EXASCALE COMPUTING PROJECT

Motivation and Overview

Patricia A. Grubel
Los Alamos National Laboratory

David E. Bernholdt
Oak Ridge National Laboratory

Anshu Dubey, Katherine M. Riley
Argonne National Laboratory

Developing a Testing and Continuous Integration Strategy for your Team tutorial @ ECP Annual Meeting, April 2021

LA-UR-21-23406

U.S. DEPARTMENT OF ENERGY
Office of Science
National Nuclear Security Administration

License, Citation and Acknowledgements

License and Citation

- **The requested citation the overall tutorial is: David E. Bernholdt, Patricia A. Grubel, and James M. Willenbring, Developing a Testing and Continuous Integration Strategy for your Team tutorial, in Exascale Computing Project Annual Meeting, online, 2021. DOI: 10.6084/m9.figshare.14376956**
- Individual modules may be cited as *Speaker, Module Title*, in Better Scientific Software tutorial…

Acknowledgements

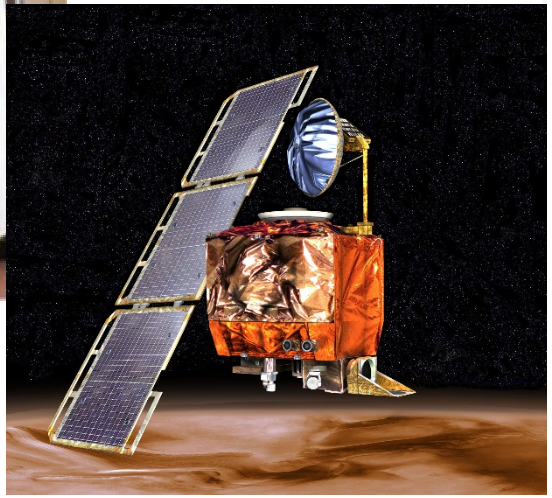
- Additional contributors include: David E. Bernholdt, Anshu Dubey, Rinku K. Gupta, Mike Heroux, Alicia Klinvex, Mark Miller, Jared O'Neal, Katherine Riley, David Rogers, Deborah Stevens, James Willenbring
- This work was supported by the U.S. Department of Energy Office of Science, Office of Advanced 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.

