EXASCALE COMPUTING PROJECT

Best Practices and Sustainability in HPC Software Development

ATPESC 2020

Katherine Riley

Director of Science, Argonne Leadership Computing Facility
Argonne National Laboratory

August 6, 2020

License, Citation and Acknowledgements

License and Citation

- **The requested citation the overall tutorial is: David Bernholdt, Software Productivity Track, in Argonne Training Program for Extreme Scale Computing 2020. DOI:** Individual modules may be cited as *Speaker, Module Title*, in Software Productivity Track, ATPESC 2020

Acknowledgements

- Additional contributors to this this tutorial include: Anshu Dubey, Katherine Riley, James M. Willenbring, Mark Miller, Mike Heroux, Alicia Klinvex, and Jared O'Neal,
- 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 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

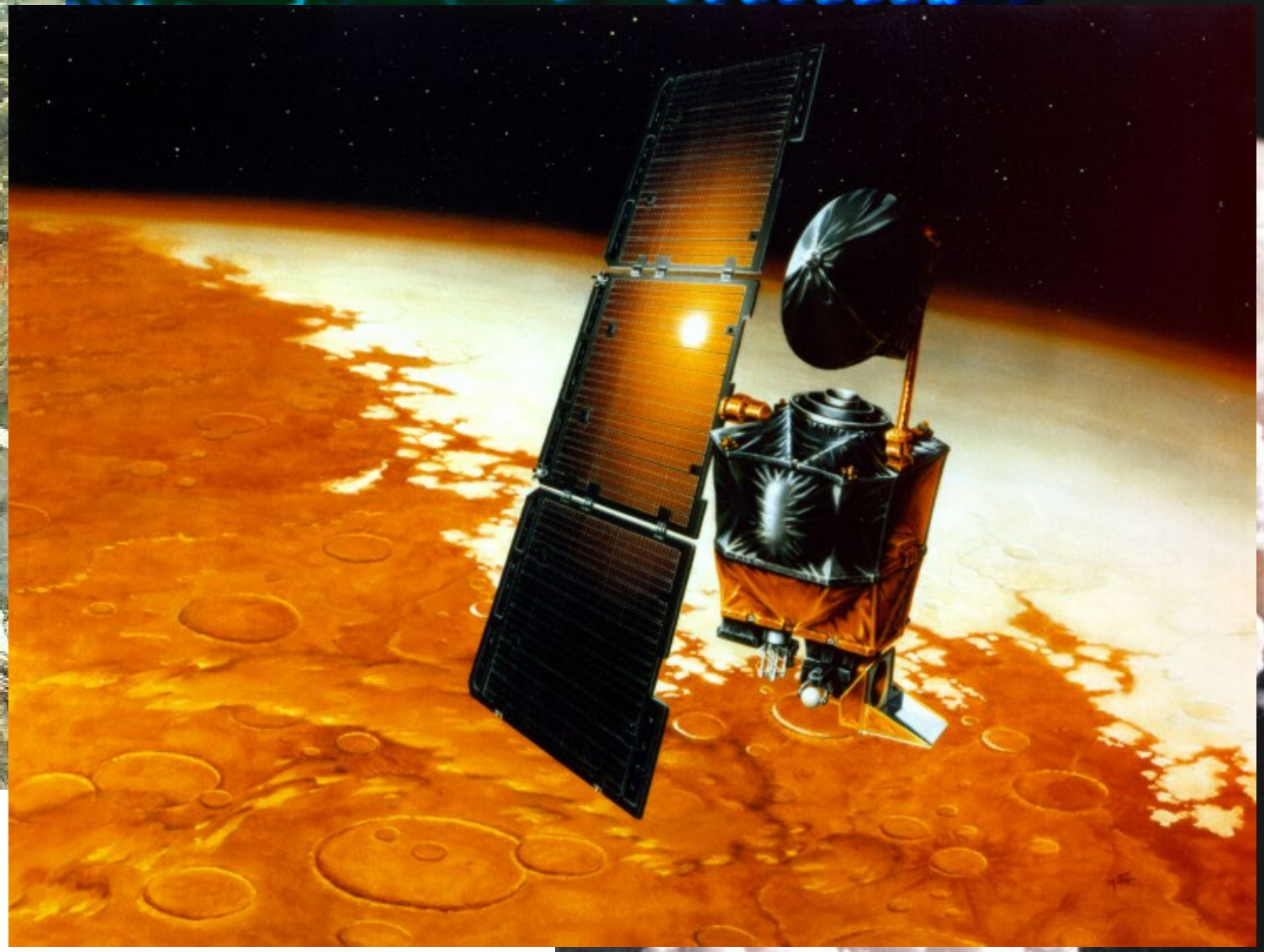
Argonne NATIONAL LABORATORY

ECP EXASCALE COMPUTING PROJECT

Good Scientific Process Requires Good Software Practices

Good Software Practices Will Increase Science Productivity


R23
CD13
R36
R24
CD5
CD7
CD8
CD14
R43

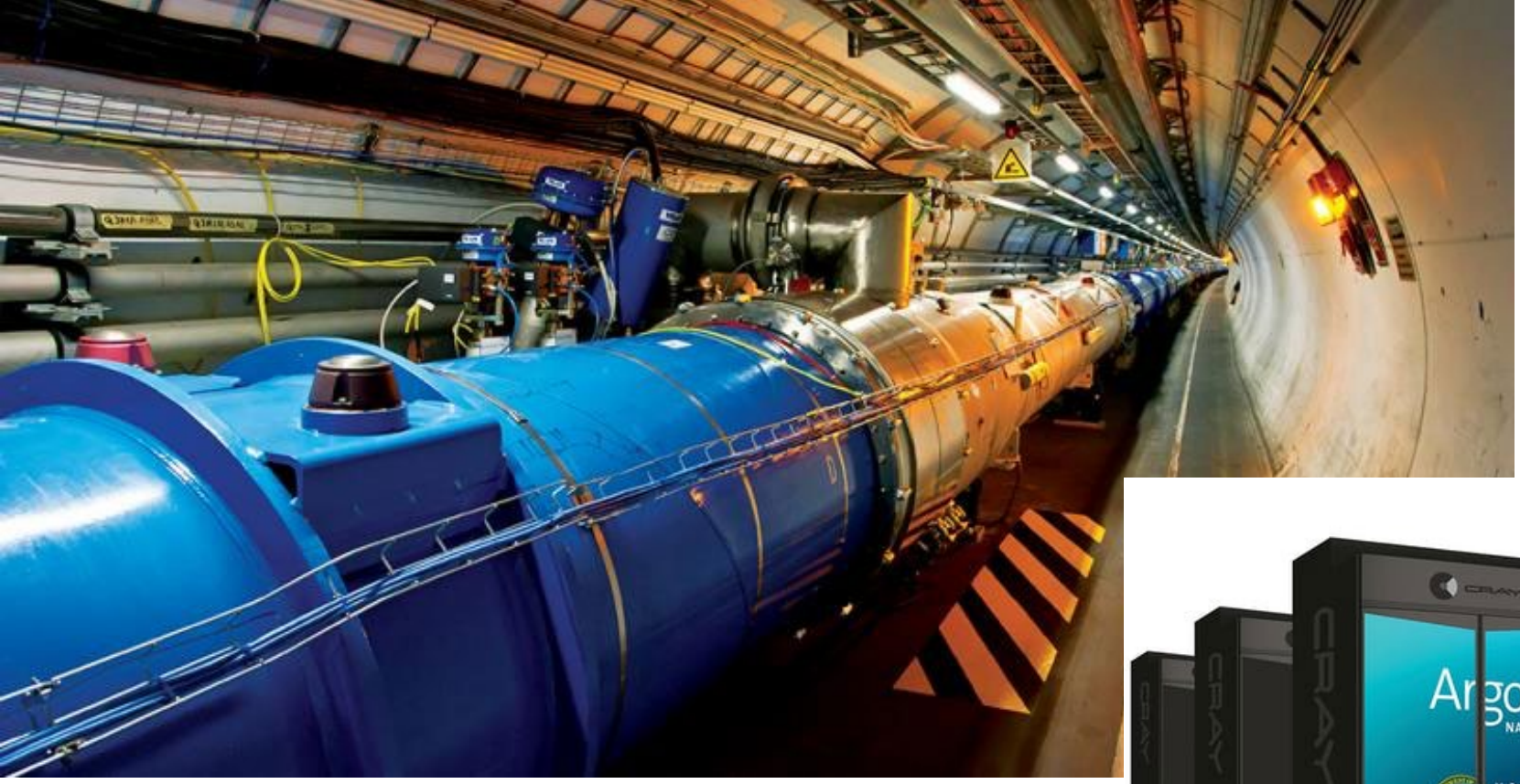


Authors retract study showing hydroxychloroquine was dangerous to hospitalized covid-19 patients

