Software Testing – Part 2

David M. Rogers
Oak Ridge National Laboratory

Anshu Dubey, Rinku Gupta
Sandia National Laboratories

Better Scientific Software Tutorial, ISS, March 2021

License, Citation and Acknowledgements

License and Citation

- This work is licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
- **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

How to build your test suite?

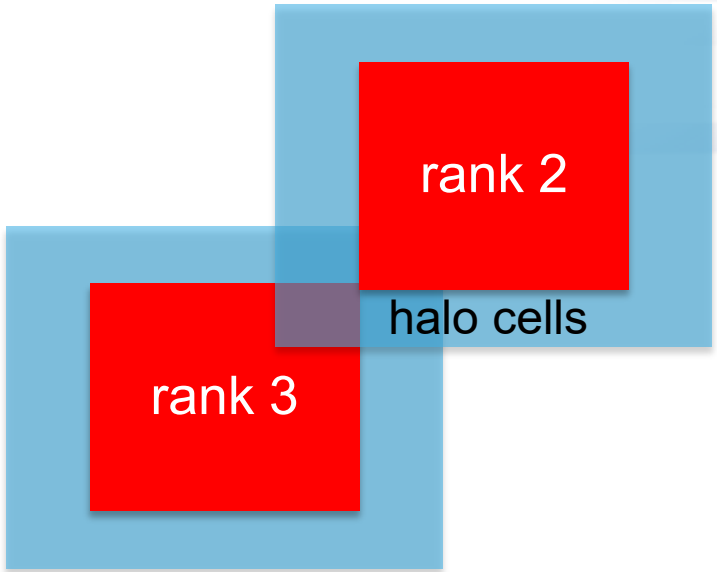
- Two “levels”
  - Regression testing
    - May be long running
    - Provide comprehensive coverage
  - Continuous integration
    - Quick diagnosis of error
- A mix of different granularities works well
  - Unit tests for isolating component or sub-component level faults
  - Integration tests with simple to complex configuration and system level
  - Restart tests

- Rules of thumb
  - Simple
  - Enable quick pin-pointing

Useful resources https://ideas-productivity.org/resources/howtos/

Why not always use the most stringent testing?

- Effort spent in devising running and maintaining test suite is a tax on team resources
- When the tax is too high…
  - Team cannot meet code-use objectives
- When is the tax is too low…
  - Necessary oversight not provided
  - Defects in code sneak through
- **Team Meeting!** Evaluate project needs:
  - Objectives: expected use of the code
  - Lifecycle stage: new or production or refactoring
  - Team: size and degree of heterogeneity
  - Lifetime: one off or ongoing production
  - Complexity: modules and their interactions



Additional Notes: Good Testing Practices

- Verify Code coverage
- Must have consistent policy on dealing with failed tests
  - Issue tracking
    - How quickly does it need to be fixed?
    - Who is responsible for fixing it?
- Someone should be watching the test suite
- When refactoring or adding new features, run a regression suite before check in
  - Add new regression tests or modify existing ones for the new features
- Code review before releasing test suite is useful
  - Another person may spot issues you didn't
  - Incredibly cost-effective

