Welcome to…

better scientific software

Anshu Dubey and Michael Heroux

Sunday 24th June 2018

Final version of tutorial slides:

Requested citation: Anshu Dubey, Michael Heroux, Better Scientific Software, tutorial, in ISC High Performance 2018

DOI:10.6084/m9.figshare.6452912

Tutorial Instructors

- Anshu Dubey, ANL
- Mike Heroux, SNL

Anshu

Mike

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

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

License, citation and acknowledgements

- Requested citation: Requested citation: Anshu Dubey, Michael Heroux, Better Scientific Software, tutorial, in ISC High Performance 2018 DOI:10.6084/m9.figshare.6452912

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 Sandia National Laboratories. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND NO SAND2017-5474 PE

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

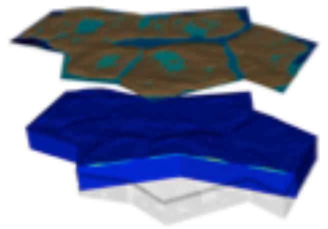
Address confluence of trends in hardware and increasing demands for predictive multiscale, multiphysics simulations.
Respond to trend of continuous refactoring with efficient agile software engineering methodologies & improved software design.

Project History

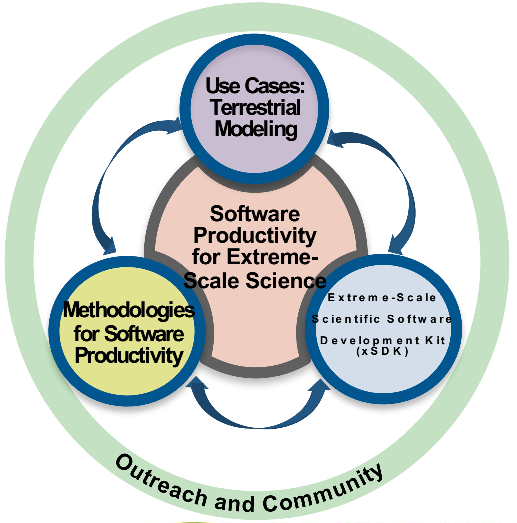
IDEAS began in 2014 as a DOE ASRC/BER partnership to improve application software productivity, quality, and sustainability. In 2017, the DOE Exascale Computing Project began supporting IDEAS to help application teams improve developer productivity and software sustainability while making major changes for exascale.

Impact on Applications & Programs

Terrestrial ecosystem use cases tied initial IDEAS activities to programs in DOE Biological and Environmental Research (BER). The Exascale Computing Project (ECP) supports a broad portfolio of applications furthering science, energy, national security, and economic competitiveness.



Approach


Use Cases: Terrestrial Modeling
Software Productivity for Extreme-Scale Science
Methodologies for Software Productivity
Extreme-Scale Scientific Software Development Kit (xSDK)
Outreach and Community

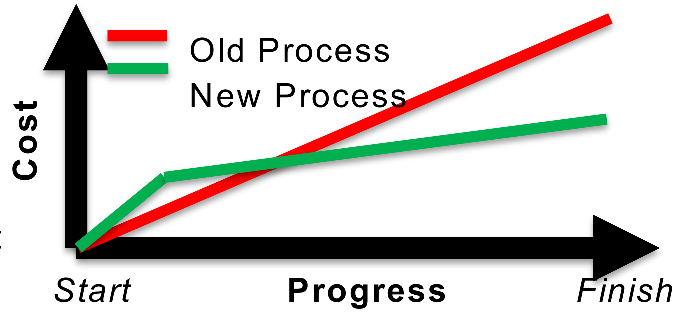
Interdisciplinary multi-institutional team (ANL, LANL, LBNL, LLNL, ORNL, PNNL, SNL, U. Oregon) with broad experience in scientific software development

Close partnerships with applications teams ensures impact on science

Identification, documentation and dissemination of **best practices** for BER and ECP software teams and the broader community

Catalyzing **software process improvements** through tailored engagement with individual projects

