SOFTWARE SYSTEM QUALITY

LECTURE # 10

STATIC TESTING

chenbo@etao.net

Summary of Previous Lecture

- Quality Assurance
- Qualification Scheme for Quality Assurance
  - Defect prevention
  - Defect Reduction
  - **Defect Containment**
- Software Fault Tolerance Techniques
  - Design Diversity technique
  - Data Diversity technique
- Failure Containment
  - Fault Tree Analysis

Today's Lecture

- Software Fault Tolerance Techniques
  - Design Diversity technique
  - Data Diversity technique
- Failure Containment
  - Fault Tree Analysis

Topics to Cover

- Introduction of static testing
- Objective of static testing
- Static Testing Methods
  - Inspections
  - Walkthroughs
  - Desk Checking
  - Peer Ratings

Introduction

- In software development, static testing, also called *dry run testing*, is a form of software testing where the authors manually read their own documents/code to find any errors.

- It is generally not detailed testing, but primarily syntax checking of the code/document and/or manually reviewing the code or document to find logic errors also.

- The term “static” in this context means “not while running” or “not while executing”

Objectives of Static Testing

- Static testing is the least expensive form of testing and has the largest potential for reducing defects in software under development.
- The primary objective of static testing is defect reduction in the software by reducing defects in the documentation from which the software is developed.

Static Testing Approach

- Consider using a two-step approach to static testing.
  - For the first step, clean up the cosmetic appearance of the document: check spelling, check grammar, check punctuation, and check formatting.
- The benefit of doing the first step is that when the document is cosmetically clean, the readers can concentrate on the content.
- The liability of skipping the first step is that if the document is not cosmetically clean, the readers will surely stop reading the document for meaning and start proofreading the content.
  - For the second step, use whatever techniques seem appropriate to focus expert review on document contents.

Static Testing Techniques

- Inspections

- Desk checking

- Walk-throughs

- Peer Ratings

Inspection

- Fagan Inspection
- Gilb Inspection
- Two Person Inspection
- N-Fold Inspection
- Meetingless Inspection

Generic Inspection Process

Generic process/steps:

1. Planning and preparation (individual)
2. Collection (group/meeting)
3. Repair (follow up)

Fagan Inspection

- Fagan inspection refers to a structured process of trying to find defects in documents such as programming code, specifications, designs and others during various phases of the software development process.
- It is named after Michael Fagan who is credited with being the inventor of formal software inspections.
- Fagan Inspection is a group review method used to evaluate output of a given process.
- In the process of software inspection the defects which are found are categorized in two categories:
  - Major Defects
  - Minor defects

Fagan Inspection

- The defects which are incorrect or even missing functionality or specifications can be classified as major defects: the software will not function correctly when these defects are not being solved.

- In contrast to major defects, minor defects do not threaten the correct functioning of the software, but are mostly small errors like spelling mistakes in documents or optical issues like incorrect positioning of controls in a program interface.

Fagan Inspection

Typical operations

In a typical Fagan inspection the inspection process consists of the following operations:

- Planning
  - Preparation of Materials
  - Arrangements of Participants
  - Arrangement of meeting place
- Overview
  - Author-Inspector Meeting
  - Assignment of Roles
- Preparation or Individual Inspection
  - Independent Analysis/examination
  - Code as well as other documents

Fagan Inspection

- ✓ Individual results:
  - ✓ Questions
  - ✓ potential defects
- ✓ The participants prepare their roles

❑Inspection meeting

- ✓ Meeting to collect/consolidate individual inspection results
- ✓ Defect identification, but not solutions, to ensure inspection effectiveness
- ✓ Fagan inspection typically involves about 4 people in the inspection team
- ✓ No more than 2 hours
- ✓ Inspection report

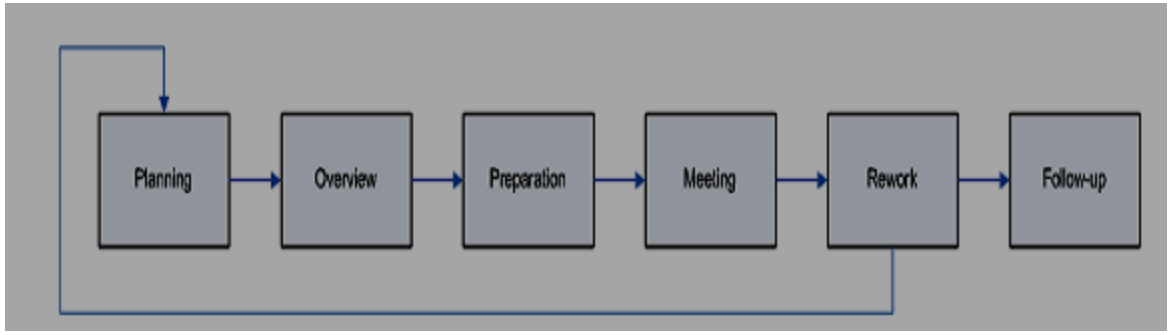
❑Rework (performed by the author)

- ✓ Rework is the step in software inspection in which the defects found during the inspection meeting are resolved by the author, designer or programmer.

