Practical Software Quality

A guide in progress

Presented at 2015 Flight Software Workshop by co-authors:
Dan Painter – NASA IV&V
Matt Rhodes - MathWorks
What’s your [mis]fortune cookie say?

Failing to plan is planning to fail.
What is quality?

“The standard of something as measured against other things of a similar kind; the degree of excellence of something.”

Oxford English Dictionary
What is quality?

“The standard of something as measured against other things of a similar kind; the degree of excellence of something.”

Oxford English Dictionary
Standards and measures

Standards can help you figure out what, and sometimes how, you should measure.

Your software quality plan **should** help you figure out when to measure.
The impetus of software quality required software quality

<table>
<thead>
<tr>
<th>[ \text{axiom} ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \propto )</td>
</tr>
</tbody>
</table>

The value of the software, from the customer’s perspective, should drive the quality requirements.
Understand the value

Why do you buy Grandma a laptop?

- It’s got great specs!
  - 9,000 mAH Li+ battery
  - 1TB SSD HD
  - 2.6 GHz Intel Core i7
  - 720p Webcam

- So she can see pictures of the kids on Facebook
Understand the value

Why do you buy Grandma a laptop?

It's got great specs!
- 9,000 mAH Li+ battery
- 1TB SSD HD
- 2.6 GHz Intel Core i7
- 720p Webcam

So she can see pictures of the kids on Facebook
Specifications to enable value

Requirements
- Customer driven requirements
- Industry driven process requirements
- Regulations
- Explicit quality requirements

Operational Constraints
- Budget – Cost
- Budget – Time
- Budget – Resources

Pre-existing solution(s)
- Legacy software
- COTS software
- Libraries

Don’t overcomplicate; focus on what is important to the customer
Value is realized as capability

Capability

Subsystem 1
- Software module 1
- Software module 4

Subsystem 2
- Software module 2
- Software module 5

Subsystem 3
- Software module 3
- Software module 6
Protecting value from risk

Risks that could prevent the customer from realizing a desired capability indicate the value of the software itself is at risk.

axiom
Unmitigated risks degrade delivered value
Per software module risk assessment

Each module is scored 1-5 per factors below

<table>
<thead>
<tr>
<th>Impact Factors</th>
<th>Likelihood Factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>Complexity</td>
</tr>
<tr>
<td>Operational S/W Control</td>
<td>Testability</td>
</tr>
<tr>
<td>Human Safety</td>
<td>Degree of Innovation</td>
</tr>
<tr>
<td></td>
<td>Developer Characteristics</td>
</tr>
</tbody>
</table>

- **Impact Factors**: Performance, Operational S/W Control, Human Safety
- **Likelihood Factors**: Complexity, Testability, Degree of Innovation, Developer Characteristics

Weighted Averages Combined & Scaled

- **Risk Level 3**: $[14-25]$
- **Risk Level 2**: $[7-13]$
- **Risk Level 1**: $[0-6]$

<table>
<thead>
<tr>
<th>1 ➔ Impact</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

$1 ➔$ Likelihood $➔$ 5
Risk flows up to capabilities

- Risk Level 1
- Risk Level 2
- Risk Level 3

Main elements:
- Capability
- Subsystem 1
- Subsystem 2
- Subsystem 3
- Software module 1
- Software module 2
- Software module 3
- Software module 4
- Software module 5
- Software module 6
<table>
<thead>
<tr>
<th>Capability</th>
<th>Software Component</th>
<th>Software Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch to Orbit</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Approach to Target</td>
<td>Trajectory Control</td>
<td>Math</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GNC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Attitude Control</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Navigation</td>
</tr>
<tr>
<td>Attitude Control</td>
<td></td>
<td>Math</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GNC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Attitude Control</td>
</tr>
<tr>
<td>Maintain Flight Systems</td>
<td>Establish and Maintain Power</td>
<td>Battery</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Power</td>
</tr>
<tr>
<td></td>
<td>Establish and Maintain Thermal Control</td>
<td>Thermal</td>
</tr>
<tr>
<td></td>
<td>Perform Fault Detection</td>
<td>FSW</td>
</tr>
<tr>
<td></td>
<td>Establish and Maintain Communication</td>
<td>Telecom</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Downlink</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Uplink</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Command</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Telemetry</td>
</tr>
</tbody>
</table>
Leverage the progress of others

There is no reason to repeat the same mistake; make sure to mitigate known risks with proven risk mitigation techniques.

