Scalable Delivery of Scalable Libraries and Tools: How ECP Delivered a Software Ecosystem for Exascale and Beyond

Michael A. Heroux, *Sandia National Laboratories, Saint John's University, MN*

Abstract—The Exascale Computing Project (ECP) was one of the largest open-source scientific software development projects ever. It supported approximately 1,000 staff from US Department of Energy laboratories, and university and industry partners. About 250 staff contributed to 70 scientific libraries and tools to support applications on multiple exascale computing systems that were also under development. Funded as a formal construction project, ECP was required to use earned-value management, based on milestones, and a key performance parameter system based, in part, on integrations. With accelerated delivery schedules and significant project risk, we also emphasized software quality using community policies, automated testing, and continuous integration. Software Development Kit teams provided cross-team collaboration. and products were delivered via E4S, a curated portfolio of libraries and tools. In this paper, we discuss the organizational and management elements of ECP that enabled the delivery of libraries and tools, our lessons learned and our next steps.

The Exascale Computing Project (ECP) represents one of the largest open-source scientific software development projects to date [1][1]. ECP funded approximately 1,000 scientists from the US Department of Energy (DOE) laboratory complex, and DOE university and industry partners. The Software Technology Focus Area sponsored the efforts of about 250 people working on contributions to 70 open-source products. The result was a collection of reusable libraries and tools (see Figure 1) to support parallel applications and their portable execution on target platforms using GPU accelerators from three different computing system vendors. All of this work was done to support application codes that were still being designed and developed and for computer systems that were still being designed and built.

ECP as a Construction Project

ECP was funded as a formal 413.3b DOE construction project [2], using a tailored earned-value management (EVM) [3] system. There are many well-established software methodologies for large, multi-team projects. Among them are the Scaled Agile Framework (SAFe) [4] and LeSS [5]. ECP was required to use EVM, with some flexibility, to structure work activities. We were able to design, develop, and deliver our software contributions across a wide range of teams, products, and organizations. Use of EVM enabled us to effectively and efficiently manage the efforts of our 35 teams, and detect potential staffing issues, technical challenges and misaligned efforts in need of correction or offramping. Each team could proceed independently in its efforts, engaging collaborators as needed with annual global planning and quarterly reporting, permitting scalable but coordinated development and delivery of capabilities. These efforts were successful in the formal sense of passing the key performance parameters (KPPs) defined for the project. Even more important, the work received mention in the *CHIPS and Science Act* [6] *(page 175)* which specifically called out the need to sustain the ECP software ecosystem we developed.

In this paper, we describe the challenges of managing the Software Technology efforts in ECP and the organizational and management approaches that enabled the development and delivery of ECP libraries and tools to the open-source software community. We also discuss the lessons learned from this effort and some future directions.

ECP Challenges

ECP was a large, multi-institutional, multi-year project to design and develop new application, library, and tool

XXXX-XXX © 2024 IEEE
Digital Object Identifier 10.1109/XXX.0000.0000000
[1]ECP Website: https://exascaleproject.org

This article has been accepted for publication in Computing in Science & Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MCSE.2024.3384937

Key themes:
- Focus: GPU node architectures, advanced memory and storage technologies
- Explore: New high-concurrency, latency tolerant algorithms
- Develop: New portable (Nvidia, Intel, AMD GPUs) software products
- Enable: Access and use via standard APIs

Legacy: A stack that enables performance portable application development on accelerated platforms, from deskside to supercomputer

Software categories:
- **Next generation established products:** Widely used HPC products (e.g., MPICH, OpenMPI, PETSc)
- **Emergent products:** Address key new requirements (e.g., Kokkos, RAJA, Spack)
- **New products:** Enable exploration of emerging HPC requirements (e.g., zfp, Variorum)

Example Products	Engagement
MPI – Backbone of HPC apps	Explore and develop MPICH and OpenMPI new features and standards
OpenMP/OpenACC –On-node parallelism	Explore and develop new features and standards
GPU Portability Libraries: Kokkos, RAJA	Explore and develop lightweight APIs for compile-time polymorphism
LLVM/ and test suites	Inject HPC features and testing for eventual vendor adoption
Perf Tools - PAPI, TAU, HPCToolkit	Explore and develop portable access to vendor performance counters
Math Libraries: BLAS, sparse solvers, etc.	Provide scalable algorithms and software, critical enabling technologies
IO: HDF5, MPI-IO, ADIOS	Provide standard and next-gen IO, leveraging non-volatile storage
Viz/Data Analysis: VTK-m, SZ, zfp	Provide *in-situ* capabilities, data compression to address IO challenges

