Toward Community Software Ecosystems for High-Performance CSE

Lois Curfman McInnes
Mathematics and Computer Science Division
Argonne National Laboratory

Workshop on Challenges in High Performance Computing
Australian National University
September 6, 2019
Ecosystem: A group of independent but interrelated elements comprising a unified whole

Diversity is essential for an ecosystem to thrive.

- No element functions in isolation.
- Each element fulfills unique roles.
- The whole is greater than the sum of its parts.
CSE: Essential driver of scientific progress

CSE = Computational Science & Engineering

Development and use of computational methods for scientific discovery

• all branches of the sciences
• engineering and technology
• support of decision-making across a spectrum of societally important applications
Rapidly expanding role of CSE: New directions toward predictive science

- Mathematical methods and algorithms
- CSE and HPC: Ubiquitous parallelism
- CSE and the data revolution
- CSE software
- CSE education & workforce development

Research and Education in Computational Science & Engineering

Software is the foundation of sustained CSE collaboration and scientific progress.

CSE cycle: Modeling, simulation, and analysis
- Software: independent but interrelated elements for various phases that together enable CSE
Audience query: CSE software and YOU

• Do you **develop** CSE software?
  – That you **use** yourself
  – That you **provide** to others
    • In your research group
    • In the broader community

• Do you **use** CSE software developed by others?

• Do you **lead** projects or organizations where teams develop and use CSE software?

• Are you a **stakeholder or supporter** of projects that develop and use CSE software?

• Are you a **fan** of CSE software?
Ecosystem: A group of independent but interrelated elements comprising a unified whole

Diversity is essential for an ecosystem to thrive.

- No element functions in isolation.
- Each element fulfills unique roles.
- The whole is greater than the sum of its parts.
We must explicitly consider community software ecosystem perspectives for next-generation CSE.

**Historically:** Organic growth of software ecosystems around packages

**What’s new now?** Bigger challenges, advances in technologies

**Let’s be intentional.**
- broader perspectives
- productivity, sustainability

Better science, Broader impact
Outline

• About me

• CSE software challenges
  – Architectures, science, research culture

• Toward community software ecosystems
  – Extreme-scale Scientific Software Development Kit (xSDK)
    • Experiences in the Exascale Computing Project (ECP)

• Improving developer productivity & software sustainability
  – International community efforts
    • Better Scientific Software (BSSw)
Acknowledgments

U.S. Department of Energy

• Office of Science, Advanced Scientific Computing Research

• 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
Thank you to my collaborators & communities

- PETSc developers and users
- SIAM Activity Group on CSE
- FASTMath SciDAC Institute
- IDEAS Software Productivity Project
- Developers of xSDK packages
- Better Scientific Software (BSSw) community
Software libraries facilitate progress in computational science and engineering

- **Software library**: a high-quality, encapsulated, documented, tested, and multiuse software collection that provides functionality commonly needed by application developers
  - Organized for the purpose of being reused by independent (sub)programs
  - User needs to know only
    - Library interface (not internal details)
    - When and how to use library functionality appropriately

- **Key advantages** of software libraries
  - Contain complexity
  - Leverage library developer expertise
  - Reduce application coding effort
  - Encourage sharing of code, ease distribution of code

- **References:**
  - What are Interoperable Software Libraries? Introducing the xSDK
PETSc/TAO
Portable, Extensible Toolkit for Scientific Computation / Toolkit for Advanced Optimization

- **Easy customization and composability of solvers at runtime**
  - Enables optimality via flexible combinations of physics, algorithmics, architectures
  - Try new algorithms by composing new/existing algorithms (multilevel, domain decomposition, splitting, etc.)

- **Portability & performance**
  - Largest DOE machines, also clusters, laptops
  - Thousands of users worldwide

Scalable algebraic solvers for PDEs. Encapsulate parallelism in high-level objects. Composable, hierarchical, nested, extensible. Active & supported user community. Full API from Fortran, C/C++, Python.

PETSc provides the backbone of diverse scientific applications.

Clockwise from upper left: hydrology, cardiology, fusion, multiphase steel, relativistic matter, ice sheet modeling

