Why effective software practices are essential for CSE projects

Presented at
Better Scientific Software tutorial

SC17, Denver, Colorado

David E. Bernholdt
Oak Ridge National Laboratory

and **Lois Curfman McInnes**
Argonne National Laboratory

NNSA National Nuclear Security Administration
U.S. DEPARTMENT OF ENERGY | Office of Science
IDEAS productivity

License, citation and acknowledgements

License and Citation

- This work is licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
- Requested citation: David E. Bernholdt and Lois Curfman McInnes, Why effective software practices are essential for CSE projects, tutorial, in SC '17: International Conference for High Performance Computing, Networking, Storage and Analysis, Denver, Colorado, 2017. DOI: 10.6084/m9.figshare.5593318.

Acknowledgements

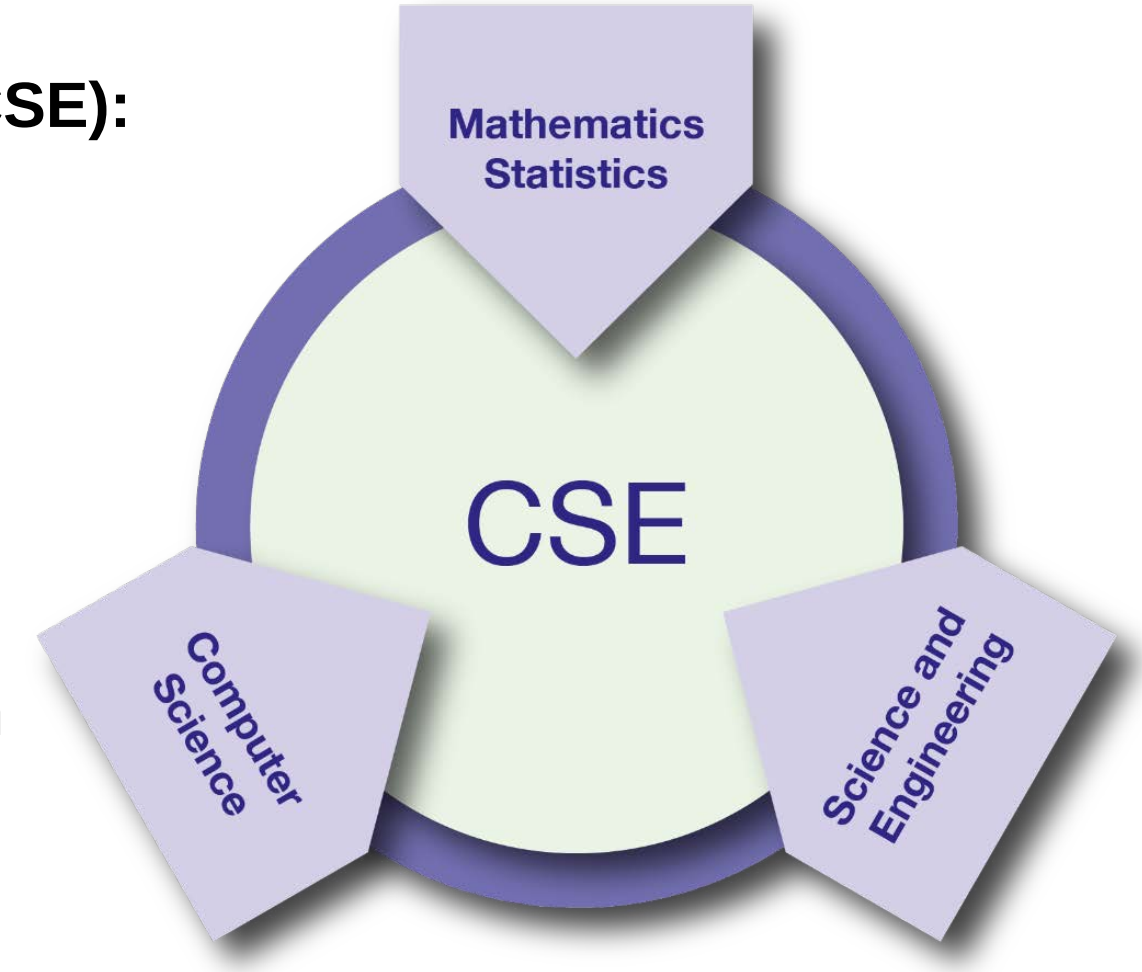
- This work was supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR), and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.
- This work was performed in part at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC for the U.S. Department of Energy under Contract No. DE-AC05-00OR22725 and at the Argonne National Laboratory, which is managed by UChicago Argonne, LLC for the U.S. Department of Energy under Contract No. DE-AC02-06CH11357.