FIGURE 1. ECP Software Technology efforts provided contributions to 70 scientific libraries and tools products. Some products were well established and needed investments to provide portable and scalable execution on the target exascale systems. Other products were still emerging at the beginning of ECP but were essential by the end. Others were new at the start of ECP. Two of these, zfp, a data compression library, and Variorum, a portable power management tool, received R&D100 awards during the last year of ECP, a recognition of their future impact in high-performance computing.

capabilities on a new generation of systems designed to execute a billion-billion (10^{18} or exa) operations per second. The key exascale systems are Frontier at Oak Ridge National Laboratory, Aurora at Argonne National Laboratory, and El Capitan at Lawrence Livermore National Laboratory. Frontier and El Capitan use AMD GPUs, and Aurora uses Intel GPUs. When ECP started, system details were still evolving and Aurora was based on a different (non-GPU) processor. Even so, we knew these systems would require disruptive changes to scientific problem formulations, algorithms, and software design. Most of the performance from these systems would come from the GPUs, each capable of trillions (10^{12}) of operations per second. GPUs had been used in previous systems but not with the diversity of architecture or the scale of performance needed to reach the exascale threshold, nor with the requirement for performance portability across multiple kinds of GPUs. The challenge for the ECP leadership team was to provide a framework for successful software development in this environment.

Organizational Elements

One key element of ECP library and tools efforts was its organizational structure, designed to provide a framework for teams to collaborate and succeed in a challenging environment with flexibility to accommodate team and project needs. The structure had three layers: product teams, Software Development Kit (SDK) teams, and the Extreme-scale Scientific Software Stack (E4S) team (Figure 2). The majority of the work was done by product teams–more than 200 people. The SDK and E4S teams each engaged 5–10 people providing support and coordination. The ECP software technology leadership team had 10 people, responsible for managing the entire process.

Product Teams

The fundamental ECP organizational building block was the product team. Milestones and integrations were defined and tracked per product. Product teams were responsible for defining and executing their activities and integrations and managing their own internal processes. Product plans were reviewed by ECP leadership to ensure feasibility and alignment with ECP end-project objectives. Product teams have always been the fundamental building block of DOE software development efforts. However, within ECP their work was coordinated and supported by the SDK and E4S teams, and ECP leadership provided a framework for planning, executing, tracking, and assessing progress. The result was a structured, but flexible, approach to software product development.

This article has been accepted for publication in Computing in Science & Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MCSE.2024.3384937

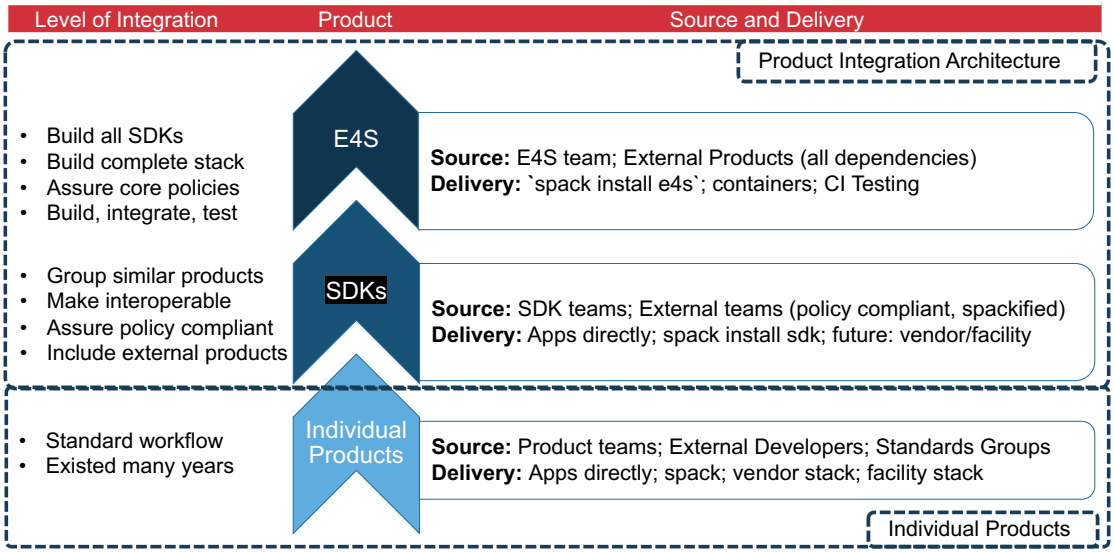

FIGURE 2. ECP software libraries and tools efforts were organized at three levels: individual products, Software Development Kits (SDKs), and the Extreme-scale Scientific Software Stack (E4S). The product teams were responsible for defining and executing their milestones and integrations using their own internal processes, as long as the teams achieved their milestones and integrations. SDK teams were responsible for coordinating activities among product teams. The E4S team was responsible for coordinating activities among SDK teams and curating the full collection of ECP libraries and tools. ECP leaders were responsible for managing the entire process.