https://www.mcs.anl.gov/petsc
## PETSc/TAO platform for experimentation

No optimality without interplay among physics, algorithmics, architectures

### Functionality

<table>
<thead>
<tr>
<th>Optimization</th>
<th>Time Integrators</th>
<th>Nonlinear Algebraic Solvers</th>
<th>Krylov Subspace Solvers</th>
<th>Preconditioners</th>
<th>Domain-Specific Interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDE Constrained</td>
<td>Adjoint Based</td>
<td>Derivative Free</td>
<td>Others</td>
<td>Pipeline methods</td>
<td></td>
</tr>
<tr>
<td>Pseudo-transient</td>
<td>Runge-Kutta</td>
<td>Strong Stability Preserving</td>
<td>Line Search Newton</td>
<td></td>
<td></td>
</tr>
<tr>
<td>General Linear</td>
<td>IMEX</td>
<td>Rosenbrock-W</td>
<td>Trust Region Newton</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line Search Newton</td>
<td>Quasi-Newton</td>
<td>Nonlinear Gauss Seidel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trust Region Newton</td>
<td>(BFGS)</td>
<td>Successive Substitutions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line Search Newton</td>
<td>Nonlinear Multigrid (FAS)</td>
<td>Nonlinear CG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trust Region Newton</td>
<td></td>
<td>Active Set VI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pipeline methods</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hierarchical Krylov</td>
<td>GMRES</td>
<td>BICG-Stabilized</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LSPR</td>
<td>TFQMR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Chebyshev</td>
<td>CG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYMMLQ</td>
<td>Others</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### More Details (Algorithms, Data Structures, etc.)

- Infrastructure networks, e.g., electrical, gas, water
- Structured mesh refinement
- Complex domains with finite element and finite volume discretizations
- SIMPLE domains and discretizations, e.g., finite difference methods
- Compressed Sparse Row (AJJ)
- Block AIJ
- Matrix Blocks (MatNest)
- Symmetric Block AIJ
- Dense
- GPU and Threading Matrices
- MPI, OpenMP, MPI-IO, CUDA, Pthreads, BLAS, LAPACK, etc.
Multiphase steel modeling using PETSc and hypre

Scale bridging: coupled 3D microscopic-macroscopic

Uses nonlinear and linear FETI-DP domain decomposition methods (in PETSc) and algebraic multigrid (in hypre)

- Demonstrates excellent performance on Blue Gene/Q at Jülich Supercomputing Centre (JUQUEEN: 458K cores) and ALCF (Mira: 1,572K MPI processes).

- Reference:
  
  - [https://doi.org/10.1137/140997907](https://doi.org/10.1137/140997907)
Disruptive changes in HPC architectures

• Extreme levels of concurrency
  – Increasingly deep memory hierarchies
  – Very high node and core counts

• Additional complexities
  – Hybrid architectures
  – GPUs, multithreading, manycore
  – Relatively poor memory latency and bandwidth
  – Challenges with fault resilience
  – Must conserve power – limit data movement
  – New (not yet stabilized) programming models
  – Etc.
Increasing complexity of CSE software

- Computer size, problem size
  - extreme-scale
  - petsascale
  - small cluster
  - 1 core

- Software lifetime
  - decades
  - years
  - months

- Problem complexity, number of developers
  - 1 person
  - small team
  - large team
  - collaborating teams
Challenges of CSE software

Technical
• All parts of the cycle can be under research
• Requirements change throughout the lifecycle as knowledge grows
• Importance of reproducibility
• Verification complicated by floating point representation
• Real world is messy, so is the software

Sociological
• Competing priorities and incentives
• Limited resources
• Perception of overhead with deferred benefit
• Need for interdisciplinary interactions

Science through computing is only as good as the software that produces it.
CSE software opportunities

• Better design, software practices, and tools are available
• Better software architectures: toolkits, libraries, frameworks
• Open-source software, community collaboration

Working toward community software ecosystems for high-performance CSE
Outline

• About me

• CSE software challenges
  – Architectures, science, research culture

• Toward community software ecosystems
  – Extreme-scale Scientific Software Development Kit (xSDK)
    • Experiences in the Exascale Computing Project (ECP)

• Improving developer productivity & software sustainability
  – International community efforts
    • Better Scientific Software (BSSw)
Multiphysics: A primary motivator for exascale

Multiphysics: greater than 1 component governed by its own principle(s) for evolution or equilibrium

– Also: broad class of coarsely partitioned problems possess similarities

nuclear reactors
A. Siegel, ANL

particle accelerators
K. Lee, SLAC

crack propagation
E. Kaxiras, Harvard

climate
K. Evans, ORNL

fusion
A. Hakim, PPPL

radiation hydrodynamics
E. Myra, Univ. of Michigan

IJHPCA, Feb 2013
Vol 27, Issue 1, pp. 4-83

Multiphysics simulations:
Challenges and opportunities

David E Keyes1,2, Lois C McInnes3, Carol Woodward4, William Gropp5, Eric Myra6, Michael Pernice7, John Bell8, Jed Brown9, Alain Clo1, Jeffrey Connors10, Emil Constantinescu11, Don Estep12, Kate Evans13, Charbel Farhat14, Ammar Hakim15, Glenn Hammon11,16, Glen Hansen14, Judith Hill17, Tobin Isaac18, Xiangmin Jiao19, Kirk Jordan20, Dinesh Keshab21, Efthimios Kaxiras22, Alice Koniges23, Kihwan Lee24, Aaron Lott25, Qiming Lu26, John Magerlein27, Reed Maxwell28, Michael McCourt29, Miriam Mehl30, Roger Pawlowski31, Amanda P Randles32, Daniel Reynolds33, Beatrice Rivière34, Ulrich Rüde35, Tim Scheibe36, John Shadid37, Brendan Sheehan38, Mark Shephard39, Andrew Siegel40, Barry Smith41, Xianzhu Tang42, Cian Wilson43 and Barbara Wohlmuth44

doi:10.1177/1094342012468181
Interoperable Design of Extreme-scale Application Software (IDEAS)

