Introducing the xSDK and Spack

Presented to
ATEPESC 2017 Participants

Lois Curfman McInnes
Senior Computational Scientist
Mathematics and Computer Science Division
Argonne National Laboratory

Q Center, St. Charles, IL (USA)
Date 08/07/2017
IDEAS: Interoperable Design of Extreme-scale Application Software

Project History

• Project began in Sept 2014 as ASCR/BER partnership to improve application software productivity, quality, and sustainability
• Partnership among ANL, LBNL LLNL, LANL, ORNL, PNNL, SNL, CoSM

Resources: https://ideas-productivity.org/resources, featuring

• WhatIs and HowTo docs: concise characterizations & best practices
  • What is Software Configuration?
  • What is CSE Software Testing?
  • What is Good Documentation?
  • How to Configure Software
  • What is Version Control?
  • How to Write Good Documentation
  • How to Add and Improve Testing in a CSE Software Project
  • How to do Version Control with Git in your CSE Project

See ATPESC presentations on Wed, Aug 9, 2017
Motivation: Next-generation modeling requires the combined use of independent packages

Use-case applications (blue boxes) and their present usage of xSDK domain components (orange boxes) and numerical libraries (green boxes). The xSDK presently includes domain components Alquimia (biogeochemistry interface) and PFLOTRAN (subsurface flow) and the numerical libraries hypre, PETSc, SuperLU, and Trilinos. CLM, Chombo, and SUNDIALS (hashed colors) are targeted for later inclusion.

For example: integrated surface-subsurface hydrology modeling
xSDK Vision

Extreme-Scale Science Application
- 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.

Sustainable SW
- Leverage from libs.
- Inform libs.

Domain components
- Reacting flow, etc.
- Reusable.

Libraries
- Solvers, etc.
- Interoperable.

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

SW engineering
- SW policies.
- Best practices.

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

website: xSDK.info
xSDK release 0.2.0: Packages can be readily used in combination by multiphysics, multiscale applications

Opportunities for more components

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

Multiphysics Application C
Application A
Application B

More contributed domain components

HDF5
BLAS
More external software

Alquimia
PFLOTRAN
PETSc
hype
SuperLU
Trilinos

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

Extreme-Scale Scientific Software Development Kit (xSDK)

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
The xSDK is using Spack to deploy its software

• The xSDK packages depend on a number of open source libraries

• Spack is a package manager for HPC

• Spack allows the xSDK to be deployed with a single command
  • User can optionally choose compilers, MPI implementation, and build options
  • Will soon support combinatorial test dashboards for all xSDK packages

Spack has grown into a thriving open source community

  • Over 140 contributors
  • Over 40 organizations
  • Over 1,400 packages
  • Over 75% of package code contributed from outside LLNL

github.com/LLNL/spack
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.

Also specify **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.

**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.

We welcome feedback. What policies make sense for your apps and packages?

https://xsdk.info/policies
Next Steps: xSDK4ECP

xSDK4ECP: Develop community policies and interoperability layers among numerical packages as needed by ECP scientific applications

• Coordinated use of on-node resources
• Integrated execution
  – Control inversion, adaptive execution strategies
• Coordinated and sustainable documentation, testing, packaging, and deployment

Current xSDK packages:

• Numerical libraries: hypre, PETSc, SuperLU, Trilinos
• Domain components: Alquimia, PFLOTRAN

Packages working toward xSDK compatibility:

• Chombo, SUNDIALS, ALExa (AMP, DTK, TASMANIAN)
• Dense linear algebra packages: initial focus: PLASMA, DPLASMA
Join the xSDK community

• We are actively soliciting contributions to the xSDK and feedback on draft xSDK 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
  – Inform potential users that an xSDK member package can be easily used with others
  – Foundation for work on performance portability, deeper levels of package interoperability