SDK Teams

The next building block was the Software Development Kit (SDK) team. Each SDK team was a collection of product teams producing compatible and complementary products. The purpose of SDKs was to facilitate interaction among product teams in a variety of activities such as requirements gathering, design space exploration, training, and the evolution of software practices and tools. Coordination of versioning, vendor interactions, design space exploration, and software delivery at the SDK level helped amortize costs across related products and reduced complexity at the top software stack level. This level also spurred the dynamics of *co-opetition* [7] where teams collaborated on some activities and learned from each other during coordinated activities. Then, to make sure they kept pace with other SDK members, teams competed to be the best in other activities. This dynamic was very effective at accelerating progress and improving quality.

E4S Team

The final building block was the E4S[2] team, which focused on building, testing, delivering, and deploying the complete software stack. E4S (the product) is a curated build of more than 100 primary products and their dependencies (hundreds more products). Using Spack, E4S incorporates external, including proprietary, products. E4S can also be configured to create custom builds that target many environments—from laptops to clusters, leadership systems, and edge computing environments. E4S comes in containerized environments and has binary caches of previously built products to enable rapid rebuilds. The E4S team manages versioning across products for reproducibility, correctness, and security patching. Actively managing a portfolio of configurable HPC software products that are built and tested on many HPC platforms, including on the DOE exascale platforms Frontier, Aurora, and El Capitan, provides a rich resource of version-compatible products, built under numerous different parameter settings, and tested on many systems.

Other Delivery Mechanisms

While many ECP libraries and tools products were (and still are) delivered through the SDKs and E4S, most products were also delivered directly to clients as "second-party"[3] software that was co-developed with

[2]E4S website: https://e4s.io

[3]Second-party software is a distinct product developed in close collaboration with another product.

their application users. Finally, some of our work was delivered to community ecosystems such as LLVM[4], a major platform for compiler and runtime library features and to community efforts like the MPI and C++ Language standards.

Efficient and Effective Management

Trusted systems are often the result of addressing two complementary concerns: doing things right and doing the right things. The first concern is about quality, and the second is about value. For ECP efforts, we focused on milestones and integrations. Milestones incentivized and supported teams in planning and execution toward "doing things right" and integrations incentivized and measured efforts toward "doing the right things." In this section, we describe how we used milestones and integrations to plan, execute, track, assess, and adapt ECP software libraries and tools efforts. Figure 3 shows the relationship between milestones and integrations and how they were used to support the ECP software lifecycle.

Milestones and Metrics

An idealized project mental model has three main components: creating something (scope), over a certain timespan (schedule), for a certain amount of effort and resources (cost). As required by DOE policy, ECP used EVM. (However, we were permitted to use a tailored approach that allowed us to refine annual plans.) EVM breaks the entire project timeline, budget and scope into a collection of smaller chunks, each of which is called an *activity*. Each activity was defined for a particular product over a particular time (usually a few months) containing a description of what would be done (its scope), the amount of budget for the activity, and the expected beginning and end dates. On a large project like ECP, many activities can be executed concurrently, and the set of in-progress activities represents a wavefront in time, as current activities are completed and new activities are started. ECP's orchestrated and concurrent execution of many activities using EVM enabled the scalable delivery of many products by many teams in an orderly manner.

If all activities were perfectly predictable, EVM would result in a smooth progression of completing milestones (the successful end of an activity) resulting in an on-time delivery of all capabilities for the estimated cost. However, most projects are not so predictable, especially software R&D projects like ECP that are producing new capabilities to support new applications for new computers that themselves are being co-designed and developed along with the software. The ability to effectively monitor and adapt activities was what made EVM so valuable to ECP.

EVM has many project management and control concepts. The two concepts that were most useful for ECP product development were *cost performance index (CPI)* and *schedule performance index (SPI)*:

- **CPI:** Measures predicted costs vs. actual costs. A CPI of 1.0 means actual costs match predicted costs at a given point in time of the project. A CPI above (below) 1.0 means costs are less (more) than predicted.
- **SPI:** Measures actual progress vs. predicted progress in completing work (scope). An SPI of 1.0 means work is on schedule. An SPI above (below) 1.0 means work is ahead (behind) schedule.

In ECP, we used many EVM metrics but CPI and SPI were particularly helpful in getting an early indication of when a particular library or tool product team was struggling. For most projects, their CPI and SPI were very close to 1.0 throughout the project. This did not necessarily mean that ECP teams were perfect at estimating. For CPI, it typically meant that team staffing was stable and the team was spending money at a consistent rate to pay salaries. For SPI, it meant that features were being delivered on time. However, because features were not always described in great detail, the features delivered could vary and still satisfy what was planned. In particular, an elaborately planned feature could be trimmed and still be declared as completed.