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

What is CSE?

- **Computational Science & Engineering (CSE): 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 apps
- **CSE: essential driver of scientific and technological progress** in conjunction with theory and experiment

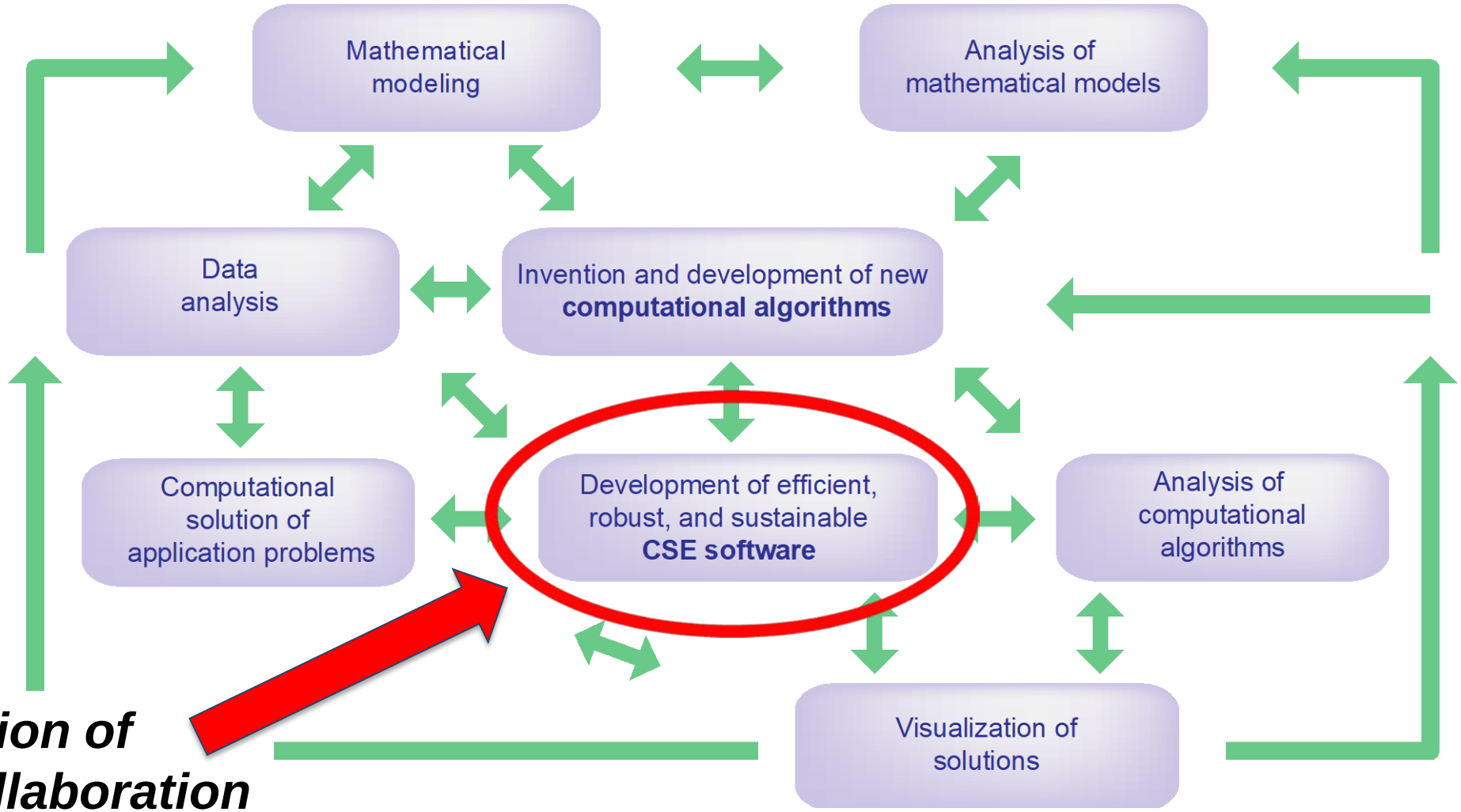


Reference: Research and Education in Computational Science and Engineering, U. Rüde, K. Willcox, L.C. McInnes, H. De Sterck, et al., Oct 2016, https://arxiv.org/abs/1610.02608

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Software is at the core of CSE



Software: foundation of sustained CSE collaboration and scientific progress

Increasing complexity of CSE software

- Multiphysics and multiscale modeling
- Coupling of data analytics
- Disruptive changes in computer hardware
  - Requires algorithm/code refactoring
- Importance of reproducibility
  - Science requirements are unfolding, evolving, not fully known a priori

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

Challenges of CSE software

Technical

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

Sociological

- Competing priorities and incentives
- Limited resources
- Perception of near-term overhead with deferred benefit
- Need for interdisciplinary interactions