Motivation
Enable increased scientific productivity, realizing the potential of extreme-scale computing, through a new interdisciplinary and agile approach to the scientific software ecosystem.

Objectives
Address confluence of trends in hardware and increasing demands for predictive multiscale, multiphysics simulations. Respond to trend of continuous refactoring with efficient agile software engineering methodologies and improved software design.

Impact on Applications & Programs
Terrestrial ecosystem use cases tie IDEAS to modeling and simulation goals in two Science Focus Area (SFA) programs and both Next Generation Ecosystem Experiment (NGEE) programs in DOE Biologic and Environmental Research (BER).

Approach
ASCR/BER partnership ensures delivery of both crosscutting methodologies and metrics with impact on real application and programs.
Interdisciplinary multi-lab team (ANL, LANL, LBNL, LLNL, ORNL, PNNL, SNL)
ASCR Co-Leads: Mike Heroux (SNL) and Lois Curfman McInnes (ANL)
BER Lead: David Moulton (LANL)
Integration and synergistic advances in three communities deliver scientific productivity; outreach establishes a new holistic perspective for the broader scientific community.

IDEAS history
DOE ASCR/BER partnership began in Sept 2014
Program Managers:
- Paul Bayer, David Lesmes (BER)
- Thomas Ndousse-Fetter (ASCR)

First-of-a-kind project: qualitatively new approach based on making productivity and sustainability the explicit and primary principles for guiding our decisions and efforts.
**Example:** Multiscale, multiphysics modeling of watershed dynamics requires combined use of independent packages.

Subsurface applications (blue boxes) and their present usage of xSDK domain components (orange boxes) and numerical libraries (green boxes).
The U.S. National Strategic Computing Initiative (NSCI) is a whole-of-nation effort to accelerate scientific discovery and economic competitiveness by maximizing the benefits of HPC research, development, and deployment.

DOE supports NSCI goals through the ECP
- DOE Office of Science (DOE-SC)
- National Nuclear Security Administration (NNSA)

By 2021-22, deliver an advanced, exascale-ready software stack and mission-critical modeling & simulation and data analytic computing applications.

What is a capable exascale computing system?
- Delivers 50× the performance of today’s 20 PF systems, supporting applications that deliver high-fidelity solutions in less time and address problems of greater complexity
- Operates in a constrained power envelope
- Is sufficiently resilient (perceived fault rate: ≤1/week)
- Includes a software stack that supports a broad spectrum of applications and workloads

https://www.exascaleproject.org
ECP’s holistic approach uses co-design and integration to achieve exascale computing

**Application Development**
Science and mission applications

**Software Technology**
Scalable software stack

**Hardware and Integration**
Relationships: facilities with AD/ST, with vendors

**Emphasis for this presentation**
ECP applications need sustainable coordination among ECP math libraries

- ECP application team interviews:
  - Astro, NWChemEx, WDMAPP, ExaFel, GAMESS, ExaSGD, Subsurface, EXAALT, WarpX, ExaAM, MFIX-Exa, ATDM (LANL, LLNL, SNL) apps, CoPA, AMREX, CEED, CODAR

- Many ECP app teams rely on math libraries, often in combination
  
  Need sustainable coordinated xSDK releases and increasing interoperability among xSDK packages to achieve good performance and easily access advanced algorithms and data structures
ECP math libraries: key themes for advances

Key themes

- Performance on new node architectures
- Extreme strong scalability
- Interoperability, complementarity: xSDK
- Optimization, UQ, solvers, discretizations
- Advanced, coupled multiphysics, multiscale
Software libraries are not enough
Apps need to use software packages in combination

“\textbf{The way you get programmer productivity is by eliminating lines of code you have to write.}”


- Need consistency of compiler (+version, options), 3rd-party packages, etc.
- Namespace and version conflicts make simultaneous build/link of packages difficult
- Multilayer interoperability requires careful design and sustainable coordination
Need software ecosystem perspective

**Ecosystem:** A group of independent but interrelated elements comprising a unified whole

Ecosystems are challenging!

“We often think that when we have completed our study of one we know all about two, because ‘two’ is ‘one and one.’ We forget that we still have to make a study of ‘and.’”

