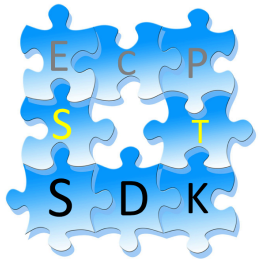
The Layers of CSE Software Sustainability

James Willenbring
Sandia National Laboratories,
North Dakota State University

2019 Collegeville Workshop on
Sustainable Scientific Software
(CW3S19)

July 23, 2019



National Nuclear Security Administration

U.S. DEPARTMENT OF ENERGY | Office of Science

SAND2019-8583 C

Software Sustainability Definitions

- Cost Efficient Maintainability and Evolvability
  -- Sehestedt, et al. [1]
- Capacity of the software to endure
  -- Software Sustainability Institute proposal [2]
- The software will continue to be available in the future, on new platforms, meeting new needs
  -- Daniel Katz [3]

Sustainability Factors

How …

extensible
interoperable
maintainable
portable
reusable
scalable
usable

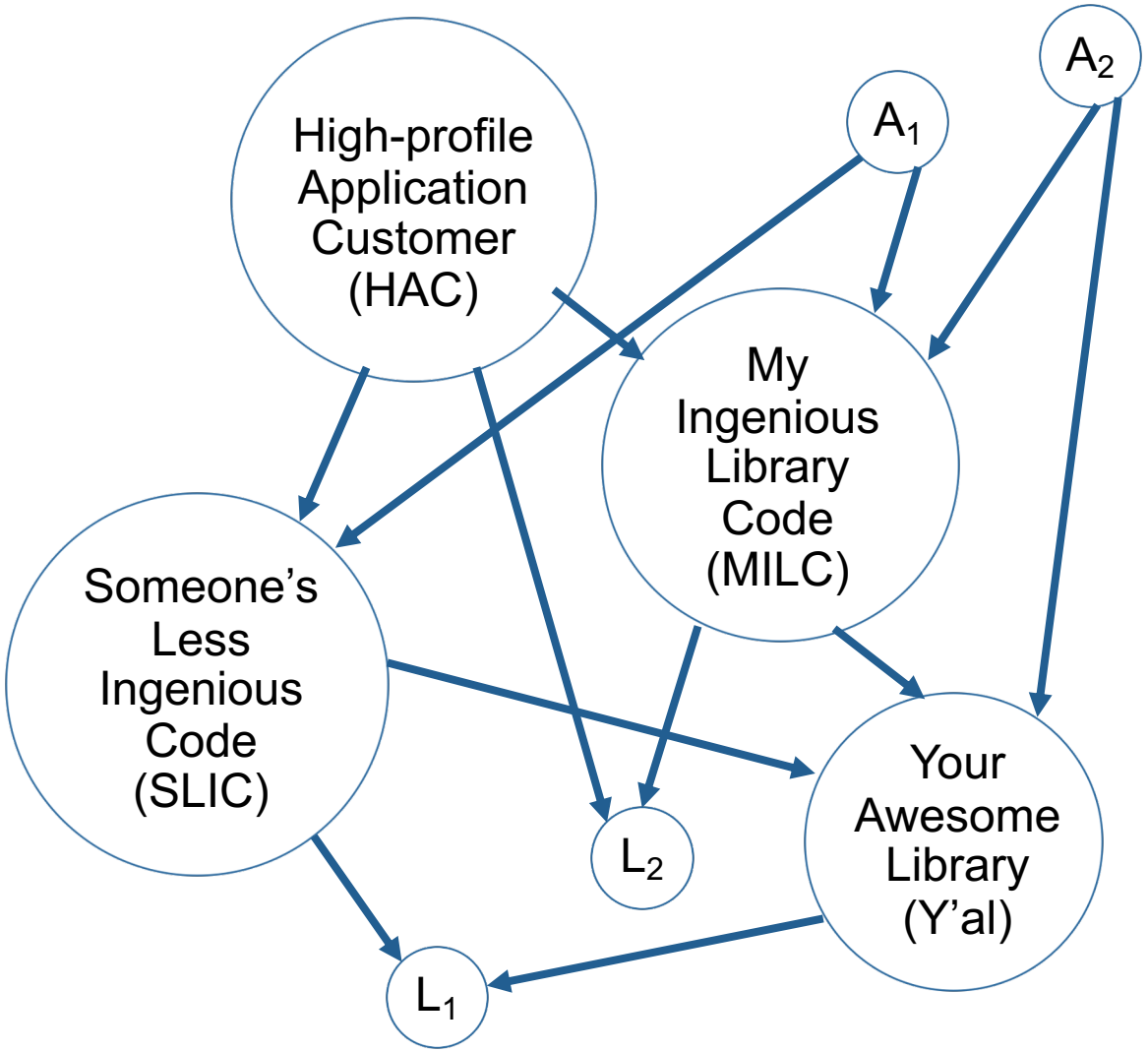
Categories of Software Sustainability

- Intrinsic: Pertaining to characteristics of the software
- Extrinsic: Pertaining to the software development environment

-- Rosado de Souza, et al. [4]

- *Cost Efficient* Maintainability and Evolvability – intrinsic/extrinsic
- *Capacity of the software* to endure - intrinsic
- The software will continue to be available in the future, on new platforms, meeting new needs - neutral

