ORNL), Michael A. Heroux (SNL), and Lois Curfman McInnes (ANL)
2018-11-28	<i>Porting Codes to New Architectures</i> , Bronson Messer (ORNL)
2018-11-08	<i>SC18: Does That Stand for "Software Conference"?</i> , David E. Bernholdt (ORNL)
2018-10-26	<i>Building Connections and Community within an Institution</i> , Gregory Watson (ORNL) and Elsa Gonsiorowski (LLNL)
2018-09-25	<i>Adopting Continuous Integration for Long-Timescale Materials Simulation</i> , Richard Zamora (ANL)
2018-09-10	<i>Applications Open for 2019 BSSw Fellowship Program. Q&A Webinar on Sept 21, 2018</i> , David E. Bernholdt (ORNL), Michael A. Heroux (SNL), and Lois Curfman McInnes (ANL)
2018-08-30	<i>Do Social Media and Science Mix? Twitter Use in a Large Research Project</i> , Tim Scheibe (PNNL)
2018-08-15	<i>Software Verification</i> , Anshu Dubey (ANL)
2018-07-30	<i>URSSI: Conceptualizing a US Research Software Sustainability Institute</i> , Daniel S. Katz (UIUC), Jeff Carver (Univ of Alabama), Sandra Gesing (Univ of Notre Dame), Karthik Ram (Berkeley Institute for Data Science), and Nic Weber (Univ of Washington)
2018-07-17	<i>Think Locally, Act Globally: Outreach for Better Scientific Software</i> , David E. Bernholdt (ORNL)
2018-06-28	<i>Building Trusted Scientific Software</i> , Michael A. Heroux (SNL)
2018-06-14	<i>Research Software Engineer: A New Career Track?</i> , Chris Richardson (Univ of Cambridge)
2018-05-31	<i>On Demand Learning for Better Scientific Software: How to Use Resources & Technology to Optimize Your Productivity</i> , Elaine Raybourn (SNL)
2018-05-17	<i>Keeping Your Vision Fit for Years of Software Development</i> , Mark C. Miller (LLNL)
2018-04-30	<i>SuperLU: How Advances in Software Practices are Increasing Sustainability and Collaboration</i> , Sherry Li (LBNL)
2018-04-17	<i>Scaling Small Teams to a Team of Teams: Shared Consciousness</i> , Elaine Raybourn (SNL) and David Moulton (LANL)
2018-03-26	<i>Can You Teach an Old Code New Tricks?</i> , Charles Ferenbaugh (LANL)
2018-03-13	<i>BSSw Fellowship Activity: Promoting Software Citation</i> , Daniel S. Katz (UIUC)
2018-02-26	<i>Call for Papers: Accelerating Scientific Discovery with Reusable Software</i> , Scott Lathrop (NCSA)
2018-02-05	<i>Introducing the 2018 BSSw Fellows</i> , David E. Bernholdt (ORNL), Michael A. Heroux (SNL), and Lois Curfman McInnes (ANL)
2018-02-02	<i>Better Science through Software Testing</i> , Tom Evans (ORNL)
2017-12-13	<i>New FAQ List for BSSw Fellowship Program. Applications Due by Jan 5, 2018</i> , Michael A. Heroux (SNL) and Lois Curfman McInnes (ANL)
2017-12-01	<i>Applications Open for New BSSw Fellowship Program. Q&A Webinar on Dec 12, 2017</i> , Michael A. Heroux (SNL) and Lois Curfman McInnes (ANL)
2017-11-13	<i>BSSw Site Launch at SC17. Contribute to Better Scientific Software!</i> , David E. Bernholdt (ORNL), Michael A. Heroux (SNL), and Lois Curfman McInnes (ANL)