MS5H - Code Complete and More: Emerging Efforts to Improve Software Quality (Part 1/2)

- 11:00 - 11:30 CEST
  - The Role of Software Platforms in Supporting Scientific Software Communities
  - *Michael A. Heroux*
- 11:30 - 12:00 CEST
  - Finding Time through User Research
  - *Hannah Cohoon*, Kazi Sinthia Kabir, Tamanna Motahar, Jason Wiese
- 12:00 - 12:30 CEST
  - Human Factors in Industrial Research Software Engineering
  - Katharina Dworatzyk, *Tobias Schlauch*
- 12:30 - 13:00 CEST
  - How Human-Centered Tools and Processes can Improve Software Development
  - *Axel Huebl*

MS6H - Code Complete and More: Emerging Efforts to Improve Software Quality (Part 2/2)

- 14:00 - 14:30 CEST
  - Socio-Technical Resilience in Research Software Engineering
  - *Caroline Jay,* Helen Sharp, Tamara Lopez, Mark Levine, Melanie Langer, Michel Wermelinger, Bashar Nuseibeh, Yijun Yu, Yo Yehudi
- 14:30 - 15:00 CEST
  - Improving Software Sharing and Impact through Software Registries
  - *Jason Maassen*, Maaike de Jong
- 15:00 - 15:30 CEST
  - Supporting Software Sustainability by Using Software Complexity Metrics to Inform Code Reviews
  - *James Willenbring*
- 15:30 - 16:00 CEST
  - Panel Discussion
  - *All speakers*

Sandia National Laboratories

Exceptional service in the national interest

The role of software platforms in supporting scientific software communities

Michael A. Heroux

PASC 2023 June 28 - 30, 2023

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.

U.S. DEPARTMENT OF ENERGY

NNSA

Outline

Exascale Computing Project (ECP)-sponsored libraries and tools, an intro

Software platforms, defined

E4S and SDKs (aka, spokes), as software platforms

Post-ECP, looking forward

Importance of community, final thoughts

ECP Software Technology (ST) Focus Area

ECP ST Stats

- 250 staff
- 70 products
- 35 L4 subprojects
- 30 universities
- 9 DOE labs
- 6 technical areas
- 1 of 3 ECP focus areas
- ~$500M total budget

WBS	WBS Name	CAM/PI	PC
2.3	**Software Technology**	**Heroux, Mike, McInnes, Lois**	
2.3.1	Programming Models & Runtimes	Thakur, Rajeev	
2.3.1.01	PMR SDK	Shende, Sameer	Shende, Sameer
2.3.1.07	Exascale MPI (MPICH)	Guo, Yanfei	Guo, Yanfei
2.3.1.08	Legion	McCormick, Pat	McCormick, Pat
2.3.1.09	PaRSEC	Bosilca, George	Carr, Earl
2.3.1.14	Pagoda: UPC++/GASNet for Lightweight Communication and Global Address Space Support	Hargrove, Paul	Hargrove, Paul
2.3.1.16	SICM	Graham, Jonathan	Turton, Terry
2.3.1.17	OMPI-X	Bernholdt, David	Grundhoffer, Alicia
2.3.1.18	**RAJA/Kokkos**	**Trott, Christian Robert**	**Trujillo, Gabrielle**
2.3.1.19	Argo: Low-level resource management for the OS and runtime	Beckman, Pete	Gupta, Rinku
2.3.2	Development Tools	Vetter, Jeff	
2.3.2.01	Development Tools Software Development Kit	Miller, Barton	Tim Haines
2.3.2.06	Exa-PAPI++: The Exascale Performance Application Programming Interface with Modern C++	Anzt, Hartwig	Jagode, Heike
2.3.2.08	Extending HPCToolkit to Measure and Analyze Code Performance on Exascale Platforms	Mellor-Crummey, John	Meng, Xiaozhu
2.3.2.10	PROTEAS-TUNE	Vetter, Jeff	Winkler, Amanda
2.3.2.11	SOLLVE: Scaling OpenMP with LLVm for Exascale	Chandrasekaran, Sunita	Oryspayev, Dossay
2.3.2.12	FLANG	McCormick, Pat	Perry-Holby, Alexis
2.3.3	Mathematical Libraries	Li, Sherry	
2.3.3.01	**Extreme-scale Scientific xSDK for ECP**	**Yang, Ulrike**	**Yang, Ulrike**
2.3.3.06	Preparing PETSc/TAO for Exascale	Munson, Todd	Munson, Todd
2.3.3.07	STRUMPACK/SuperLU/FFTX: sparse direct solvers, preconditioners, and FFT libraries	Li, Sherry	Li, Sherry
2.3.3.12	Enabling Time Integrators for Exascale Through SUNDIALS/ Hypre	Woodward, Carol	Woodward, Carol
2.3.3.13	CLOVER: Computational Libraries Optimized Via Exascale Research	Anzt, Hartwig	Carr, Earl
2.3.3.14	ALExa: Accelerated Libraries for Exascale/ForTrilinos	Prokopenko, Andrey	Grundhoffer, Alicia
2.3.3.15	**Sake: Solvers and Kernels for Exascale**	**Rajamanickam, Siva**	**Trujillo, Gabrielle**
2.3.4	Data and Visualization	Ahrens, James	
2.3.4.01	Data and Visualization Software Development Kit	Atkins, Chuck	Bagha, Neelam
2.3.4.09	ADIOS Framework for Scientific Data on Exascale Systems	Klasky, Scott	Hornick, Mike
2.3.4.10	DataLib: Data Libraries and Services Enabling Exascale Science	Ross, Rob	Ross, Rob
2.3.4.13	ECP/VTK-m	Moreland, Kenneth	Moreland, Kenneth
2.3.4.14	VeloC: Very Low Overhead Transparent Multilevel Checkpoint/Restart/Sz	Cappello, Franck	Ehling, Scott
2.3.4.15	ExaIO - Delivering Efficient Parallel I/O on Exascale Computing Systems with HDF5 and Unify	Byna, Suren	Bagha, Neelam
2.3.4.16	ALPINE: Algorithms and Infrastructure for In Situ Visualization and Analysis/ZFP	Ahrens, James	Turton, Terry
2.3.5	Software Ecosystem and Delivery	Munson, Todd	
2.3.5.01	**Software Ecosystem and Delivery Software Development Kit**	**Willenbring, James M**	**Willenbring, James M**
2.3.5.09	**SW Packaging Technologies**	**Gamblin, Todd**	**Gamblin, Todd**
2.3.5.10	ExaWorks	Laney, Dan	Laney, Dan
2.3.6	NNSA ST	Mohror, Kathryn	
2.3.6.01	LANL ATDM	Tim Randles	Montoya, RoseMary
2.3.6.02	LLNL ATDM	Becky Springmeyer	Gamblin, Todd
2.3.6.03	**SNL ATDM**	**Jim Stewart**	**Trujillo, Gabrielle**

A Sampler of Products

MPICH is a high perforr
portable implementation o
Interface (MPI) standard.