Axiom
Start with known mitigations for known risks
Known risks: Enemies of Quality

- Unrealistic expectations
- Incorrect specification
- Bad coding practices and constructs
- Inadequate testing
## Example risk mitigation activities

### Building the right thing
- Prototyping
- Simulation
- Requirements tracing
- Design reviews

### Building the thing well
- Code generation
- Unit testing
- Coding standards
- Code reviews
- Monte Carlo testing

### Confirmation of building the right thing well
- Static analysis
- Integration testing
- Coverage testing
- Code metrics

### Other Risk Mitigations
- Independent verification and validation
- Experience and training
- Continuous quality plan re-evaluation

Each of these mitigation activities combats one or more of enemies of quality.
Scope with the risk assessment

Since exhaustive testing is out of reach, risk mitigation activities should be scoped relative to the determined risk levels.

Risk mitigation $\propto$ risk level
Example risk mitigation scoping

Each risk level dictates risk mitigation methods applied with specific levels of rigor.

- **Risk Level 3 ≈ [14-25]**
  - Dynamic verification activities, medium rigor
  - Static verification activities, high rigor
  - Validation activities, high rigor

- **Risk Level 2 ≈ [7-13]**
  - Static verification activities, medium rigor
  - Validation activities, medium rigor

- **Risk Level 1 ≈ [0-6]**
  - Validation activities, medium rigor
An analogy for rigor

How = Rigor of strategy application ≈ How many stones do I need to look under?
Rigor applied to static code analysis

- Identify systematic run-time errors
- Analyze non-terminating constructs
- Analyze first set of potential run-time errors

- Identify external interfaces and perform tainted data analysis
- Analyze unreachable branches
- Analyze second set of potential run-time errors

- Apply coding standard X checks
- Analyze third set of potential run-time errors
- Prove code safe
# Rigor applied to static code analysis

<table>
<thead>
<tr>
<th>Assurance Task</th>
<th>SQO Level 1</th>
<th>SQO Level 2</th>
<th>SQO Level 3</th>
<th>SQO Level 4</th>
<th>SQO Level 5</th>
<th>SQO Level 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Develop Quality Plan (AT-1)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Identify Software Build Information (AT-2)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Identify Source Code Metrics (AT-3)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Apply Standards Based Rules (AT-4)</td>
<td>OPTIONAL per IV&amp;V effort</td>
<td>OPTIONAL per IV&amp;V effort</td>
<td>OPTIONAL per IV&amp;V effort</td>
<td>OPTIONAL per IV&amp;V effort</td>
<td>OPTIONAL per IV&amp;V effort</td>
<td>OPTIONAL per IV&amp;V effort</td>
</tr>
<tr>
<td>Identify Systematic Runtime Errors (AT-5)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Analyze Non Terminating Constructs (AT-6)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Analyze Unreachable Branches (AT-7)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Identify External Interfaces (AT-8)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Analyze First Subset of Potential Runtime Errors (AT-9)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Analyze Second Subset of Potential Runtime Errors (AT-10)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Analyze Third Subset of Potential Runtime Errors (AT-11)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Prove Code Safe (AT-12)</td>
<td>X (targeted modules)</td>
<td>X (targeted modules)</td>
<td>X (targeted modules)</td>
<td>X (targeted modules)</td>
<td>X (targeted modules)</td>
<td>X (targeted modules)</td>
</tr>
<tr>
<td>Perform Tainted Data Analysis (AT-13)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Perform Dataflow Analysis (AT-14)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
# Rigor applied to static code analysis