− Sir Arthur Stanley Eddington (1892–1944), British astrophysicist
Extreme-scale Science Applications

Domain component interfaces
- Data mediator interactions.
- Hierarchical organization.
- Multiscale/multiphysics coupling.

Native code & data objects
- Single use code.
- Coordinated component use.
- Application specific.

Shared data objects
- Meshes.
- Matrices, vectors.

Library interfaces
- Parameter lists.
- Interface adapters.
- Function calls.

Documentation content
- Source markup.
- Embedded examples.

Testing content
- Unit tests.
- Test fixtures.

Build content
- Rules.
- Parameters.

Shared data objects
- Meshes.
- Matrices, vectors.

Extreme-scale Science Applications

Domain components
- Reacting flow, etc.
- Reusable.

Libraries
- Solvers, etc.
- Interoperable.

Frameworks & tools
- Doc generators.
- Test, build framework.

SW engineering
- Productivity tools.
- Models, processes.

Extreme-scale Scientific Software Development Kit (xSDK)
Building the foundation of a highly effective extreme-scale scientific software ecosystem

**Focus:** Increasing the functionality, quality, and interoperability of important scientific libraries, domain components, and development tools

**Impact:**
- Improved code quality, usability, access, sustainability
- Inform potential users that an xSDK member package can be easily used with other xSDK packages
- Foundation for work on performance portability, deeper levels of package interoperability
xSDK History: Version 0.1.0: April 2016

Notation: A → B:
A can use B to provide functionality on behalf of A

xSDK functionalty, April 2016
Tested on key machines at ALCF, NERSC, OLCF, also Linux, Mac OS X

April 2016
• 4 math libraries
• 1 domain component
• PETSc-based xSDK installer
• 14 mandatory xSDK community policies

https://xsdk.info

Multiphysics Application C

Application A

Application B

xSDK Installer

HDF5
BLAS
More external software

Domain components
• Reacting flow, etc.
• Reusable.

Libraries
• Solvers, etc.
• Interoperable.

Frameworks & tools
• Doc generators.
• Test, build framework.

SW engineering
• Productivity tools.
• Models, processes.

Extreme-Scale Scientific Software Development Kit (xSDK)

Alquimia
PETSc
hypre
SuperLU
Trilinos

More domain components

More contributed libraries

More external software

xSDK Installer

xSDK

PETSc-based xSDK installer

Multiphysics Application C

Application A

Application B

xSDK

4 math libraries
1 domain component
PETSc-based xSDK installer
14 mandatory xSDK community policies

https://xsdk.info

Extreme-Scale Scientific Software Development Kit (xSDK)
### xSDK community policies

**xSDK compatible package:** Must satisfy mandatory xSDK policies:

- **M1.** Support xSDK community GNU Autoconf or CMake options.
- **M2.** Provide a comprehensive test suite.
- **M3.** Employ user-provided MPI communicator.
- **M4.** Give best effort at portability to key architectures.
- **M5.** Provide a documented, reliable way to contact the development team.
- **M6.** Respect system resources and settings made by other previously called packages.
- **M7.** Come with an open source license.
- **M8.** Provide a runtime API to return the current version number of the software.
- **M9.** Use a limited and well-defined symbol, macro, library, and include file name space.
- **M10.** Provide an accessible repository (not necessarily publicly available).
- **M11.** Have no hardwired print or IO statements.
- **M12.** Allow installing, building, and linking against an outside copy of external software.
- **M13.** Install headers and libraries under `<prefix>/include/ and `<prefix>/lib/.
- **M14.** Be buildable using 64 bit pointers. 32 bit is optional.
- **M15.** All xSDK compatibility changes should be sustainable.
- **M16.** The package must support production-quality installation compatible with the xSDK install tool and xSDK metapackage.

---

**Also recommended policies,** which currently are encouraged but not required:

- **R1.** Have a public repository.
- **R2.** Possible to run test suite under valgrind in order to test for memory corruption issues.
- **R3.** Adopt and document consistent system for error conditions/exceptions.
- **R4.** Free all system resources it has acquired as soon as they are no longer needed.
- **R5.** Provide a mechanism to export ordered list of library dependencies.
- **R6.** Provide versions of dependencies.

