xSDK: Foundations of a Numerical Software Ecosystem for High-performance CSE

Ulrike Meier Yang

CASC
Center for Applied Scientific Computing

Feb 26, 2019

SIAM CSE 2019

LLNL-PRES-768139

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

xSDK Project Members:

- Satish Balay
- Cody Balos
- Jim Demmel
- Veselin Dobrev
- Jack Dongarra
- Rob Falgout
- Aaron Fisher
- David Gardner
- Mike Heroux
- Tzanio Kolev
- Ruipeng Li
- Sherry Li
- Piotr Luszczek
- Lois Curfman McInnes
- T. Moore
- Sarah Osborn
- Slaven Peles
- Ben Recht
- Bjorn Sjogreen
- Barry Smith
- Keita Teranishi
- Carol Woodward
- Jim Willenbring
- Ulrike Meier Yang
- …

xSDK Collaborators

hypre high performance preconditioners
PETSc/TAO
SuperLU
TRILINOS
DTK
STRUMPACK
Omega_h
MAGMA
mfem
AMReX
PUMi
deal.II
SLEPc
PLASMA
PHIST
TASMANIAN
sundials

- **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 …

Outline

- **Motivation**
  - Math libraries and scientific software ecosystems
  - Building community and sustainability
  - xSDK history and goals to fulfill ECP needs

- **About the xSDK (eXtreme-scale Scientific software Development Kit)**
  - xSDK community policies
  - xSDK release process
  - Installing the xSDK
  - Using the xSDK in ECP applications

- **Lessons learned**

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:**
  - https://en.wikipedia.org/wiki/Library_(computing)
  - What are Interoperable Software Libraries? Introducing the xSDK

Mutual benefits for users and library developers

User perspective

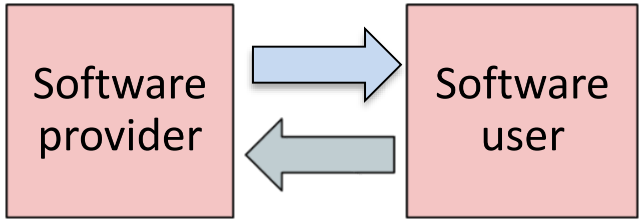
Focus on primary interests

- Reuse algorithms and data structures developed by experts
- Customize and extend to exploit application-specific knowledge
- Cope with complexity and changes over time

Provider perspective:

Share capabilities

- Broader impact of work
- Improved code quality
- Motivate new research directions



- **More efficient, robust, reliable, sustainable software**
- **Improve developer productivity**
- **Better science**

Individual software libraries are not enough.

- Well-designed libraries provide critical functionality … But alone are not sufficient to address all aspects of next-generation scientific simulation and analysis.

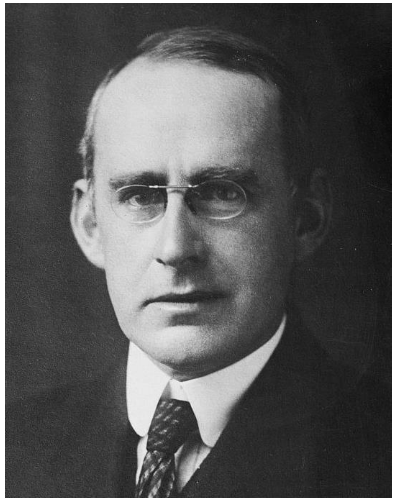
- Applications need to use software packages **in combination** on ever evolving architectures

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

Difficulties in combined use of independently developed software packages

Challenges:

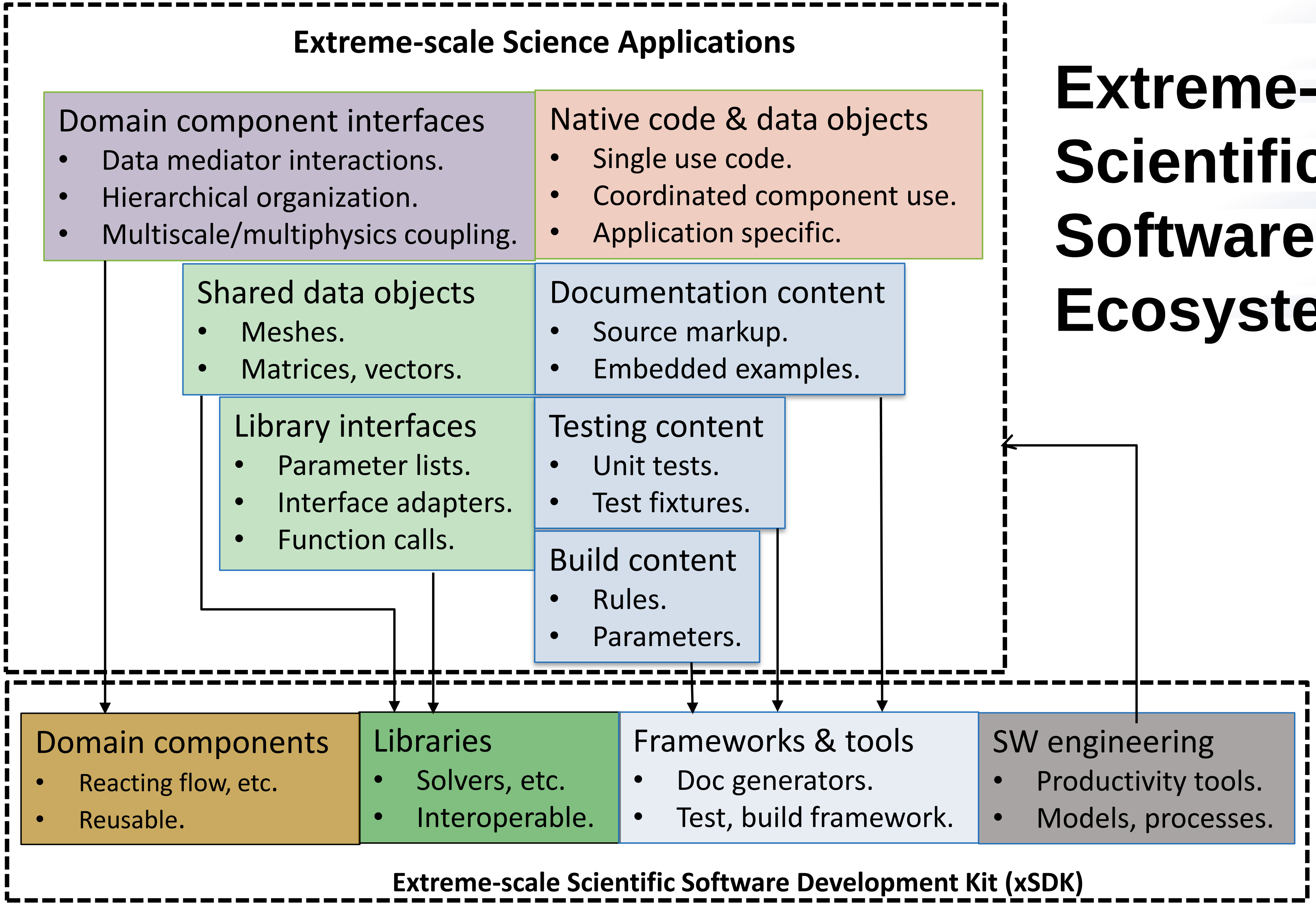
- Obtaining, configuring, and installing multiple independent software packages is tedious and error prone.
  - Need consistency of compiler (+version, options), 3rd-party packages, etc.
- Namespace conflicts
- Incompatible versioning
- And even more challenges for deeper levels of interoperability

Ref: [What are Interoperable Software Libraries? Introducing the xSDK]

Levels of package interoperability:

- **Interoperability level 1**
  - Both packages can be used (side by side) in an application
- **Interoperability level 2**
  - The libraries can exchange data (or control data) with each other
- **Interoperability level 3**
  - Each library can call the other library to perform unique computations

Extreme-scale Scientific Software Ecosystem



IDEAS productivity

www.ideas-productivity.org

Interoperable Design of Extreme-scale Application Software (IDEAS)

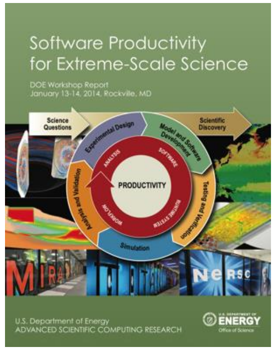
Motivation

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

Objectives

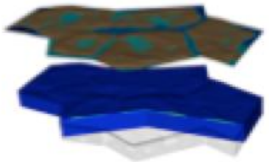
Address confluence of trends in hardware and increasing demands for predictive multiscale, multiphysics simulations.

