Motivation and Overview of Best Practices in HPC Software Development

David E. Bernholdt
Oak Ridge National Laboratory

Anshu Dubey, Katherine M. Riley
Argonne National Laboratory

Better Scientific Software Tutorial, ISS, March 2021

License, Citation and Acknowledgements

License and Citation

- **The requested citation the overall tutorial is: David E. Bernholdt, Anshu Dubey, Rinku K. Gupta, and David M. Rogers, Better Scientific Software tutorial, in Improving Scientific Software conference, online, 2021. DOI: 10.6084/m9.figshare.14256257**
- Individual modules may be cited as *Speaker, Module Title*, in Better Scientific Software tutorial…

Acknowledgements

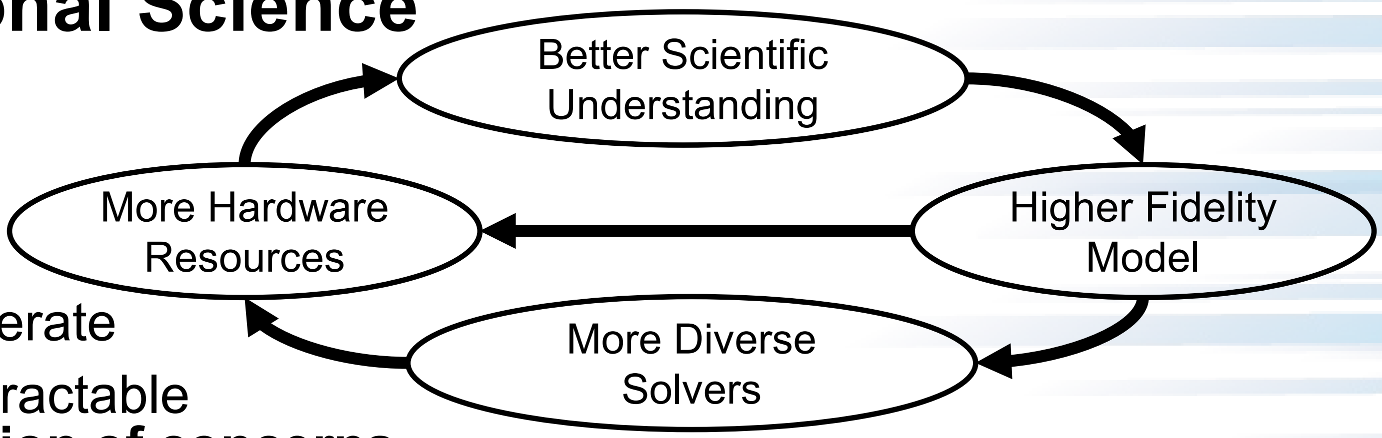
- Additional contributors include: Mike Heroux, Alicia Klinvex, Mark Miller, Jared O'Neal, Katherine Riley, David Rogers, Deborah Stevens, James Willenbring
- This work was supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR), and by the 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 Argonne National Laboratory, which is managed by UChicago Argonne, LLC for the U.S. Department of Energy under Contract No. DE-AC02-06CH11357.
- 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.
- This work was performed in part at the Lawrence Livermore National Laboratory, which is managed by Lawrence Livermore National Security, LLC for the U.S. Department of Energy under Contract No. DE-AC52-07NA27344.
- This work was performed in part at the Los Alamos National Laboratory, which is managed by Triad National Security, LLC for the U.S. Department of Energy under Contract No.89233218CNA000001
- 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.

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Science through computing is,
at best,
as credible as the software that produces it!

The Success of Computational Science Creates the Challenges of Computational Science

