CISC 327 - Software Quality Assurance

Lecture 20

Inspection
Logistical Note

- **Two weeks** from now (Nov. 14th and 15th), we will be in a different lecture hall (due to Convocation)
- Check the course website for details
- I will try to remind you repeatedly during next week’s lectures
But first...

- E2 almost completely marked
- Haven’t entered totals yet but average pretty clearly lower than E1
- A “feature” of non-memorization-based testing is that you can have the right idea but flub the execution
THE MANAGEMENT RESPECTFULLY WISHES
TO CLARIFY THE FOLLOWING POINTS

• **Completeness criterion** for systematic testing: knowing when you have enough test cases, **not** when you have run the tests

• Even though my intended answers for parts (ii) and (iii) were nearly identical, you could get full marks by saying all the right things somewhere

• **Decision coverage**: not **one** test per decision

• **Recursion depth coverage**: not **one** test per recursive function/method

• Unexpected number of people missed functionality testing, but got full marks on decision coverage and recursion depth
THE MANAGEMENT RESPECTFULLY WISHES TO CLARIFY THE FOLLOWING POINTS

• On Q2, “requirements tests” was intended to mean **functionality** tests, but since many of you lost marks on A1 for missing boundary robustness tests, I understand why you thought you needed them here.

• I didn’t intend for you to write them, but if you did I ignored them.

• I did not ignore completely redundant tests, like “abc” / “a” and then “xyz” / “y”
THE MANAGEMENT RESPECTFULLY WISHES TO CLARIFY THE FOLLOWING POINTS

• On Q2(c), exhaustive output coverage, many people either missed the word “output” or didn’t make the connection from input to output (you can derive the number of possible outputs by reasoning about the number of inputs, but in general, “infinite” inputs alone does NOT imply “infinite” outputs: String.isEmpty)

• Many people claimed that the set of strings is infinite, which I forgave...
Uphill in the snow, both ways, finite

128K RAM = about $2^{(1 \text{ million})}$ possible RAM states

(All About Apple Museum, CC BY-SA 3.0)
128 bytes RAM = about $2^{1024}$ possible RAM states

Still a lot of strings to test...but not infinite.
THE MANAGEMENT RESPECTFULLY WISHES TO CLARIFY THE FOLLOWING POINTS

• Q3: A few people went overboard...
THE MANAGEMENT RESPECTFULLY WISHES TO CLARIFY THE FOLLOWING POINTS

• Q4:
  You’ve got to know when to hold ’em
  Know when to fold ’em

  You’ve got to know when to write more
  Know when to desist
THE MANAGEMENT RESPECTFULLY WISHES TO CLARIFY THE FOLLOWING POINTS

• Q4:
  You needed only three cases :-/

• The entire 0 = wildcard business was irrelevant to the answer, because it didn’t affect the output partitions at all!

• The question wants output partitions, not a complete, thorough set of black box test cases

• Each testing method has different strengths and weaknesses: know what each one does, and what it doesn’t do

• Thorough testing often involves applying multiple testing methods!
THE MANAGEMENT RESPECTFULLY WISHES TO CLARIFY THE FOLLOWING POINTS

• Q5(a), or, No Good Deed Goes Unpunished
  – I originally had wider indentation, but I made it narrower because I wanted to give you more room to draw a flow graph (or two (or three))
  – And I thought that braces around the inner if-elseif might be confusing; how wrong I was...
  – This led to the Great Misparsing (−1 mark)
THE MANAGEMENT RESPECTFULLY WISHES TO CLARIFY THE FOLLOWING POINTS

• Q5(b)
  – Generally good but suffered from a lack of specificity (writing “x < 0” instead of –1, etc.)
  – I respect the desire to think abstractly about the equivalence class of input, but that is not an actual test case that you could run (unless you’re using abstract interpretation technology!)
Inspections

- Today we begin our look at inspection as a quality assurance technique
  - Statistical Process Control
  - What is inspection?
  - Informal vs. formal inspection
  - Inspection in the software process
  - Inspection roles
  - Effectiveness of inspections vs. testing
Statistical Process Control

- Historical background: W.A. Shewhart
  - 1924: Bell wanted more reliable phone equipment → control chart
  - 1939: *Statistical Method from the Viewpoint of Quality Control*
  - Influential in US during the war → postwar Japan
First Law of Software Development

• Earlier is Cheaper
  – The later in the development cycle a fault is detected, the more expensive it is to fix
  • Methods that find faults earlier deliver more bang for the buck

(Phillips, RMC 1999)
Software Development Products

• What do we produce when making software?
  – Plans, procedures, requirements specifications, design specifications, source code, comments, test cases, test reports, user documentation, technical documentation

• Of all these, we can only test one of them (code), and only when we are already far along (at least partially runnable)

• So how can we discover and address quality and detect faults earlier?
Reviews, Walkthroughs, and Inspections

• Terminology
  – Unfortunately, there is no good agreement on precise definitions for these terms, but...

