CSE Collaboration through Software: Improving Productivity and Sustainability

SIAM CSE17
Atlanta, GA
February 28, 2017

Tutorial slides available at: http://bit.ly/siam-cse17-mt3

Lois Curfman McInnes, Argonne National Laboratory

Tutorial instructors

U.S. DEPARTMENT OF ENERGY
Office of Science

- David Bernholdt, ORNL
- Anshu Dubey, ANL
- Mike Heroux, SNL
- Alicia Klinvex, SNL
- Lois Curfman McInnes, ANL


David


Anshu


Mike


Alicia


Lois

Argonne NATIONAL LABORATORY

Sandia National Laboratories

OAK RIDGE National Laboratory

Members of the IDEAS Scientific Software Productivity Project: www.ideas-productivity.org

- **Focus: Increasing CSE software productivity, quality, and sustainability**

IDEAS productivity

Interoperable Design of Extreme-scale Application Software (IDEAS)

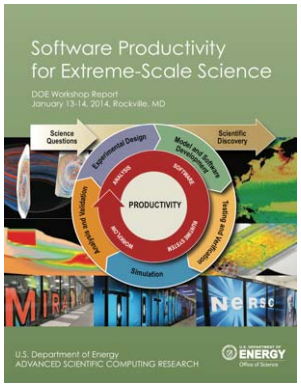
Motivation

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

Objectives

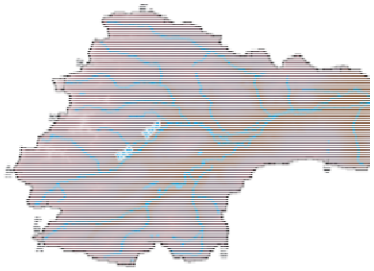
Address confluence of trends in hardware and increasing demands for predictive multiscale, multiphysics simulations.

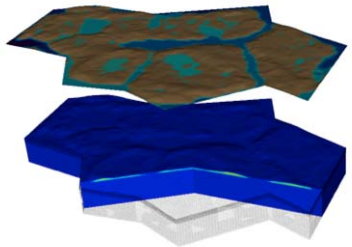
Respond to trend of continuous refactoring with efficient agile software engineering methodologies & improved software design.

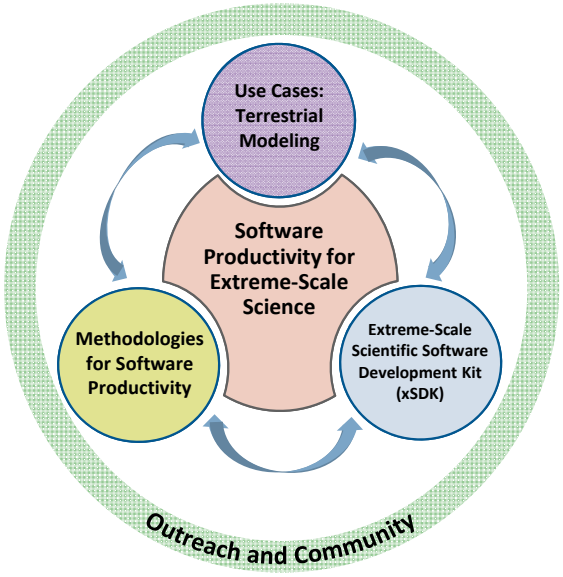


Impact on Applications & Programs

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.

www.ideas-productivity.org

U.S. DEPARTMENT OF ENERGY
Office of Science

Tutorial objectives

Overview of best practices in software engineering explicitly tailored for CSE

- **Why:** Increase CSE software quality, sustainability, productivity
  - Better CSE software > better CSE research > broader CSE impact
- **Who:** Practices relevant for projects of all sizes
  - **emphasis on small teams**, e.g., a faculty member and collaborating students
- **Approach:**
  - Information, examples, exercises, pointers to other resources
  - Not to prescribe any set of practices as "*must use*"
    - Be informative about practices that have worked for some projects
    - Emphasis on adoption of practices that help productivity rather than put unsustainable burden
  - Customize as needed for each project