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 exisiting 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

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Software process for CSE

Baseline

- **Invest in extensible code design**
  - Most uses need additions and/or customizations
  - Use version 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 on them
- First reaction usually is resistance to change and suspicion of new processes
- Many skeptics get converted when they see the benefit

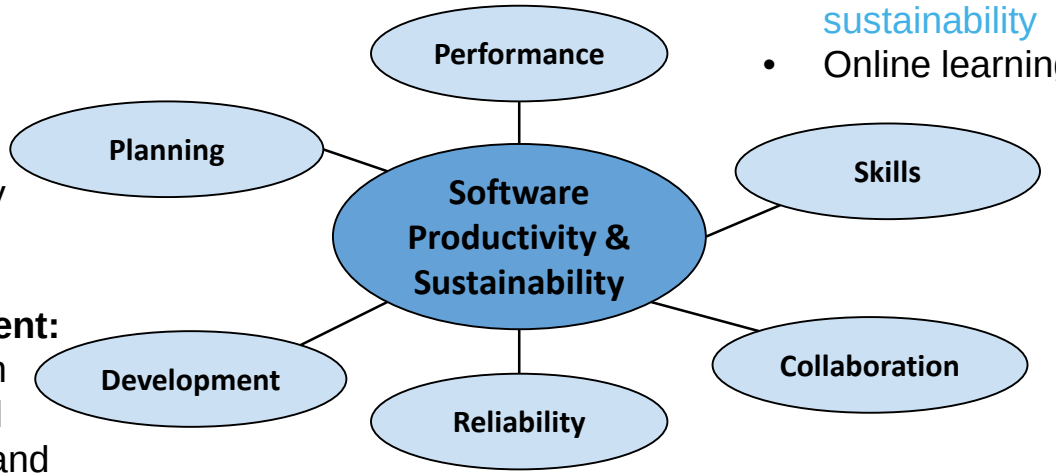
Resources

Key:
Blue text: covered in this tutorial
Black text: pointers to other resources

better scientific software

Better Performance:
- High-performance computing
- Performance at LCFs
- Performance portability

Better Planning:
- Requirements
- Design
- Software interoperability



Better Skills:
- Personal productivity and sustainability
- Online learning

Better Development:
- Documentation
- Version control
- Configuration and builds
- Deployment
- Issue tracking
- Refactoring
- Software engineering
- Development tools

Better Reliability:
- Testing
- Continuous integration testing
- Reproducibility
- Debugging

Better Collaboration:
- Licensing
- Strategies for more effective teams
- Funding sources and programs
- Projects and organizations
- Software publishing and citation
- Discussion forums, Q&A sites