Computational Science and Engineering (CSE) Software Sustainability

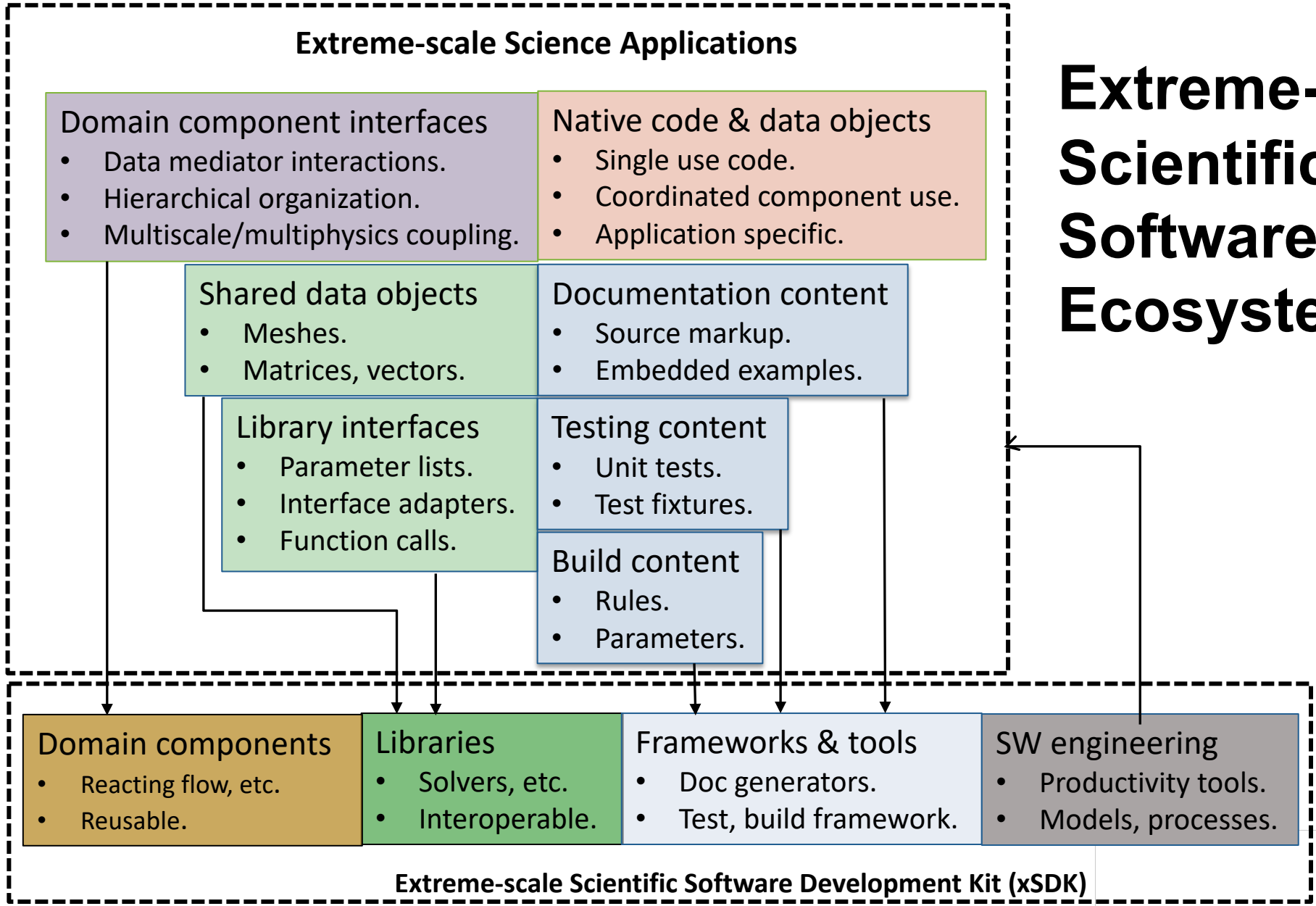


MILC needs to be:

extensible
interoperable
maintainable
portable
reusable
scalable
usable

A quick and dirty proof of concept

Extreme-scale Scientific Software Ecosystem



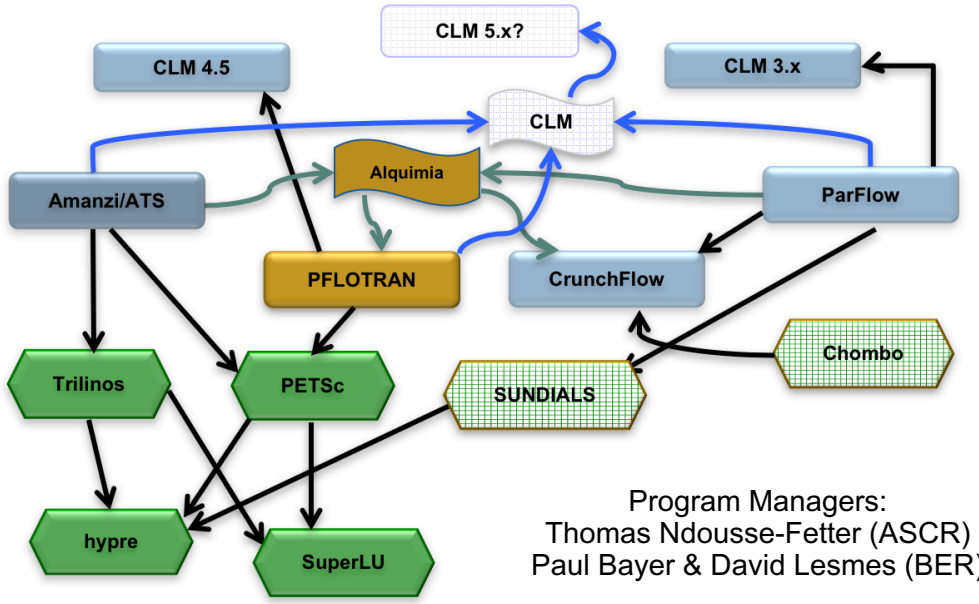
Motivation and history of xSDK

Next-generation scientific simulations require combined use of independent packages

- Installing multiple independent software packages is tedious and error prone
  - Need consistency of compiler (+version, options), 3rd-party packages, etc.
  - Namespace and version conflicts make simultaneous build/link of packages difficult
- Multilayer interoperability among packages requires careful design and sustainable coordination
- **Prior to xSDK effort, could not build required libraries into a single executable due to many incompatibilities**