*We welcome feedback.*

What policies make sense for your software?

https://xsdk.info/policies

Version 0.5.0, July 2019

---

Lois Curfman McInnes, ANU, September 6, 2019
Compatibility with xSDK community policies

To help developers who are considering compatibility with xSDK community policies:

- **Template** with instructions to record compatibility progress
- **Examples** of compatibility status for xSDK packages
  - Approaches used by other packages for compatibility

https://github.com/xsdk-project/xsdk-policy-compatibility
The xSDK is using Spack to deploy its software

- The xSDK packages depend on a number of open source libraries
- Spack is a flexible package manager for HPC
- Spack allows the xSDK to be deployed with a single command
  - User can optionally choose compilers, build options, etc.
  - Will soon support combinatorial test dashboards for xSDK packages

Spack has grown into a thriving open source community

- Over 300 contributors
- Over 2,800 software packages
- Key component of ECP strategy for software deployment
xSDK History: Version 0.2.0: February 2017

Notation: A ➔ B:
A can use B to provide functionality on behalf of A

https://xsdk.info

Multiphysics Application C

Application A

Application B

Extreme-Scale Scientific Software Development Kit (xSDK)

HDF5

BLAS

More external software

xSDK functionality, Feb 2017

Tested on key machines at ALCF, NERSC, OLCF, also Linux, Mac OS X

February 2017
- 4 math libraries
- 2 domain components
- Spack xSDK installer
- 14 mandatory xSDK community policies

xSDK History: Version 0.2.0: February 2017

Tested on key machines at ALCF, NERSC, OLCF, also Linux, Mac OS X

xSDK functionalty, Feb 2017

- HDF5
- BLAS
- More external software

xSDK

Alquimia

PFLOTRAN

More domain components

PETSc

hype

SuperLU

Trilinos

More contributed libraries

Application A

Application B

Domain components
• Reacting flow, etc.
• Reusable.

Libraries
• Solvers, etc.
• Interoperable.

Frameworks & tools
• Doc generators.
• Test, build framework.

SW engineering
• Productivity tools.
• Models, processes.

https://xsdk.info

Notation: A ➔ B:
A can use B to provide functionality on behalf of A

February 2017
- 4 math libraries
- 2 domain components
- Spack xSDK installer
- 14 mandatory xSDK community policies

xSDK

Application A

Application B

Domain components
• Reacting flow, etc.
• Reusable.

Libraries
• Solvers, etc.
• Interoperable.

Frameworks & tools
• Doc generators.
• Test, build framework.

SW engineering
• Productivity tools.
• Models, processes.

Extreme-Scale Scientific Software Development Kit (xSDK)
**xSDK Version 0.3.0: December 2017**

**Notation: A → B:**
- A can use B to provide functionality on behalf of A

**xSDK functionality, Dec 2017**
- Tested on key machines at ALCF, NERSC, OLCF, also Linux, Mac OS X

**https://xsdk.info**

**Domain components**
- Reacting flow, etc.
- Reusable.

**Libraries**
- Solvers, etc.
- Interoperable.

**Frameworks & tools**
- Doc generators.
- Test, build framework.

**SW engineering**
- Productivity tools.
- Models, processes.

**Extreme-Scale Scientific Software Development Kit (xSDK)**

**December 2017**
- 7 math libraries
- 2 domain components
- Spack xSDK installer
- 16 mandatory xSDK community policies

**Impact:**
- Improved code quality, usability, access, sustainability
- Foundation for work on performance portability, deeper levels of package interoperability
xSDK Version 0.4.0: December 2018

https://xsdk.info

Each xSDK member package uses or can be used with one or more xSDK packages, and the connecting interface is regularly tested for regressions.

Multiphysics Application C

Application A

Application B

xSDK functionality, Dec 2018

Tested on key machines at ALCF, NERSC, OLCF, also Linux, Mac OS X

Alquimia
SLEPc
hypre
PETSc
SuperLU
More domain components

STRUMPACK

December 2018
• 17 math libraries
• 2 domain components
• 16 mandatory xSDK community policies
• Spack xSDK installer

Domain components
• Reacting flow, etc.
• Reusable.

Libraries
• Solvers, etc.
• Interoperable.

Frameworks & tools
• Doc generators.
• Test, build framework.

SW engineering
• Productivity tools.
• Models, processes.

Extreme-Scale Scientific Software Development Kit (xSDK)

Impact: Improved code quality, usability, access, sustainability

Foundation for work on performance portability, deeper levels of package interoperability
xSDK collaborators

xSDK Release 0.4.0, Dec 2018

• xSDK release lead: Jim Willenbring, SNL

• xSDK planning
  – Lois Curfman McInnes (ANL)
  – Ulrike Meier Yang (LLNL)

• Leads for xSDK testing
  – Satish Balay, ANL: ALCF testing
  – Piotr Luszczek, UTK: OLCF testing
  – Aaron Fischer, LLNL: NERSC testing
  – Cody Balos, LLNL: general testing
  – Keita Teranishi, SNL: general testing