Working to bend the curve of software development costs downwards



www.ideas-productivity.org

Tutorial objectives

Overview of best practices in software engineering explicitly tailored for CSE

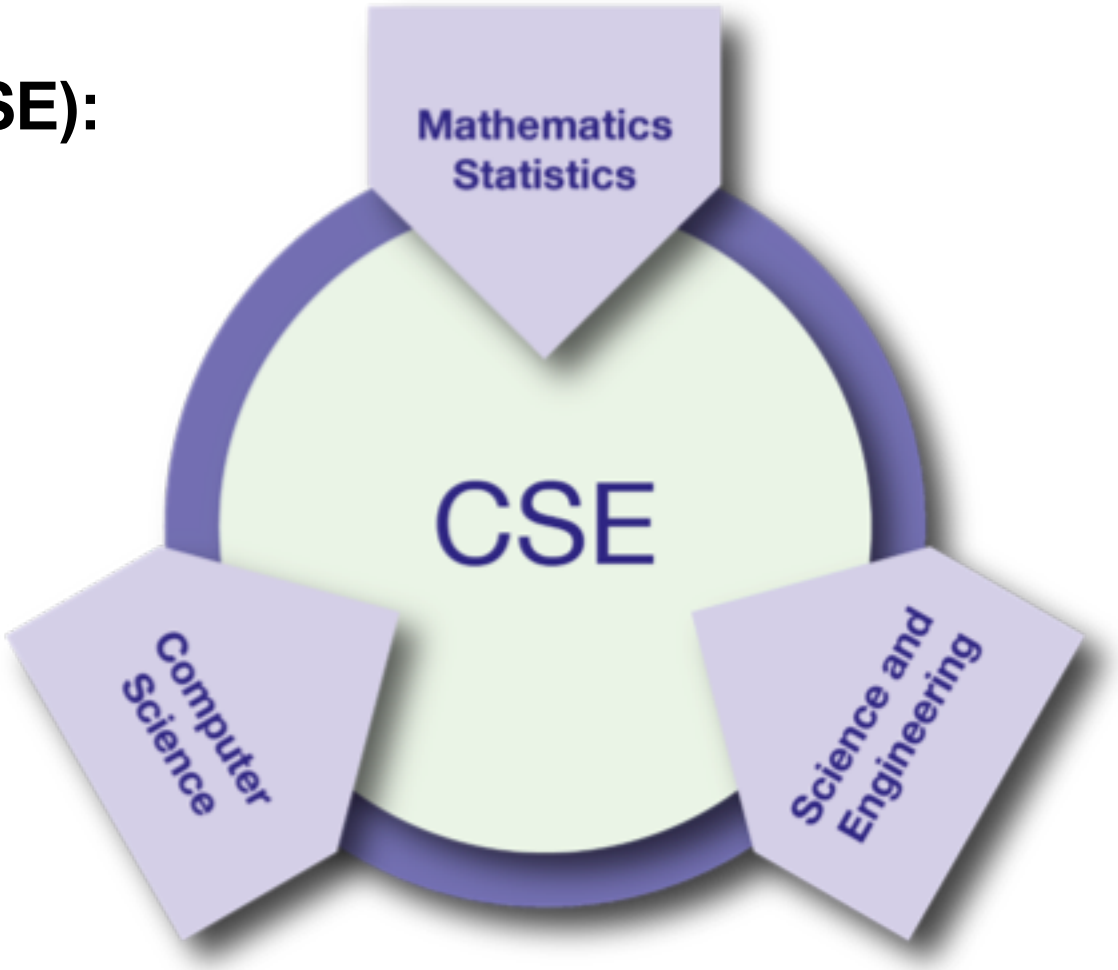
- **Why:** Increase CSE software quality, sustainability, productivity
  - Better CSE software > better CSE research > broader CSE impact
- **Who:** Practices relevant for projects of all sizes
  - **emphasis on small teams**, e.g., a faculty member and collaborating students
- **Approach:**
  - Information, 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

Agenda

Time	Topic	Speaker
2:00pm-2:30pm	Why Effective Software Practices are Essential for CSE Projects	Anshu Dubey, ANL
2:30pm-3:00pm	Introduction to Software Licensing	Michael A. Heroux, SNL
3:00am-3:30pm	Better (Small) Scientific Software Teams	Michael A. Heroux, SNL
3:30am-4:00pm	Improving Reproducibility Through Better Software Practices	Michael A. Heroux, SNL
4:00pm-4:30pm	*Break*	
4:30pm-5:00pm	Testing HPC Scientific Software – Part 1	Anshu Dubey, ANL
5:00pm-5:30pm	Testing HPC Scientific Software – Part 2	Anshu Dubey, ANL
5:30pm-6:00pm	Code Coverage Hands-on and CI Demo	Anshu Dubey, ANL