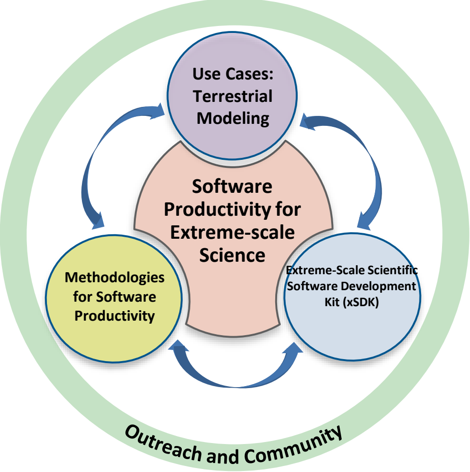
Respond to trend of continuous refactoring with efficient agile software engineering methodologies and improved software design.



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

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.

xSDK for ECP: Project goals, description, scope

Goals: Create a value-added aggregation of ECP mathematics libraries, to increase the combined usability, standardization and interoperability of these libraries, as needed to support large-scale multiphysics and multiscale problems.

Project Description

- Develop **community policies** and **interoperability** layers among xSDK component packages
- Determine xSDK sustainability strategy for ECP
- Work with ECP applications to motivate and test xSDK

Project Scope

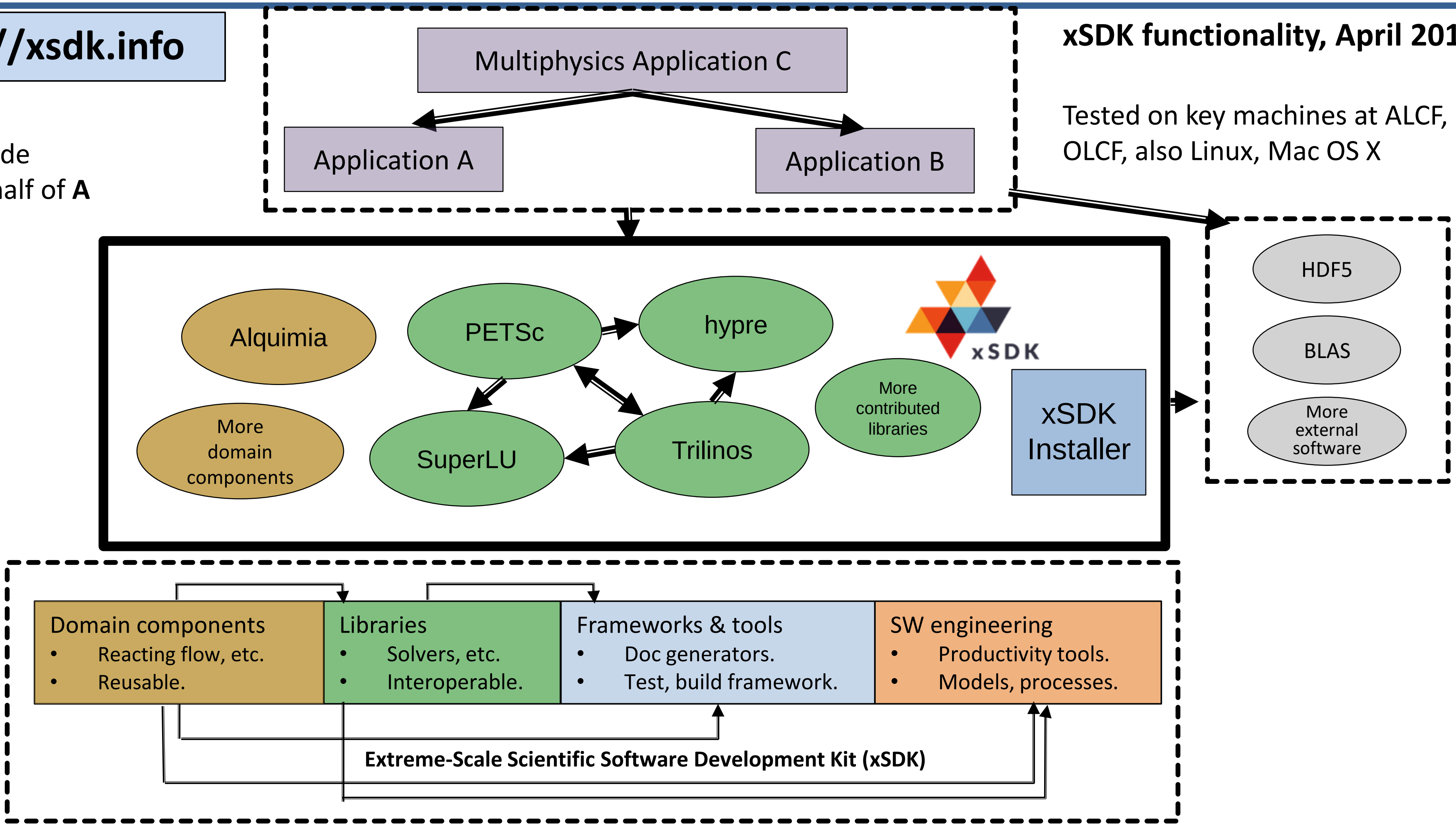
- Enable the seamless combined use of diverse, independently developed software packages as needed by ECP applications
  - coordinated use of on-node resources
  - integrated execution
  - coordinated & sustainable documentation, testing, packaging, and deployment