While most teams consistently had CPI and SPI values near 1.0, some teams had CPI, SPI, or both, values significantly less than 1.0. Usually, these teams needed help to refine scope, increase funding, relax schedules, or make some other adjustment, to make it possible for them to succeed. The ECP leadership team was able to identify these concerns early and work with teams to make adjustments. In other instances, poor CPI and SPI values signaled deeper problems such as the loss of key staff or poor team morale, problems that were harder to address but important to track. Monthly reviews of CPI and SPI were key elements of the ECP project management approach. Tracking SPI and CPI was used constructively (not punitively) to help teams succeed and keep track of progress at all levels of the project.

The end objective of ECP for the scientific libraries and tools teams was very clear: Produce a software

[4]LLVM: https://llvm.org

This article has been accepted for publication in Computing in Science & Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MCSE.2024.3384937

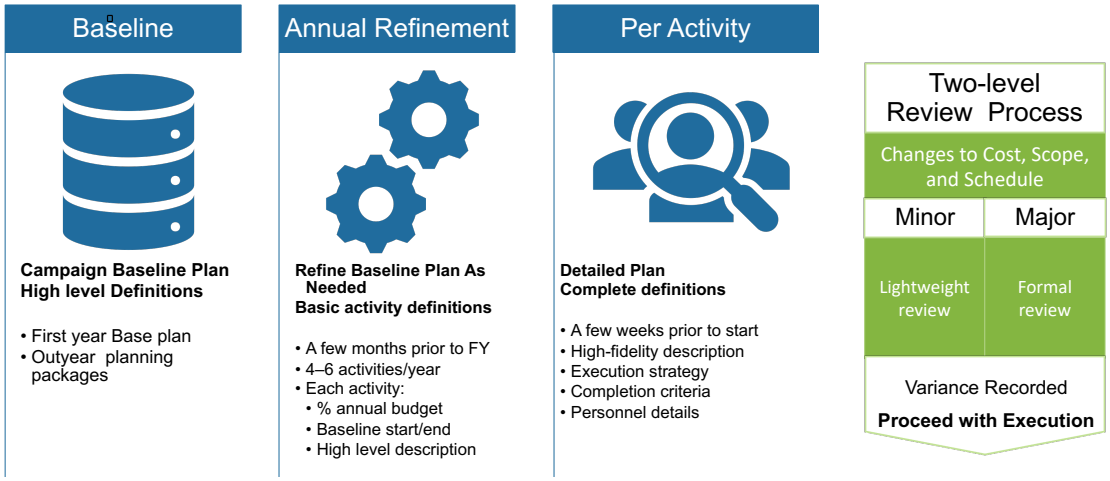

FIGURE 3. ECP library and tool development and delivery process: The above diagram shows the idealized workflow for ECP library and tool development and delivery. The process began with a high-level objective described in per-product, per-year planning packages (short paragraphs describing what will be done) to provide libraries and tools for ECP applications to use on exascale systems by the end of the project. Each year, planning packages were refined to describe specific activities as a collection of milestones. As capabilities were developed, they were integrated into the ecosystem and demonstrated in the client environment. The process was repeated until the project was completed.

stack that would support applications on the target exascale systems before the end of the project. However, the exact path to that objective was not known at the beginning of the project and was disrupted regularly as the project proceeded. We had changes in the target systems, application requirements, development team staffing, and external risk factors that arose or diminished as the project proceeded.

Because of the certainty of our objective and the uncertainty about our path toward that objective, the ECP activity (milestone) planning process contained end-goal planning objectives and coarse grain plan descriptions that were refined as the project proceeded. Each year, we would define approximately 300 activities and at any given time more than 100 activities would be in progress.

As shown in Figure 4, the process began with a multi-year baseline plan that provided coarse-grain descriptive paragraphs (called planning packages), one planning package per product per year, for each year of the project. The final objective for each product was a description of the expected final product capabilities for the exascale systems. A few months before the start of a new fiscal year, each product team would refine that year's planning package into 4 - 6 activities for the year with an estimate of the percent annual budget for each activity, a baseline start/end date and a high-level description. A few weeks before the start of an activity, the remaining details needed for staffing, completion criteria, etc., were added. Outyear planning packages could also be updated. The process was repeated until ECP was completed.