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

IDEAS *WhatIs* and *HowTo* documents

- **Motivation:** Software teams have a wide range of levels of maturity in SW engineering practices.
- **Resources:**
  - ***'What Is'*** docs: 2-page characterizations of important topics for CSE software projects
  - ***'How To'*** docs: brief sketch of best practices
    - Emphasis on ``bite-sized'' topics enables CSE software teams to consider improvements at a small but impactful scale
  - Current topics:
    - *What Is CSE Software Productivity?*
    - *What Is Software Configuration?*
    - *How to Configure Software*
    - *What Is Performance Portability?*
    - *How to Enable Performance Portability*
    - *What Is CSE Software Testing?*
    - *What Are Software Testing Practices?*
    - *How to Add and Improve Testing in a CSE Software Project*
    - *What Is Good Documentation?*
    - *How to Write Good Documentation*
    - *What Are Interoperable Software Libraries?*
    - *What Is Version Control?*
    - *How to Do Version Control with Git*
  - More topics under development
  - See: https://ideas-productivity.org/resources/howtos

Impact: Provide baseline nomenclature and foundation for next steps in software productivity and software engineering for CSE teams.

Other Tutorials: Slides and video

Best Practices for HPC Software Developers

- On-going monthly webinar series
  - https://ideas-productivity.org/events/hpc-best-practices-webinars/
  - Topics to date:
    - *What All Codes Should Do: Overview of Best Practices in HPC Software Development*
    - *Developing, Configuring, Building, and Deploying HPC Software*
    - *Distributed Version Control and Continuous Integration Testing*
    - *Testing and Documenting your Code*
    - *How the HPC Environment is Different from the Desktop (and Why)*
    - *Best Practices for I/O on HPC Systems*
    - *Basic Performance Analysis and Optimization*
    - *Python in HPC*
    - *Intermediate Git*
    - *Using the Roofline Model and Intel Advisor*

Argonne Training Program on Extreme-Scale Computing

- Annual two-week short course
  - https://extremecomputingtraining.anl.gov/
  - **Software Engineering and Community Codes** track (2016) – *6 presentations*
  - **Software Productivity** track (2017)
    - *What All Codes Should Do: Overview of Best Practices in HPC Software Development*
    - *Git Introduction*
    - *Better (Small) Scientific Software Teams*
    - *Improving Reproducibility through Better Software Practices*
    - *Testing and Verification*
    - *Code Coverage and Continuous Integration*
    - *Software Lifecycle with an Example. Community Impact*
    - *An Introduction to Software Licensing*

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

More resources

- **Software Carpentry**: http://software-carpentry.org
  - Since 1998, Software Carpentry has been teaching researchers in science, engineering, medicine, and related disciplines the computing skills they need to get more done in less time and with less pain.
  - Lessons: https://software-carpentry.org/lessons/
    - freely reusable under the Creative Commons Attribution license
- **Software Sustainability Institute**: http://www.software.ac.uk
  - UK national facility for cultivating and improving research software to support world-class research
  - Guides: https://www.software.ac.uk/resources/guides-everything
- **Computational Sci. Stack Exchange**: https://SciComp.StackExchange.com
  - Question and answer site for scientists using computers to solve scientific problems

Agenda

Tutorial evaluation form: http://bit.ly/sc17-eval

Time	Topic	Speaker
8:30am-8:45am	Why effective software practices are essential for CSE projects	David E. Bernholdt, ORNL
8:45am-9:15am	Introduction to software licensing	David E. Bernholdt, ORNL
9:15am-9:45am	Better (small) scientific software teams	Michael A. Heroux, SNL
9:45am-10:00am	Improving Reproducibility Through Better Software Practices	Michael A. Heroux, SNL
10:00am-10:30am	*Break*	
10:30am-10:45am	Testing of HPC Scientific Software: Introduction	Alicia M. Klinvex, SNL
10:45am-11:15am	Verification	Anshu Dubey, ANL
11:15am-11:45am	Evaluating project testing needs	Anshu Dubey, ANL
11:45am-12:00pm	Code coverage demo and CI demo	Alicia M. Klinvex, SNL