• Spack liaison: Todd Gamblin, LLNL

• Package compatibility with xSDK community policies and software testing:
  – AMReX: Ann Almgren, Michele Rosso (LBNL)
  – DTK: Stuart Slattery, Bruno Turcksin (ORNL)
  – deal.II: Wolfgang Bangerth (Colorado State University)
  – hypre: Ulrike Meier Yang, Sarah Osborn, Rob Falgout (LLNL)
  – MAGMA and PLASMA: Piotr Luszczek (UTK)
  – MFEM: Aaron Fischer, Tzanio Kolev (LLNL)
  – Omega_h: Dan Ibanez (SNL)
  – PETSc/TAO: Satish Balay, Alp Denner, Barry Smith (ANL)
  – PUMI: Cameron Smith (RPI)
  – SUNDIALS: Cody Balos, David Gardner, Carol Woodward (LLNL)
  – SuperLU and STRUMPACK: Sherry Li and Pieter Ghysels (LBNL)
  – TASMANIAN: Miroslav Stoyanov, Damien Lebrun Grandie (ORNL)
  – Trilinos: Keita Teranishi, Jim Willenbring, Sam Knight (SNL)
  – PHIST: Jonas Thies (DLR, German Aerospace Center)
  – SLEPc: José Roman (Universitat Politècnica de València)
  – Alquimia: Sergi Mollins (LBNL)
  – PFLOTRAN: Glenn Hammond (SNL)

and many more …
xSDK: Primary delivery mechanism for ECP math libraries’ continual advancements toward predictive science

As motivated and validated by the needs of ECP applications:

Performance on new node architectures
Extreme strong scalability
Advanced, coupled multiphysics, multiscale
Optimization, UQ, solvers, discretizations
Interoperability, complementarity: xSDK

Next-generation algorithms
Advances in data structures for new node architectures
Improving library quality, sustainability, interoperability

Toward predictive scientific simulations
Increasing performance, portability, productivity

Timeline:
- xSDK release 1
- xSDK release 2
- …
- xSDK release n

As motivated and validated by the needs of ECP applications:
Ecosystem imperative for CSE software

Dialogue with broader CSE / HPC community

• Classic app development approach:
  – Application developers write most code; source code considered private
  – Occasionally use libraries, but typically only those “baked into” the OS
  – Portability challenges, unmanaged disruptions: Low risk. But…

• Ecosystem-based app development approach:
  – App developers use composition, write glue code & unique functionality
  – Source code includes substantial 3rd-party packages
  – Risks (if 3rd-party code is poor):
    • Dependent on portability of 3rd-party code
    • Upgrades of 3rd-party package can be disruptive
  – Opportunities (if 3rd-party code is good):
    • 3rd party improvements are yours (for free!)
    • Portability to new architectures is seamless

Webinar track

Scientific Software Ecosystems

https://bluewaters.ncsa.illinois.edu/webinars

Objectives:
• Promote quality reusable research software for computational and data-enabled discovery
• Promote community efforts to improve research software quality, culture, credit, collaboration, …

Suggestions welcome!
Contact Scott Lathrop (NCSA Blue Waters Technical Program Manager for Education)
IEEE CiSE special issue
March/April 2019

Accelerating Scientific Discovery with Reusable Software

Goals

- Raise awareness about reusable software among computational- and data-enabled researchers
- Identify the challenges and opportunities facing the community
- Document high-quality, sustainable, reusable software to enhance quality and reproducibility
- Identify the benefits of reusable software that follow standards of quality and good practices
- Foster communities of practice

Papers

- Community organizations: Changing the culture in which research software is developed and sustained
- The role of scientific communities in creating reusable software: Lessons from geophysics
- A community of practice around peer review for long-term research software sustainability
- Developing a computational chemistry framework for the exascale era
- Fostering reuse in scientific computing with embedded components

Outline

• About me

• CSE software challenges
  – Architectures, science, research culture

• Toward community software ecosystems
  – Extreme-scale Scientific Software Development Kit (xSDK)
    • Experiences in the Exascale Computing Project (ECP)

• Improving developer productivity & software sustainability
  – International community efforts
    • Better Scientific Software (BSSw)
Community software ecosystems require high-quality software

• Complex, intertwined challenges

• Need community efforts:
  – Improve software sustainability
  – Change research culture
  – Promote collaboration
  – Etc.

Get involved!

Resources … and opportunities to get involved

• **WSSSPE:** [http://wssspe.researchcomputing.org.uk](http://wssspe.researchcomputing.org.uk)
  - International community-driven organization that promotes sustainable research software

• **NUMFocus:** [https://www.numfocus.org](https://www.numfocus.org)
  - Umbrella nonprofit that supports and promotes open-source scientific computing

• **Software Carpentry:** [http://software-carpentry.org](http://software-carpentry.org)
  - Volunteer non-profit organization dedicated to teaching basic computing skills to researchers.
  - Lessons: [https://software-carpentry.org/lessons](https://software-carpentry.org/lessons)
Resources … and opportunities to get involved

- **Software Sustainability Institute**: [http://www.software.ac.uk](http://www.software.ac.uk)
  - Institute to support UK’s research software community: cultivating better, more sustainable, research software to enable world-class research
  - Guides: [https://www.software.ac.uk/resources/guides-everything](https://www.software.ac.uk/resources/guides-everything)

  - Question and answer site for scientists using computers to solve scientific problems

  - Leading a conversation on impactful sw engineering practices for HPC
Advancing scientific productivity through better scientific software
Science through computing is only as good as the software that produces it.