To manage changes in scope, schedule, or costs that were needed outside of the annual planning process, we had a two-level change management process. The key objective of the process was to make sure we clearly articulated proposed changes, reviewed changes with transparency (especially with sponsors and lab management), and then recorded our decisions for later reference. The process was not intended to prevent changes but to ensure that changes were agreed to by project stakeholders and well understood by everyone involved in the project to retain transparency and trust. We consistently advised product teams that they should always be doing the most important work and, if what was most important changed, then they needed to change their plans to support the change in priorities. We also used the process to expand the project scope for emerging needs such as new products or new capabilities that were needed to address gaps in our portfolio. Examples included additional investments in using low-precision arithmetic and new workflow capabilities.

Integrations

ECP had four key performance parameters (KPPs) which were metrics used to assess overall project success. Two KPPs were focused on applications, another on hardware, and one on software libraries and tools.

This article has been accepted for publication in Computing in Science & Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MCSE.2024.3384937

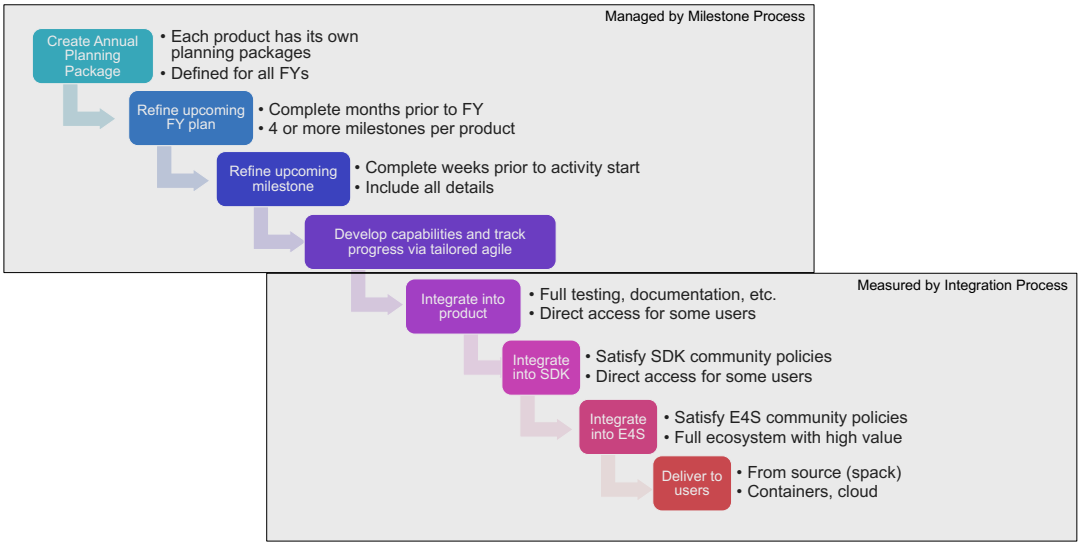

FIGURE 4. ECP libraries and tools activity/milestone planning process with annual refinement and change management: The above diagram shows the planning process for ECP library and tool development and delivery. The process started with a multi-year, coarse-grain plan for the entire project, for each product. A few months before the start of a new fiscal year, each product team would refine that year's planning package. A few weeks before the start of an activity, the remaining details were added. The process was repeated until the project was completed. For any mid-year changes, a change management process was used to ensure that the changes were well understood, agreed to by all stakeholders, and made visible to all staff.

The software KPP, called KPP-3 and sketched in Figure 5, focused on demonstrating that a new library or tool feature was used and could be used in the future, for example, a new GPU solver in PETSc [8] was used by an application and the solver was available in the latest PETSc release. The KPP-3 definition was intended to be accurate at a high level and flexible, to ensure that the software developed by ECP teams was integrated into client environments in a meaningful and sustainable manner while allowing for a variety of integration approaches.

Activities and milestones were how ECP addressed the "doing things right" goal. Integrations were how ECP addressed the "doing the right things" goal. Integrations were the process of taking the features developed by the teams and sustainably integrating them into client environments.

The primary emphasis of KPP-3 was to ensure a "capable and sustainable software ecosystem," a long-term view, where software capabilities are not just developed but are maintained and usable over time. The integration goal prioritized real-world applicability and long-term use, starting with pre-exascale environments and then transitioning to exascale environments. The integration goals were product-focused and success in integration was quantified by counting the number of "integrations," the number of successful capability integrations of a product into client environments providing a concrete metric to evaluate doing the right things.

Annual Lifecycle

Using the organizational elements of milestones and integrations, ECP implemented annual planning, executing, tracking, reporting, assessing, and adapting lifecycle. The lifecycle centered on the product teams and their milestones and integrations.

