Building In Quality

Ten Years Later
Some software just has to work
Who Owns Quality?

Fundamental beliefs in high-reliability fields:

- Developers are responsible for quality.
- Top management is responsible for quality (ISO 9000)
- Test group is not responsible for creating quality, simply for assessing it.
Quality Strategies in High-Reliability Software

- Find problems soon after creation
- Fix problems immediately
- Prevent creation of problems

High quality from start to finish
How Defects Are Created

- Business req’ts (customer wants)
  - omit things the customer wants
  - assume customer is able to tell you what they want

- Specifications (features)
  - neglect some requirements
  - ambiguous or unclear specifications

- Design
  - make logical design errors
  - don’t define precisely enough
  - don’t cover stated requirements
  - don’t cover unstated requirements
  - ignore environment behavior

- Coding
  - make logical coding errors
  - misinterpret design
  - leave out part of design

- Integration
  - use the wrong files
  - errors in config files
The V-Model

Business req’ts (customer wants)

Reviews

Specifications (features)

Spec Reviews

Design

Design Reviews

Code

TIME

Acceptance testing

System Testing

Integration testing

Functional Testing

Unit testing

RELEASE!
Quality Gate is a Set of Exit Criteria

Typical quality gate in medical products during 1990s:

- Complies with coding standards
- Compiles with no flags level 3 or above.
- Cyclomatic complexity at or below 20
- Modules with complexity above 10 have been peer-reviewed
- Unit testing completed with 100% path coverage
- Passes all unit tests
- Integration test scripts written
A High-Reliability Definition of Done

Checklist for Technical Completeness for User Stories (Herman 2016)

Design:
- Design covers everything in the user story and acceptance criteria.
- Design reviewed by area experts and feedback is incorporated.
- User story has a link to the design.

Code:
- Code implements the design.
- Unit tests cover the design (includes use cases, API contracts).
- Code compiles and runs, on the build server, without errors, warnings, or unit test failures.
- Code and unit tests have been peer reviewed and adjustments made per comments.
- No new defects.

Acceptance Testing
- Acceptance tests in a form listed below have been written and entered into project management system
  - May include manual, automated, and unit or integration tests
  - Verification Procedure and/or SMART for high risk stories, SMART for stories with medium risk
- Acceptance tests have been reviewed by a developer and feedback has been incorporated.
- Acceptance tests pass on a branch or main build; unresolved issues found on main build have been logged into defect tracking system.
- Executed results have been attached to project management system.

Defect Fixing:
- Minimal steps to reproduce are documented in the defect description.
- Root cause analysis is documented in the defect description.
- Fix approach is documented in the defect description.
Requirements  Design  Coding  Integration
System Testing  Acceptance Testing  User
Turn off the water!
Strategies for Preventing Defects

1) Make the structure & logic visible
2) Mistake-proof with tools
3) Maintain intellectual control
4) Know your domain
5) Design away the opportunity for error
Make the Logic Visible

106 END

105 WRITE(*,*) 'x is positive but x < y'

104 GOTO 105

103 WRITE(*,*) 'x is positive and x >= y'

102 IF (X .LT. Y) GOTO 105

101 IF (X .LT. 0) GOTO 106
### Better Visibility

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>IF (x .LT. 0) GOTO 106</td>
<td>If x is less than zero, go to line 106</td>
</tr>
<tr>
<td>102</td>
<td>IF (x .LT. y) GOTO 105</td>
<td>If x is less than y, go to line 105</td>
</tr>
<tr>
<td>103</td>
<td>WRITE(<em>,</em>) 'x is positive and x &gt;= y'</td>
<td>Write the message 'x is positive and x &gt;= y'</td>
</tr>
<tr>
<td>104</td>
<td>GOTO 106</td>
<td>Go back to line 106</td>
</tr>
<tr>
<td>105</td>
<td>WRITE(<em>,</em>) 'x is positive but x &lt; y'</td>
<td>Write the message 'x is positive but x &lt; y'</td>
</tr>
<tr>
<td>106</td>
<td>END</td>
<td>End of the program</td>
</tr>
</tbody>
</table>
## Mistake-Proof with Tools

```
101 IF (x .GT. 0) THEN
102 IF (x .GE. y) THEN
103 WRITE(*,*) 'x is positive and x >= y'
104 ELSE
105 WRITE(*,*) 'x is positive but x < y'
106 END IF
107 END IF
```
More Visibility