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

Why? ECP’s legacy rests on the quality and sustainability of ECP software.

• Levels of complexity and change in ECP software far exceed any prior eras and efforts.
• Software is a primary means of ECP collaboration to achieve key performance parameters & overall success.

IDEAS-ECP project works with ECP teams to:

• Understand current software practices, productivity challenges, preferred approaches for collaboration
• Identify crosscutting, high-priority needs for training
• Collaborate on Productivity and Sustainability Improvement Planning
• Improve software practices while maintaining scientific productivity; share best practices experiences

Interviews with ECP teams
1. Applications and software technologies
   • Identify crosscutting productivity challenges and opportunities

Customization & curation of methodologies
3. Targeting scientific software productivity & sustainability
   • Create user stories to prioritize & plan work

Productivity & Sustainability Improvement Planning
2. Lightweight, iterative workflow: teams identify most urgent software bottlenecks and work to overcome them

Outreach & community
4. In partnership with DOE computing facilities & broader community
   • Webinar series, tutorials: https://ideas-productivity.org/events
   • Better Scientific Software site: https://bssw.io

References:
• IDEAS-ECP events: https://ideas-productivity.org/events
IDEAS-ECP team

Michael Heroux (SNL), Co-Lead PI
Lois Curfman McInnes (ANL), Co-Lead PI
David Bernholdt (ORNL), Co-PI, Outreach Lead
Elsa Gonsiorowski (LLNL), Co-PI
Osni Marques (LBNL), Co-PI
David Moulton (LANL), Co-PI
Boyana Norris (Univ of Oregon), Co-PI
Satish Balay (ANL)
Roscoe Bartlett (SNL)
Anshu Dubey (ANL)
Patricia Grubel (LANL)
Rinku Gupta (ANL)
Stephen Hudson (ANL)
Reed Milewicz (SNL)
Mark Miller (LLNL)
Jared O’Neal (ANL)
Elaine Raybourn (SNL)
Barry Smith (ANL)
Greg Watson (ORNL)
Jim Willenbring (SNL)
Paul Wolfenbarger (SNL)
Lisa Childers (ALCF)
Rebecca Hartman-Baker (NERSC)
Judy Hill (OLCF)
Hai Ah Nam (LANL)
Jean Shuler (LLNL)

Computing Facilities Liaisons

https://www.ideas-productivity.org
Best practices for CSE software

**Modern learning theory:**
- Build from knowledge base: Elaboration and Models
- Vast body of SE content from broad community
- Learn, adapt, adopt, assimilate

**Approach:**
Community collaboration to curate, create, and disseminate resources for **better developer productivity and software sustainability**—key element of overall scientific productivity
Resources for software productivity & sustainability: key element of overall scientific productivity

Webinar Series: Best Practices for HPC Software Developers


Prior Sessions:
- Modern C++ for High-Performance Computing
- Jupyter Notebooks; Managing Defects in HPC Software
- Bringing Best Practices to a Long Lived Production Code
- Barely Sufficient Project Management: A few techniques for improving your scientific software efforts
- Intermediate Git
- Python in HPC

Impact: Helping science teams to achieve:
Better: Science, portability, robustness, composability
Faster: Execution, development, dissemination
Cheaper: Fewer staff hours and lines of code

And more resources …
Coming soon to https://bssw.io

Tutorials

- What All Codes Should Do: Overview of Best Practices in HPC Software Development
- Software Design
- Introduction to Git
- Better (Small) Scientific Software Teams
- Improving Reproducibility through Better Software Practices
- Testing and Verification
- Code Coverage and Continuous Integration
- Software Refactoring and Documentation
- An Introduction to Software Licensing

And more …

Slides and videos available via https://ideas-productivity.org/events
Taking stock: Understanding what you want from your CSE software and how to achieve it

• Software architecture and process design
  – Managing complexity and avoiding technical debt (future saving)
  – Worthwhile to understand trade-offs

• Issues to consider
  – The target of the software
    • Proof-of-concept
    • Discard once you’re done with it (or the student/postdoc leaves)
    • Long-term research tool that successive group members will extend
    • Others …
  – How important are performance, scalability, portability to you?
  – Buy vs. build: can you achieve your goals by contributing to existing software, or do you need to start from scratch?
  – What 3rd-party software are you willing to depend on?

• Target should dictate the rigor of the design and development process
  – Considering resource constraints
Technical debt

Consequence of Choices
Quick and dirty collects interest, which means more effort required to add features.