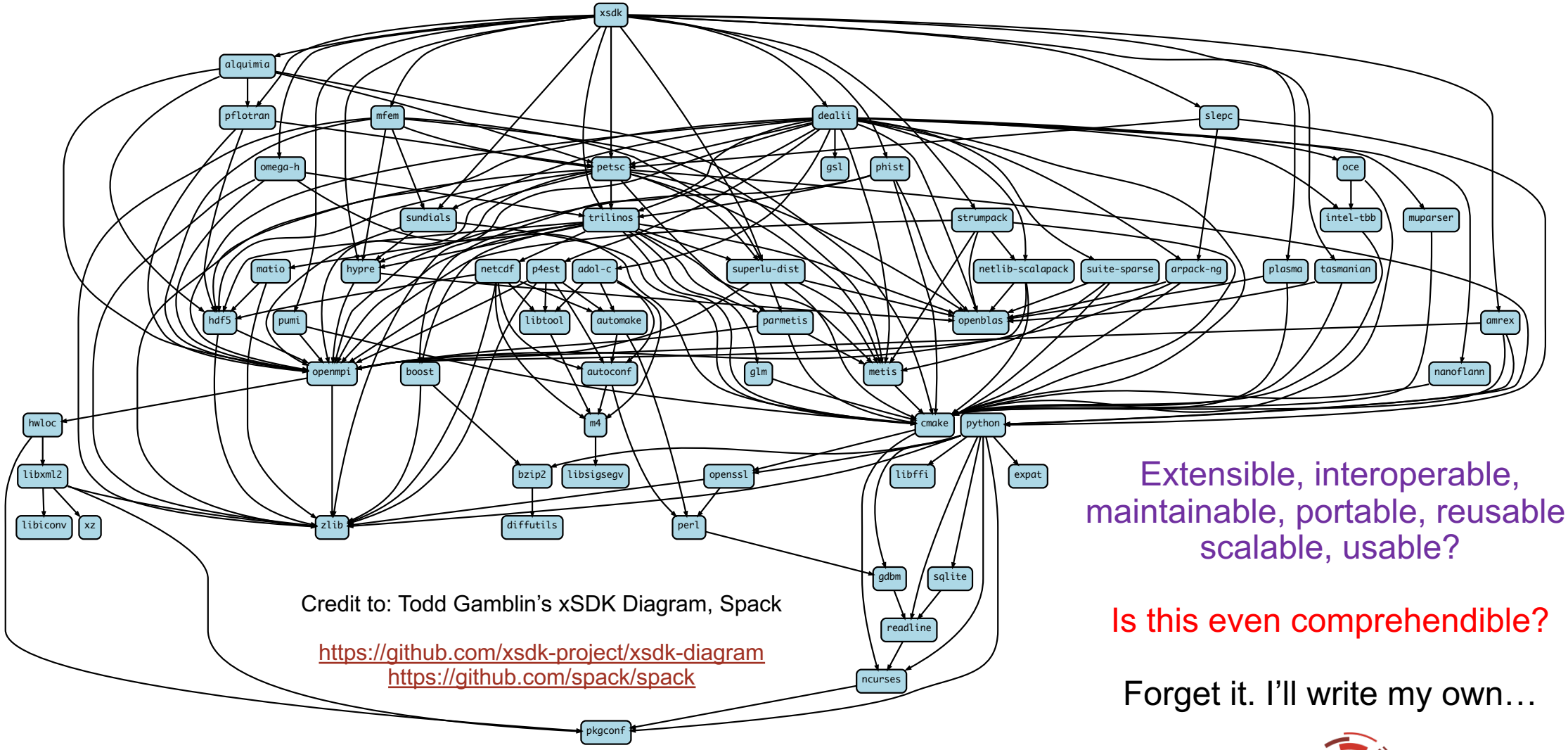
xSDK history: Work began in ASCR/BER partnership, IDEAS project (Sept 2014)

Needed for BER multiscale, multiphysics integrated surface-subsurface hydrology models



Program Managers:
Thomas Ndousse-Fetter (ASCR)
Paul Bayer & David Lesmes (BER)

Complexity of CSE Software Sustainability – the xSDK



Credit to: Todd Gamblin's xSDK Diagram, Spack

https://github.com/xsdk-project/xsdk-diagram
https://github.com/spack/spack

Extensible, interoperable, maintainable, portable, reusable, scalable, usable?