Fagan Inspection

❑Follow-up

- ✓ In the follow-up phase, defects fixed in the rework phase should be verified.
- ✓ The moderator is usually responsible for verifying rework. Sometimes fixed work can be accepted without being verified, such as when the defect was trivial. In non-trivial cases, a full re-inspection is performed by the inspection team (not only the moderator).
- ✓ If verification fails, go back to the rework process.



Gilb Inspection

1. Planning (same to Fagan inspection)
2. Kickoff (Overview)
3. Individual Checking (Preparation)
4. Logging Meeting (Inspection)
5. a) Edit (Rework) b) Process brainstorming
6. Edit Audit (Follow-up)

- Process brainstorming is added right after the inspection meeting. The focus of this step is root cause analysis aimed at preventive actions and process improvement in the form of reduced defect injections for future development activities
- The team size is typically about four to six people
- Checklists are extensively used, particularly for step 3

Inspection Session

- During the session, two activities occur:
  1. The programmer narrates, statement by statement, the logic of the program. During the address, other participants should raise questions, and they should be followed to determine whether errors exist. It is likely that the programmer rather than the other team members will find many of the errors found during this narration.
  2. The program is analyzed with respect to a checklist of historically common programming errors.
- The moderator is responsible for ensuring that the discussions proceed along productive lines and that the participants focus their attention on finding errors, not correcting them. (The programmer corrects errors after the inspection session.)

Inspection Session

- The ideal time for the inspection session appears to be from **90 to 120 minutes.** Since the session is a mentally taxing experience, longer sessions tend to be less productive.
- Most inspections proceed at a rate of approximately **150 program statements per hour.**
- For that reason, large programs should be examined in multiple inspections, each inspection dealing with one or several modules or subroutines.
- Note that for the inspection process to be effective, the appropriate attitude must be established. If the programmer views the inspection as an attack on his or her character and adopts a defensive posture, the process will be ineffective. Rather, the programmer must approach the process with an ego less attitude so the session can be productive

Two Person Inspection

- Some software artifacts are small enough to be inspected by one or two inspectors
- Similarly, such reduced size inspection teams can be used to inspect software artifacts of limited size, scope or complexity
- The so called Two Person Inspection was proposed to simplify the Fagan inspection, with an Author-Inspector pair
- This technique is cheaper and more suitable for smaller scale programs, small increments of design and/or code in the incremental development, or other software artifacts of similarly smaller size
- A typical implementation of two-person inspection is the reversible author-inspector pair
- This technique is easier to manage because of the mutual benefit to both individuals

Meeting less Inspection

- Experimental evidences indicates that most of the discovered defects are indeed discovered by individual inspectors during the preparation step of Formal Inspections like Fagan and Gilb
- The defect detection ratio in the meeting session lies in the range of 5% to 30%
- Therefore there is a possibility of eliminating inspection meetings entirely, thus significantly reducing the overall inspection cost
- This results in a so called meetingless inspection, where individual inspectors do not communicate with each other

Meeting less Inspection

- One of the main drawback of this approach is the high False Alarm rate
- Another drawback of this approach is duplication of errors
- Various ways of communication can be used to pass the individual inspection results to the author, e.g through direct communication with the author, or through some defect repository

N-Fold Inspection

- Tsai et al. [1], developed the N-fold inspection process, in which N teams each carry out independent inspections of the entire artifact.
- N-Fold inspection uses formal inspections but replicates these inspection activities using N independent teams.
- The same software artifact (e.g., a URD (User Requirements Document) is given to all N teams.
- An appointed moderator supervises the efforts of all N teams.
- N-fold inspections will find more defects than regular inspections as long as the teams don't completely duplicate each other's work.
- However, they are far more expensive than a single team inspection.

N-Fold Inspection

- Each team performs formal inspection using a checklist and analyzes the software artifact.
- Several teams may identify the same fault, but the moderator gathers all results of the independent inspection efforts and records each fault once in a database.

