CISC327 - Software Quality Assurance

Lecture 21

Inspection
E2 Bonus

• Bonus mark logistics:
  – Bonus marks are not equivalent to regular marks and have no pre-set weight
  – On exam 2, bonus marks are written in blue or green and are not included in the total
  – In December, I will total all bonus marks and decide how to translate them into an increase (if any) in your course grade
  – On exam 2, the highest anyone got was 4
Does this program halt?

```c
int main (int argc, char *argv[])
{
    return 0;
}
```
Does this program halt?

```c
int main (int argc, char *argv[])
{
    while (1) {
    }
    return 0;
}
```
Can you imagine a program that also gives those answers for those 2 programs?
Can you imagine a program that also gives those answers for those 2 programs?

```python
if C_program_input is literally "int main (...) { return 0; }")" then
    return "1" (halts)
else
    return "0" (doesn't halt)
```
Did we just solve the halting problem?
E2 Bonus

• What I was not looking for:
  – Noticing that I didn’t specify the encoding of the Turing machine or its input.
  – I’ve said “integer” on exams and you’ve all played along, even though I didn’t say whether it was a 32-bit, 64-bit, arbitrary-precision, or mathematical integer. The encoding of the TM is similarly irrelevant.
E2 Bonus

• Knowing anything about Turing machines probably **didn’t** help
  – one student wrote a “disclaimer” saying they hadn’t taken any courses about Turing machines... and then gave a substantially above-average answer!

• The undecidability of the problem is to a large extent a meaningless distraction: the CTO’s program **cannot be totally correct** but that does not mean it can’t be tested!
E2 bonus

• Could you do requirements testing?
  – Yes, you can partition the requirements:
    • one test case (M1, x1) for which M1 indeed halts on x1
    • one test case (M0, x0) for which M0 never halts on x0
  – Is this useful? The Bogosys CTO only needs his program to work for two cases (one halting, one looping) to claim he has passed these requirements tests. So probably not. But you can definitely partition the requirements and write those two tests.
E2 bonus

• Could you do input coverage testing?
  – I didn’t give the encoding of the TM but, in any case, there are lots and lots of Turing machines, so exhaustive input coverage isn’t going to work.
  – But could you partition the inputs? Not in a useful way. You could partition them by the length of the encoding string, but that has no clear relation to whether the machine halts.
  – So: **No**, you couldn’t do input coverage testing.
  – Note: A program to generate **useful** input partitions would **itself** solve the halting problem!
E2 bonus

• Could you do output coverage testing?
  – Yes! There are only two possible outputs, so we need two test cases. Really the same as requirements testing.
  – Would it be useful? Not particularly, because all it would show is that there are two cases for which the CTO’s program works. Doesn’t tell us anything about the billions of other cases!
E2 bonus

• Could you do path coverage testing?
  – Some answers: “No, because we don’t have the code.”
  – Some other answers: “Yes, because we have the code.”
  – The question didn’t say...
  – Common misconception that the CTO’s program consists of a single if statement.
By the way...

• Compilers include some approximation of Magic to do optimizations
  – all code following an infinite loop is dead and can be omitted from the executable

• Recognizing some non-halting programs does not mean the compiler recognizes all of them

• The CTO could, conceivably, write a program that recognizes lots of them
Here’s one idea

```c
int magic(M, x) {
    timeout = 10000;
    while (timeout > 0) {
        if (M is in the halted state)
            return 1;
        else {
            timeout--;
            step(M, x); // simulate 1 step
        }
    }
    // didn’t halt in 10,000 steps,
    // so assume it never will
    return 0;
}
```
Inspection Process

• Today we look at the inspection process
  – Steps in a formal inspection process
  – Example inspection documents
Inspection At Any Stage

• Inspections may be used at any stage of software development
  – Requirements, design, coding, testing, acceptance

• Ideally, inspections can be applied at every stage, to catch problems as early as they appear

• No matter what stage inspection is applied to, the inspection process is roughly the same
A Generic Inspection Process

- The basic process of formal inspection is always the same, no matter the artifact being inspected.

Planning
- Choose team, materials, schedule for inspection

Orientation
- Introduce artifact, process, goals to learn

Preparation
- Individually check artifact, note issues

Review Meeting
- Meet to discuss and consolidate issues

Rework
- Correct defects noted

Verify
- Verify artifact and process quality
(Recall) Inspection Roles

• **Moderator**
  – Chairs the meeting, *records* faults found
  – Helps others stick to the job, at the right *pace*
  – Keeps proceedings *objective*, professional, friendly

• **Inspectors (2 or 3)**
  – Knowledgeable *peers* who examine the artifact, in detail

• **Author**
  – Silent *observer* who assists or clarifies only when asked
Planning

- Objectives
  - Gather **review package**: artifact being inspected, references for it, checklists of inspection criteria, data sheets to record
  - Form inspection **team**
  - Set **schedule**
Planning

• Procedure
  – Moderator assembles team and review package
  – Moderator customizes checklist to artifact
  – Moderator plans schedule
  – Moderator checks artifact is ready for review
  – Moderator helps Author prepare overview of artifact
## Example Planning Document

<table>
<thead>
<tr>
<th>Planning</th>
<th>1. Inspection ID</th>
<th>Date:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2. Team</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Moderator</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Authors</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reviewers</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Documents</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Work Product</td>
<td></td>
</tr>
<tr>
<td></td>
<td>References</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Checklist</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Meetings</td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td>Orientation</td>
<td>Location</td>
</tr>
<tr>
<td></td>
<td>Review Meeting</td>
<td>Start</td>
</tr>
<tr>
<td></td>
<td></td>
<td>End</td>
</tr>
<tr>
<td></td>
<td>5. Planning Objectives</td>
<td></td>
</tr>
<tr>
<td></td>
<td>References obtained for work product.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Checklists obtained for work product.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Moderator is trained in TekInspect method.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Team members agree to proposed times/dates.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Moderator’s quick review yields less than 5 major issues.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reviewers understand responsibilities and are committed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6. Plan. Effort</td>
<td>min</td>
</tr>
</tbody>
</table>

