Los Alamos National Laboratory (LANL)
Response to Stewardship of Software for Scientific and High-Performance Computing RFI
LA-UR-21-32162

(1) Software dependencies and requirements for scientific application development and/or research in computer science and applied mathematics relevant to DOE's mission priorities:

What software packages and standardized languages or Application Programming Interfaces (APIs) are current or likely future dependencies for your relevant research and development activities? What key capabilities are provided by these software packages? What key capabilities, which are not already present, do you anticipate requiring within the foreseeable future? Over what timeframe can you anticipate these requirements with high confidence? What are the most-significant foreseeable risks associated with these dependencies and what are your preferred mitigation strategies? When responding to these questions, please describe the scope of the relevant research and development activities motivating the response.

LANL conducts research, development, and long-term maintenance on multi-physics codes each of which can approach a million lines of source code and contain tens to a hundred second and third party dependencies. Currently there is little ability to rely upon stable software application binary interfaces (ABIs) so our codes are routinely rebuilt on various platforms as dependencies are updated or software APIs change. This results in extremely complex build systems and significant infrastructure to manage dependencies as source across open and secure computing environments. Further research and development is needed in dependency management, particularly in methods that do not rely upon a brute force approach of carrying dependencies as source with no ability to rely upon stable ABIs.

Below is a list of third party software /APIs used in LANL codes along with an estimate of the priority.

Software Package	Purpose	Importance at LANL
MPI	Distributed memory message passing	High
OpenMP	Shared memory parallelism (including GPU)	Moderate
HDF5	I/O	Moderate
Parallel NetCDF	I/0	Moderate / Low
Kokkos	Shared memory parallelism (including GPU)	Moderate
Trillinos & Hypre	Solvers	High/Moderate
Legion	Distributed and shared memory programming system	Moderate

ParMETIS	Mesh partitioning	Moderate
Paraview/VTK/VTKm	Visualization and analysis	High
Caliper	Performance analysis	Low
FleCSI	Computational Science Infrastructure	High

There are a number of gaps in capabilities provided by software packages used at LANL. There are no standardized tools available for data management and workflow management that are scalable, performant, and tractable for a diverse set of codes and applications to easily adopt. Current tooling in this area is generally monolithic which can inhibit incremental adoption and limit needed customization. Performance portability tools are currently reliant on template metaprogramming techniques (in C++) which can add significant complexity to applications and software developers ability to effectively debug large scale applications. Modern programming systems (such as Legion) are still evolving and will require further research, development, and maturation towards stable/standard APIs. Machine learning software can be ill-suited to HPC environments and Scientific computing needs, further research in ML software is required to close this gap. General computational science infrastructure (FleCSI, MFEM,..) requires further research, development, and maturation to realize their potential of supporting a broad set of scientific computing requirements.

(3) Infrastructure requirements for software development for scientific and high-performance computing:

What infrastructure requirements do you have in order to productively develop state-of-the-art software for scientific and high-performance computing? These requirements might include access to testbed hardware, testing allocations on larger-scale resources, hosting for source-code repositories, documentation, and other collaboration tools. What are the key capabilities provided by this infrastructure that enables it to meet your needs? What key capabilities, which are not already present, do you anticipate requiring within the foreseeable future? Over what timeframe can you anticipate these requirements with high confidence? What are the most-significant foreseeable risks associated with this infrastructure and what are your preferred mitigation strategies? When responding to these questions, please describe the scope of the relevant research and development activities motivating the response.

LANL has transitioned most of our codes to modern CI infrastructure (gitlab) and modern software development practices. Current gaps include the ability to test software commits on production hardware resources both on-prem and across the Trilab and broader DOE facilities. The Trilab has embarked on a remote compute enablement project to address this (and other limitations) across LANL, LLNL, and Sandia. LANL also participates in current ECP efforts to reduce these barriers across DOE compute facilities. A significant challenge is the potential security implications of cross-realm authentication. In many ways this is not a technology challenge, rather it is an artifact of important (and sometimes differing) security requirements at each of the laboratories. Addressing this gap is critical however as a major barrier to deployment of software across facilities is the additional burden it places on software developers to maintain a large number of credentials and deal with a litany of platform and site specific idiosyncrasies.