Outline

Part I: 9:10-10:50 am

- [10 min] **Background, introductions, objectives, setup**
- [15 min] **Why effective software practices are essential for CSE projects**
- [25 min] **Software licensing**
- [50 min] **Effective models, tools, processes, and practices for small teams, including agile workflow management**
  - Interactive exercises: Need Github ID

Part II: 1:30-3:10 pm

- [25 min] **Reproducibility**
- [75 min] **Scientific software testing**
  - Automated testing and continuous integration
  - Interactive exercises for code coverage
    - Access to Linux environment with Git and GNU compiler suite

Tutorial setup: Hands-on

Part I:

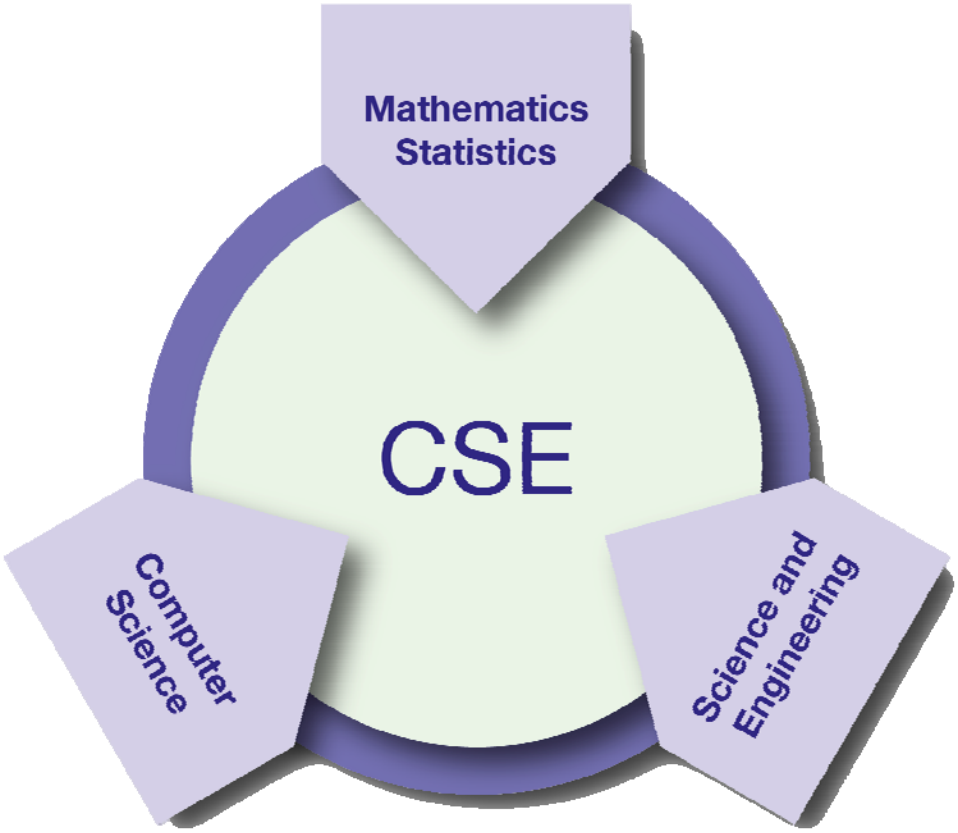
- **Need Github ID**
  - Create a free account at **https://github.com/**
- More Git info: CSE17 tutorial: ***Version Control with Git***, R. LeVeque et al.
  - https://github.com/uwescience/git-tutorials-siamcse2017

Part II:

- Need Git and access to Linux environment with GNU compiler suite

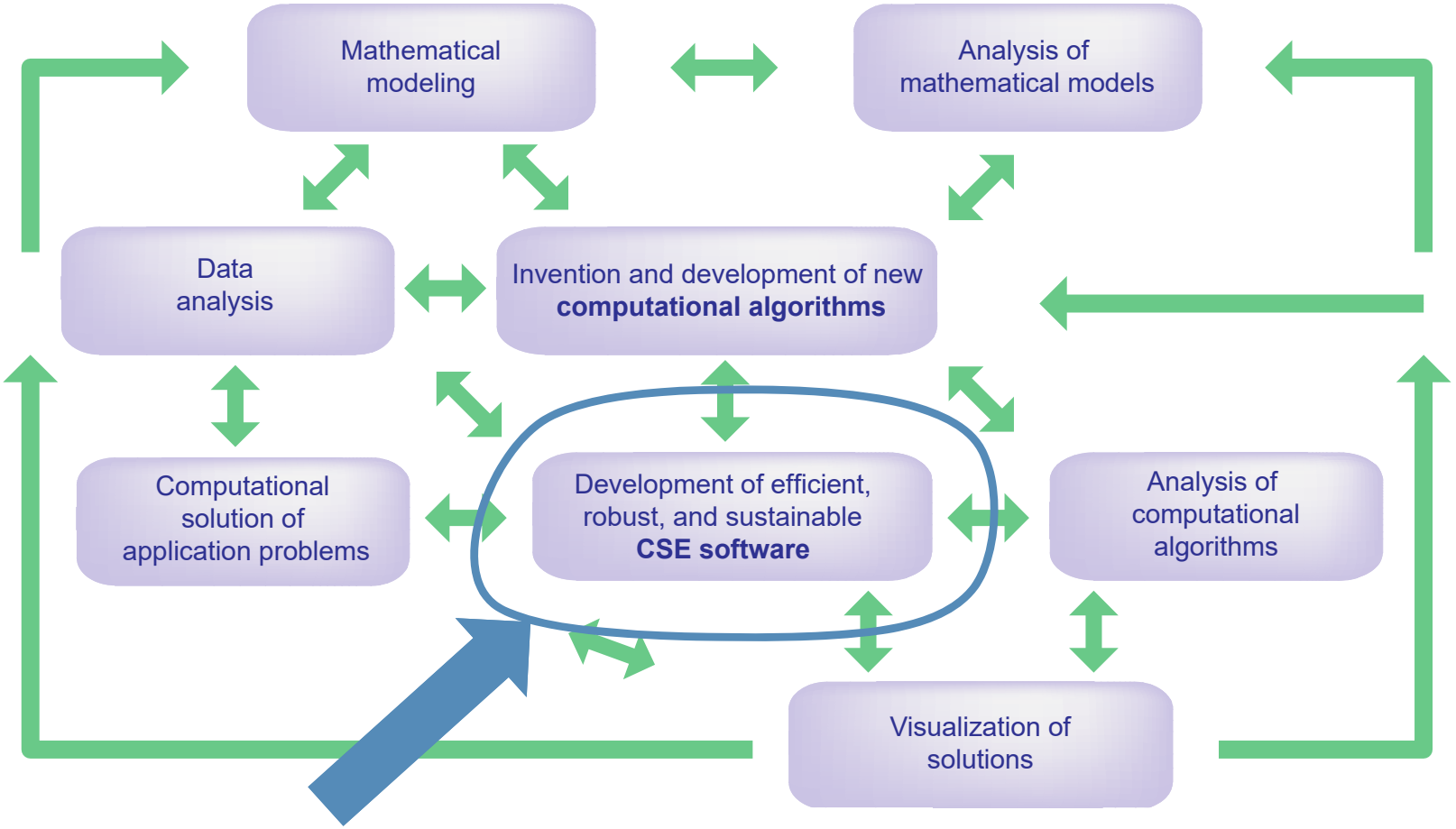
What is CSE?