xSDK History: Version 0.1.0: April 2016

https://xsdk.info

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

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



xSDK functionality, April 2016

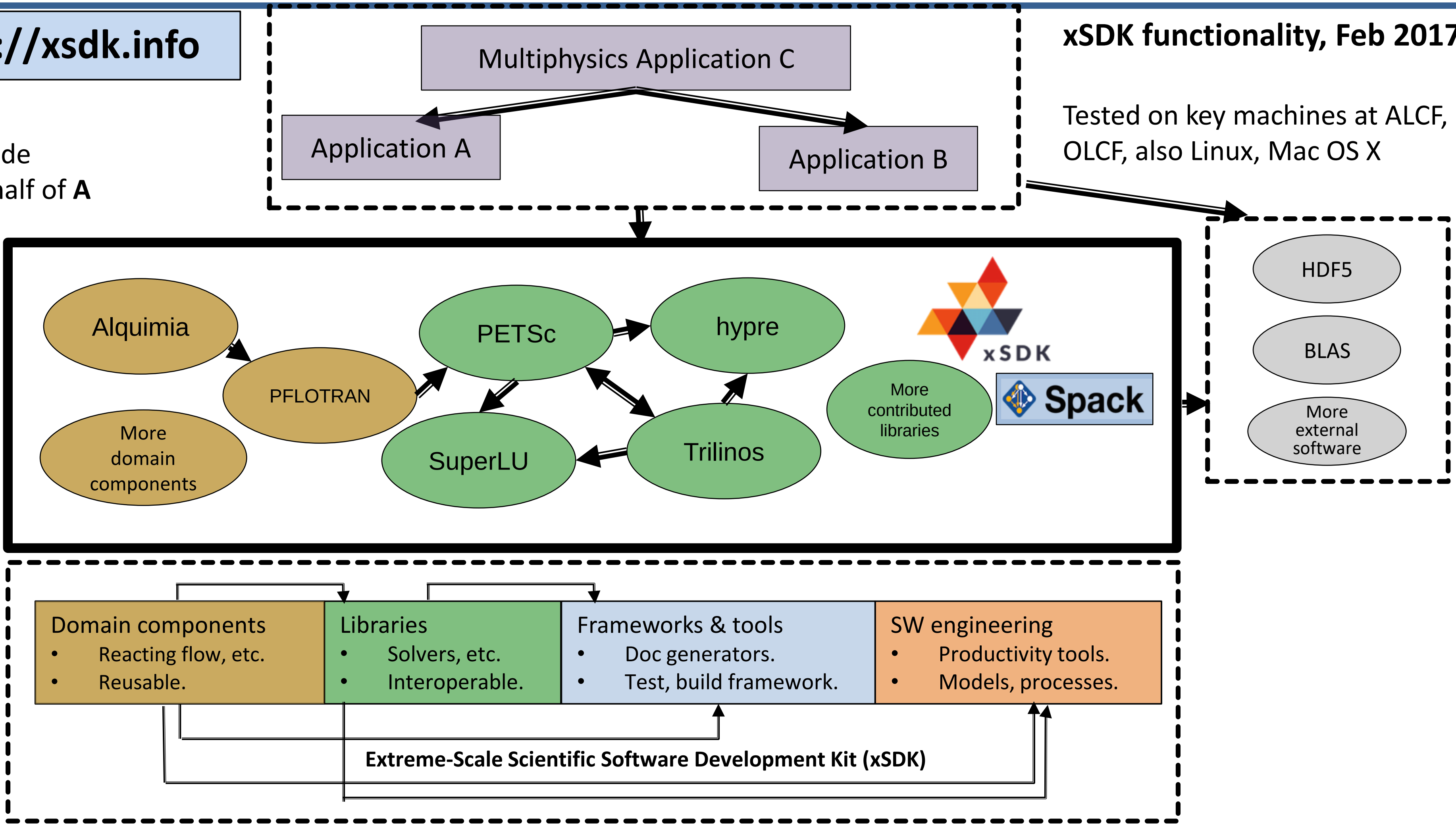
Tested on key machines at ALCF, NERSC, OLCF, also Linux, Mac OS X