No two project alike

Some personality driven

Some community driven

Small, medium, large

ECP Software Technology works on products that apps need now and in the future

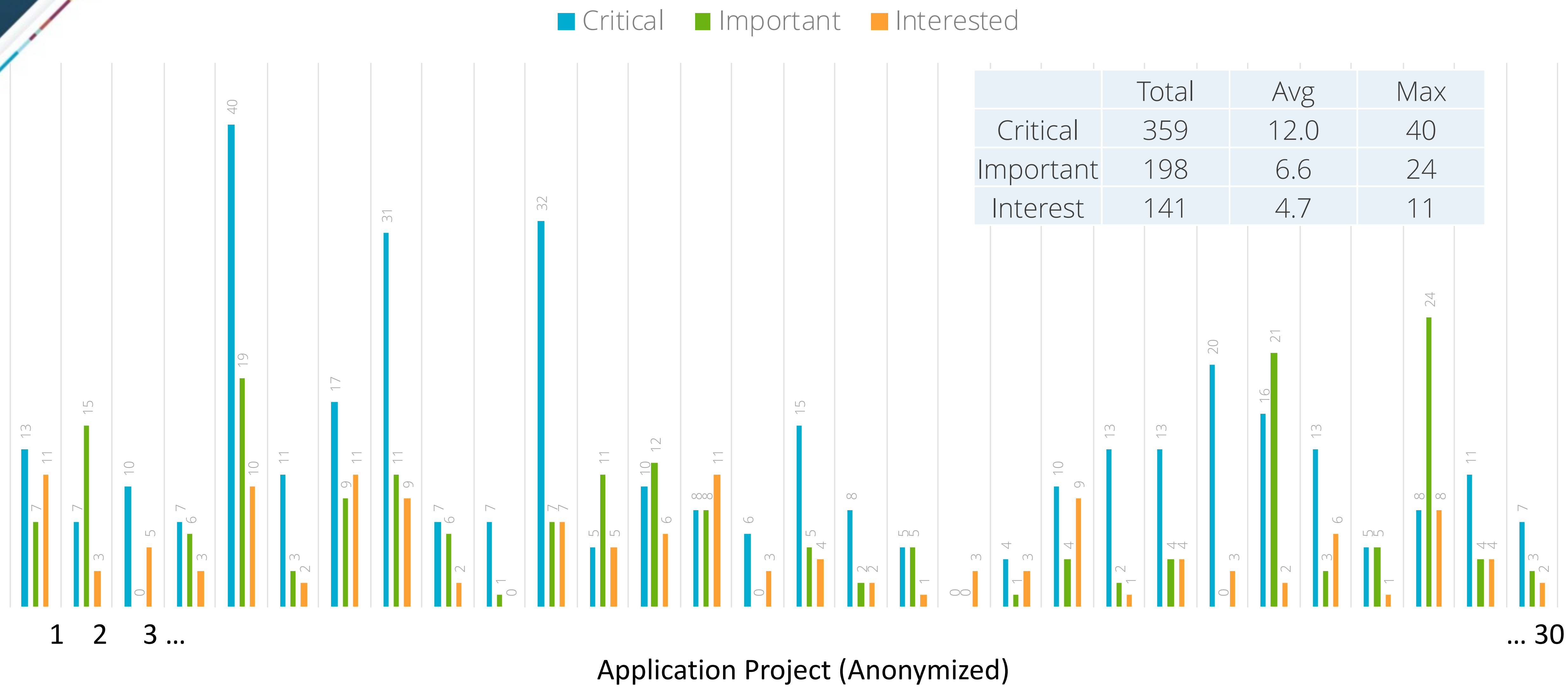
Legacy: A stack that enables performance portable application development CPU, GPU and related platforms, now and in the future

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

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

Example Products	Engagement
MPI – Backbone of HPC apps	Explore/develop MPICH and OpenMPI new features & standards
OpenMP/OpenACC –On-node parallelism	Explore/develop new features and standards
Performance Portability Libraries	Lightweight APIs for compile-time polymorphisms
LLVM/Vendor compilers	Injecting HPC features, testing/feedback to vendors
Perf Tools - PAPI, TAU, HPCToolkit	Explore/develop new features
Math Libraries: BLAS, sparse solvers, etc.	Scalable algorithms and software, critical enabling technologies
IO: HDF5, MPI-IO, ADIOS	Standard and next-gen IO, leveraging non-volatile storage
Viz/Data Analysis	ParaView-related product development, node concurrency

THE NUMBER OF ECP SOFTWARE TECHNOLOGY PROJECT DEPENDENCIES FOR EACH ECP APPLICATION PROJECT (ANONYMIZED)



	Total	Avg	Max
Critical	359	12.0	40
Important	198	6.6	24
Interest	141	4.7	11

Takeaways from product sampler

Wide range of products and teams: libs, tools, small personality-driven, large community-driven

Varied user base and maturity: widely used, new, emerging

Variety of destinations: direct-to-user, facilities, community stacks, vendors, facilities, combo of these

Wide range of dev practices and workflows: informal to formal

Wide range of tools: GitHub, GitLab, Doxygen, Readthedocs, CMake, autotools, etc.

Question at this point might (should?) be:

- *Why are you trying to make a portfolio from this eclectic assortment of products?*

Answer:

- Each product team charged with challenging tasks:
  - Provide capabilities for next-generation leadership platforms
  - Address increasing software quality expectations
  - While independently developed, product compatibility and complementarity improvements matter
- **Working together on these frontiers is better than going alone**

Software Platforms: “Working in Public” Nadia Eghbal

Platforms in the software world are digital environments that intend to improve the value, reduce the cost, and accelerate the progress of the people and teams who use them

Platforms can provide tools, workflows, frameworks, and cultures that provide a (net) gain for those who engage

Eghbal Platforms:

	HIGH USER GROWTH	LOW USER GROWTH
HIGH CONTRIBUTOR GROWTH	**Federations** (e.g., Rust)	**Clubs** (e.g., Astropy)
LOW CONTRIBUTOR GROWTH	**Stadiums** (e.g., Babel)	**Toys** (e.g., ssh-chat)

Eghbal, Nadia. Working in Public: The Making and Maintenance of Open Source Software (p. 60). Stripe Press. Kindle Edition.