(4) Developing and maintaining community software:

How much additional effort is needed to develop and maintain software packages for use by the wider community above the effort needed to develop and maintain software packages solely for use in specific research projects or for internal use? What tasks are the largest contributors to that additional effort? What are the largest non-monetary impediments to performing this additional work? How is any such additional effort currently funded? How does that funding compare to a level of funding needed to maximize impact?

The additional effort required to develop and maintain software packages for a wider community is largely a function of the breadth of the software packages capabilities, the stability of the APIs that present those capabilities to users, and the relative breadth of the internal user community that already uses the software package. For more mature software the additional costs are often related to user engagement (tech support, documentation, training, etc.) and potential feature requests. For less mature software the additional costs can be significant (at least in the near term) as a broader user community will exercise more of the software and can be negatively impacted by unstable APIs resulting in slowing of feature development. Non-monetary impediments often include the need for infrastructure to manage user engagement / communication, infrastructure to test on a more diverse set of platforms that external users require and impediments to debugging user issues such as lack of access to their platform (related to 3). The level of funding required for maintaining a broader external user community is project specific.

(5) Challenges in building a diverse workforce and maintaining an inclusive professional environment:

What challenges do you face in recruiting and retaining talented professionals to develop software for scientific and high-performance computing? What additional challenges exist in recruiting and retaining talented professionals from groups historically underrepresented in STEM and/or individuals from underserved communities? What challenges exist in maintaining inclusivity and equity [8] in the development community for scientific and high-performance-computing software? What successful strategies have you employed to help overcome these challenges? What opportunities for professional recognition and career advancement exist for those engaged in developing scientific and high-performance-computing software?

LANL believes diversity fuels our innovative, agile, and principled workforce that is essential to solving problems of global importance across the entire breadth of our science and engineering. To address challenges in building a diverse workforce and maintaining an inclusive professional environment will require a multi-faceted, continuously evolving approach. Recent efforts at LANL include building partnerships with historically minority serving institutions to establish a future pipeline of a more diverse candidate pool and raising the visibility of opportunities at LANL. At the early stages of the pipeline we have partnered with high schools to help supplement their STEM curriculum with targeted programs that include lecturing, mentoring, and projects led by LANL staff. Addressing broader challenges in the STEM pipeline remains a gap however and could benefit from larger-scale national programs that build upon the national laboratories' long standing experience in educational engagement at the primary, undergraduate, and graduate levels.

In general, the increasing importance of software in scientific research, and the increasing complexity of many research-oriented codes, has created significant new demand for expertise in scientific software development across the DOE complex as well as at academic research institutions. This creates challenges as well as opportunities for recruiting and retaining a talented and diverse scientific computing workforce.

Scientific software development is becoming increasingly recognized as an area of expertise in its own right, and practitioners are seeking to define and solidify career paths in this area. One manifestation of this is the increasing prominence of the term "research software engineer" (RSE) as a professional category (Cohen et al. 2017). There is a US-RSE association (https://us-rse.org/) that appears to be growing rapidly and counts many DOE laboratory staff among its membership. However, there are many other categories of scientific software professionals within DOE laboratories who may not fit well into the RSE framework, including those who work primarily with the HPC software stack and infrastructure, those who focus on code optimization for hardware, and other specialized roles (Sims 2021).

Organizations like LANL have historically supported a significant scientific computing workforce in the context of large multiphysics code projects. In these projects, scientists with PhDs, in both domain sciences and computer science, often spend significant portions of their careers working on software development and infrastructure. In addition to this scientist-developer approach, which has worked well for LANL, there may be a need to establish alternate career paths for research professionals who primarily identify as software engineers rather than scientists. This is particularly relevant for smaller-scale basic science research projects, where an RSE workforce might be a valuable addition to research teams where software is currently developed and maintained primarily by graduate students and postdocs. One consideration here is that some institutions provide research software support as an institutional resource, while others employ RSEs directly on research teams. LANL may face institutional choices about whether one or both of these strategies meets our needs. These choices are likely to be strongly tied to sponsors' funding models for software development.