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

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

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

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

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

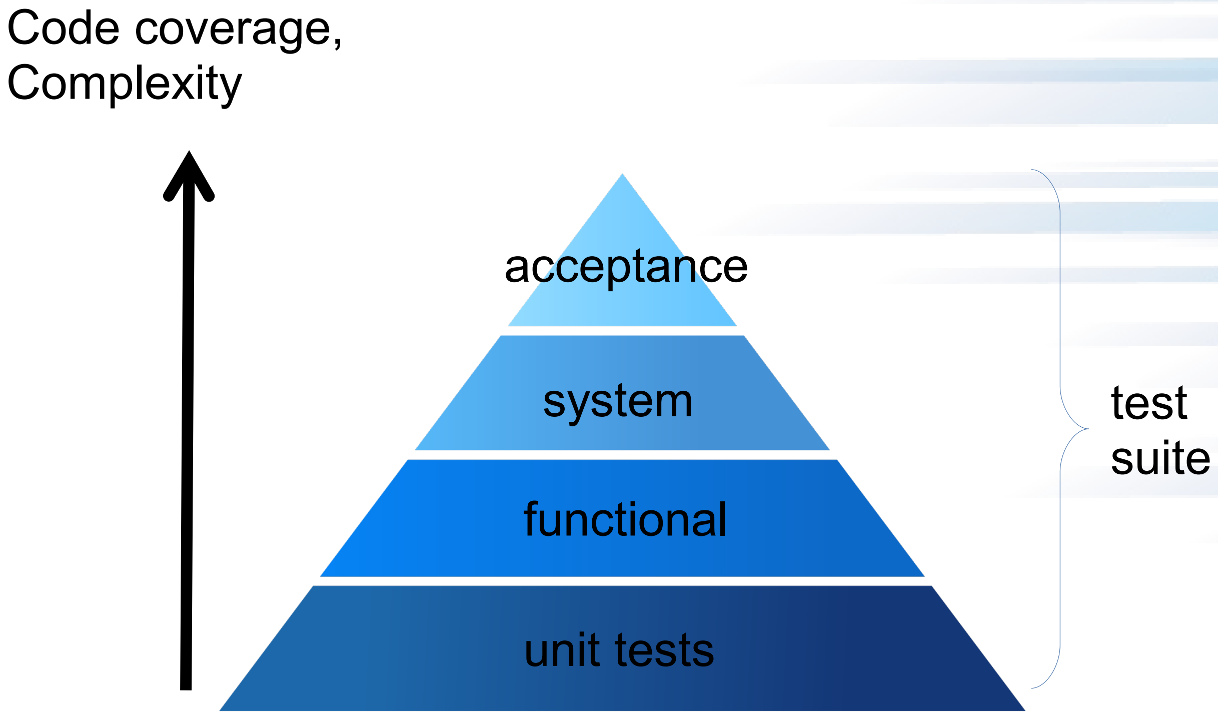
Desirable

- 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

General Categories of Testing

- Development tests
  - Tests run to protect stability while making changes to the code
  - Can include: unit, functional, integration, system, regression, verification, performance, etc.
- Post-installation “smoke” tests
  - Simple tests to ensure the build/install process has succeeded
  - Typically take only a few minutes
  - Could be a *subset* of development tests
- Continuous integration tests
  - Rapid feedback aimed at preventing changes from breaking key branches of the code
  - Run quickly, fail fast, catch problems that would impact other developers
  - Usually associated with automation



(Some) Challenges of Testing Complex Software Systems

- Designing tests
  - Complex software tends to have an extensive network of interdependencies
  - For complex scientific software it may be hard to construct a priori tests for some cases
- Implementing tests
  - Introducing testing into legacy code (legacy == untested)
  - Understanding and progressively improving code coverage
- Automating tests
  - Just get started – easy to get lost in all of the options
  - You have to *have* tests to be able to automate them!
- What to run where, and when?
  - Consider what resources are required, and what the tests are used for

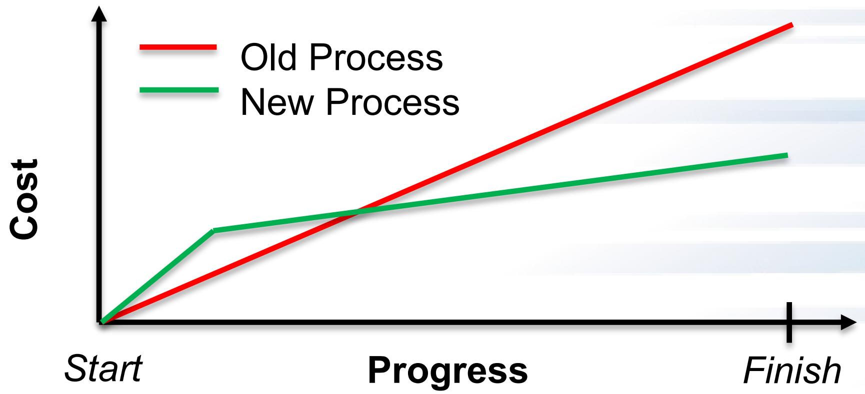
Testing is a very large subject. This is what we have time for today

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 (EDT)	Module	Topic	Speaker
2:30-2:35pm	00	Introduction	David E. Bernholdt, ORNL
2:35pm-2:40pm	01	Motivation and Overview	Patricia A. Grubel, LANL
2:40pm-3:00pm	02	Software Testing 1	Patricia A. Grubel, LANL
3:00pm-3:25pm	03	Software Testing 2	David E. Bernholdt, ORNL
3:25pm-3:55pm	04	Continuous Integration	James M. Willenbring, SNL
3:55pm-4:00pm	05	Summary	James M. Willenbring, SNL