Hannah Jewell looks back at plague and yellow fever outbreaks in Europe and New Orleans, which revealed stark divides between the rich and poor. (The Washington Post)

Two major journals retract papers because data could not be validated. Lancet and NEJM.
https://www.sciencemag.org/news/2020/06/two-elite-medical-journals-retract-coronavirus-papers-over-data-integrity-questions

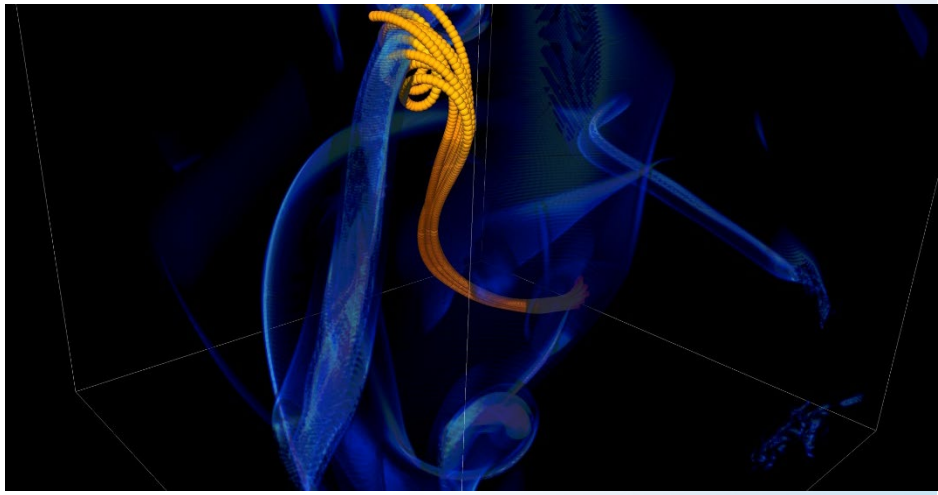
Stanford antibody study attacked for statistics – not published, but a lot of splash
https://www.medrxiv.org/content/10.1101/2020.04.14.20062463v2




Argonne
NATIONAL LABORATORY
U.S. DEPARTMENT OF
ENERGY
Aurora
intel
CRAY

FRONTIER

Mitigate Risk But It Is Never Zero



- Short notice availability of one of the biggest machines of it's time
  - **< 1month to get ready, run was 1.5 weeks**
- 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.

Understand Limitations

- Good and bad science can be done with simple models
- Are the numerics and stability understood?
- Is it statistically sound?
- Is it reproducible? Is the data sound?
- The more complex a model(s), the easier it is to make an incorrect assumption
- … and many more …

Software Practices and Sustainability are Tightly Coupled

Improve Developer Productivity and Software Sustainability

1 Customize and curate methodologies

- Target scientific software productivity and sustainability
- Use workflow for best practices content development

2 Incrementally and iteratively improve software practices

- Determine high-priority topics for improvement and track progress
- *Productivity and Sustainability Improvement Planning (PSIP)*

3 Establish software communities

- Determine community policies to improve software quality and compatibility
- Create Software Development Kits (SDKs) to facilitate the combined use of complementary libraries and tools

4 Engage in community outreach

- Broad community partnerships
- Collaboration with computing facilities
- Webinars, tutorials, events
- *WhatIs* and *HowTo* docs
- Better Scientific Software site (https://bssw.io)

Objectives of the Session

- **Good software practices** are important for scientific productivity, quality, and reliability of computational science
- **Challenges are increasing**
- Help CSE researchers **increase effectiveness** as well as leverage and impact
- **Facilitate CSE collaboration via software** in order to advance scientific discoveries

Your code will live longer than you expect.
Prepare for this.

Your science campaigns have real costs.
Think of the consequences.

Agenda

Time (Central TZ)	Module	Topic	Speaker
9:30am-9:45am	00	Introduction	David E. Bernholdt, ORNL
9:45am-10:15am	01	Overview of Best Practices in HPC Software Development	Katherine M. Riley, ANL
10:15am-10:45am	02	Agile Methodologies	James M. Willenbring, SNL
10:45am-11:00am	03	Git Workflows	James M. Willenbring, SNL
11:00am-11:15am		*Break (and Q&A with speakers)*	
11:15am-12:00pm	04	Software Design	Anshu Dubey, ANL
12:00pm-12:45pm	05	Software Testing	Anshu Dubey, ANL
12:45pm-1:45pm		*Lunch (and Q&A with speakers)*	
1:45pm-2:00pm	06	Agile Methodologies Redux	James M. Willenbing, SNL
2:00pm-3:00pm	07	Refactoring	Anshu Dubey, ANL
3:00pm-3:15pm		*Break (and Q&A with speakers)*	
3:15pm-3:45pm	08	Continuous Integration	Mark C. Miller, LLNL
3:45pm-4:30pm	09	Reproducibility	David E. Bernholdt, ORNL
4:30pm-4:45pm	10	Summary	David E. Bernholdt, ORNL

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.