Is this even comprehendible?

Forget it. I'll write my own…

CSE Software Sustainability Break-down

- **Three layers**
  - **First layer**: Those aspects of sustainability relating directly and specifically to the code base and project circumstances, such as staffing, funding, tools, processes, etc.
  - **Second layer:** Sustainability issues related to the (direct and indirect) dependencies of a software project
  - **Third layer:** Concerned with the interoperability of a well-defined ecosystem of software

First Layer of Sustainability – My Project

- Is my project (intrinsically and extrinsically) sustainable?
  - Project team members/leaders typically have a big impact on (but not complete control of) the first layer
    - Software design
    - Software testing
    - Funding
    - Emphasis placed on documentation
    - Coding guidelines
    - Process for committing changes
    - …

1st Layer Metric Categories

- Complexity
- Coupling
- Cohesion
- Size
- Sarkar metrics [5]
- …
- These are useful, but do not capture the complexity of CSE software sustainability

Second Layer of Sustainability – Project Dependencies

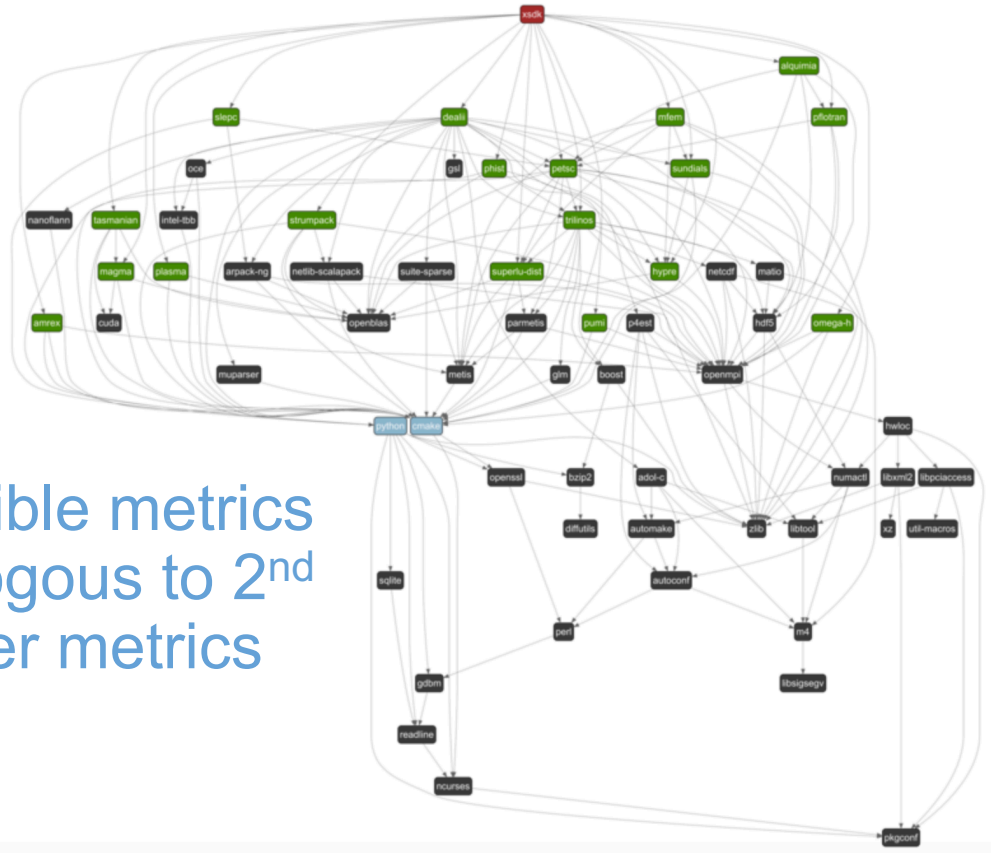
- Can my project “safely” accept a dependence on other pieces of software?
  - Interface stability
  - User support
  - Documentation
  - Funding stability
  - Sustainability of its dependencies
  - “-ability” list
  - …