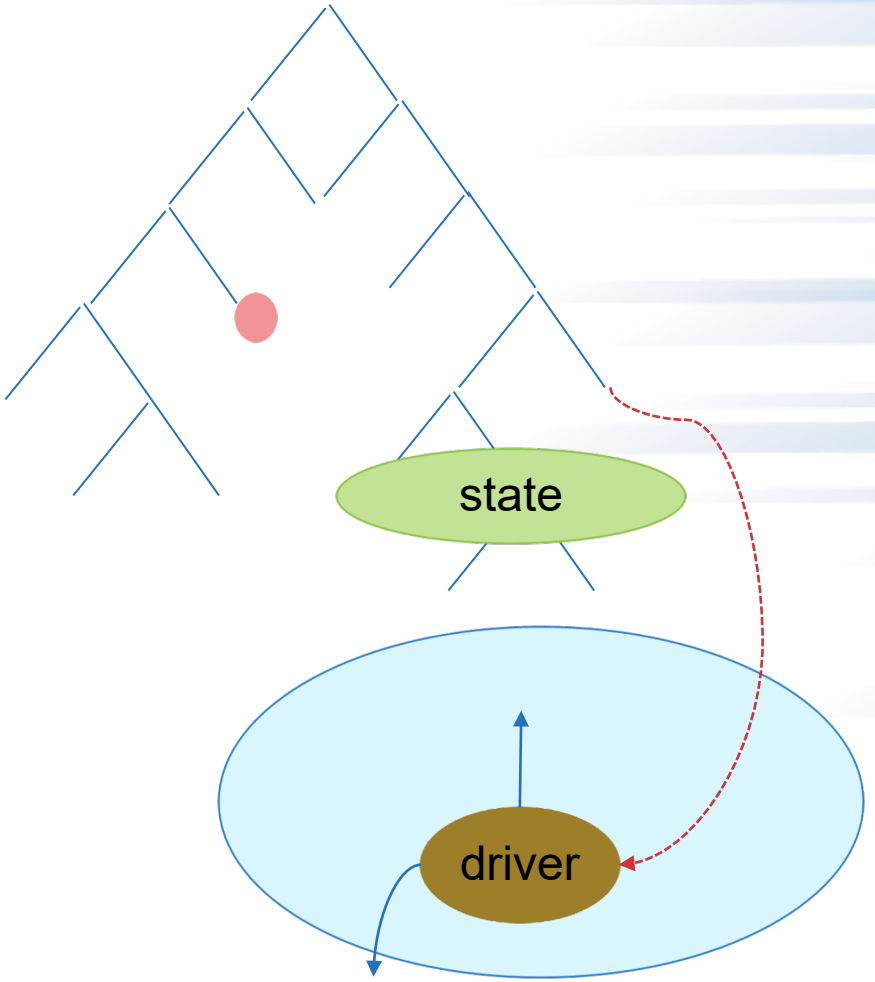
Example 1: Test Development For a New Code

- Development of tests and diagnostics goes hand-in-hand with code development
  - Compare against simpler analytical or semi-analytical solutions
  - Build granularity into testing
  - Use scaffolding ideas to build confidence
  - Always inject errors to verify that the test is working
  - Non-trivial to devise good tests, but extremely important