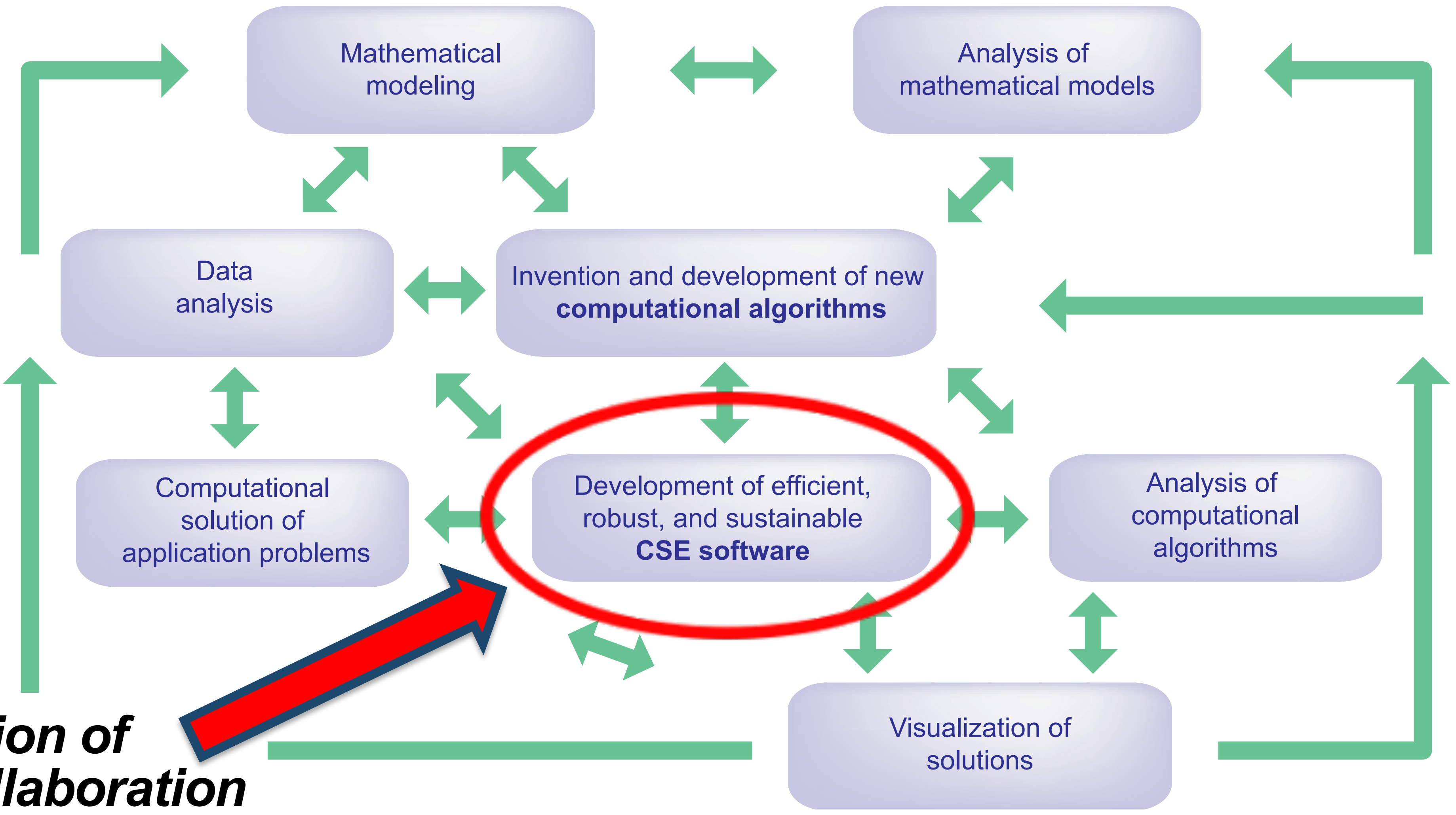
What is CSE?

- **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


Mathematical modeling
Analysis of mathematical models
Data analysis
Invention and development of new **computational algorithms**
Computational solution of application problems
Development of efficient, robust, and sustainable **CSE software**
Analysis of computational algorithms
Visualization of solutions

Software: foundation of sustained CSE collaboration and scientific progress

Heroic Programming

Usually a pejorative term, is used to describe the expenditure of huge amounts of (coding) effort by talented people to overcome shortcomings in process, project management, scheduling, architecture or any other shortfalls in the execution of a software development project in order to complete it. Heroic Programming is often the only course of action left when poor planning, insufficient funds, and impractical schedules leave a project stranded and unlikely to complete successfully.