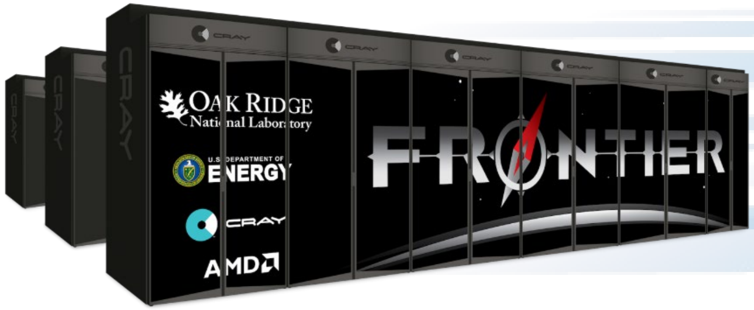


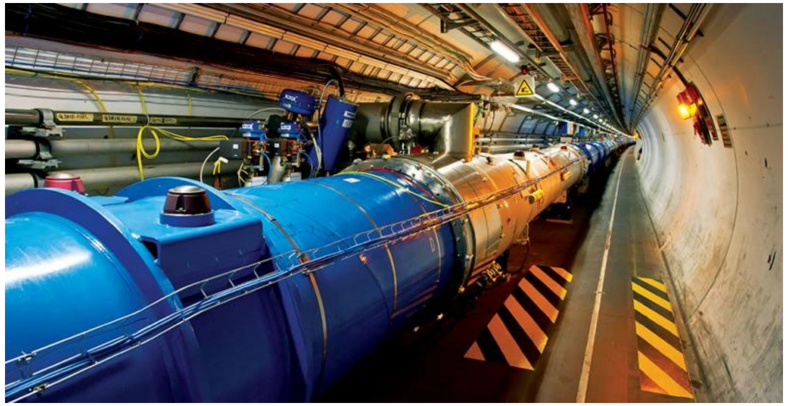
- Positive feedback loop
  - More complex codes, simulations and analysis
  - More moving parts that need to interoperate
  - Variety of expertise needed – the only tractable development model is through **separation of concerns**
  - **It is more difficult to work on the same software in different roles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

Supercomputers change fast
Especially now!

Scientific Facilities Provide Valuable Resources

- Major supercomputers often cost O($100M)
- All cost millions more to operate, annually
- Significant allocations on large supercomputers can be worth millions
- Even if you don’t pay the $ you have to spend the time and effort to get the allocation
- **<u>Sponsors’ concern</u>: Are you being a good steward of the resources?**
- **<u>Your concern</u>: Are you getting the most science possible out of your work (aka scientific productivity)?**





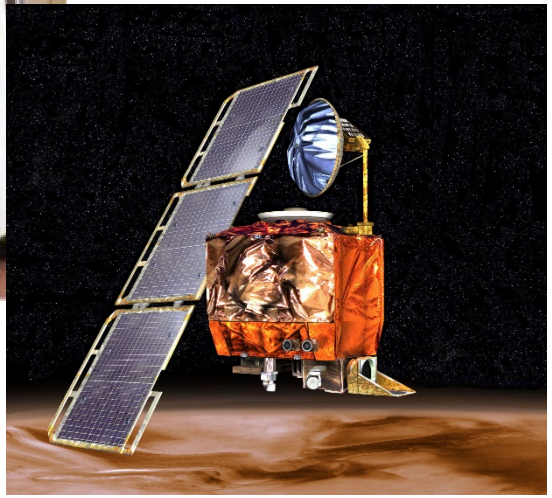
IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

High-Consequence Software-Related Scientific Failures

Therac-25 (1985-1987)

- Computer-controlled radiation therapy system
- **Poor software design, development and testing practices** allowed flaws that let to at least six cases of substantial radiation overdoses, three fatal

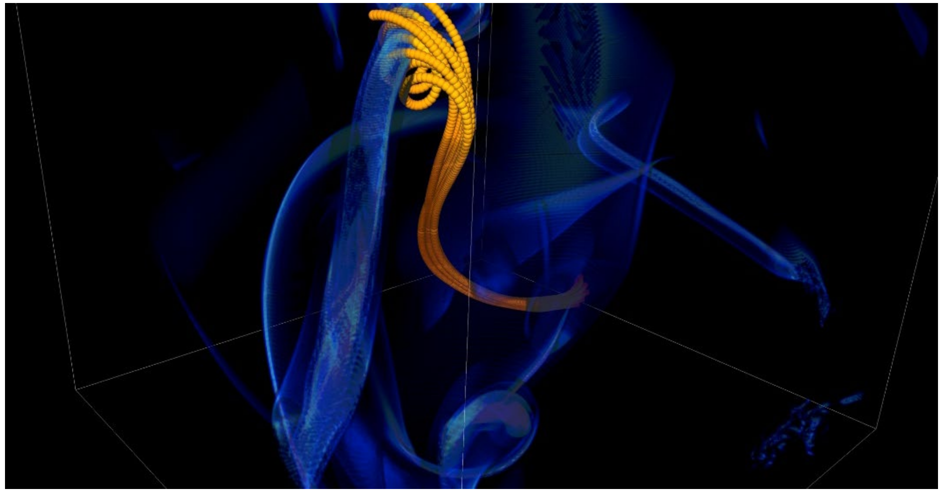


Mars Climate Orbiter (1999)

- Incorrect trajectory adjustment caused loss of the orbiter as it was supposed to enter Martian orbit
- Discrepancy in the units used in two different software components
- One component **didn’t follow specifications**
- **Inadequate testing** at the interface
- Concerns raised earlier in the mission were ignored because they **weren’t properly documented**

Just two of many examples

More Subtle Impacts on Scientific Productivity



- In 2005, the FLASH astrophysics team was offered a unique opportunity to access one of the biggest machines in the world at that time (BG/L) for a dedicated run
- Short notice to prepare
  - **< 1month to get ready for 1.5 week run**
- Quick and dirty development of particle capability in code
- Error in tracking particles resulted in duplicated tags from round-off
- Had to develop post-processing tools to correctly identify trajectories
  - **6 months to process results**

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.