xSDK History: Version 0.2.0: February 2017

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



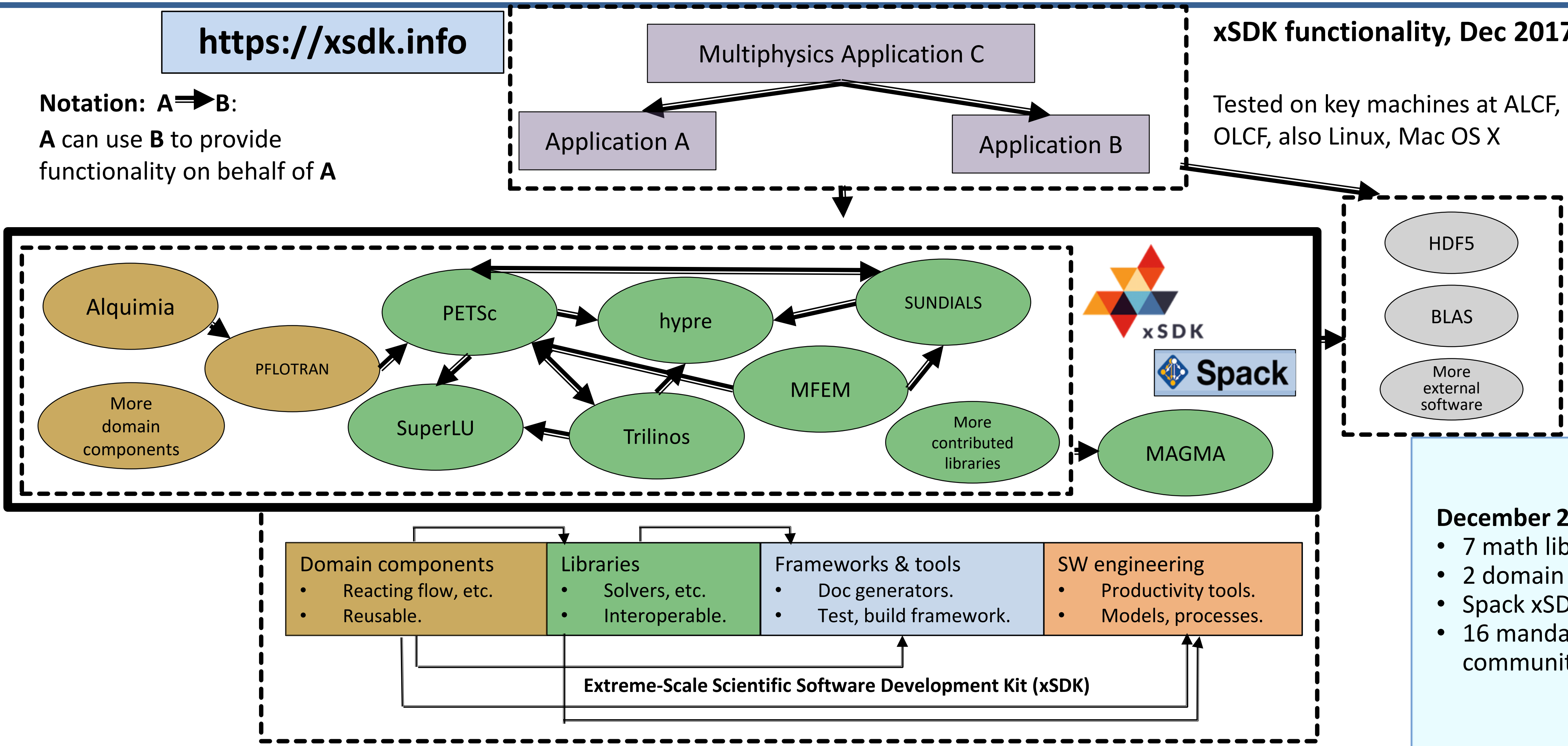
xSDK functionality, Feb 2017

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

xSDK History: Version 0.3.0: December 2017

https://xsdk.info

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



xSDK functionality, Dec 2017

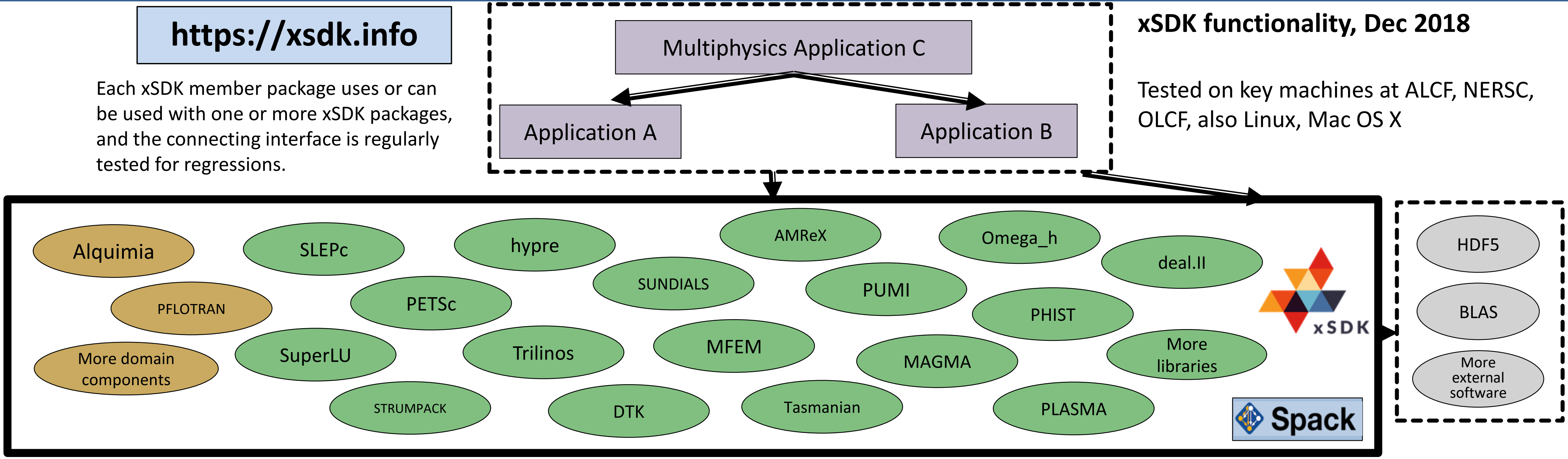
Tested on key machines at ALCF, NERSC, OLCF, also Linux, Mac OS X

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

xSDK History: 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.



xSDK functionality, Dec 2018

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

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

Domain components	Libraries	Frameworks & tools	SW engineering
• Reacting flow, etc. • Reusable.	• Solvers, etc. • Interoperable.	• Doc generators. • Test, build framework.	• 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: https://xsdk.info

Building the foundation of an extreme-scale scientific software ecosystem

xSDK community policies: Help address challenges in interoperability and sustainability of software developed by diverse groups at different institutions

<u>xSDK compatible package</u>: must satisfy the mandatory xSDK policies (M1, ..., M16)

Topics include: configuring, installing, testing, MPI usage, portability, contact and version information, open source licensing, namespacing, and repository access

Also specify **recommended policies**, which currently are encouraged but not required (R1, ..., R6)

Topics include: public repository access, error handling, freeing system resources, and library dependencies

<u>xSDK member package</u>:

(1) Must be an xSDK-compatible package, *and*

(2) it uses or can be used by another package in the xSDK, and the connecting interface is regularly tested for regressions.

xSDK policies 0.4.0: Dec 2018

- Facilitate combined use of independently developed packages

Impact:

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

We encourage feedback and contributions!

xSDK community policies

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

https://xsdk.info/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.