Example 2:
Test Development For a Legacy Code

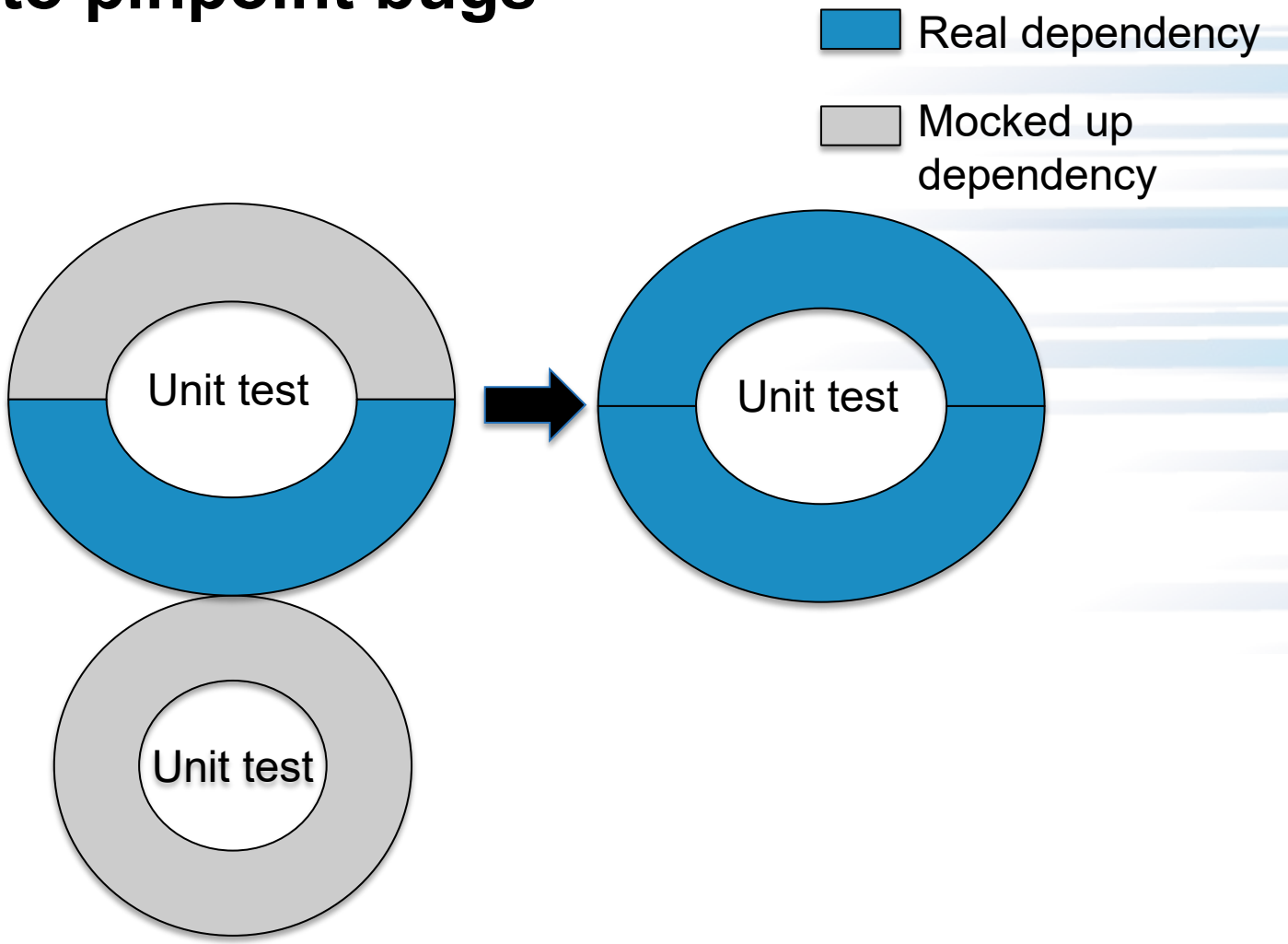
There may not be existing tests

- Isolate a small area of the code
- Dump a useful state snapshot
- Build a test driver
  - Start with only the files in the area
  - Link in dependencies
    - Copy if any customizations needed
- Read in the state snapshot
- Restart from the saved state
- Verify correctness
  - Always inject errors to verify that the test is working



Example 3: Structuring Tests to pinpoint bugs

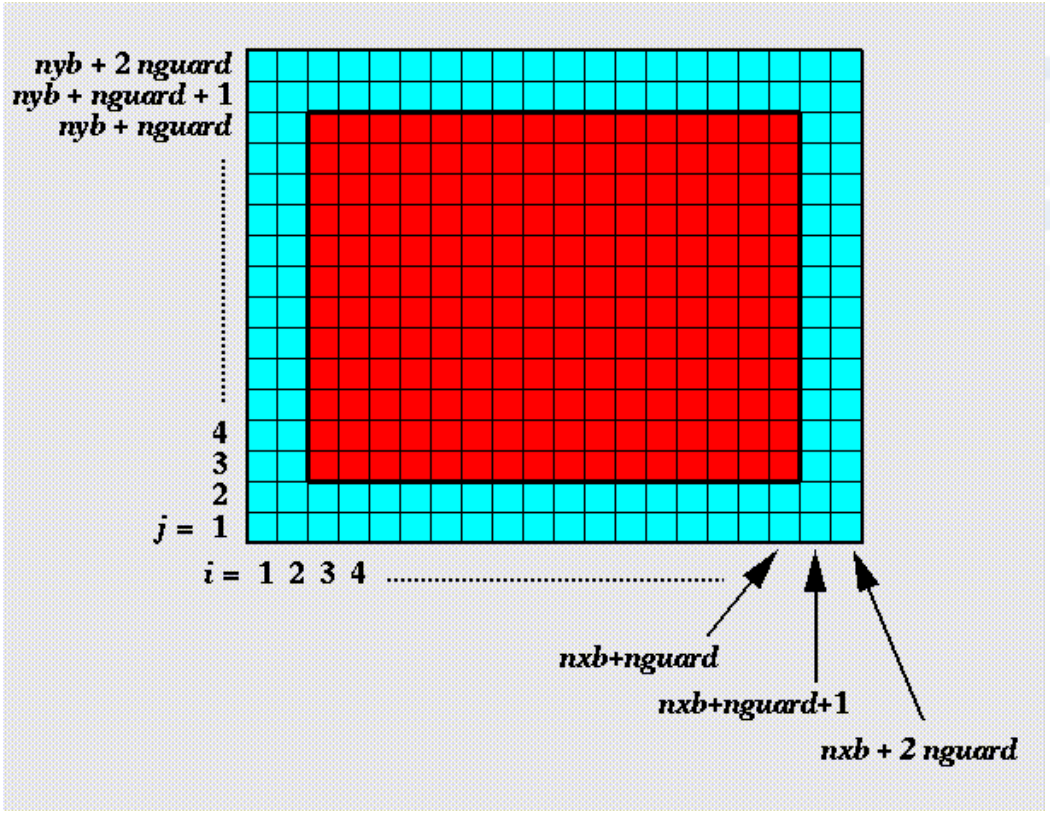
- Bottom-up picture (shim if needed)
  - Components can be exercised against known simpler applications
  - Same applies to combination of components
- Build a scaffolding of verification tests to gain confidence