- **Planning:** The lifecycle began with the activities planning process. Each product team refined its activities and resulting milestones annually, adding final details just before starting the execution of an activity. The SDK and E4S teams did the same. The planning process was reviewed by the ECP project leadership team to help ensure rigorous and realistic plans that were consistent across products and in line with the long-term ECP objective of a capable and sustainable ecosystem.
- **Executing:** Execution of activities toward milestone completion progressed during the year. Teams generally worked independently, engaging with their clients as needed. The SDK and E4S teams engaged with product teams to support their efforts and to coordinate activities among teams.
- **Tracking:** Progress was tracked using Jira and

This article has been accepted for publication in Computing in Science & Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MCSE.2024.3384937

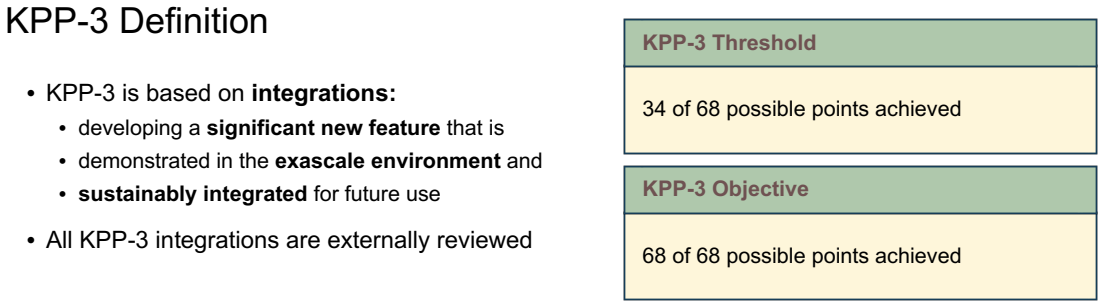

FIGURE 5. The definition of KPP-3 was centered on the concept of a *sustainable integration of significant capabilities*, called an *integration*. Each product team had to demonstrate four (or in some cases eight) integrations during the lifetime of the project. Integrations were reviewed by external subject-matter experts (SMEs). Artifacts such as screenshots of output and client letters were used as evidence of integrations and SME reports were used to determine if the KPP-3 requirements were satisfied. Final approval was given by the ECP Federal Project Director, who had formal approval authority. KPP-3 threshold is the value (34) required to pass the KPP. The objective (68) is the value that would be required to achieve the highest possible score. The objective was set to be challenging but achievable.

Confluence, two widely used, web-based project management platforms. ECP used custom Jira issue types that aligned with and could automatically synchronize with a tool called Primavera[5]. Monthly reporting of CPI and SPI helped identify potential problems early.

- **Reporting:** Concurrent with the assessment process was the production of a Capability Assessment Report (CAR) [9]. The CAR provided a detailed assessment of the ECP Software Technology efforts including a description of activities, progress, and plans for each product. The CAR was used to inform the annual project reviews and identify gaps in the project. It also provided stakeholders and the scientific community with project updates.
- **Assessing:** Annual project reviews were conducted with project leadership and external subject-matter experts (SMEs). The SMEs reviewed the progress of the project and provided feedback to the project leadership team. The project leadership team used the SME feedback to make adjustments to the project.
- **Adapt and Repeat:** The annual review process led to numerous project changes over the years of ECP, including restructuring teams, off-ramping products, and addressing product gaps using contingency funds[6]. The lifecycle was repeated annually until the project was completed.

Complementing Product Efforts

The ECP approach to the development and delivery of capabilities through a milestone and integration process was designed to flexibly complement the software engineering practices of individual product teams. While product teams were expected to address ECP community policies[7] for software quality, user support,

[5]Use of Primavera–a powerful project management platform–was a requirement of our ECP sponsors but development teams never had to work with it directly because we mirrored and synced content between Primavera and Jira. This approach was an organizational innovation that permitted most staff to use the more familiar Jira platform

[6]By monitoring milestone and integration progress, the ECP leadership team was able to see when products were not progressing (as evidenced by repeatedly missing milestones), or had little probability of making an impact, as evidenced by a lack of pre-exascale integrations, or both. In some cases, corrections in planning and strategy led to improvements. In other cases, we created an exit plan to stop funding the project. Similarly, sometimes we identified gaps in our portfolio and initiated new activities for new products to address the gaps.

[7]E4S Policies: https://e4s-project.github.io/policies.html

and integration, the ECP leadership team did not direct product teams to use specific software engineering practices, tools, or processes beyond the use of Jira and Confluence for ECP-specific efforts.

ECP was a project with a fixed timeline. Most product teams existed before ECP and continued their efforts after ECP finished, and many received concurrent funding from non-ECP sources for the development of other features not within the scope of ECP. By requesting only high-level, coarse-grain information from project teams, we minimized duplicate record-keeping. ECP's use of milestones and integrations addressing the specific funding scope of ECP could easily integrate with the finer-grain workflows of a product team. Also, because we used the web-based tool Jira for ECP issue tracking, cross-referencing ECP issues with other systems, e.g., GitHub Issues, was easily done through web links.