About Platforms and ECP

ECP is commissioned to

- Provide new scientific software capabilities
- On the frontiers of apps, algorithms, software and hardware

ECP provides two platforms to foster collaboration and cooperation as we head into the frontier:

- **E4S:** a comprehensive portfolio of HPC products and dependencies
- ***SDKs:** Domain-specific collaborative and aggregate product suites

E4S and SDKs are:

- novel meta-organizational structures (software platforms) that
  - enable the coordinated development and delivery of capabilities for
    - 70 software products on HPC systems,
      - especially leadership platforms

*In post-ECP discussions, we are proposing a "hub-and-spoke" model. SDKs are "spokes"

Delivering an open, hierarchical software ecosystem

More than a collection of individual products

Levels of Integration | **Product** | **Source and Delivery**

ECP ST Open Product Integration Architecture

- Build all SDKs
- Build complete stack
- Assure core policies
- Build, integrate, test

Source: ECP E4S team; Non-ECP Products (all dependencies)
Delivery: spack install e4s; containers; CI Testing

- Group similar products
- Make interoperable
- Assure policy compliant
- Include external products

Source: SDK teams; Non-ECP teams (policy compliant, spackified)
Delivery: Apps directly; spack install sdk; future: vendor/facility

- Standard workflow
- Existed before ECP

Source: ECP L4 teams; Non-ECP Developers; Standards Groups
Delivery: Apps directly; spack; vendor stack; facility stack

ECP ST Individual Products

Extreme-scale Scientific Software Stack (E4S)

- E4S: HPC software ecosystem – a curated software portfolio
- A **Spack-based** distribution of software tested for interoperability and portability to multiple architectures
- Available from **source**, **containers**, **cloud, binary caches**
- Leverages and enhances SDK interoperability thrust
- Not a commercial product – an open resource for all
- Growing functionality: May 2023: E4S 23.05 – 100+ full release products

Community Policies Commitment to SW quality	DocPortal Single portal to all E4S product info	Portfolio testing Especially leadership platforms
Curated collection The end of dependency hell	Quarterly releases Release 23.2 – February	Build caches 10X build time improvement
Turnkey stack A new user experience	https://e4s.io	Post-ECP Strategy PESO, LSSw

https://spack.io

Spack lead: Todd Gamblin (LLNL)

https://e4s.io

E4S lead: Sameer Shende (U Oregon)

Also includes other products, e.g.,
AI: PyTorch, TensorFlow, Horovod
Co-Design: AMReX, Cabana, MFEM

E4S Community Policies: ***A commitment to quality improvement***

Enhance sustainability and interoperability

Serve as membership criteria for E4S

- Membership is not required for *inclusion* in E4S
- Also includes forward-looking draft policies

Modeled after xSDK community policies

Multi-year effort led by SDK team

- Included representation from across ST
- Multiple rounds of feedback incorporated from ST leadership and membership

SDK lead: Jim Willenbring (SNL)

xSDK

Policies: Version 1

https://e4s-project.github.io/policies.html

- **P1: *Spack-based Build and Installation***
- **P2: *Minimal Validation Testing***
- **P3: *Sustainability***
- **P4: *Documentation***
- **P5: *Product Metadata***
- **P6: *Public Repository***
- **P7: *Imported Software***
- **P8: *Error Handling***
- **P9: *Test Suite***

P1 *Spack-based Build and Installation* Each E4S member package supports a scriptable Spack build and production-quality installation in a way that is compatible with other E4S member packages in the same environment. When E4S build, test, or installation issues arise, there is an expectation that teams will collaboratively resolve those issues.

P2 *Minimal Validation Testing* Each E4S member package has at least one test that is executable through the E4S validation test suite (https://github.com/E4S-Project/testsuite). This will be a post-installation test that validates the usability of the package. The E4S validation test suite provides basic confidence that a user can compile, install and run every E4S member package. The E4S team can actively participate in the addition of new packages to the suite upon request.

P3 *Sustainability* All E4S compatibility changes will be sustainable in that the changes go into the regular development and release versions of the package and should not be in a private release/branch that is provided only for E4S releases.

P4 *Documentation* Each E4S member package should have sufficient documentation to support installation and use.

P5 *Product Metadata* Each E4S member package team will provide key product information via metadata that is organized in the *E4S DocPortal* format. Depending on the filenames where the metadata is located, this may require *minimal setup*.

P6 *Public Repository* Each E4S member package will have a public repository, for example at GitHub or Bitbucket, where the development version of the package is available and pull requests can be submitted.

P7 *Imported Software* If an E4S member package imports software that is externally developed and maintained, then it must allow installing, building, and linking against a functionally equivalent outside copy of that software. Acceptable ways to accomplish this include (1) forsaking the internal copied version and using an externally-provided implementation or (2) changing the file names and namespaces of all global symbols to allow the internal copy and the external copy to coexist in the same downstream libraries and programs. This pertains primarily to third party support libraries and does not apply to key components of the package that may be independent packages but are also integral components to the package itself.

P8 *Error Handling* Each E4S member package will adopt and document a consistent system for signifying error conditions as appropriate for the language and application. For e.g., returning an error condition or throwing an exception. In the case of a command line tool, it should return a sensible exit status on success/failure, so the package can be safely run from within a script.

P9 *Test Suite* Each E4S member package will provide a test suite that does not require special system privileges or the purchase of commercial software. This test suite should grow in its comprehensiveness over time. That is, new and modified features should be included in the suite.