Accretion leads to unmanageable software:
- Increases cost of maintenance
- Parts of software may become unusable over time
- Inadequately verified software produces questionable results
- Increases ramp-on time for new developers
- Reduces software and science productivity due to technical debt
Software process for CSE

Baseline

• Invest in extensible code design
  – Most uses need additions and/or customizations
  – Use revision control and automated testing
  – Institute a rigorous verification and validation regime
  – Define coding and testing standards

• Clear and well defined policies for
  – Auditing and maintenance
  – Distribution and contribution
  – Documentation

Desirable

• Provenance and reproducibility
• Lifecycle management
• Open development and frequent releases
Customize according to your needs

• There is no “all or nothing”
• Focus on improving productivity and sustainability rather than purity of process
• Danger of being too dismissive too soon
  – Examine options with as little bias as possible
• Fine balance between getting a buy-in from the team and imposing process
• First reaction usually is resistance to change and suspicion of new processes
• Many skeptics get converted when they see the benefit
We want and need contributions from the community … Join us!

Collaborative content development on topics related to developer productivity and software sustainability for CSE

Better Scientific Software (BSSw)

Software—the foundation of discovery in computational science & engineering—faces increasing complexity in computational models and computer architectures. BSSw provides a central hub for the community to address pressing challenges in software productivity, quality, and sustainability.

https://bssw.io
We want and *need* contributions from the community ... Join us!

Types of content

- **Curated links**: A brief article that highlights other web-based articles or content. Your article should describe why the CSE community might find value.

- **Original experience**: An original article to inform the CSE community about how to improve developer productivity and software sustainability.

- **Event**: A brief description of an event relevant to better scientific software.

- **"What Is" document**: Define terms and concepts in a particular topic area.

- **"How To" document**: Describe a process for improving productivity and sustainability.

- **Blog article**: An original article in the form of a blog of @250 - 500 words. We will solicit contributions from thought leaders in the community and welcome proposals from anyone.


- **Unit Testing C++ with Catch**, Mark Dewing

- **Best Practices for HPC Developers Webinar Series**

- **What is software configuration?**
- **What is version control?**
- **What are software testing practices**
- **What is good documentation?**
- **What are interoperable software libraries?**
- **How to configure software**
- **How to do version control with Git in your CSE project**
- **How to add and improve testing in your CSE software project**
- **How to write good documentation**
- **And more ...**
STRUMPACK & SuperLU: Advancing software practices as a key aspect of increasing overall scientific productivity

Productivity & Sustainability Improvement Planning (PSIP):
• PSIP workflow helps a team identify areas for improving software practices, select a specific area and topic for a single improvement cycle, and then develop those improvements with specific metrics for success.
• Low-overhead, iterative process, improving long-term productivity while balancing the need to continually achieve scientific advances.

https://bssw.io/items/planning-for-better-software-psip-tools

Advances already:
• Namespacing
• Build system
• Convert from SVN to Git for revision control

Next steps: advances for
• Documentation
• Examples
• CI testing

https://bssw.io/blog_posts/superlu-how-advances-in-software-practices-are-increasing-sustainability-and-collaboration
Collaborative community software ecosystems help improve CSE productivity and sustainability

**Impact**
- **Better:** Science, portability, robustness, composability
- **Faster:** Execution, development, dissemination
- **Cheaper:** Fewer staff hours and lines of code

What makes sense for your software?
- Consider resources to help improve software sustainability
- What community software ecosystems do you want to use and be a part of?
Software ecosystems: Lessons learned

• Working toward shared understanding of issues and perspectives is essential … and takes time.
  – Need regular opportunities for exchanging ideas, persistence, patience, informal interaction
  – Must establish common vocabulary

• Lots of fun, too 😊

- It takes all kinds.
- Think outside the box.
- Face the bumps with a smile.

The pursuit is the reward.

It takes all kinds.
Think outside the box.
Face the bumps with a smile.
Opportunities to collaborate … Join us!

Scientific software in xSDK

We seek contributions to the xSDK and feedback on community policies. See https://xsdk.info/faq

• xSDK compatible package
  – Must satisfy mandatory xSDK policies
• xSDK member package
  – Must be an xSDK-compatible package, and it uses or can be used by another package in the xSDK, and the connecting interface is regularly tested for regressions

Why participate?

• Improved code quality, usability, access, sustainability
• Foundation for work on performance portability, deeper levels of package interoperability

We welcome feedback. What policies make sense for your software?

Opportunities for member packages

Contribute to

Curate, create, & disseminate resources that lead to better scientific software

• key aspect of overall scientific productivity

What topics interest you?

Scientific Libraries Community

The developers of scientific libraries provide widely used software that is the foundation of long-term CSE collaboration and discovery.

Learn More About Communities
Tech tattoo

SO MY CODE WILL SEE THE FUTURE

better scientific software

https://bssw.io

... for your laptop
actual size 2”x4”
Thank you!