If research software engineering becomes an increasingly relevant and in-demand area of expertise on scientific projects, LANL may need to develop additional recruiting strategies and job categories to support this element of the future workforce. A positive aspect of these potential workforce developments is that they may provide opportunities to rethink roles and recruiting pipelines in ways that could enhance inclusivity and equity in hiring.

(6) Requirements, barriers, and challenges to technology transfer, and building communities around software projects, including forming consortia and other non-profit organizations:
ASCR recognizes that successful software for scientific and high-performance computing often has many stakeholders, including academic research activities, research laboratories, and industry. Moreover, while DOE has provided funding for the development of a significant number of foundational software packages within the modern software ecosystem for scientific and high-performance computing, as the complexity of the software ecosystem continues to increase, and number of stakeholders has grown, ASCR seeks to understand how it might encourage sustainable, resilient, and diversified funding and development models for the already-successful software within the ecosystem. Such models include, depending on circumstances that ASCR seeks to better understand, both the private sector and non-profit organizations. Non-profit organizations include both charitable organizations (e.g., those with 501(c)(3) status) and R&D consortia (e.g., those with 501(c)(6) status). What are the important characteristics and components of sustainable models for software for scientific and high-performance computing? What are key obstacles, impediments, or bottlenecks to the establishment and success of these models? What development practices and other factors tend to facilitate successful establishment of these models?

Sustainability of software continues to pose challenges in scientific and high-performance computing. Vendor supported software, often incubated at the laboratories and over time further developed and maintained by private industry has proven successful in a number of cases, particularly when the software

is more broadly adopted and standardized such as is the case for MPI and OpenMP. Other technologies such as parallel file systems have seen mixed success with this model, having at times required significant investments from DOE to maintain viability. Other technologies such as HPSS have required long term sustained investments by DOE to maintain viability. In some cases non-profit organizations have helped steward important software technologies but in most cases required long-term sustained investment/contributions.

As the communities and technologies evolve the software ecosystem will need to balance evolution of standardized tools with adoption and inclusion of new core technologies. The computing hardware landscape and computational science techniques will both likely continue to innovate at a fast past. To harness these new innovations software ecosystems will need to pace these innovations. In some cases we have been able to realize broader adoption of core technologies such as CMake, and technologies which originated externally and have proven beneficial in HPC such as Docker containers. Success in long term sustainability could be improved through adoption and alignment with technologies developed in the broader cloud and ML ecosystem and more targeted and sustained investment in technologies that are necessarily differentiated from these broader markets. Partnering with other FFRDCs and federal agencies to support these differentiated technologies could reduce total costs and duplication of effort.

(7) Overall scope of the stewardship effort:
The section labeled Potential Scope, mentioned earlier in the RFI, outlines activities that ASCR currently anticipates potentially including in future programs stewarding the software ecosystem for scientific and high-performance computing. Are there activities that should be added to, or removed from, this list? Are there specific requirements that should be associated with any of these activities to ensure their success and maximize their impact?

The potential scope of this proposed effort could vary significantly based on the prioritization of software technologies that require stewardship and the duration and magnitude of investment required for these technologies. If we consider other large-scale efforts such as the Advanced Simulation and Computing Initiative (ASCI) or the Exascale Computing Project, the annual scope of tens of millions of dollars per year for software stewardship is not excessive. Careful consideration for what is essential, necessary, and optional can have major ramifications.

Beyond simply considering scope as a budgetary concern, there is an inherent tension between research, development and production. As more scientific software becomes “production” and garners external users, the level of rigor required to support external users can slow fundamental development. Historically, much of ASCR’s software remained research quality, and was often the artifact of research, not the intended end product. This has provided researchers the agility to address fundamental challenges in algorithms, performance, and scalability that have been critical to the success of high performance computing. Proper scoping and balance between research innovation and stewardship of existing technology will be important or we risk slowing the entire enterprise of advancing scientific computing.

(10) Other:
What are key obstacles, impediments, or bottlenecks to progress by, and success of, future development of software for scientific and high-performance computing? Are there other factors, issues, or opportunities, not addressed by the questions above, which should be considered in the context of stewardship of the ecosystem of software for scientific and high-performance computing?

