How to Develop High Quality Software

John J. Paliotta
CTO, Vector Software
How to Develop High Quality Software

- Is Everything Ok?
- How Did We Get Here?
- The Quality Improvement Process
- A Practical Approach to Quality
- Business Advantages of Quality Improvement
- Conclusion and Q&A
Is Everything OK?
Lots of Money is Being Spent on the Problem

Cambridge University research estimates the global cost of debugging software to be $312 Billion annually.

Maintenance costs are generally double the original development and testing costs.
How Did We Get Here?
State of the Art, 1977

TORONADO BROUGHAM
Front-wheel drive and the inner world of Toronado—a rare driving pleasure.

It's a leisurely world. The roomy, padded, and "carsickness-proof" interior is a joy. The comfort chairs, heated and cooled, give you a feeling of being at home. The rear view mirror is as big as a basketball, and power controls will leave you wanting more when you're finished.

The search for excellence goes deep in the 1977 Oldsmobile.
Why Isn’t Software Quality Better
Software is Different

- Best Feature of Software: *It’s Easy to Create and Change*
- Worst Feature of Software: *It’s Easy to Create and Change*
- Building a Physical Product

- Building Software

![Diagram showing the stages of building a physical product and software]
We Can Fix That in the Next Release ...

- Release cycles have shrunk from years to days to hours
- Technical Debt results from the trade-offs made during development
What’s the Solution?
If We Just Wait Long Enough ...
Software Engineering
The Quality Improvement Process
Best Practices from Industry

- Tests should prove that an application complies with its formal requirements
- Code coverage should be measured to ensure that testing is complete
- Each deployed version should undergo complete testing
Best Practices from Experience

- Requirements must be complete and correct
- Coding style and architecture should be easy-to-understand
- Testing must be part of everyone’s responsibility
- Testing must be Automated and Continuous
Ensure Requirement Completeness

- The `square_root()` function shall return the square root of its input for all valid values.
- The `square_root()` function shall return the square root of its input for all valid inputs, and 0 for all invalid inputs. Valid inputs are positive 32-bit floating point numbers, zero and positive infinity. Invalid inputs are negative numbers, negative infinity and NaN. In the event of an error, the type of error and system time should be logged using the common error logging system.

How about a simple table or diagram

<table>
<thead>
<tr>
<th><strong>square_root()</strong></th>
<th><strong>return the square root of valid input values</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Parameter(s)</td>
<td>...</td>
</tr>
<tr>
<td>Valid Input Values</td>
<td>...</td>
</tr>
<tr>
<td>Return Value</td>
<td>...</td>
</tr>
<tr>
<td>Error Handling</td>
<td>...</td>
</tr>
</tbody>
</table>
Write Simple Code

- “This code is legal”
- “It’s obvious what this does”
- “This implementation will be much faster”

Avoid

- Complex functions (complexity > 10)
- Functions with complex APIs (> 5 parameters)
- Complex conditionals (> 5 sub-conditions)
- Direct access to hardware addresses
- Direct access to RTOS features
- Undefined language features (un-initialized variables)

Do This Instead

- Create Small and Modular functions
- Factor Complex logic into Sub-Functions
- Access hardware through a single project API
- Access RTOS features via a single project API
- Verbose Coding versus Implicit behavior (short circuiting)
Implement Quality Gates

- Cyclomatic Complexity <10 for all Functions
- Static Analysis must be clean
- All existing Tests must pass (Correctness Testing)
- New Tests must map to Requirements
- Code and Tests must pass Peer Review
- Testing must produce 100% Statement Coverage (Completeness Testing)
Design Achievable Goals and Communicate Rationale

GOAL SETTING

SMART

SPECIFIC
MEASURABLE
ATTAINABLE
RELEVANT
TIME-BOUND

WHY
Publish Metrics

- Metrics have the Ability to Motivate

- The numbers don’t matter, the trends do
  - Bugs found by Customers, QA, and Developers
  - Average time for a branch to get through QA
  - Percentage of staff time used for bug fixing versus new
Test the Requirements Not the Code

- Ensure that tests validate what the software is supposed to do, not what it does do
- Ensures that there are not any forgotten cases
- Promotes the testing of functional boundary cases
Measure Code Coverage

- 100% Code Coverage should not be the goal of software testing
  - It should be the result of complete testing
Speak the Language of Test
Throw Away Your Source Code

Software Artifacts

- Requirements
- API
- Tests
- Code
Implement Test Automation

- Easy
- Fast
- Automated
- Shared
Making Testing Faster
Massively Parallel Testing
Change Based Testing

Continuous Test Process

- Design Requirements
- Implement Code
- Run Affected Tests
- Fix Bugs
- Evaluate Code Coverage
- Add New Tests
Business Advantages of Improved Quality
The Business Case for Quality Improvement

- **Faster Release Cycles**
  - Few Bugs get into the Release Cycle
  - Release Cycle is Faster
  - Customer Get New Features Faster

- **Lower Life-Cycle Costs**
  - Less Time Fixing Bugs
  - More Time Building New Product Features

- **Brand Loyalty**
  - Fewer Bugs get to the Customer
  - Customers are Happier
Conclusion
Final Thoughts

- If you are in the business of building software, you have a quality problem
  - Quality is part of everyone’s job, not just QA

- There are no silver bullets, nor are any “on the way”

- The first step to improvement is to recognize the problem
  - Plan for improvement
  - Implement that plan
  - Measure results

- Quality requires Constant Process Improvement
Questions and Answers