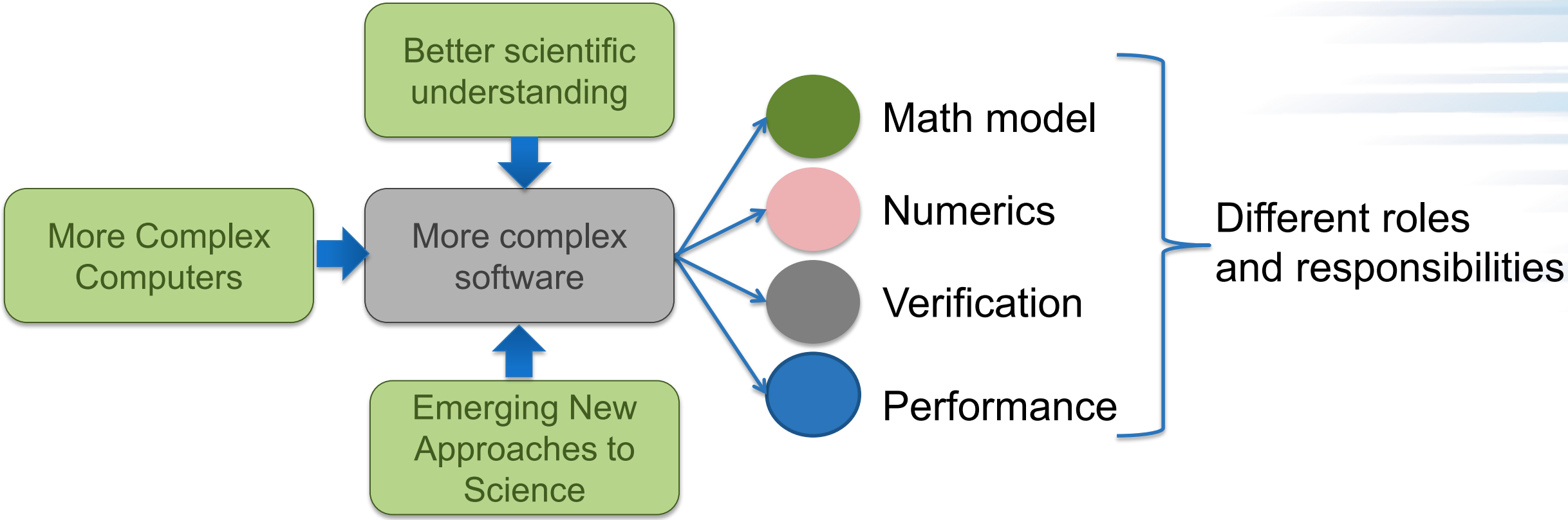
From http://c2.com/cgi/wiki?HeroicProgramming

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:



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

In Extreme-Scale science

- Codes aiming for higher fidelity modeling and new modeling
  - 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**
- **Complexity of workflows and of computational approaches**
- 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**

In Extreme-Scale science

- Codes aiming for higher fidelity modeling
  - 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 p**
- **Complexity of workflows and**
- Onset of higher platform heterog
  - Requirements are unfolding, not known *a priori*
  - **The only safeguard is investing in flexible design and robust software engineering process**

Computers change fast

Technical Debt

Consequence of Choices

Quick and dirty collects interest which means more effort required to add features

Thoughtful solutions that are limited in applicability

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

- "... it seems likely that significant software contributions to existing scientific software projects are not likely to be rewarded through the traditional reputation economy of science. Together these factors provide a reason to expect the over-production of independent scientific software packages, and the underproduction of collaborative projects in which later academics build on the work of earlier ones."

  - Howison & Herbsleb (2011)

Challenges Developing a Scientific Application

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

Consider During This Track

- What risks/challenges are you currently facing?
- What approaches might be easier to start in your development process?
- What choices are made in your application out of momentum vs planning?
- What support would you need for a more challenging change?
- If you consider software changes, are you considering the new risks?

Questions