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.

Compatibility with xSDK community policies

To help developers of packages who are considering compatibility with xSDK community policies, we provide:

- Template with instructions to record compatibility progress
- Examples of compatibility status for xSDK packages
  - Explain approaches used by other packages to achieve compatibility with xSDK policies
- Available at **https://github.com/xsdk-project/xsdk-policy-compatibility**

xSDK Community Policy Compatibility for PETSc

This document summarizes the efforts of current and future xSDK member packages to achieve compatibility with the xSDK community policies. Below only short descriptions of each policy are provided. The full description is available here and should be considered when filling out this form.

Please, provide information on your compability status for each mandatory policy, and if possible also for recommended policies. If you are not compatible, state what is lacking and what are your plans on how to achieve compliance. For current xSDK member packages: If you were not compliant at some point, please describe the steps you undertook to fulfill the policy.This information will be helpful for future xSDK member packages.

Website: https://www.mcs.anl.gov/petsc

Mandatory Policies

Policy	Support	Notes
M1. Support xSDK community GNU Autoconf or CMake options.	Full	PETSc uses the GNU Autoconf options. The implementation is done with python code.
M2. Provide a comprehensive test suite for correctness of installation verification.	Full	PETSc has over 1000 test examples and a test harness that can execute the examples in parallel. It also collects information on the failures and can display them graphically, e.g., see ftp://ftp.mcs.anl.gov/pub/petsc/nightlylogs/archive/2017/09/19/master.html
M3. Employ userprovided MPI communicator (no MPI_COMM_WORLD).	Full	All PETSc objects take a MPI communicator in the constructor, allowing the user complete control over where each object exists and performs its computations.
M4. Give best effort at portability		

Processes for xSDK release and delivery

- **2-level release process**
  - **xSDK member packages**
    - Achieve compatibility with xSDK community policies prior to release
      - https://github.com/xsdk-project/xsdk-policy-compatibility
    - Have a Spack package
    - Port to target platforms
    - Provide user support
  - **xSDK**
    - Ensure and test compatibility of mostly independent package releases
- **Obtaining the latest release: https://xsdk.info/releases**
- **Draft xSDK package release process checklist:**
  - https://docs.google.com/document/d/16y2bL1RZg8wke0vY8c97ssvhRYNez34Q4QGg4LolEUk/edit?usp=sharing

xSDK delivery process

- Regular releases of software and documentation, primarily through member package release processes
- Anytime open access to production software from GitHub, BitBucket and related community platforms

Downloading

https://xsdk.info/download

1. Obtain xSDK using Spack.

xSDK is distributed primarily with the Spack package manager.

You can obtain Spack from the github repository using this command:

```
git clone https://github.com/spack/spack.git
```

Installing xSDK

https://xsdk.info/installing-the-software

1. After cloning spack git repo, setup spack environment

```
# For bash users
$ export SPACK_ROOT=/path/to/spack
$ . $SPACK_ROOT/share/spack/setup-env.sh

# For tcsh or csh users (note you must set SPACK_ROOT)
$ setenv SPACK_ROOT /path/to/spack
$ source $SPACK_ROOT/share/spack/setup-env.csh
```

2. Setup spack compilers

```
spack compiler find
```

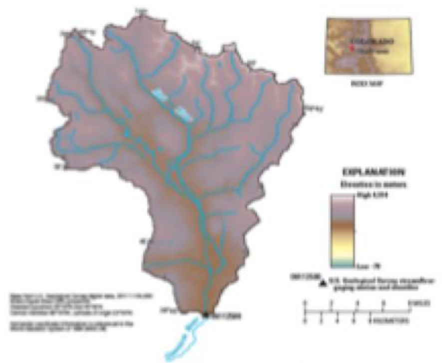
Spack compiler configuration is stored in **$HOME/.spack/$UNAME/compilers.yaml** and can be checked with

```
spack compiler list
```

