CS314 Software Engineering
Peer Reviews

Dave Matthews

The only valid measurement of code quality: WTFs/minute

(c) 2008 Focus Shift/OSNews/Thom Holwerda - http://www.osnews.com/comics
Peer Reviews

• Informal
  – Over the shoulder
  – Tool assisted (like GitHub pull request reviews)
  – Email pass around
  – Pair Programming

• Formal
  – Inspections

Inspections

• Process to systematically examine project artifacts (documents or code) to identify as many problems as possible.
  – Careful line by line review.
  – May use a checklist of common programming errors, local standards, and practices.
  – Multiple people with different points of view.
  – Review independently to find problems, meet to review and report findings.
Inspections – Why?

- Inspections can find multiple defects at a time.
- Inspections can be done on incomplete systems.
- Inspections can consider broader quality aspects, such as non-functional requirements.
- Inspections can find defects difficult to detect by testing.
- Inspecting can save you a lot of rework / technical debt.

Inspections – What?

- Requirements
- Architecture
- Design
- Code
- Documentation
- Tests
Inspections – When?

• The earlier problems are found, the less expensive they are to fix.
• Inspecting before extensive testing allows you to quickly get rid of many defects, rather than removing them one at a time.
• Inspecting can save you a lot of rework / technical debt.

Inspections – Roles

• End user
  – User experience, useful messages, input validation, responsiveness, ...
• Maintainer
  – Descriptive names, consistent exception handling, small classes/methods, good comments, ...
• Tester
  – Documentation, well defined interfaces, ...
• Moderator
  – Makes the process efficient and effective at finding problems, ensures only logging occurs – no fixing.
**Inspections – Principles**

- Advance preparation – identify problems before meeting
- Only inspect things that are ready
- Re-inspect when changes are made
- No blame – work together, next time it will be yours
- Keep the discussions open – don’t involve managers
- Avoid discussing how to fix the problems
- Avoid discussing style issues
- Do not rush (200 lines code / hour, 10 pages of text)
- Avoid extended sessions (<2 hours at a time)
- Keep inspection logs to track quality of processes

**Inspections – sample checklist**

<table>
<thead>
<tr>
<th>problem class</th>
<th>inspection check</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data faults</td>
<td>Are all program variables initialized before their values are used?</td>
</tr>
<tr>
<td></td>
<td>Have all constants been named?</td>
</tr>
<tr>
<td></td>
<td>Should the upper bound of arrays be equal to the size of the array or size-1?</td>
</tr>
<tr>
<td></td>
<td>If character strings are used, is a delimiter explicitly assigned?</td>
</tr>
<tr>
<td></td>
<td>Is there any possibility of a buffer overflow?</td>
</tr>
<tr>
<td>Control faults</td>
<td>For each conditional statement, is the condition correct?</td>
</tr>
<tr>
<td></td>
<td>Is each loop certain to terminate?</td>
</tr>
<tr>
<td></td>
<td>Are compound statements correctly bracketed?</td>
</tr>
<tr>
<td></td>
<td>In case statements, are all possible cases accounted for?</td>
</tr>
<tr>
<td></td>
<td>If a break is required after each case in case statements, has it been included?</td>
</tr>
<tr>
<td>Input/output faults</td>
<td>Are all input variables used?</td>
</tr>
<tr>
<td></td>
<td>Are all output variables assigned a value before they are output?</td>
</tr>
<tr>
<td></td>
<td>Can unexpected inputs cause corruption?</td>
</tr>
<tr>
<td>Interface faults</td>
<td>Do all functions and methods have the correct number of parameters?</td>
</tr>
<tr>
<td></td>
<td>Do formal and actual parameter types match?</td>
</tr>
<tr>
<td></td>
<td>Are the parameters in the right order?</td>
</tr>
<tr>
<td></td>
<td>Do all components use a consistent model for shared memory structure?</td>
</tr>
<tr>
<td>Storage faults</td>
<td>If a linked structure is modified, have all links been correctly diagnosed?</td>
</tr>
<tr>
<td></td>
<td>If dynamic storage is used, has space been allocated correctly?</td>
</tr>
<tr>
<td></td>
<td>Is space explicitly deallocated after it is no longer required?</td>
</tr>
<tr>
<td>Exception faults</td>
<td>Have all possible error conditions been taken into account?</td>
</tr>
</tbody>
</table>

Ian Sommerville, Software Engineering, 2016
Formal Inspection Activity

- Pick a some code in your project
  - The piece of code in your project that scares you the most
  - Find a problem area with low test coverage
  - Around 200-400 lines
- Plan an inspection and create team/inspection.md
  - Assign roles to team members
  - Pick at time to meet
- Do your preparation ahead of time
  - Add what you find to team/inspection.md
- Meet and discuss the items in inspection.md
  - Create github issues for that you need to address
  - Add the github issue numbers to team/inspection.md