- **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

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 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 software process**
  - Cognizant of resource constraints

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 none”
- 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
- Many skeptics get converted when they see the benefit
- First reaction usually is resistance to change and suspicion of new processes

Resources

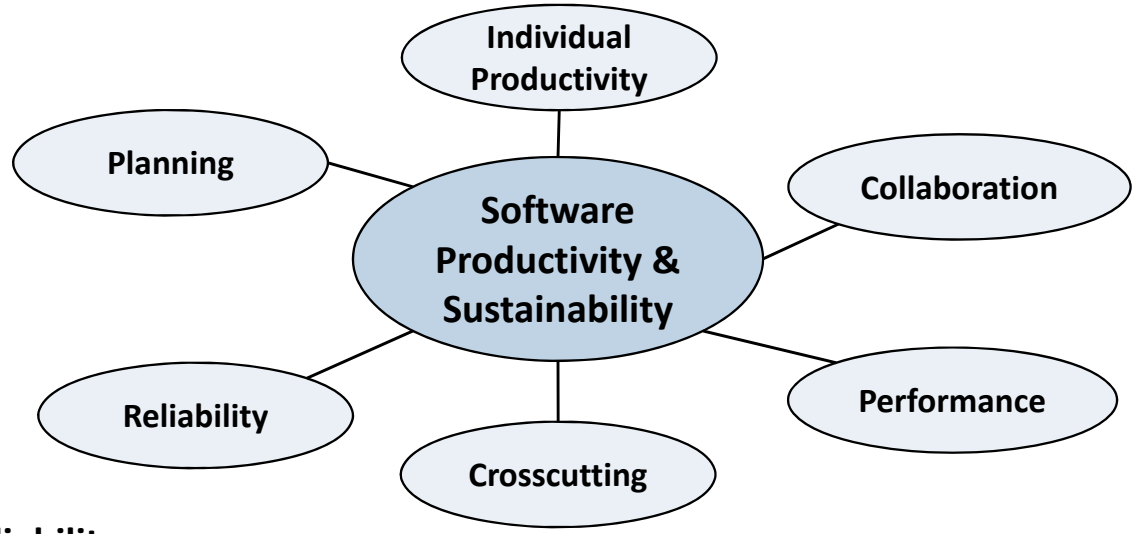
better scientific software

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

Individual Productivity:
- Personal kanban
- Individual learning plans

Collaboration:
- Version control
- Licensing
- Strategies for more effective teams
- Documentation
- Issue tracking

Planning:
- Requirements
- Design
- Development
- Configuration and builds
- Legacy code
- Refactoring



Performance:
- Performance portability
- Software interoperability
- High-performance computing

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

Crosscutting:
- Projects and organizations
- Discussion forums, Q&A sites
- Software publishing and citation
- Funding sources and programs

IDEAS productivity

IDEAS *WhatIs* and *HowTo* docs

- **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.

Tutorials: Slides and video

- **Webinar series: *Best Practices for HPC Software Developers***
  - Presented May – July 2016
  - https://ideas-productivity.org/resources/training-events
    - What All Codes Should Do: Overview of Best Practices in HPC Software Development
    - Developing, Configuring, Buildling, 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)
    - Basic Performance Analysis and Optimization
    - Best Practices for I/O on HPC Systems
- **Argonne Training Program on Extreme-Scale Computing**
  - **Session: *Software Engineering and Community Codes***
  - Presented Aug 8-9, 2016
  - https://extremecomputingtraining.anl.gov/agenda-2016
    - Good Scientific Process Requires Software Engineering Practices
    - Tools for Controlling Change in Your Software
    - Introduction to Make and GNU Autotools
    - Documenting Your Code
    - Testing Your Code
    - Software Refactoring

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 Science Stack Exchange: SciComp.StackExchange.com**
  - Question and answer site for scientists using computers to solve scientific problems

Outline

Part I: 9:10-10:50 am

- [10 min] **Background, introductions, objectives, setup**
- [15 min] **Why effective software practices are essential for CSE projects**
- [25 min] **Software licensing**
- [50 min] **Effective models, tools, processes, and practices for small teams, including agile workflow management**
  - Interactive exercises

Part II: 1:30-3:10 pm

- [25 min] **Reproducibility**
- [75 min] **Scientific software testing**
  - Automated testing and continuous integration
  - Interactive exercises for code coverage
    - Access to Linux environment with Git and GNU compiler suite