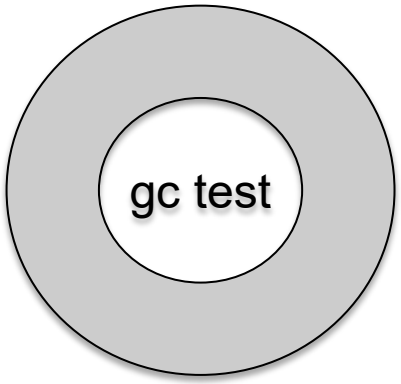


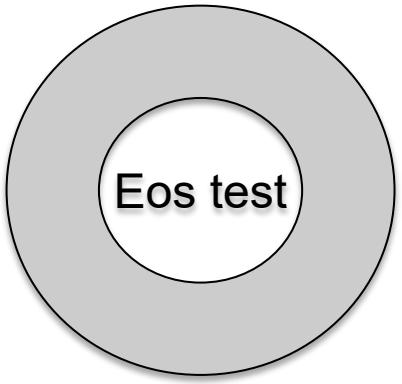
Example 3: Structured Testing

Unit test for Grid halo cell fill

- Verification of guard/ghost/halo cell fill
- Initialize field on interior cells (red)
- Apply guard cell fill
- Check for equivalence with known fill pattern