Technical Debt

> The cost implied cost of additional rework caused by choosing an easy (limited) solution now instead of using a better approach that would take longer.
> -- Wikipedia

Like monetary debt, the more you accumulate, the harder it is to pay off

- Increases cost of maintenance
- Parts of software may become unusable over time
- Inadequately verified software produces questionable results
- Increases ramp-up time for new developers
- **Overall, reduces software and science productivity**

Challenges Developing Scientific Applications Today

Technical

- All parts of the model and software system 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
- Increasing architectural diversity

Sociological

- Competing priorities and incentives
  - Sponsors often care more about scientific publications than software per se
- Limited resources
- Need for interdisciplinary interactions
  - Many different kinds of expertise to be successful

Good scientific process requires good software practices

Good software practices increase software sustainability

Good software practices increase scientific productivity

Software sustainability increases scientific productivity

Best Practices for Scientific Software Development

Baseline

- Invest in extensible code design
- Use version control and automated testing
- Institute a rigorous verification and validation regime
- Define and enforce 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

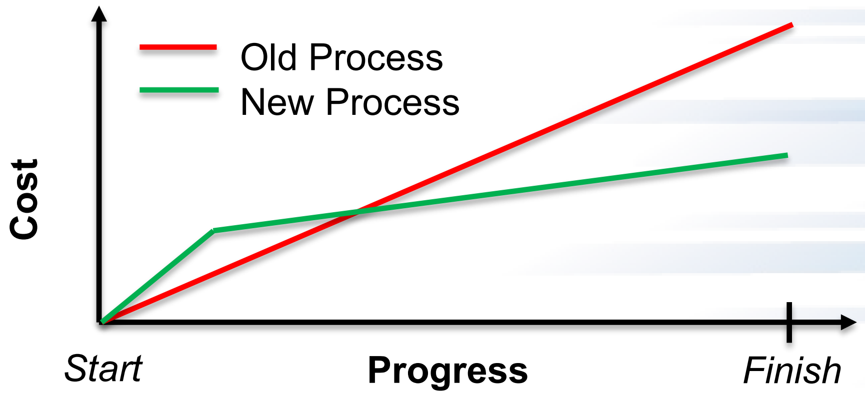
This tutorial will focus primarily on scientific software as distinct from more generic software engineering best practices

Continual, Incremental Software Process Improvement

Target: your project should include “just enough” software engineering so that you can meet your short-term and longer-term scientific goals effectively

1. Identify your team’s “pain points” in your software development processes
2. Set a goal for something to improve
   - Target processes and behaviors, not just tasks
   - Pick something that you can address in a few months that will give you a noticeable benefit
3. Agree on a plan to address it, identify markers of progress and what is “done”
   - Write them down
4. Work your plan, track your progress
5. When you are done, celebrate…

…then pick a new pain point to address



The new process costs something to implement, but it pays off over time

Productivity and Sustainability Improvement Planning
https://bssw.io/psip

Agenda

Time (MDT)	Module	Topic	Speaker
1:00pm-1:05pm	00	Introduction	David E. Bernholdt, ORNL
1:05pm-1:15pm	01	Motivation and Overview of Best Practices in HPC Software Development	David E. Bernholdt, ORNL
1:15pm-1:45pm	02	Agile Methodologies	Rinku K. Gupta, ANL
1:45pm-2:00pm	03	Git Workflows	Rinku K. Gupta, ANL
2:00pm-2:20pm	04	Software Testing 1	David M. Rogers, ORNL
2:20pm-2:40pm		*Break (optional Q&A)*	*All*
2:40pm-3:00pm	05	Software Design	Anshu Dubey, ANL
3:00pm-3:15pm	06	Software Testing 2	David M. Rogers
3:15pm-3:40pm	07	Refactoring	Anshu Dubey, ANL
3:40pm-3:55pm	08	Reproducibility	David E. Bernholdt, ORNL
3:55pm-4:00pm	09	Summary	David E. Bernholdt, ORNL