Spack

- E4S uses the Spack package manager for software delivery
- Spack provides the ability to specify versions of software packages that are and are not interoperable
- Spack is a build layer for not only E4S software, but also a large collection of software tools and libraries outside of ECP ST
- Spack supports achieving and maintaining interoperability between ST software packages
- https://spack.io

Frank – Designed for Libs & Tools Developers

- Prep system for ECP libs & tools
- Access to latest non-NDA HW/SW
- Shared file system – 1 copy of SW
- Port to many device types at once
- Porting support from E4S team
- CI testing workhorse (500K builds)
- Next: Bare metal, BIOS-changing support for low-level software work

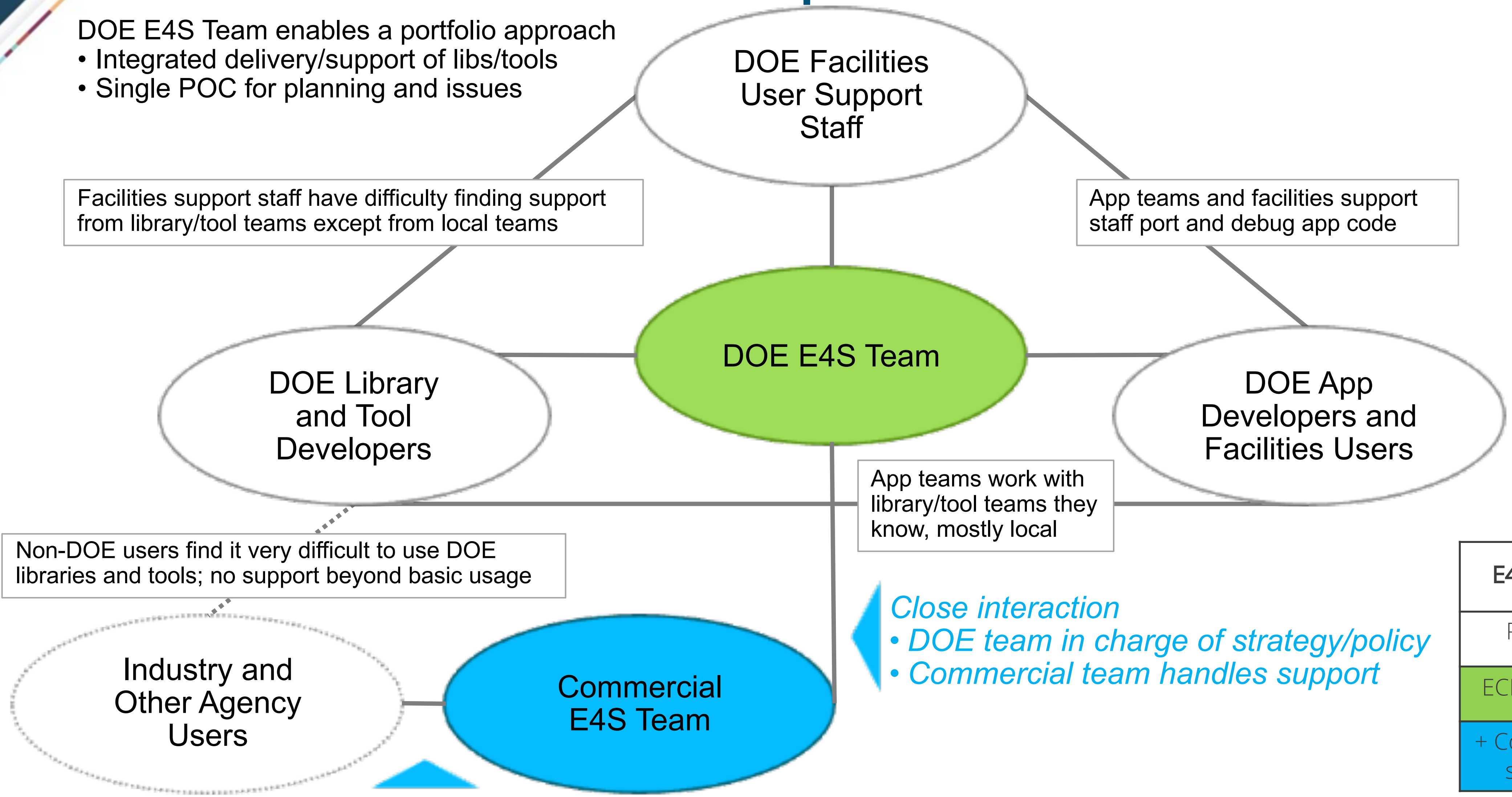
Name	Description	Processors	Local Network	Physical location
Compute: Orthus	**Primary login gateway**	2 x 8c Xeon E5-2667 v2 @ 3.3GHz	10GbE	
Compute: Jupiter	Quad Cooper lake + Intel DG1	4 x 24c Xeon Gold 6438 @ 2.3GHz	100GbE + EDR	R86.U10
Compute: Saturn	Quad Cooper lake + A100 (80GB)	4 x 26c Xeon Platinum 8367HC @ 3.2GHz	100GbE + EDR	R86.U10
Compute: Reptar	Cascade lake 6248 node	2 x 24c Xeon Gold 6248R @ 2.9GHz	10GbE + 100GbE	R84.U37
Compute: Illyad	AMD + 2 A100 (40GB)	2 x 24c Epyc Rome 7402 @ 2.8GHz	100GbE + 2xEDR	R85.U22
Compute: Gilgamesh	AMD + 2 MI50 + A100 (40GB)	2 x 24c Epyc Milan 7413 @ 2.6GHz	100GbE + 2xEDR	R85.U26
Compute: Instinct	Intel + 2 AMD MI100 + MI50	2 x 14c Xeon E5-2660 v4 2.0GHz	100GbE	R85.U6
Compute: Voltar	A100 (80GB) + P100 + V100 GPU node	2 x 16c Xeon Gold 6226R @ 2.9GHz	10GbE + EDR	R86.U26
Compute: Cyclops	IBM Power9 + 4 V100	2 x 20c Power9 @ 3.66GHz	10GbE + 2xHDR (200 Gbps)	R86.18
Compute: Gorgon	IBM Power9 + 4 V100	2 x 20c Power9 @ 3.66GHz	10GbE + 2xHDR (200 Gbps)	R86.U16
Compute: Medusa	IBM Power9	2 x 20c Power9 @ 3.66GHz	10GbE	R86.U14
Compute: Typhon	IBM Power9	2 x 20c Power9 @ 3.66GHz	10GbE	R86.U12
Compute: Delphi	Intel + GV100	2 x 18c Xeon E5-2697 v4	100GbE	R86.U35
Compute: Aurora	NEC SX-Aurora demo machine	8c Xeon 4108 Silver @ 1.8GHz	10GbE + EDR	R85.U31
Compute: Godzilla	Intel DG1 + 2 x K80 node	2 x 14c Xeon E5-2680v4 @ 2.3GHz	40GbE + EDR	R85.U6
Compute: Centaur	IBM Power8 + 2 K80	2 x 20c Power8 @ 3.5GHz	10GbE	R85.U18
Compute: Minotaur	IBM Power8 + 2 K80	2 x 20c Power8 @ 3.5GHz	10GbE	R85.U20
Compute: Eagle	IBM Power9 + 3 x T4	2 x 16c Power9 @ 2.1GHz	10GbE + 2xEDR	R86.U24
Compute: Pegasus	Compute node	2 x 18c Xeon Gold 6140 @ 2.3GHz	100GbE + EDR	R86.U22
Compute: Vina	Raptor Talos II	2 x 22c Power9 @ 2.2GHz	10GbE	R84.U44
Compute: Pike	Raptor Talos II + MI25	2 x 22c Power9 @ 2.2GHz	10GbE	R84.U29
Compute: Cirrus-AIX.stor	AIX machine			
Compute: Cumulus-AIX.stor	AIX machine			
Compute: Nimbus-AIX.stor	AIX machine			
Compute: KNL Grover	Intel Phi system			
Compute: Axis cluster (axis1-8)	DL580 G7 nodes			