Possible 2nd Layer Metrics

- What percentage of the CSE-related software dependencies (direct or indirect) for a given software library or application are interoperable with one another and
  - periodically versioned for interoperability? (e.g., through Spack/E4S)
  - regularly tested for continued interoperability? (e.g., through Spack/E4S)
- What percentage of lines of source code of the CSE-related software dependencies (direct or indirect) for a given software library or application are interoperable with one another and
  - periodically versioned for interoperability?
  - regularly tested for continued interoperability?
- What percentage of interface calls to dependency libraries support backward compatibility?
- What percentage of days in the past three months have the dependencies of a given software library or application been interoperable at a development version level?

Third Layer of Sustainability – An Ecosystem of Software

- What set of software products can be used “safely” & interoperably?
  - What packages, features, and interoperability is necessary and useful?
    - Chicken/egg problem
  - Long-term view
- Not all of the software in the ecosystem needs to survive indefinitely
  - Graceful retirement



Possible metrics analogous to 2nd layer metrics

EXASCALE COMPUTING PROJECT

Between the Layers

- Layers feed one another
  - More sustainable packages make for a more sustainable ecosystem
  - Better testing infrastructure and coverage makes it easier to sustain packages
- Need a vocabulary to discuss intricacies of CSE software sustainability.
  - Can these layers or a modified version of them enable those discussions?
- A package that is perfectly first layer sustainable may not be “safe” to use without higher levels of sustainability
  - Unless it depends on no other CSE software, which devastates productivity
  - Higher levels involve a lot of extrinsic factors
- Need ways to quantify sustainability

Sources

- [1] Stephan Sehestedt, Chih-Hong Cheng, and Eric Bouwers. 2014. Towards quantitative metrics for architecture models. In Proceedings of the WICSA 2014 Companion Volume (WICSA '14 Companion). ACM, New York, NY, USA, , Article 5, 4 pages. DOI: http://dx.doi.org/10.1145/2578128.2578226
- [2] Stephen Crouch, Neil Chue Hong, Simon Hettrick, Mike Jackson, Aleksandra Pawlik, Shoaib Sufi, Les Carr, David De Roure, Carole Goble, and Mark Parsons. Nov-Dec 2013. "The Software Sustainability Institute: Changing Research Software Attitudes and Practices," Computing in Science & Engineering , vol.15, no.6, pp.74,80. DOI:10.1109/MCSE.2013.133.
- [3] Daniel Katz. 2016. Defining Software Sustainability. Retrieved from https://danielskatzblog.wordpress.com/2016/09/13/defining-software-sustainability/.
- [4] Mário Rosado de souza, Robert Haines, and Caroline Jay. 2014. Defining Sustainability through Developers' Eyes: Recommendations from an Interview Study. DOI: https://doi.org/10.6084/m9.figshare.1111925.v1
- [5] Santonu Sarkar Avinash C. Kak Girish Maskeri Rama "Metrics for Measuring the Quality of Modularization of Large-Scale Object-Oriented Software" IEEE Transactions on Software Engineering vol. 34 no. 5 Sep–Oct 2008. DOI: 10.1109/TSE.2008.43
- *"Works As Coded." Abbreviations.com.* STANDS4 LLC, 2019. Web. 18 Jul 2019. <https://www.abbreviations.com/Works%20As%20Coded>.