101 IF (x .GT. 0) THEN
102 IF (x .GE. y) THEN
103 WRITE(*,*) 'x is positive and x >= y'
104 ELSE
105 WRITE(*,*) 'x is positive but x < y'
106 END IF
107 END IF
Today’s Tools Rock!

```
365  color: #215968;
366 }
367 .auto-style19 {
368   color: #000000;
369 }
370 -->
371 </style>
372
373 <meta content=document name=resource-type>
374 <meta content="software testing, software quality" name=keywords>
375 </head>
376
377 <body lang=EN-US link=blue vlink=purple>
378
379 <div class=WordSection1>
380 
381 <table class=MsoNormalTable border=0 cellspacing=0 width="95%"
382 style='width:95.0%'>
383 <tr>
384   <td width=150 cellspacing=2 valign=top style='width:112.5pt;background:#D4EEF3;
385     padding:.75pt .75pt .75pt .75pt'>
386     <div align=center>
387       <table class=MsoNormalTable border=0 cellspacing=0 width=135
388         style='width:101.25pt'>
389         <tr>
390           <td width=131 style='width:98.25pt;padding:1.5pt 1.5pt 1.5pt 1.5pt'></td>
391         </tr>
392         </tr>
```

© 2017 Iberle Consulting Group, Inc.
Maintain Intellectual Control

No more than seven things at once!

Abstractions which simplify:

• Classes and objects
• Design notation – UML, DFDs
• Design patterns
Test Driven Development

High quality from start to finish

Find problems soon after creation

Fix problems immediately

Prevent creation of problems

© 2017 Iberle Consulting Group, Inc.
Know Your Domain

Coworkers

Classes

Books

Webinars

Conferences
Design Away Opportunity for Error

User interface field must be long enough for longest value in corresponding database field.

A) Code UI field length to match database field length and write test for longest possible value.

B) Test tool runs through UI code, checking UI field lengths against database field lengths.

C) User interface reads field length from database and creates UI field to match.

Credit: James Shore, *Art of Agile Development*, “No Bugs”
Design Away Opportunity for Error

11. Set up faxing and take Product Tour

Make sure your PC is on. Follow the onscreen instructions to enter your name, phone number, and other important settings. See the reference guide for more details. Take the Product Tour to learn about your new HP OfficeJet.

Defect: They don’t match
Design Away Opportunity for Error

Set up faxing and take Product Tour

Make sure your PC is on. Follow the onscreen instructions to enter your name, phone number, and other important settings. See the reference guide for more details. Take the Product Tour to learn about your new HP OfficeJet.
Design Away Opportunity for Error

Set up faxing and take Product Tour

Make sure your PC is on. Follow on-screen instructions.
The Agile V
The Agile V

- Business req’ts (customer wants)
- Specifications (features)
- Design
- Code & Refactor
- Unit testing
- Functional Testing
- Integration testing
- System Testing
- Continuous Integration
- Reviews
- User Story
- Design Reviews
- TIME
- RELEASE!
- User Demo
- Time
- Reviews
Agile Quality Gates

Definition of Done
Agile Quality Gates
Agile Quality Gates

Releasable

Definition of Done
Agile Quality Gates

Definition of Done: Release

Definition of Done: Sprint

Definition of Done: Story
How Agile Affects Software Quality

The Good

• Fast feedback
• Focus on user and customer
• TDD
• ATDD
• Design patterns
• Pair programming
• Continuous integration

The Problematic

• Demolishing too many quality gates
• Ignoring non-functional requirements
• Expecting user demos to find design problems
Building In Quality – Then and Now

• Quality is **built in** during development
  • Find problems early and fix right away
  • An ounce of prevention is worth a pound of cure

• Some prevention strategies
  • Make the structure & logic visible
  • Mistake-proof with tools
  • Maintain intellectual control
  • Know your domain
  • Design away the opportunity for error

• There’s more in my paper: see PNSQC proceedings or [www.kiberle.com](http://www.kiberle.com)
Q & A