Name	Description
Visualization: Chymera	Drives 8K display in 472
Visualization: Cerberus	**Secondary login gateway**; Jetson/Nucs + NFS
NUC cluster	Intel NUCs (16)
Jetson ARM64 cluster	Tegra TX-1
Jetson ARM64 cluster	Tegra TX-2
Compute: Xavier	NVidia Tegra 3
Compute: OD1K	ARM64 v8
Compute: Omicron	M1 Mac
Compute: Sever	Intel Xe
Compute: Silicon	VLSI simulation node

Name
Infrastructure:orion
Infrastructure:mecha
Infrastructure:newstorage
Infrastructure:mnemosyne
Infrastructure:lighthouse

E4S Business Model: Optimize Cost & Benefit Sharing

DOE E4S Team enables a portfolio approach
- Integrated delivery/support of libs/tools
- Single POC for planning and issues



First of a kind interactions
- *Industry/agencies can acquire support*
- *Shared costs and benefits with DOE*

E4S Phase	Cost & Benefit Scope
Pre-E4S	Local Facility
ECP support	DOE complex
+ Commercial support	Universal

Commercial E4S Support Essential for non-DOE Users
Provides vehicle for sustainable non-DOE user support

Support Phase	Primary Scope	Primary Cost and Benefit Sharing Opportunities
Pre-E4S	Local facility	**Local costs and benefits:** Prior to ECP and E4S, libraries and tools were typically strongly connected to the local facility: ANL libs and tools at ALCF, LBL at NERSC, LLNL at Livermore Computing, etc.
+ ECP E4S	All DOE facilities	**DOE complex-shared costs and benefits:** ECP requires, and E4S enables, interfacility availability and use of libs across all facilities: First-class support of ANL libs and tools at other facilities, etc.
+ Commercial E4S	DOE facilities, other US agencies, industry, and more	**Universal shared costs and benefits:** Commercial support of E4S expands cost and benefit sharing to non-DOE entities: DOE costs are lower, software hardening more rapid. US agencies, industry and others can contract for support, gaining sustainable use of E4S software and contributing to its overall support.

E4S 23.05: What's New?

- E4S includes support for Intel oneAPI 2023.1 software (BaseKit and HPCToolkit) in containers on x86_64 with support for HPC packages built with Intel compilers
- E4S includes support for CUDA architectures
  - 70 (V100), 80 (A100), and 90 (H100) under x86_64
  - 70 under ppc64, and
  - 75 and 80 under aarch64
- E4S includes supports ROCm for gfx908 (MI100) and gfx90a (MI200) architectures under x86_64
- E4S includes support for DOE LLVM under x86_64, ppc64le, and aarch64
- E4S includes new applications: Xyce (with pymi), LBANN, Quantum Espresso, LAMMPS, WARPX, Deal.ii, and OpenFOAM.
- E4S includes support for AI/ML frameworks such as TensorFlow and PyTorch support for A100 as well as H100 GPUs is integrated in E4S 23.05
- E4S supports updates to 100+ HPC packages on x86_64, aarch64, and ppc64le, 100K+ binaries in E4S Spack Build Cache
- **New E4S tools: e4s-alc (à la carte) customizes container images, e4s-cl (container launch) replaces MPI at runtime!**
- Detailed documentation for installing E4S on bare-metal and using containers

E4S Engagements: DOE, Other US Agencies

DOE:

- NERSC, OLCF, ALCF – Active porting on leadership, exascale platforms
- Multiple ECP apps: ExaWind, WDPApp, Cinema
- Emerging Sandia effort: Xyce on E4S on AWS for a summer class

NSF:

- E4S installed on Frontera, TACC; Bridges-2, PSC; BlueWaters, NCSA; Expanse, SDSC
- SDSC: E4S Singularity containers available on Open Science Grid High Throughput Computing (https://OSG-HTC.org)

NOAA:

- E4S base images being used in production on AWS and in custom containers.

DoD:

- Testing installation of E4S on Narwhal, Navy DSRC

NASA:

- Singularity support for E4S on Pleiades
- Custom E4S images exploration
- Visit to NASA Ames on April 11, 2023

E4S Engagements: International

- CEA, France: E4S engagement discussed with CEA
  - Workshop planned in July 2023 with ParaTools, SAS
- CSC, Finland: Lumi Supercomputer
  - E4S Workshop in March 2023
  - https://ssl.eventilla.com/event/WL761
  - E4S 23.05 installed on Lumi
- Pawsey Supercomputing Center, Perth, Western Australia
  - E4S workshop planned in April 2023
  - https://pawsey.org.au/event/evaluate-application-performance-using-tau-and-e4s-april-4-5/
  - E4S 23.05 installed on Setonix

- E4S provides a large stack of reusable software libraries and tools
- Build from scratch using Spack, or use via containers, cloud, build caches
- Includes Trilinos, Kokkos: building blocks for many Sandia codes
- Makes stack management easier, portable, lower cost
- Promises to reduces complexity for higher-level analysis tools (Teresa's talk)

Steady Stream of E4S to ALL DOE Science Facilities!