Lessons Learned

ECP was an ambitious project that was successful in many ways and yet still had significant room for improvement. It was successful in the formal sense of passing the key performance parameters (KPPs) defined for the project. We learned many lessons that will be useful for future projects. The following are some of the most important:

1) **Organizing independently-developed products into a coherent software ecosystem provides value:** ECP delivered its milestones, integrations, and a full stack of portable GPU libraries and tools to support ECP applications on exascale systems. Success is evident from the explicit support for sustaining this ecosystem after ECP ended.
2) **Organizing a research software project using a tailored EVM structure can work:** Using EVM with a two-level refinement planning process enabled keeping the long-term project objective (a capable software stack for exascale systems) in focus and permitted adaptation due to uncertainty and unforeseen changes in external factors. The EVM structure also provided a mechanism for identifying teams that were struggling and needed help.
3) **The use of milestones and integrations resulted in efficient and effective efforts:** Milestones supported doing things right. Integrations supported doing the right things.
4) **ECP vocabulary and workflows were difficult for many staff to understand and use:** ECP used terms and workflows that were used heavily by project leadership but only occasionally by most staff. This was a significant barrier to entry for new staff and a source of persistent challenge for existing staff. We did introduce the role of project coordinator, bringing in project management specialists, but we think more improvements would be needed in a future project.
5) **Heavyweight verification processes were expensive to support:** ECP was a large and visible project with very specific reporting requirements. Most requirements were handled by ECP leadership but KPP content required individual technical staff members to produce evidence and formal reports for subject matter experts (SMEs). This process was necessary for assuring our federal project director that we were meeting our KPP-3 requirements but was costly, requiring the capture of detailed runtime artifacts and producing reports with particular formatting and content. In the future, we will look for ways to reduce the overhead of producing evidence and formal reports.

Next Steps

The Exascale Computing Project finished its development and delivery of libraries and tools in December 2023. However, the end of ECP is the beginning of the "Exascale Era." The following are some of the next objectives for the software ecosystem ECP started:

- **Achieving 100X performance and efficiency:** ECP demonstrated 100X or more increase in performance and energy efficiency [10]. However, ECP impacted only tens of applications and software technologies directly and focused only on very high-end systems. Hundreds of scientific software products can realize this "100X" potential, so the work continues. We need to make the ECP libraries and tools available for high-end systems, deskside and rack systems, and computing centers with limited power budgets where energy efficiency is key. Many public and private sector scientific software stacks are primarily running on CPU-based parallel computers, limiting performance and energy efficiency improvements. Bridging performance and efficiency gaps for these communities is essential.
- **Delivering a turnkey community scientific software stack:** The ECP project has been delivering a near-turnkey scientific software stack to the HPC/AI community for the past

three years. Spack-enabled software stacks, with E4S as a primary deliverable, have been key components of the ECP software delivery strategy. Part of our vision is making this stack available ubiquitously—critical as we build toward the emerging area of HPC/AI for science—incorporating and complementing vendor-provided libraries and tools.

- **Realizing the cross-cutting potential of AI/ML for science:** AI/ML workflows for scientific discovery will be physics-informed machine learning, graph neural networks, deep reinforcement learning, and multimodal large language models, to enable digital twins, analysis of scientific data for discovery and acceleration of traditional simulations. Presently, many scientists manage their own AI/ML software stacks. The ECP software ecosystem can be readily augmented beyond its existing support of mainstream AI/ML libraries (e.g., TensorFlow, PyTorch) to include new libraries and tools that support AI for science.
- **Sustaining a collaborative approach for scientific software development:** ECP demonstrated successful large-scale scientific software development delivering a curated software stack to the HPC/AI community. The ECP leadership approach has been successful in providing a framework for planning, executing, tracking, assessing, and reporting on the project. We will sustain these efforts and expand them to other communities.
- **Engaging software foundations:** The ECP software ecosystem has been successful in delivering a curated software stack to the HPC/AI community. To further expand the cost and benefit sharing of software investments, we will engage with software foundations, in particular the High-Performance Software Foundation (HPSF)[8] to sustain and expand the ecosystem.

Conclusions

The Exascale Computing Project provided a unique opportunity for the DOE scientific libraries and tools communities by sponsoring sustained multi-year funding that promoted cross-institutional collaboration at a scale the DOE high-performance computing community had not experienced before.