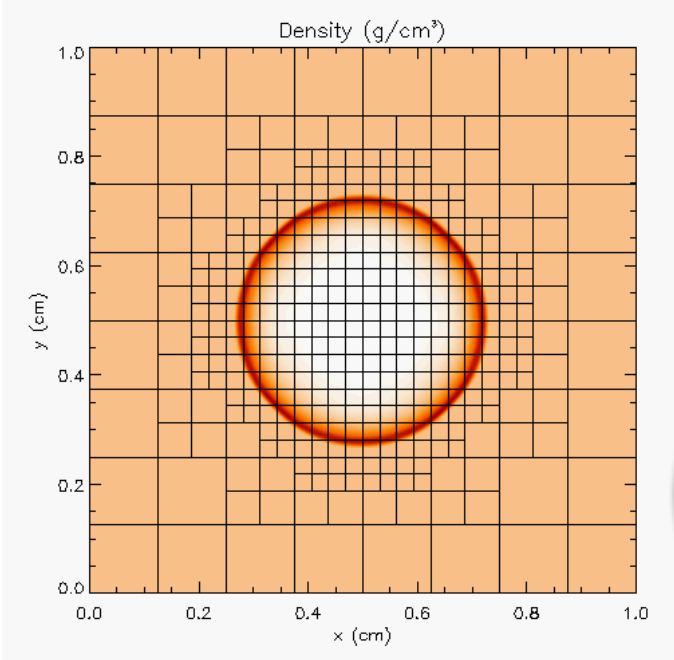


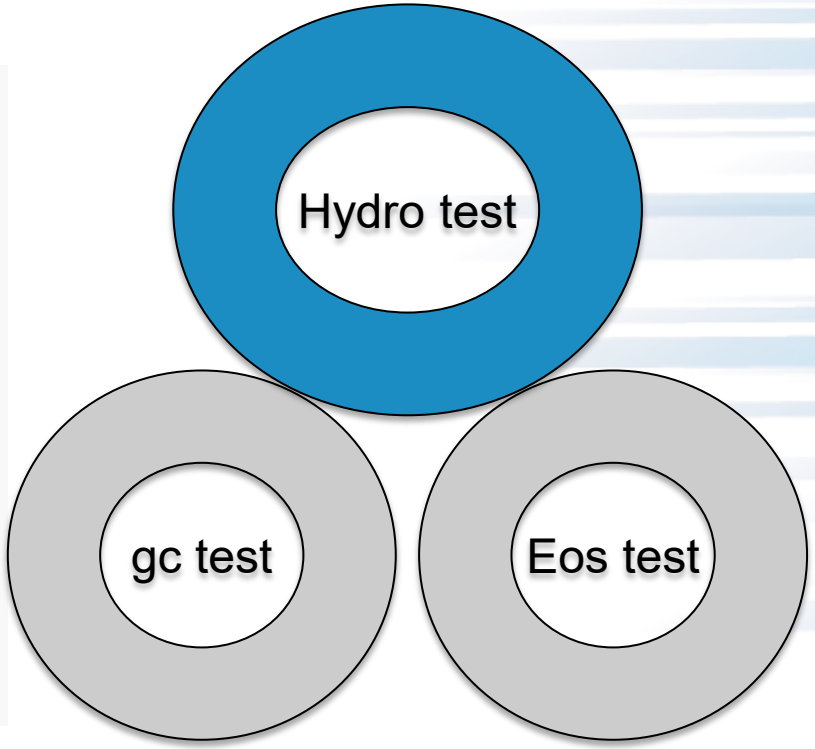
Next, build an EOS Test – is E(V,T) consistent with P(V,T)?

Example 3: Structured Testing

Unit test for Hydrodynamics

- Sedov blast wave
- High pressure at the center
- Shock moves out spherically
- FLASH with AMR and hydro
- Known analytical solution





Though it exercises mesh, hydro and eos, if mesh and eos are verified first, then this test verifies hydro

More testing needed for Grid using AMR
Flux correction and regridding

Example 3: Structured Testing

For AMR, correct behavior of flux conservation and regridding should also be verified.

Reason about correctness for testing Flux correction and regridding

IF Guardcell fill and EOS unit tests passed

- Run Hydro without AMR
  - If failed fault is in Hydro
- Run Hydro with AMR, but no dynamic refinement
  - If failed fault is in flux correction
- Run Hydro with AMR and dynamic refinement
  - If failed fault is in regridding

Example 4: Coverage Matrix (physics vs. functionalities)

First line of defense – code coverage tools

- Code coverage tools necessary but not sufficient
- Do not give any information about interoperability

	Hydro	EOS	Gravity	Burn	Particles
AMR	CL	CL		CL	CL
UG	SV	SV			SV
Multigrid	WD	WD	WD	WD	
FFT			PT		

- Map your tests and examples – what do they do?
- Follow the order
  - All unit tests – including full module tests (e.g. CL)
  - Tests sensitive to perturbations (e.g. SV)
  - Most stringent tests for solvers (e.g. WD, PT)
  - Least complex test to cover remaining spots (**Aha!**)

TAKEAWAYS

- CONTEXT: UNDERSTAND TESTING NEEDS AND COSTS
- DEVISE TESTS TO ENABLE QUICK PINPOINTING OF ERRORS THROUGH REASONING ABOUT THEIR BEHAVIOR
- TEST AT VARIOUS GRANULARITIES – BOTTOM-UP (UNIT/VERIFICATION) THROUGH TOP-DOWN (INTEGRATION/VALIDATION)
- TESTS AT VARIOUS COMPLEXITIES – CI VS. REGRESSION
- MAINTAIN A HOLISTIC VALIDATION STRATEGY: THINK GLOBALLY, ACT LOCALLY

…….QUESTIONS ?

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