GitHub

ST Development Team

ST Development Team works any issues reported

PHASE I

Product establishes:

1. Spack-based package and build
2. Validation testing in E4S testsuite
3. Documentation for install and use
4. Accessible public repository

E4S community policies

E4S

PHASE II

- E4S establishes install at facilities
- E4S packages get tested and validated in facility environment
- New E4S releases automatically tested through ECP CI infrastructure

OUTPUT

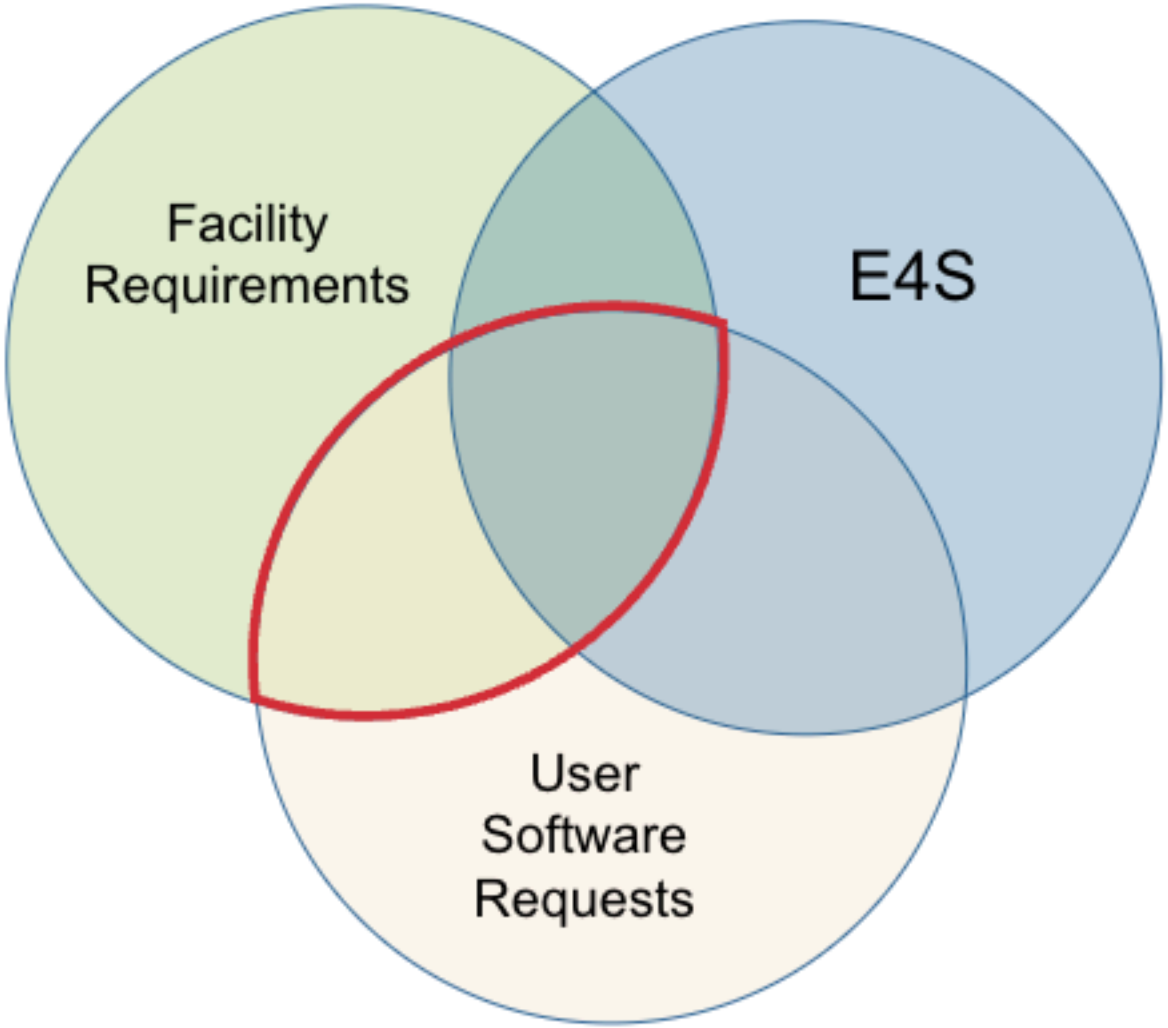
- **High-quality Spack recipes, for ECP products, ready for facility systems**

E4S

FEEDBACK PHASE

- **Software Integration team integrates packages into facility system**
- **New E4S release up-streamed and support requests from facility generated as needed**
- **Issues/Fixes/changes worked with developers as needed**

Facility Software Integration



- Not all E4S products can be maintained by facility staff (there are a lot!)
- User requests drive facility priorities
- Compatibility and maintainability, with facility environments, are essential
- Red area depicts area of focus from perspective of software integration staff at facilities
- ‘Level 2’ Support from ParaTools helps!

Final Thoughts on E4S

Does E4S contain too much?

- Yes, few people use everything

Does E4S not contain enough?

- Yes, most serious users include more products via Spack

Is E4S ported to and tested on too many configurations?

- Yes, most users need only a few

Is E4S not ported to and tested on enough configurations?

- Yes, almost everyone needs to tweak it for their environment

Is E4s useful?

- A big yes

For almost all users,

- The gap between what they want and what E4S is can be addressed through incremental efforts
- The stable E4S suite available on many platforms enables “better, faster, cheaper, pick all 3”

PESO: Toward a Post-ECP Software-Sustainability Organization

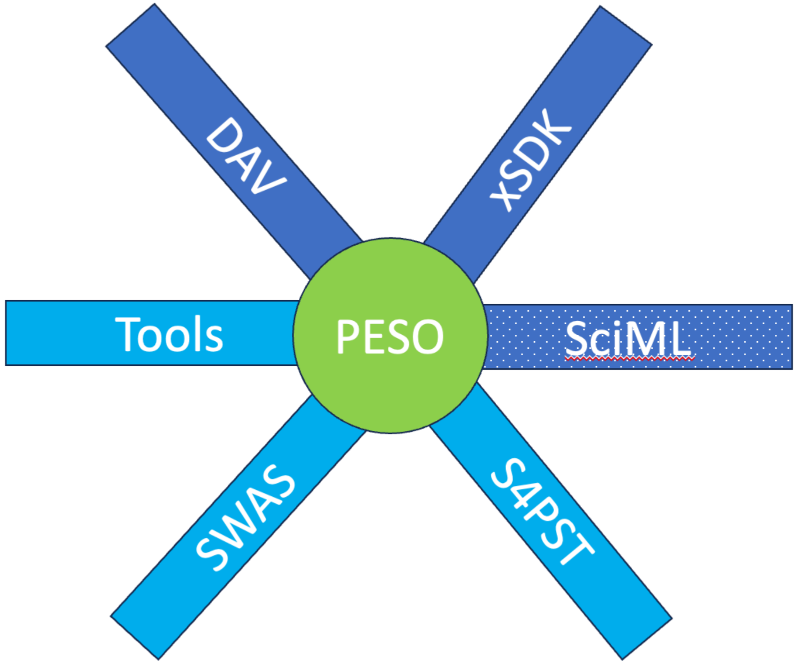
- Michael Heroux (Sandia National Laboratories; PI)
- James Ahrens (Los Alamos National Laboratory)
- Todd Gamblin (Lawrence Livermore National Laboratory)
- Timothy Germann (Los Alamos National Laboratory)
- Xiaoye Sherry Li (Lawrence Berkeley National Laboratory)
- Lois Curfman McInnes (Argonne National Laboratory)
- Kathryn Mohror (Lawrence Livermore National Laboratory)
- Todd Munson (Argonne National Laboratory)
- Sameer Shende (University of Oregon)
- Rajeev Thakur (Argonne National Laboratory)
- Jeffrey Vetter (Oak Ridge National Laboratory)
- James Willenbring (Sandia National Laboratories)