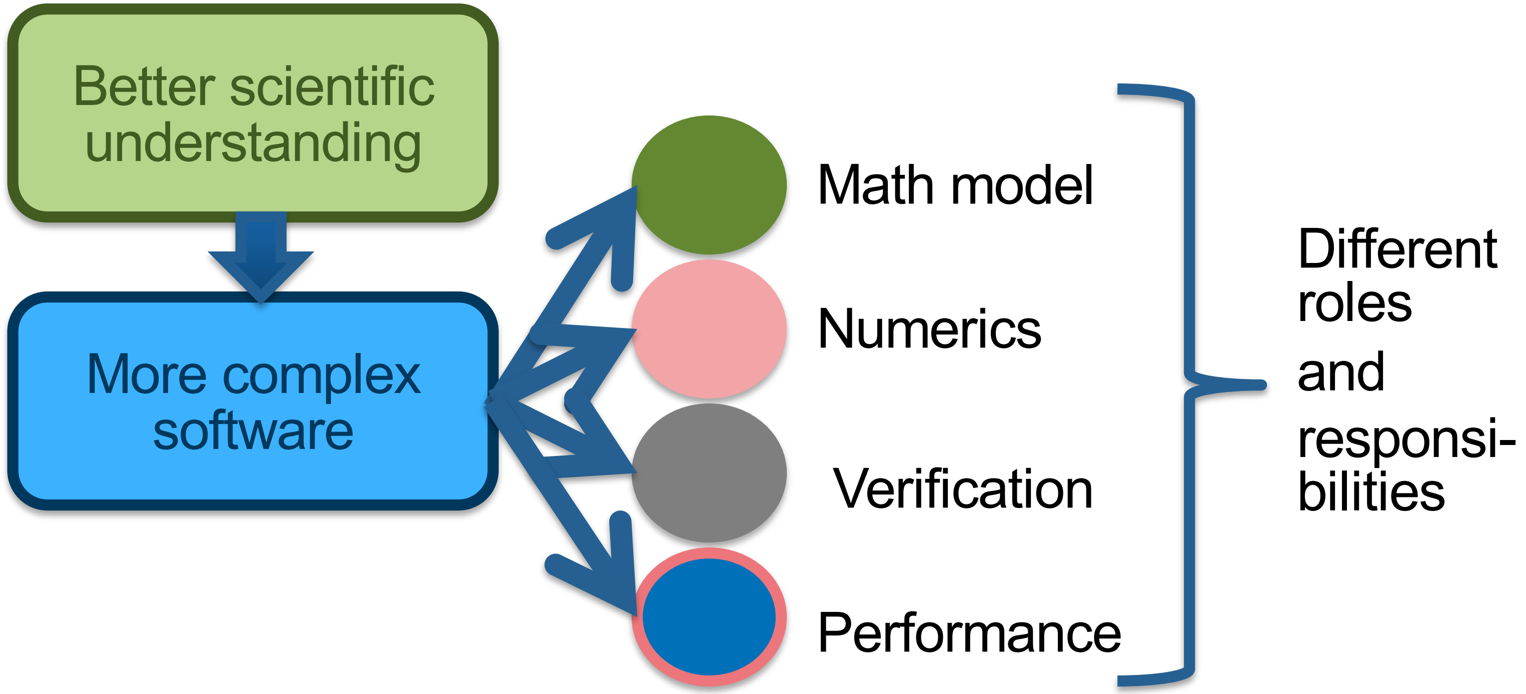
From http://c2.com/cgi/wiki?HeroicProgramming

Science teams often resemble heroic programming

Many do not see anything wrong with that approach

What is wrong with heroic programming

Scientific results that could be obtained with heroic programming have run their course, because:


Better scientific understanding
More complex software
Math model
Numerics
Verification
Performance
Different roles and responsi-bilities

It is not possible for a single person to take on all these roles

In High Performance Computing Science

- Codes aiming for higher fidelity modeling
  - More complex codes, simulations and analysis
  - Numerous models, more moving parts that need to interoperate
  - Variety of expertise needed – the only tractable development model is through separation of concerns
  - **It is more difficult to work on the same software in different roles without a software engineering process**
- Onset of higher platform heterogeneity
  - Requirements are unfolding, not known apriori
  - **The only safeguard is investing in flexible design and robust software engineering process**

Other reasons

Accretion leads to unmanageable software

- Increases cost of maintenance
- Parts of software may become unusable over time
- Inadequately verified software produces questionable results
- Increases ramp-on time for new developers
- Reduces software and science productivity due to technical debt

consequence of choices – quick and dirty incurs technical debt, collects interest which means more effort required to add features.