(Johnson, U. Hawaii 2000)
Orientation Meeting

• Objectives
  – Author provides overview of artifact
  – Inspectors obtain review package
  – Preparation goals set
  – Inspectors commit to participating
Orientation Meeting

• Procedure
  – Moderator distributes review package
  – Author presents overview
  – Moderator outlines preparation procedure
Example Orientation Document

<table>
<thead>
<tr>
<th>Orientation</th>
<th>7. Prep. Goals</th>
<th>5 min/pg × 20 pg = prep time/revie...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8. Orient.</td>
<td>Reviewers understand scope and pur...</td>
</tr>
<tr>
<td></td>
<td>Objectives</td>
<td>Reviewers understand checking process, checklists, and references:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Work product, references, checklists, and checking forms provided</td>
</tr>
<tr>
<td></td>
<td>9. Orient. Effort</td>
<td>2 min/meet × 10 particip. = min</td>
</tr>
</tbody>
</table>

(Johnson, U. Hawaii 2000)
Preparation

• Objectives
  – Find the maximum number of non-minor defects in the artifact
Preparation

• **Procedure (for Inspectors only)**
  - Allocate scheduled time
  - Do detailed *individual inspection* of the artifact
  - Use *checklists* as a guide to focus on potential issues
  - Use *references* for calibration of what is expected or needed
  - Note *critical*, *severe*, and *moderate* level defects on reviewer report form
  - Note *minor* defects and questions for author clarification on artifact document
Example Defect Classification

• **Critical**
  – Defects that will cause the system to *hang, crash*, or produce *incorrect results* or behaviour, with no known workarounds

• **Severe**
  – Defects that will cause *incorrect results* or behaviour, but have known workarounds

• **Moderate**
  – Defects that affect limited areas of functionality that can either be worked around or ignored

• **Minor**
  – Defects that can be overlooked without loss of functionality
Example Checklists and References

• Checklists
  – Checklists often include questions concerning completeness, style, adherence to company standards, etc.
  – Code inspection checklists often include detailed questions about use of language features (e.g., no gotos), naming of variables, methods and classes, depth of nesting, etc.
Example Checklists and References

• References
  – May include:
    – Company standards documents
    – High quality examples of artifacts similar to the one being inspected
    – Chapters of reference textbooks on quality practice for artifacts
    – Online resources on quality practice for artifacts
Example Preparation Document

- Reviewer Report Form

![Reviewer Report Form Image]

(Johnson, U. Hawaii 2000)
Why Not Write On Artifact Directly?

• Advantages of Reviewer Report Form
  – Minor issues pre-filtered, saving review meeting time, focusing review meeting on **important** issues
  – Forces inspectors to write down issues clearly, saving meeting time
  – Defects can be considered in order of **importance**
  – Easy to gather inspection stats
Why Not Write On Artifact Directly?

• Disadvantages (?) of Reviewer Report Form
  – Requires more preparation time (15 minutes?)
  – Discourages last minute preparation
  – Makes quality of inspector preparation more visible
Review Meeting

• Objectives
  – Make consolidated, comprehensive list of non-minor defects to be addressed
  – Help provide group synergy
  – Help provide shared knowledge of artifacts
Review Meeting

• **Procedure**
  – Moderator requests defects sequentially, in order of *importance*
  – Inspectors point out defects found, compare notes
  – Moderator (or note taker) writes down consolidated list of defects found and summarizes results of meeting in *review summary defect report*
Example Review Summary Defect Report

(plus a detailed description of each defect)
Rework

• Objectives
  – Assess each defect listed in the review defect report, determine if really a defect, and repair as necessary
  – Written report on handling of each non-minor defect
  – Resolve minor issues as necessary and appropriate
Rework

• **Procedure (for Author)**
  - Author gets *review defect summary report* as well as marked-up copies of inspected artifact with details
  - Author assesses each defect, categorizes root cause and notes actions taken in an *author action report*
  - When finished, Author provides *author action report* and *reworked artifact* to Moderator for verification
Example Author Action Report

1. Inspection ID ________ 2. Document ____________ 3. Author ________

4. Issue Disposition
   Item  Fixed  Type  Explanation
   _____  _____  _____  ________________________________
   _____  _____  _____  ________________________________
   _____  _____  _____  ________________________________

5. Effort ________ min

6. Rework Objectives
   □ Outcome of all Review Meeting Data Sheet issues are noted on this form.
   □ All minor issues have been addressed.
   □ No known defects remain in the work product.

(Johnson, U. Hawaii 2000)
Verify

• Objectives
  – Assess reworked artifact quality
  – Assess inspection process
  – Pass or fail the artifact
Verify

• **Procedure (for Moderator)**
  
  – Obtain reworked artifact and author action report
  
  – Review reworked artifact and action report for remaining problems
  
  – Provide **recommendation** for artifact (pass / fail)
  
  – With inspectors, **sign off** on artifact
  
  – Compute summary **statistics** for inspection and archive review documents in quality database
  
  – Generate process improvement proposals (if any)
Summary

• **Inspection Process**
  - No matter what artifact of development is being inspected, inspection *process* is much the same
  - **Six steps**: planning, orientation meeting, preparation, review meeting, rework, verify

• **Reference**
  - O’Regan, Ch. 2.1-2.5
    “Overview of Fagan Inspections”

• **Next Time**
  - Inspections in practice: Code inspections