https://pesoproject.org

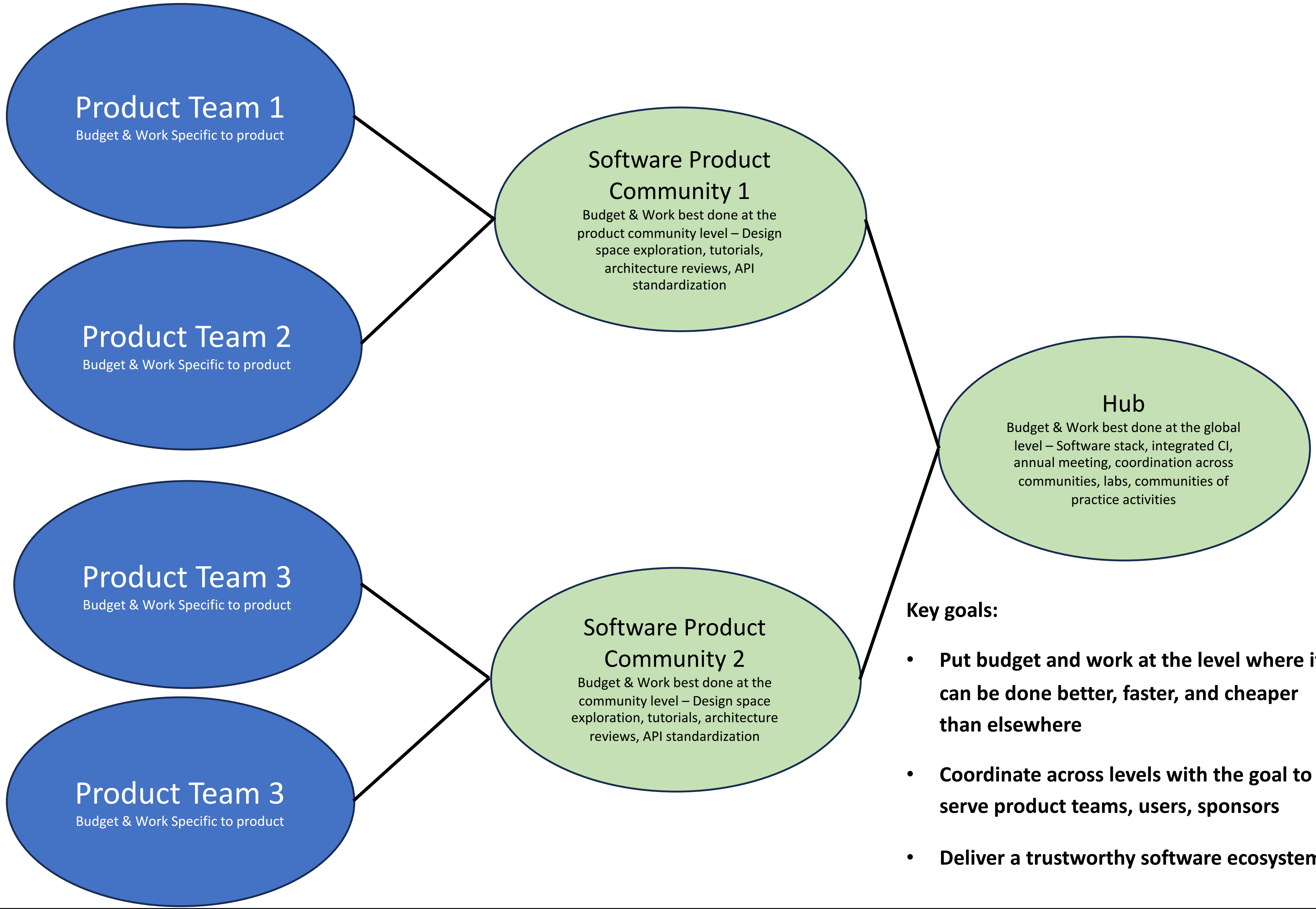
PESO Project Hub-and-Spoke Model

PESO will

- **Serve as a hub** for software-ecosystem sustainment efforts for DOE's open-source libraries and tools for advanced scientific computing
- **Work with spokes (groups of software project teams)** to coordinate development activities for long-term sustainability and benefit to stakeholc
- Work with **communities of practice (COPs)**
  - To provide cross-cutting services and support that are broadly needed by developer users, and stakeholders
- Realize the full potential of DOE investments in the scientific libraries and t ecosystem:
  - By taking a broad, strategic view
  - Through project growth, improved software quality and availability, and sustainable delivery, deployment, and support.
  - Realizing the 100X potential enabled by ECP investments



PESO Proposed Organization Strategy



Key goals:

- **Put budget and work at the level where it can be done better, faster, and cheaper than elsewhere**
- **Coordinate across levels with the goal to serve product teams, users, sponsors**
- **Deliver a trustworthy software ecosystem**

- A lot of work is best done at the individual product team level
  - Everyday development work
  - Delivery of capabilities that contribute part of the whole
  - Testing and product improvement
- Some work is best done at a product community level:
  - Portfolio planning, coordination
  - Holistic tutorial delivery
  - Design space exploration for next-gen platforms
- Some work is best done at hub level:
  - Software stack management
  - Specialized CI testing
  - All-team meetings
  - Coordinated planning across portfolio
  - Community of practice activities: working with software foundations, improving software skills, community engagement

It Takes a Community

The Natural Selection of Bad Science – or why we need communities

Use of narrow metrics, e.g., pub count, impact factor, leads to poor results

Furthermore, successful teams under these metrics train the next generation

Who build their own teams, further propagating poor results

Conclusion: Progress requires change at institutional [community] level

Smaldino, Paul E. and McElreath, Richard, 2016, *The natural selection of bad science*, R. Soc. open sci.3160384160384 http://doi.org/10.1098/rsos.160384

Other McElreath work (good stuff):

Research as Amateur Software Development: https://youtu.be/zwRdO9_GGhY

Science is Like a Chicken Coop: https://youtu.be/d8LqFO1dk-w

IDEAS – First ASCR-funded productivity and sustainability project

Foundation for establishing communities, and capabilities for the past 9 years

Subject: RE: Project name
Date: Sunday, June 1, 2014 at 9:07:08 AM Central Daylight Time
From: Heroux, Michael A
To: Scheibe, Timothy D, bernholdtde@ornl.gov, Hans Johansen, Lois McInnes, Boyana Norris, Moulton, John D. (LANL), Peter Thornton, Carver, Jeffrey C., Ulrike Yang

I really like Tim's direction with the name. Any other ideas?

