Improving Scientific Software Quality

Christopher Harrop & Mark Govett
Why do we need to talk about software quality?

What is software quality?

How can test driven development improve software quality?
Why do we need to talk about it?

“Existing models are known to have high levels of software quality”

(B. N. Lawrence et al, Crossing the Chasm: how to develop weather and climate models for next generation computers?, 2018)
Why do we need to talk about it?

“The [nuclear engineering code], in spite of the aspirations of its designers, amounted to no more than a very expensive random number generator”

(L. Hatton, The T Experiments: Errors in scientific software, 1997)
Why do we need to talk about it?

- Poor quality has far reaching consequences
- Wide range of scientific software development practices
- Software quality throttles scientific progress
- Time to develop a new model is longer than the lifespan of the new hardware
What is it?

Scientific Software

Quality

do j=2,ny-1
do i=2,nx-1
  u_new[i,j] = ((u(i+1,j) + u(i-1,j) + u(i,j+1) - 0.5_r8kind * dtdx * ((u(i+1,j)**2 - 0.5_r8kind * dtdy * (v(i,j)) * (u - 0.5_r8kind * g * dtdx * (h(i+1,j)))) - 0.5_r8kind * g * dtdx * (h(i+1,j))))
What is it?

- The bad news
  - There is no objective definition of software quality
  - Can’t be measured accurately/easily
  - Hard to prove that software engineering practices determine quality
What is it?

● The bad news
  ○ There is no objective definition of software quality
  ○ Can’t be measured accurately/easily
  ○ Hard to prove that software engineering practices determine quality

● The good news
  ○ There are some good ideas and standards
  ○ Some things are quantifiable
  ○ Studies show rigorous testing decreases defect density
What is it?

IEEE Std 1061™-1998 (R2009) - A Software Quality Metrics Methodology
What is it?

- Functional Suitability
- Usability
- Reliability
- Maintainability
- Performance Efficiency
- Compatibility
- Security
- Portability

What is it?

Scientific Software Quality Attribute Priorities

1. Functional Suitability
2. Usability
3. Reliability
4. Maintainability
5. Performance Efficiency
6. Compatibility
7. Security
8. Portability

Scientific Software Quality Attribute Priorities
What is it?

Scientific Software Quality Attribute Priorities

1. Functional Suitability
2. Usability
3. Reliability
4. Maintainability
5. Performance Efficiency
6. Compatibility
7. Security
8. Portability

No one knows I exist...

Why does everyone ignore me?
What is it?

Perceived Scientific Software Quality Tradeoffs
What is it?

Perceived Scientific Software Quality Tradeoffs

Performance Efficiency

Maintainability

Portability
What is it?

- Scientific software quality models do exist
- Does/should anyone use them?
- Metrics for quality components
  - Complexity → Maintainability
  - Defect density → Reliability
- Some scientific quality models lack important attributes

Challenges Specific to Scientific Software

- Capacity for scientific insight is an important quality attribute
- “Scientific software quality” conflates science and software
  - Theoretical system, computational system, software implementation
- Requirements are often poorly defined up front
- Requirements driven by scientific discovery process
- Evolving requirements make extensibility and reproducibility difficult
- Oracle & tolerance problems make correctness difficult to measure
How Can Test Driven Development Help?

Input

10^6 LOC

Baseline Output

Typical Scenario
How Can Test Driven Development Help?

Typical Scenario

Input

$10^6$ LOC

Code Change(s)

Baseline Output

Are these forecasts equivalent?

Input

$10^6$ LOC

Test Output
How Can Test Driven Development Help?

Several problems with reliance on system level tests

- Focus is on testing the “model” instead of the “software”
- Does not provide error localization when failures are detected
- Trillions of operations performed exacerbate comparison of results
- High levels of test coverage are difficult to achieve
- Often masks serious errors
- Undetected bugs are allowed into the “stable” repository branches
How Can Test Driven Development Help?

A better way....

- Test the science AND the software
  - Theoretical system, computational system, software implementation

- Test multiple quality factors
  - Performance, reliability, correctness, portability

- Test at all granularities
  - Unit tests, integration tests, system tests

- Write new code → Write new tests
How Can Test Driven Development Help?

Rules of engagement

- Automate tests / continuous integration
- Require pull requests for all merges
- Require reviews for all pull requests
- No pull requests are merged unless all tests pass
- Pull requests must supply tests for all new code
Conclusions

- We can learn from commercial software engineering industry
- Maintainability should be prioritized
- Test-driven development should be adopted to reduce defect density
- Test automation should be maximized to minimize human error
ISO/IEC 25010:2011 Quality Model

Functional Suitability

- Functional Completeness
- Functional Correctness
- Functional Appropriateness
What is it?

ISO/IEC 25010:2011 Quality Model

Performance Efficiency

- Time Behavior
- Resource Utilization
- Capacity
What is it?

ISO/IEC 25010:2011 Quality Model

Compatibility

- Co-existence
- Interoperability
What is it?

ISO/IEC 25010:2011 Quality Model

Usability
- Appropriateness
- Recognizability
- Learnability
- Operability

- User Error Protection
- User Interface Aesthetics
- Accessibility
What is it?

ISO/IEC 25010:2011 Quality Model

Reliability

- Maturity
- Availability
- Fault Tolerance
- Recoverability
What is it?

ISO/IEC 25010:2011 Quality Model

Security

- Confidentiality
- Integrity
- Non-repudiation

- Authenticity
- Accountability
What is it?

ISO/IEC 25010:2011 Quality Model

Maintainability

- Modularity
- Reusability
- Analysability
- Modifiability
- Testability
What is it?

ISO/IEC 25010:2011 Quality Model

Portability

- Adaptability
- Installability
- Replaceability
What is it?

IEEE Std 1061™-1998 (R2009) - A Software Quality Metrics Methodology

● Goals
  ○ Assess achievement of quality goals;
  ○ Establish quality requirements for a system at its outset;
  ○ Establish acceptance criteria and standards;
  ○ Evaluate the level of quality achieved against the established requirements;
  ○ Detect anomalies or point to potential problems in the system;
  ○ Predict the level of quality that will be achieved in the future;
  ○ Monitor changes in quality when software is modified;
  ○ Assess the ease of change to the system during product evolution;
  ○ Validate a metrics set