One key consideration for future development of software for scientific and high-performance computing is the increasing need to support larger development teams, as well as coordination of code development across multiple teams and projects. This need to scale up software collaboration across the complex is a significant challenge, and there are still many questions about what approaches will be most helpful. The Exascale Computing Project (ECP) has made significant progress in this area by bringing together code developers from across the complex, and by pursuing coordinated development of widely used software libraries, including through scientific software development kits like xSDK (https://xsdk.info/). However, it is less clear how this increased scale of coordination can best be managed from a social and organizational perspective. ECP has included initial efforts to investigate and define best practices in these areas through its IDEAS (Interoperable Design of Extreme-scale Application Software) and PSIP (Productivity and Sustainability Improvement Planning) components. These are multi-laboratory efforts that LANL has been actively involved in.

One approach to scaling up team interactions that has gained particular traction in both the IDEAS project and at LANL is the concept of a team of teams (McChrystal et al. 2015; Raybourn, Moulton and Hungerford 2019; Moulton et al. 2021; Sims 2019). This concept, which draws on a wide range of organizational and social science research, emphasizes the importance of regular virtual forums where teams can communicate, having key individuals who serve as liaisons between teams, and maintaining an aligning narrative across teams. A team of teams also requires close attention to operating rhythm and delegating decisions to the appropriate level, so day-to-day decisions can be made autonomously at the team level, while larger strategic directions are coordinated at higher levels.

The PSIP component of ECP has conducted some initial studies of software repository activity across ECP components in order to better understand how these components work together and assess whether they function as a team of teams (Rogers et al. 2021, Raybourn et al. 2021). These studies indicate that there may be an emerging cadre of software developers that contribute across multiple projects and could potentially operate as liaisons between teams. More qualitatively, ECP has had some success in building an aligning narrative across teams, and meets regularly in virtual forums at various levels of coordination. Examining these issues from a team-of-teams perspective helps identify potential opportunities for further developing and enhancing best practices in cross-team coordination across the complex.

LANL will continue to collaborate with other institutions across the complex to build on this initial work by extending research collaborations and supporting efforts to develop best practices. Much of the work in this area has been pushed forward by a small group of social and computational scientists at LANL and other DOE laboratories. We anticipate developing additional collaborations, both within and outside the complex, to better define and address organizational and human aspects of scientific computing.

Comments containing references, studies, research, and other empirical data that are not widely published should include copies of the referenced materials. Note that comments will be made publicly available as submitted. Any information that may be confidential and exempt by law from public disclosure should be submitted as described below.

J. Cohen, D.S. Katz, M. Barker, N.C. Hong, R. Haines and C. Jay (2020). “The four pillars of research software engineering,” *IEEE Software,* vol. 38, no. 1, pp. 97-105.

S. McChrystal with T. Collins, D. Silverman, and C. Fussell (2015). Team of Teams: New Rules of Engagement for a Complex World. Portfolio/Penguin, New York, NY.

J.D. Moulton, E.M. Raybourn, B. Sims, and R. Milewicz (2021) “A Team of Teams Approach to Sustainable Software Ecosystems,” SIAM Conference on Computational Science and Engineering, March 2021.

E.M. Raybourn, R. Milewicz, D. Rogers, B. Sims, G. Watson, E. Gonsiorowski, and J. Willenbring (2021) “A Data-Driven Approach to Rethinking Open Source Software Organizations as a Team of Teams,” CSCE Conference on Software Engineering Research and Practice. (To be published in conference proceedings.)

E. M. Raybourn, J. D. Moulton, and A. Hungerford (2019). Scaling productivity and innovation on the path to exascale with a “team of teams” approach. In HCI in Business, Government and Organizations: Information Systems and Analytics, Lecture Notes in Computer Science 11589: 408-421.

D. Rogers, R. Milewicz, B. Sims, G. Watson, and E.M. Raybourn, “Querying the Exascale Compute Project – How do Cross-Team Collaborations Enable Interoperable Software?,” SIAM Conference on Computational Science and Engineering.

B. Sims (2021) “Research Software Engineer as an Emergent Professional Identity: A Sociological Perspective,” presented at the United States Research Software Engineer Association Virtual Workshop.

B. Sims (2019) “Enabling coordinated, distributed development of scientific software:
A research agenda for adapting a team of teams approach,” Los Alamos National Laboratory report LA-UR-19-31893.