• Reviews
  – ...are the management practice of meetings to informally consider state of the project at certain stages, to gain confidence in project direction
    • e.g., preliminary design review, critical design review
  – Used to provide confidence that the design is sound
  – Often attended by management and customers
Reviews, Walkthroughs, and Inspections

• Walkthrough
  – ...refers to an informal technical review, normally carried out by developers
  – Used by development teams to improve product quality by involving whole team in quality assurance at each stage
  – Focus is on critical analysis of artifacts, in an attempt to find or predict defects
Reviews, Walkthroughs, and Inspections

• Inspection
  – ...refers to a completely *formal* process of review, also known as *formal technical reviews*
  – A formal system used to identify and remove *defects*, and improve the overall *quality* of the development process
  – *Involves*: Formal written *reports*, defect data collection and *analysis*, required standards and measures
    • Emphasis on *documenting* process progress and defects
  – First introduced by *Fagan* (IBM) about *1976*, now *required* by some customers (e.g., U.S. military)
Inspections in the Software Process

- Requirements definition
- System and software design
- Implementation and unit testing
- Integration and system testing
- Operation and maintenance

- Requirements Review
- Design Inspection
- Code Inspection
- Functional Audit
Kinds of Inspections

• A Generic Technique
  – Inspections can assist at every stage, the earlier the better
  – E.g., U.S. Mil-Std-1521B, "Technical Reviews and Audits for ... Computer Software" identifies 10 separate kinds to be carried out

<table>
<thead>
<tr>
<th>System requirements review (SRR)</th>
<th>Test readiness review (TRR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>System design review (SDR)</td>
<td>Functional configuration audit (FCA)</td>
</tr>
<tr>
<td>Software specification review (SSR)</td>
<td>Physical configuration audit (PCA)</td>
</tr>
<tr>
<td>Preliminary design review (PDR)</td>
<td>Formal qualification review (FQR)</td>
</tr>
<tr>
<td>Critical design review (CDR)</td>
<td>Production readiness review (PRR)</td>
</tr>
</tbody>
</table>
Example: PDR

• 3.4 Preliminary Design Review (PDR).
  – This review shall be conducted for each configuration item or aggregate of configuration items to
    • (1) evaluate the progress, technical adequacy, and risk resolution (on a technical, cost, and schedule basis) of the selected design approach,
    • (2) determine its compatibility with performance and engineering specialty requirements of the Hardware Configuration Item (HWCI) development specification,
    • (3) evaluate the degree of definition and assess the technical risk associated with the selected manufacturing methods/processes, and
    • (4) establish the existence and compatibility of the physical and functional interfaces among the configuration item and other items of equipment, facilities, computer software, and personnel.
Example: PDR

• 3.4 Preliminary Design Review (PDR).
  – ...
  – For CSCIs, this review will focus on:
    • (1) the evaluation of the progress, consistency, and technical adequacy of the selected top-level design and test approach,
    • (2) compatibility between software requirements and preliminary design, and
    • (3) on the preliminary version of the operation and support documents.
The Prevention Principle

Prevention is better than cure.

OR

An ounce of prevention is worth a pound of cure.
Cost of Fixing Errors

- **3-6X**
- **10X**
- **15-40X**
- **30-70X**
- **40-1000X**

**Point at which error is fixed**

**Relative cost to fix an error**

- Requirements
- Design
- Coding
- Development Testing
- Acceptance Testing
- Operation
Inspection

• **IEEE Definition** of Inspection
  – "... a **formal** evaluation technique in which software requirements, design, or code are examined in detail by a person or group other than the author to detect faults, violations of development standards, and other problems..."

• **IEEE Objective** of Inspection
  – "... to detect and identify software element defects. This is a rigorous, formal peer examination..."
Inspection

• Verifies that the software elements satisfy its specifications
• Verifies that the software elements conform to applicable standards
• Identifies deviations from standards and specifications
• Collects software engineering data (for example, defect and effort data)
• Does not examine alternatives or stylistic issues
Inspection

• But Inspection (capital i) is a formal process!
  – One study found that 84% of surveyed organizations performed reviews or inspections, but 0% performed inspections entirely correctly
  – Even a walkthrough or a poorly done Inspection can be effective at improving software quality
  – Inspection is not only about defect correction, but also importantly about defect prevention
Fagan Inspections (e.g., for Code)

- Source code listings
- Overview meeting
- Development documents
- Trained experts

Line-by-line paraphrasing at 150 lines/hour

- Rework
- Defect statistics

Prevention Database

Inspected source code for testing

(Phillips, RMC 1999)
Inspection Roles (Fagan, Code Inspection)

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

• **Inspectors (2 or 3)**
  – Knowledgeable *peers* who paraphrase the code, line by line
  – Must have attended *overview* meeting, reviewed *requirements* and *design* documents, must understand *context* of code

• **Author**
  – Silent *observer* who assists or clarifies only when asked
Choosing Inspectors (Fagan)

- **Good Choices**
  - Review specialists (e.g., QA analysts)
  - Technical people from the same team as author
  - Technical people with special expertise in subject matter of code
  - People with a special interest in the product
  - People from other parts of the org. or outside it

- **Bad Choices** *(exclude!)*
  - Managers, supervisors, or appraisers of the author
  - Anyone with a personality clash with the author or other reviewers
  - All management
  - Anyone with a conflict of interest
Inspection Efficiency

(Phillips, RMC 1999)
Side Benefits of Inspection

• Cultural
  – Team members gain a **broader perspective** on the software system as they review each other’s work
  – Promotes a shared "**quality culture**", joint responsibility

• Organizational
  – Coding **standards** and practices are learned and enforced
  – Consistency improves

• Educational
  – Quality **improves** over time, as authors become more aware of the kinds of faults they are **prone to make**
Inspection in Context

- Static faults
- Execution faults
- Interaction faults

Faults

- Inspection
- Unit Test
- System Test
Summary

• Inspections, Walkthroughs, and Reviews
  – Designed to catch faults earlier than possible using testing, to reduce costs and increase quality
  – Informal or formal meetings in which reviewers examine work of authors in detail
  – Very effective in practice

• References
  – Gilb & Graham, Ch. 3, "Overview of Software Inspection"

• Next time
  – Inspection processes