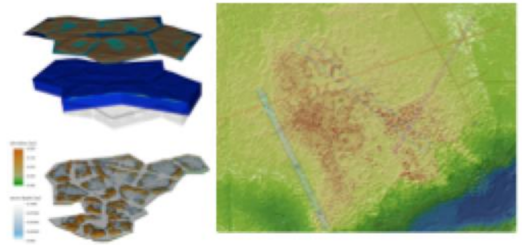

xSDK EXASCALE COMPUTING PROJECT

Application interactions with xSDK

- PFLOTRAN and Alquimia
  - Multiscale & multiphysics modeling of watershed dynamics
  - Provided as part of xSDK
  - Spack script for individual application packages

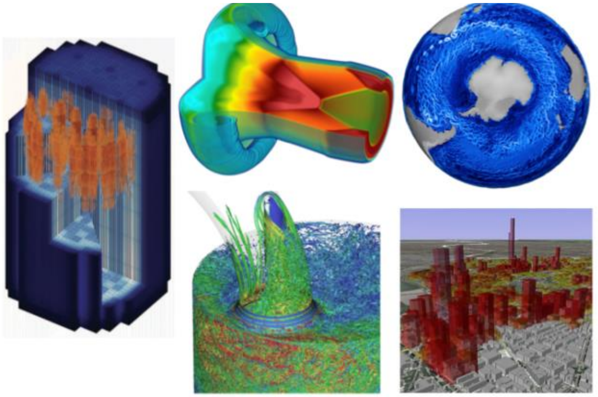


- Nalu in ExaWind
  - Learned about hypre through Trilinos (xSDK Trilinos)



- Laghos in CEED
  - MFEM and hypre
  - Planning to use SuperLU, SUNDIALS and PUMI

- AMPE and Truchas in ExaAM
  - SUNDIALS and hypre
  - Wrote Spack script for AMPE and Truchas

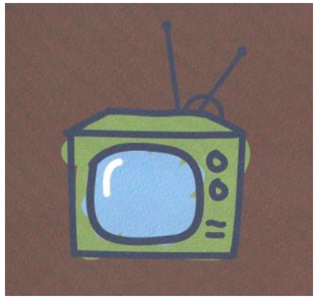


xSDK lessons learned: General observation

- **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 … xSDK: Life is good ☺**

It takes all kinds.



Think outside the box.

Face the bumps with a smile.

The pursuit is the reward.

xSDK lessons learned: Users' perspective

- Building the whole xSDK takes time and produces a very large executable.
  – Future releases should allow building of a subsection.
- Need better documentation for xSDK
- Application developers might use **their own versions** of xSDK libraries.
  – Some capabilities might no longer be supported, but necessary for their applications.
  – It will be important to provide flexibility through the xSDK to allow users to use their own versions of some xSDK libraries.
- xSDK member libraries should also pursue improved compatibilities where possible to **avoid for users to have building their own versions**.
  – New version typically provides improvement performance and interoperability (compilers, and other libraries)

xSDK lessons learned: Developers' perspective

- Requires some code modifications to **eliminate naming conflicts**
  - Namespaces
  - Unique prefix for function names and preprocessor macros
- Maintaining interoperability needs close communication with the developers of other packages
  - Coordination for release scheduling is challenging
- Work toward better, faster, more people-efficient workflow for development and testing is important!
  - Continuous and integrated testing
  - Multiple compilers
  - Multiple parallel runtime setting (OpenMP, CUDA, etc.)

Upcoming xSDK releases for ECP

FY19-FY20: Regular releases of xSDK for ECP

Theme throughout ECP timeframe: Expanding ECP math library capabilities for predictive science: Sustainable coordination and delivery of math libraries across independent development efforts, with enhanced capabilities as needed by ECP applications

- **Additional math packages** compatible with xSDK community policies
- **Deeper multilevel interoperability**, including control inversion and adaptive execution
- Coordination with broader ECP software ecosystem

CASC

Center for Applied
Scientific Computing

Lawrence Livermore
National Laboratory