From: <Scheibe>, Timothy D <tim.scheibe@pnnl.gov>
Date: Saturday, May 31, 2014 5:00 PM
To: Michael A Heroux <maherou@sandia.gov>, "bernholdtde@ornl.gov" <bernholdtde@ornl.gov>, Hans Johansen <hjohansen@lbl.gov>, Lois McInnes <curfman@mcs.anl.gov>, Boyana Norris <norris@cs.uoregon.edu>, David Moulton <moulton@lanl.gov>, Peter Thornton <thorntonpe@ornl.gov>, "Carver, Jeffrey C." <carver@cs.ua.edu>, Ulrike Yang <yang11@llnl.gov>
Subject: [EXTERNAL] RE: Project name

IDEAS: Interoperable Design/Development of Exascale Application Software

-Tim

From: Heroux, Michael A [mailto:maherou@sandia.gov]
Sent: Friday, May 30, 2014 3:14 PM
To: bernholdtde@ornl.gov; Hans Johansen; Lois McInnes; Boyana Norris; Moulton, John D. (LANL); Scheibe, Timothy D; Peter Thornton; Carver, Jeffrey C.; Ulrike Yang
Subject: Project name

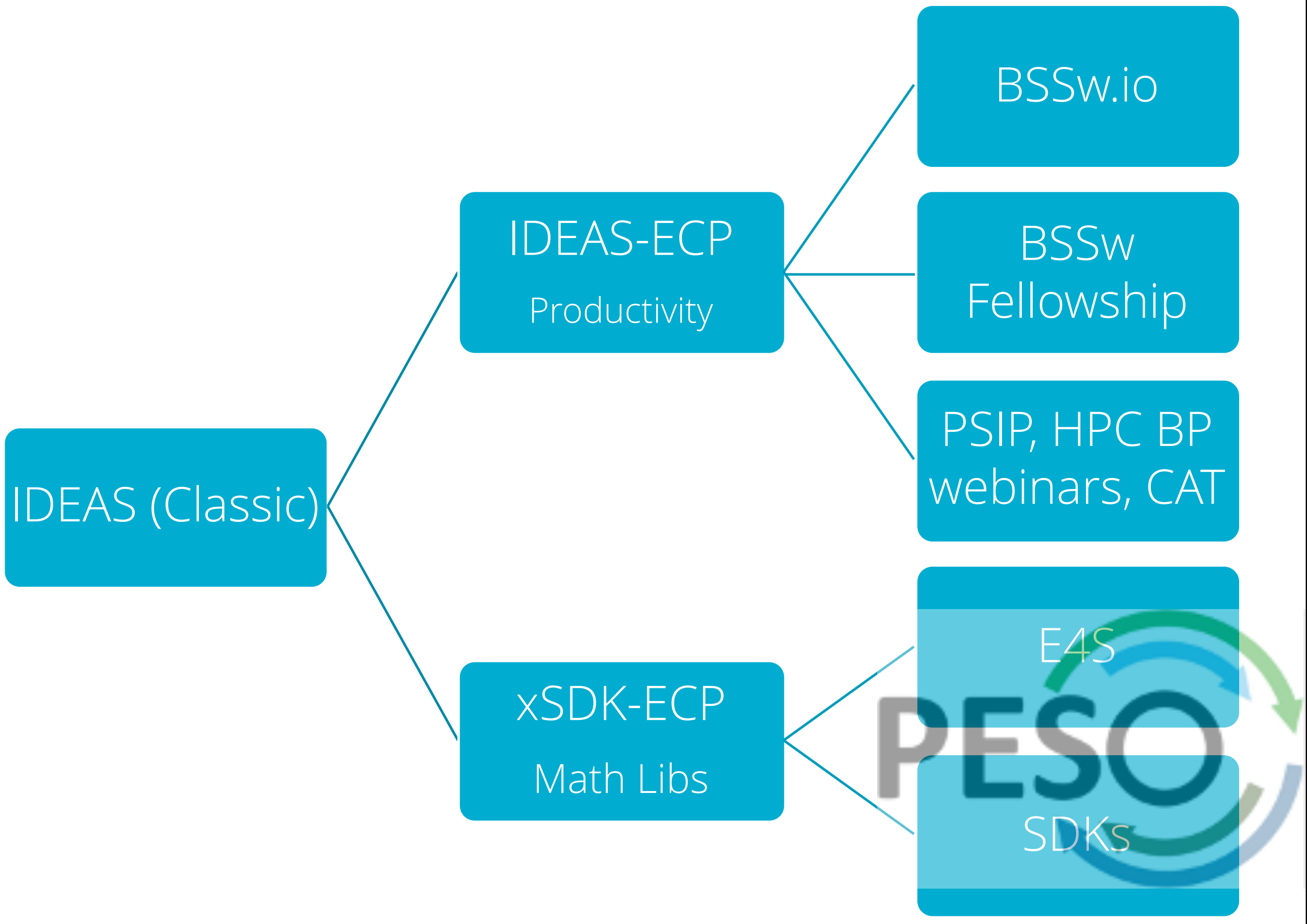
Forgot one thing: We need a project name.

Here are two ideas:

Extreme-scale Software Productivity Institute: ESPI , pronounce espee.
Better, Faster and Cheaper Institute: BFCI, no pronunciation.

Please come up with something better!

Mike



Building Community Takeaways

Elevating trust is fundamentally a community activity

Risky for single team to invest heavily – costs higher, pub rate lower

IDEAS Project focused on cross-lab and partners collaboration

The IDEAS legacy within DOE is community-based R&D

We plan to continue with this approach going forward

Takeaway Points

The US Exascale Computing Project has enabled a holistic community approach to developing and delivering scientific software

The organizational elements of SDKs (product communities or “spokes”) and E4S (curated stack) enable fundamentally new relationships in the scientific computing community

Agency, international, and industry partners can now play a larger role in delivering and supporting US DOE software

The broad impact of these community solutions is unconstrained cost and benefit sharing in the development and use of our software libraries and tools