Challenges

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 without 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

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

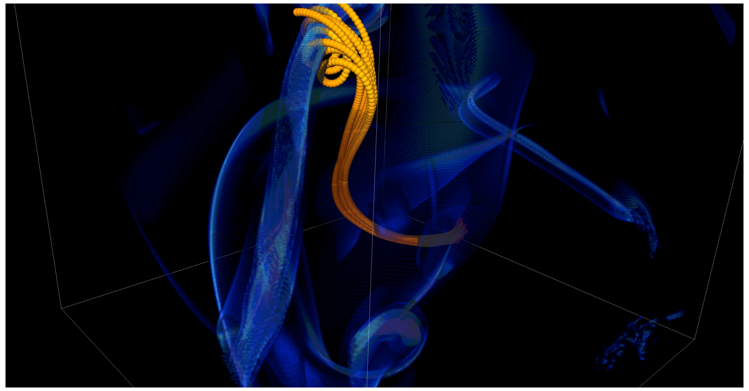
Desirable

- 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

What can happen without software process FLASH experience



- In 2005 BG/L was made available at short notice
- Quick and dirty development of particles
- Many in-flight corrections of defects
- One was adding tags to track individual particles
  - **Got many duplicated tags due to round-off**
- Had to develop post-processing tools to correctly identify trajectories

FLASH had a software process in place. It was tested regularly. This was one instance when the full process could not be applied because of time constraints. We got ready for the run in less than a month, the run went for 1.5 weeks, and it took over 6 months before we could trust the processed results.

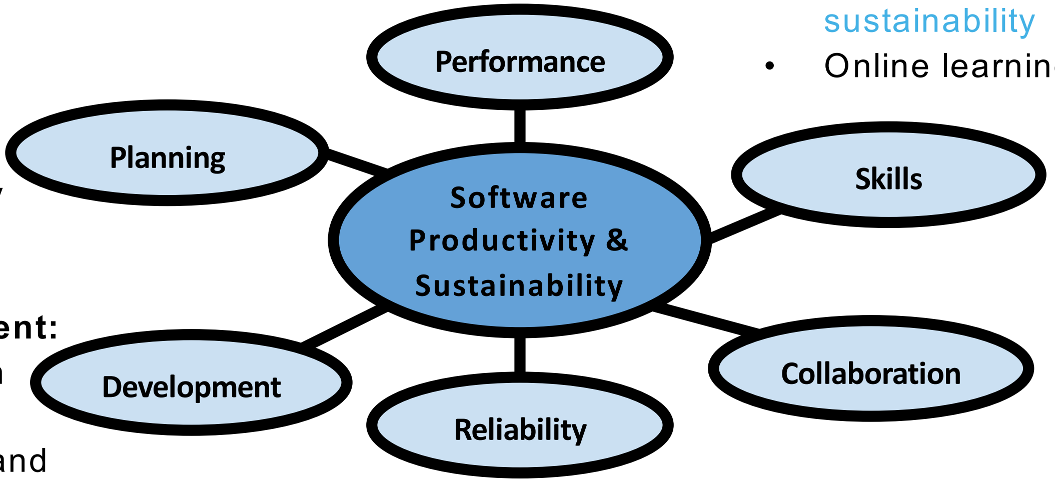
Resources

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

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

Better Skills:
- Personal productivity and sustainability
- Online learning

Better Planning:
- Requirements
- Design
- Software interoperability



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

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

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

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

What Is Performance Portability for CSE Applications?

IDEAS productivity

[illegible]

How to Enable Performance Portability for CSE Applications

IDEAS productivity

[illegible]

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/
  - Selected Topics:
    - *On-demand Learning for Better Scientific Software: How to Use Resources & Technology to Optimize your Productivity*
    - *Software Citation Today and Tomorrow*
    - *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*
    - *Jupyter and HPC: Current State and Future Roadmap*
    - *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 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

Time	Topic	Speaker
2:00pm-2:30pm	Why Effective Software Practices are Essential for CSE Projects	Anshu Dubey, ANL
2:30pm-3:00pm	Introduction to Software Licensing	Michael A. Heroux, SNL
3:00am-3:30pm	Better (small) Scientific Software Teams	Michael A. Heroux, SNL
3:30am-4:00pm	Improving Reproducibility Through Better Software Practices	Michael A. Heroux, SNL
4:00pm-4:30pm	*Break*	
4:30pm-5:00pm	Testing HPC Scientific Software – Part 1	Anshu Dubey, ANL
5:00pm-5:30pm	Testing HPC Scientific Software – Part 2	Anshu Dubey, ANL
5:30pm-6:00pm	Code Coverage Hands-on and CI Demo	Anshu Dubey, ANL

Dr. Louis Tao
Center for Bioinformatics
School of Life Sciences
Peking University

Dear Louis,

I would like to invite you to visit the University of Chicago at some
time during the period of August 6th – 14th this year. During your
visit, I hope we will be able to start a collaboration on large-scale
networks.

I am looking forward to your visit.

Yours sincerely,

Anshu Dubey