Using a three-tier organizational structure (product, SDK, E4S), ECP enabled mostly autonomous activities for product teams, rapid design space exploration via SDKs, and delivery of a curated software stack via E4S. Using a tailored EVM system ECP was able to execute hundreds of activities in parallel and still carefully monitor progress. The use of EVM framed the expected scope, schedule, and cost, providing a foundation for expected outcomes and allowing us to focus our attention on monitoring and addressing variations and realize the scalable delivery of our libraries and tools. The outcome is a GPU-capable, performance-portable stack of 70 libraries and tools, representing 1,700 completed milestones and nearly 300 documented integrations over six years.

The legacy of these efforts is proof of the potential of portable accelerated computing, project leadership structures and strategies for realizing success, and a collection of libraries and tools that, in combination, can deliver two orders of magnitude performance and energy efficiency improvements for scientific applications, the result of the efforts of hundreds of scientists who were part of ECP. We look forward to building on this legacy in the years to come.

Acknowledgment

We thank the ECP Software Technology Leadership team: Lois Curfman McInnes, Jonathan Carter, Rajeev Thakur, Jeffrey Vetter, Xiaoye (Sherry) Li, James Ahrens, Todd Munson, Rob Neely, and Kathryn Mohror for their contributions to implementing and executing the work described in this paper. We also thank the ECP review committee members who supported and helped us define the framework described in this paper. Finally, we thank the many ECP staff members who contributed to the success of the ECP software libraries and tools efforts. Their investments of time and energy made the plans and processes work despite challenges and initial uncertainties. There would be nothing to write about without their efforts. This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy, Office of Science, and the National Nuclear Security Administration. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

[8] https://hpsf.io

REFERENCES

1. D. Kothe, S. Lee, and I. Qualters, "Exascale computing in the United States," *Computing in Science and Engineering*, vol. 21, no. 1, pp. 17–29, 2019, doi:10.1109/MCSE.2018.2875366.
2. U.S. Department of Energy, "DOE O 413.3B Chg 7 (LtdChg)," https://www.directives.doe.gov/directives-documents/400-series/0413.3-BOrder-B-chg7-ltdchg, 2024, accessed: 2024-01-27.
3. U.S. Department of Energy, Office of Project Management, "Earned Value Management," https://www.energy.gov/projectmanagement/earned-value-management, accessed: 2023-10-01.
4. R. Knaster and D. Leffingwell, *SAFe® 5.0 Distilled: Achieving Business Agility with the Scaled Agile Framework*, 1st ed. Addison-Wesley Professional, 2020.
5. C. Larman and B. Vodde, *Large-Scale Scrum: More with LeSS*, 1st ed. Addison-Wesley Professional, 2016.
6. "CHIPS and Science Act," https://science.house.gov/chipsandscienceact and https://science.house.gov/imo/media/doc/the_chips_and_science_act.pdf, united States Congress, July 2022.
7. A. Brandenburger and B. Nalebuff, "The Rules of Co-opetition," *Harvard Business Review*, January 2021, https://hbr.org/2021/01/the-rules-of-co-opetition.
8. S. Balay, S. Abhyankar, M. F. Adams, S. Benson, J. Brown, P. Brune, K. Buschelman, E. M. Constantinescu, L. Dalcin, A. Dener, V. Eijkhout, J. Faibussowitsch, W. D. Gropp, V. Hapla, T. Isaac, P. Jolivet, D. Karpeev, D. Kaushik, M. G. Knepley, F. Kong, S. Kruger, D. A. May, L. C. McInnes, R. T. Mills, L. Mitchell, T. Munson, J. E. Roman, K. Rupp, P. Sanan, J. Sarich, B. F. Smith, S. Zampini, H. Zhang, and J. Zhang, "PETSc/TAO Users Manual (Rev. 3.20)," *OSTI Technical Report*, 11 2023, https://www.osti.gov/biblio/2205494. [Online]. Available: https://www.osti.gov/biblio/2205494
9. M. A. Heroux, L. C. McInnes, R. Thakur, J. S. Vetter, X. S. Li, J. Ahrens, T. Munson, K. Mohror, T. L. Turton, and ECP Software Technology teams, "ECP Software Technology Capability Assessment Report, V3.0," June 2022, https://www.osti.gov/biblio/1888898. [Online]. Available: https://www.osti.gov/biblio/1888898
10. M. A. Heroux, "100X: Leveraging the future potential of US Exascale Computing Project investments," https://www.nas.nasa.gov/pubs/ams/2023/06-20-23.html, Sandia National Laboratories, 6 2023, presentation at the NASA Advanced Modeling & Simulation Seminar Series.

Michael A. Heroux is a senior scientist at Sandia National Laboratories and Scientist in Residence at St. John's University, MN. His research interests include all human and technical aspects of scalable scientific and engineering software for new and emerging parallel computing architectures.