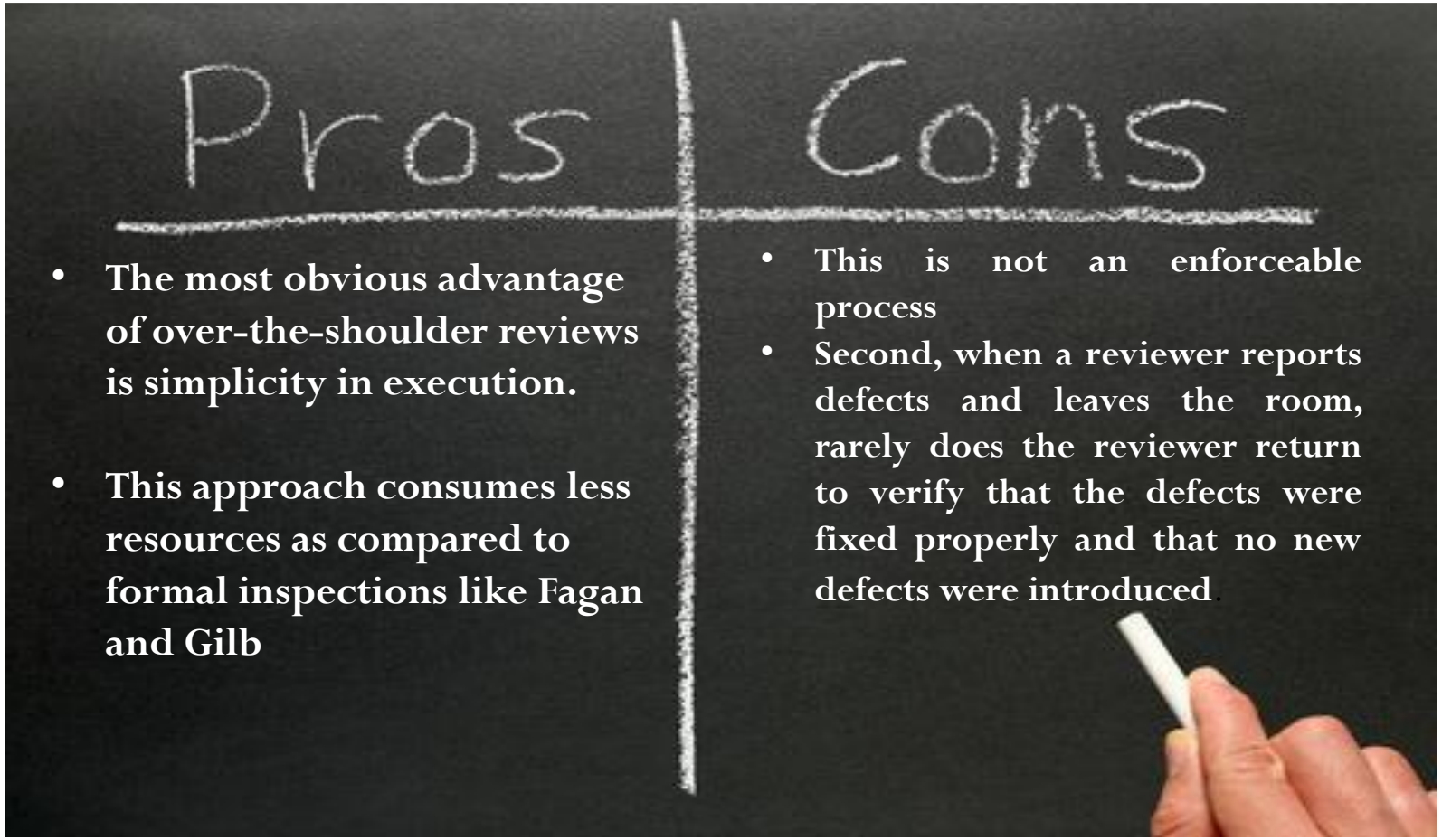
Over the Shoulder Reviews

- “over-the-shoulder” review is an informal code review technique
- An “over-the-shoulder” review is just that – a developer standing over the author’s workstation while the author walks the reviewer through a set of code changes.
- Typically the author “drives” the review by sitting at the keyboard and mouse, opening various files, pointing out the changes and explaining why it was done this way.
- The author can present the changes using various tools and even run back and forth between changes and other files in the project.
- If the review sees something a miss, they can engage in a little “spot pair programming” as the author writes the fix while the reviewer waits and watch the fix.
- Bigger changes where the reviewer doesn’t need to be involved are taken off-line.