<table>
<thead>
<tr>
<th>Impact List</th>
<th>Impact Definition</th>
<th>Impact Level</th>
<th>SQO Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>MISRA or standards compliance</td>
<td>Neither causes harm to the system nor a programmer mistake. These are simply good practices or generally accepted standards to follow.</td>
<td>Trivial</td>
<td>1,2,3,4,5,6</td>
</tr>
<tr>
<td>Deadlock</td>
<td>Two or more threads are waiting for each other to finish causing the process to freeze. These are related to Semaphores, Mutexes, and Race conditions.</td>
<td>Critical</td>
<td>2,3,4,5,6</td>
</tr>
<tr>
<td>Memory leak</td>
<td>Improper memory management. These involve improper or neglected deallocation or use of memory.</td>
<td>Minor - Major</td>
<td>2,3,4,5,6</td>
</tr>
<tr>
<td>System crash</td>
<td>Impacts system/crew safety which could lead to Loss of vehicle, loss of mission, or loss of life.</td>
<td>Critical</td>
<td>2,3,4,5,6</td>
</tr>
<tr>
<td>Undefined behavior</td>
<td>Code defects whose behavior is not specified under certain conditions. The behavior may vary depending on the implementation, environment, or semantics. Resulting behavior can range from benign to critical.</td>
<td>Major - Critical</td>
<td>2,3,4,5,6</td>
</tr>
<tr>
<td>Possible programmer mistake</td>
<td>Does not cause any major or critical issues, but areas in code that may be worth a look to determine if code was intentional or not.</td>
<td>Minor</td>
<td>3,4,5,6</td>
</tr>
<tr>
<td>Unexpected behavior or results</td>
<td>Suspicious code that may negatively affect the behavior, code flow, or calculation result if the code was not programmed as intended.</td>
<td>Minor - Major</td>
<td>3,4,5,6</td>
</tr>
<tr>
<td>Unreachable code</td>
<td>Written code that will not be executed. These could either be commented out code or defensive code. Worth investigating to see if intentional.</td>
<td>Minor - Major</td>
<td>3,4,5,6</td>
</tr>
<tr>
<td>Data loss</td>
<td>Chance to truncate data when assigning between objects, storing results of a calculation, or passing data as arguments, when the new storage type is smaller.</td>
<td>Major</td>
<td>4,5,6</td>
</tr>
<tr>
<td>Data exposure</td>
<td>Security vulnerability allowing supposedly inaccessible or private data to be modified by a malicious user.</td>
<td>Minor - Major</td>
<td>5,6</td>
</tr>
<tr>
<td>Security</td>
<td>Security vulnerabilities that do not overlap with another impact category. These include the use of unsafe functions, unverified or tainted inputs, or weaknesses prone to user exploitation.</td>
<td>Minor - Major</td>
<td>5,6</td>
</tr>
<tr>
<td>Code cleanliness</td>
<td>Good practices to observe near the completion of a project such as declaring objects as const or non-const when appropriate.</td>
<td>Trivial</td>
<td>6</td>
</tr>
<tr>
<td>Performance</td>
<td>Impacts system performance such as timing or memory usage.</td>
<td>Minor</td>
<td>6</td>
</tr>
<tr>
<td>Portability or cause compile issues</td>
<td>Code defects that may not be an issue on the current system but may not work if compiled on a different environment or if implementation was not well understood.</td>
<td>Trivial</td>
<td>6</td>
</tr>
<tr>
<td>Readability and maintainability</td>
<td>No impact on the system other than the possibility of confusion if code was shared/maintained by multiple developers or reused in another project without proper rationale included.</td>
<td>Trivial</td>
<td>6</td>
</tr>
<tr>
<td>Redefined behavior</td>
<td>Built-in commands or operators are overloaded or redefined to have new behavior. May cause confusion, however it is a non-issue if the system is well understood and documented.</td>
<td>Trivial - Minor</td>
<td>6</td>
</tr>
<tr>
<td>Unused data</td>
<td>Possible development oversight. A parameter, status or calculation result was not used, indicating there may have been an initial intent but forgotten.</td>
<td>Minor</td>
<td>6</td>
</tr>
</tbody>
</table>
Putting it all together

- Known Risks
- Requirements
- Risk Assessment Criteria
- Possible risk mitigation activities
- Pre-existing solution(s)
- Operational Constraints
- Risk mitigation adjustments
- Propose Solution
- Risk Analysis
- Software Quality Plan Creation
- Software Quality Plan
Your quality plan is an evolving, living process.
What can IV&V provide?

Assurance
- Code analysis
- Simulation
- Proven evidence based approach

Cost savings
- Subject matter experts

Confidence
- Safe
- Error Free
- Meet your needs
What can MathWorks provide?

Tools
- MATLAB, Simulink, Polyspace and more

Expertise
- Consulting
- Training
- Process assessment
- Model Based Design guidance

Community
- File Exchange
- MATLAB Answers
- Blogs
Contribute or ask questions

NASA IV&V
Dan Painter ::
 joseph.d.painter@nasa.gov

MathWorks
Matt Rhodes ::
 matt.rhodes@mathworks.com