Software Quality

It doesn’t happen by accident.
• You’ve written a function that takes three lengths and returns true if they could be the sides of a triangle, and false otherwise.
• What tests do you write to check if your code is correct?
Modern Software Quality

• Quality occurs at every stage of a project

• Quality starts with requirements
  • Feasible
  • Measurable
  • Verified by client
• Design also needs to exhibit quality
  • Connect to requirements
  • Implementability

• Testing still plays an important role
Processes

- Processes need to be high quality too!
- Many activities contribute to software quality:
  - Software engineering process
  - Architecture and design
  - Coding standards and practices
  - Version control
  - Testing
  - Code reviews

- Quality does not happen by accident!
Agile ≠ Quality

- Agile provides a *process* framework
  - Individual activities largely not addressed
  - Customize for your quality standards

- Activities you (hopefully) already do:
  - Architecture and design
  - Good coding practices
  - Version control
  - Unit testing (one kind of testing)
  - Pair programming (sort-of continuous code review)

- Is it enough?
  - How will you know when your product is high quality?
Quality Characteristics ("-ilities")

- Functional suitability
- Performance efficiency
- Compatibility
- Usability
- Reliability
- Security
- Maintainability
- Portability
Having a Plan

• For this course:
  • How will you validate that you have met the requirements?
  • How will you address project risks?
  • Will your code be high quality? maintainable?
  • Do you need to address performance, security, etc.?

• Make a plan
  • List activities you will do
  • Tie each activity to a quality aspect it addresses
  • E.g.,
    • Unit testing: verify basic functionality, provide repeatable detection of previously seen defects
Testing
“If you don’t like testing your product, most likely your customers won’t like to test it either.”

- Anonymous
(Some) Types of Testing

- Code style
- Code coverage
- Unit testing
  - These should mostly be developed in TDD
  - Adding after the fact is a pain and hard to get right
- Functional testing
  - Testing requirements/user stories
  - Generally requires interaction with UI
- Build, integration, deployment testing
  - Do all the parts work together?
  - Does it work in production?
- Load testing
- Security testing
- Usability testing

There are tools for automating most of these!
What to Test

• User stories
  • The expected use cases
• Edge cases
  • What if the numbers are really big?
• Invalid inputs
  • Users can break anything...
When to Test

• Validate requirements in functional review
• Validate functionality in design review
• Regression test with every code submission
  • to avoid taking a step backwards
  • fix bugs as they happen, not at the end
• Acceptance tests to determine client satisfaction
Test Plans

- Written plans are repeatable plans
  - automated is even better!
- Make each test count - test one thing so you don’t mask one error with another error

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Category</th>
<th>Environment</th>
<th>Setup</th>
<th>Action</th>
<th>Expected Result</th>
</tr>
</thead>
</table>
Final Thoughts
“The bitterness of poor quality remains long after the sweetness of meeting the schedule has been forgotten.”
- Karl Wiegers