Over the Shoulder Reviews

- With modern desktop-sharing software a so-called "over-the shoulder" review can be made to work over long distances.
- This complicates the process because you need to schedule these sharing meetings and communicate over the phone.
- Standing over a shoulder allows people to point, write examples, or even go to a whiteboard for discussion; this is more difficult over the Internet. Many of the face to face interactions are lost in this case

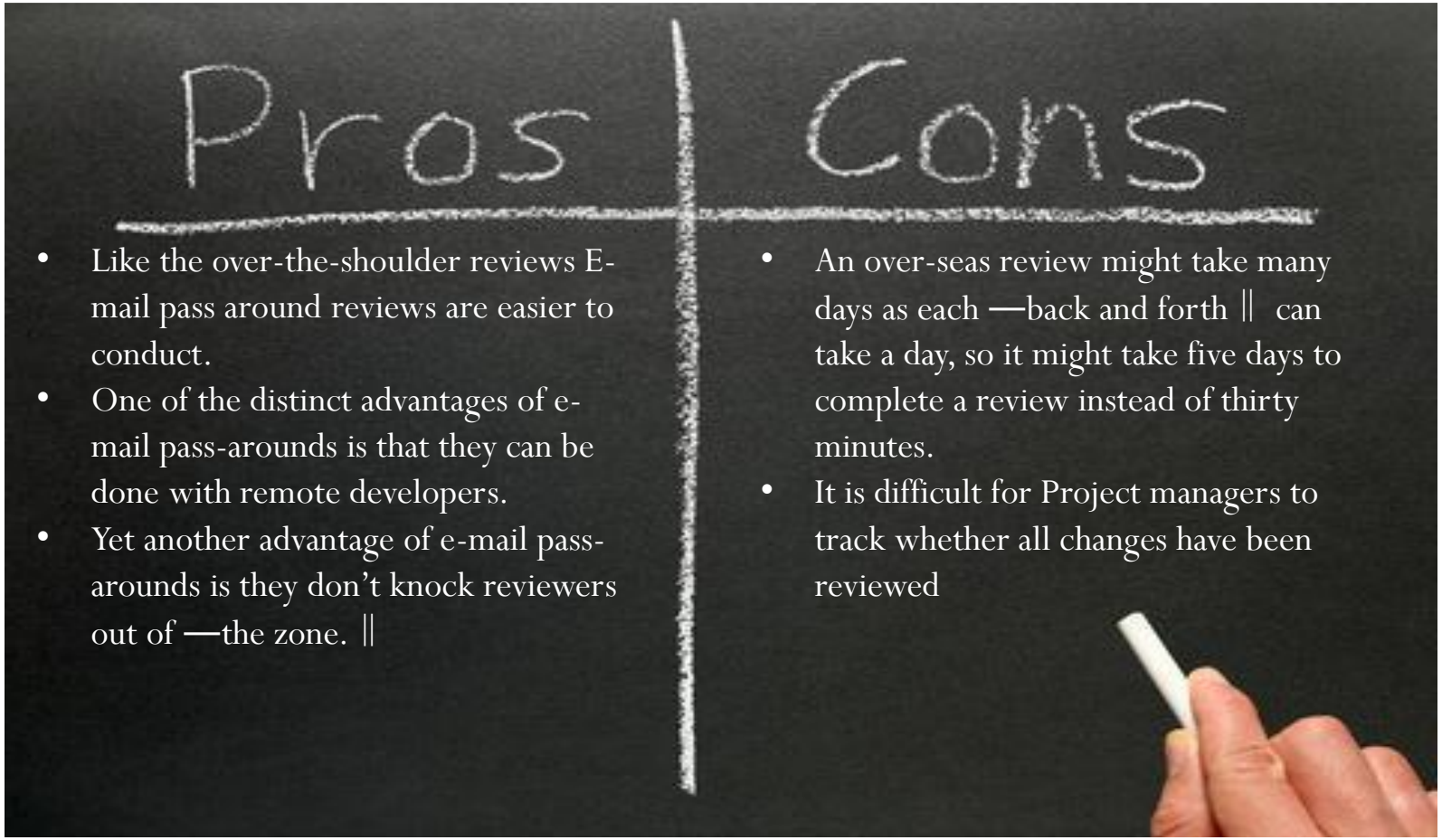
Over the Shoulder Reviews



E-mail pass-around reviews

- This is the second-most common form of informal code review.
- Here, whole files or changes are packaged up by the author and sent to reviewers via e-mail.
- Reviewers examine the files, ask questions and discuss with the author and other developers, and suggest changes.

E-mail pass-around reviews



Desk Checking

- Desk Checking is one of the older practice of human error-detection process. A desk check can be viewed as a one-person inspection or walkthrough: A person reads a program, checks it with respect to an error list, and/or walks test data through it.
- In other words you can say Manually testing the logic of a program.

Desk Checking Process

- Desk checking involves first running a spellchecker, grammar checker, syntax checker, or whatever tools are available to clean up the cosmetic appearance of the document.
- Then, the author reviews the document trying to look for inconsistencies, incompleteness, and missing information.
- Problems detected in the contents should be corrected directly by the author with the possible advice of the project manager and other experts on the project.
- Once all corrections are made, the cosmetic testing is rerun to catch and correct all spelling, grammar, and punctuation errors introduced by the content corrections.

Desk Checking Drawbacks

- Desk checking is the least formal and least time-consuming static testing technique.
- Of all the techniques, desk checking is the only one whereby the author test his or her own document.
- For most people, desk checking is relatively unproductive.
  - ✓ One reason is that it is a completely undisciplined process.
  - ✓ A second, and more important, reason is that it runs counter to a testing principle ("that people are generally ineffective in testing their own programs"). For this reason, you could deduce that desk checking is best performed by a person other than the author of the program.

Code Walkthrough

- The code walkthrough, like the inspection, is a set of procedures and error-detection techniques for group code reading.
- It shares much in common with the inspection process, but the procedures are slightly different, and a different error-detection technique is employed.
- Like the inspection, the walkthrough is an uninterrupted meeting of one to two hours in duration.
- The walkthrough team consists of three to five people.
- One of these people plays a role similar to that of the moderator in the inspection process, another person plays the role of a secretary (a person who records all errors found), and a third person plays the role of a tester.

Code Walkthrough

- Suggestions as to who the three to five people should be vary.
- Of course, the programmer is one of those people. Suggestions for the other participants include:
  1. a highly experienced programmer
  2. a programming-language expert
  3. A new programmer (to give a fresh, unbiased outlook)
  4. the person who will eventually maintain the program

Code Walkthrough

- The initial procedure is identical to that of the inspection process:
- The participants are given the materials several days in advance to allow them to bone up on the program.
- However, the procedure in the meeting is different. Rather than simply reading the program or using error checklists, the participants "play computer".
- The person designated as the tester comes to the meeting armed with a small set of paper test cases—representative sets of inputs (and expected outputs) for the program or module.

Code Walkthrough

- During the meeting, each test case is mentally executed. That is, the test data are walked through the logic of the program. The state of the program (i.e., the values of the variables) is monitored on paper or whiteboard.
- Of course, the test cases must be simple in nature and few in number, because people execute programs at a rate that is many orders of magnitude slower than a machine.
- The walkthrough should have a follow-up process similar to that described for the inspection process.

Peer Ratings

- Peer rating is a technique of evaluating anonymous programs in terms of their overall quality, maintainability, extensibility, usability, and clarity. The purpose of the technique is to provide programmer self-evaluation.
- A programmer is selected to serve as an administrator of the process. The administrator, in turn, selects approximately 6 to 20 participants(6 is the minimum to preserve anonymity). The participants are expected to have similar backgrounds (you shouldn't group Java application programmers with assembly language system programmers, for example).
- Each participant is asked to select two of his or her own programs to be reviewed. One program should be representative of what the participant considers to be his or her finest work; the other should be a program that the programmer considers to be poorer in quality.
- Once the programs have been collected, they are randomly distributed to the participants.

Peer Ratings

- Each participant is given four programs to review. Two of the programs are the —finest ‖ programs and two are" poorer ‖ programs, but the reviewer is not told which is which.
- Each participant spends 30 minutes with each program and then completes an evaluation form after reviewing the program. After reviewing all four programs, each participant rates the relative quality of the four programs. The evaluation form asks the reviewer to answer, on a scale from 1 to 7 (1 meaning definitely “yes”, 7 meaning definitely “no”),
- The reviewer also is asked for general comments and suggested improvements.

Peer Ratings

- After the review, the participants are given the anonymous evaluation forms for their two contributed programs. The participants also are given a statistical summary showing the overall and detailed ranking of their original programs across the entire set of programs, as well as an analysis of how their ratings of other programs compared with those ratings of other reviewers of the same program.
- The purpose of the process is to allow programmers to self-assess their programming skills.

References

1. Software Engineering by Roger S Pressman
2. http://en.wikipedia.org/wiki/Revision_control
3. http://project-management-knowledge.com/definitions/c/change-control-boardccb/
4. *http://en.wikipedia.org/wiki/Change_control_board*
5. *http://en.wikipedia.org/wiki/List_of_revision_control_software*
6. *http://en.wikipedia.org/wiki/Distributed_revision_control*