Static Testing

Saeed Siddik

IIT University of Dhaka

What is Static Testing?

- Static testing techniques do not execute the software and they do not require the bulk of test cases.
- Purpose: Early detection of errors, improving code quality and reducing cost of fixes
- Example: A developer performs a code review to check for syntax errors, adherence to coding standards, and potential logic flaws before compiling the code

Why Static Testing ?

- The efficiency of code coverage decreases with the increase in size of the system
- Dynamic testing is expensive and time-consuming, as it needs to create, run, validate, and maintain test cases.
- While dynamic testing is an important aspect of any quality assurance program, it is not a universal remedy.
- static testing is a complimentary technique to dynamic testing technique to acquire higher quality software

Types of Static Testing

- Software inspections
- Walkthroughs
- Technical reviews

Software inspections

- Instead of observing how the software behaves when executed, inspections focus on logic, structure, completeness, and clarity of what's written or designed.
- Inspections rely on structured peer reviews where a team examines software artifacts to uncover defects, inconsistencies, or non-compliance with standards.
- These artifacts can include:
  - Source code (uncompiled)
  - Design diagrams
  - Requirements specifications
  - Interface definition

Inspection Process

Planning: Define scope and prepare materials.

Overview: Brief reviewers on objectives and context.

Preparation: Reviewers study the material independently.

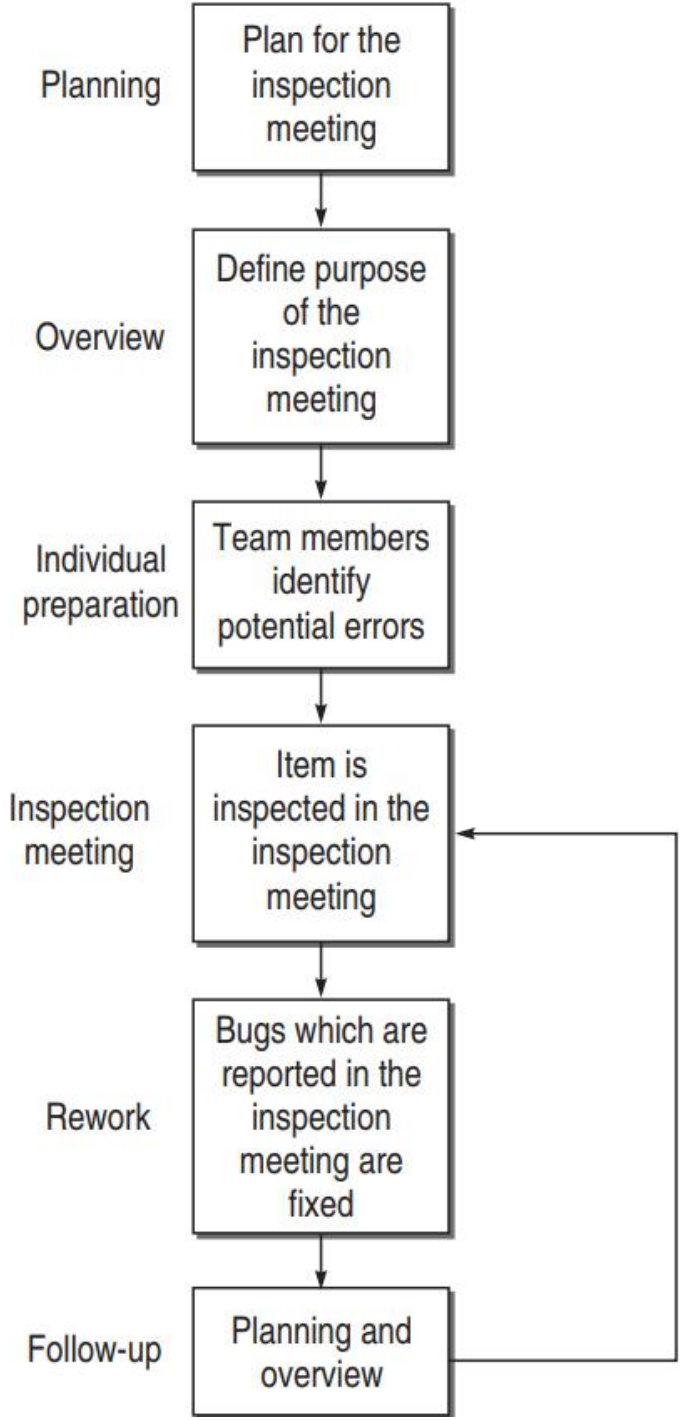
Inspection Meeting: Group discussion led by a moderator, where issues are logged.

Rework: Author fixes identified defects.

Follow-up: Moderator verifies corrections.

Inspection Process

A general inspection process has the following stages.



Practical Example

Suppose we are reviewing a Python function that calculates variance. Instead of running the code, you:

- Check if the formula is correctly implemented.
- Ensure variable naming is clear.
- Verify that edge cases (e.g. empty list) are handled.
- Confirm adherence to your coding standards.

If there's a missing check or a logic flaw in the formula, you can catch it early—**before anyone presses "Run."**

Example Snapshot: Variance Function in Python

```
def calc_variance(data):
    mean = sum(data) / len(data)
    return sum((x - mean)**2 for x in data) / len(data)
```

- Logic is correct for population variance.
- No error handling if data is empty (ZeroDivisionError risk).
- No docstring.
- Variable naming is clear.
- Could benefit from comment explaining why `len(data)` isn't `(len(data)-1)`

Code Inspection Checklist (Logic & Functionality)

- [ ] Does the algorithm implement the correct logic or formula (e.g., correct variance equation)?
- [ ] Are all inputs, edge cases, and expected conditions accounted for?
- [ ] Is error handling in place for invalid or missing data?

Code Inspection Checklist (Code Clarity & Style)

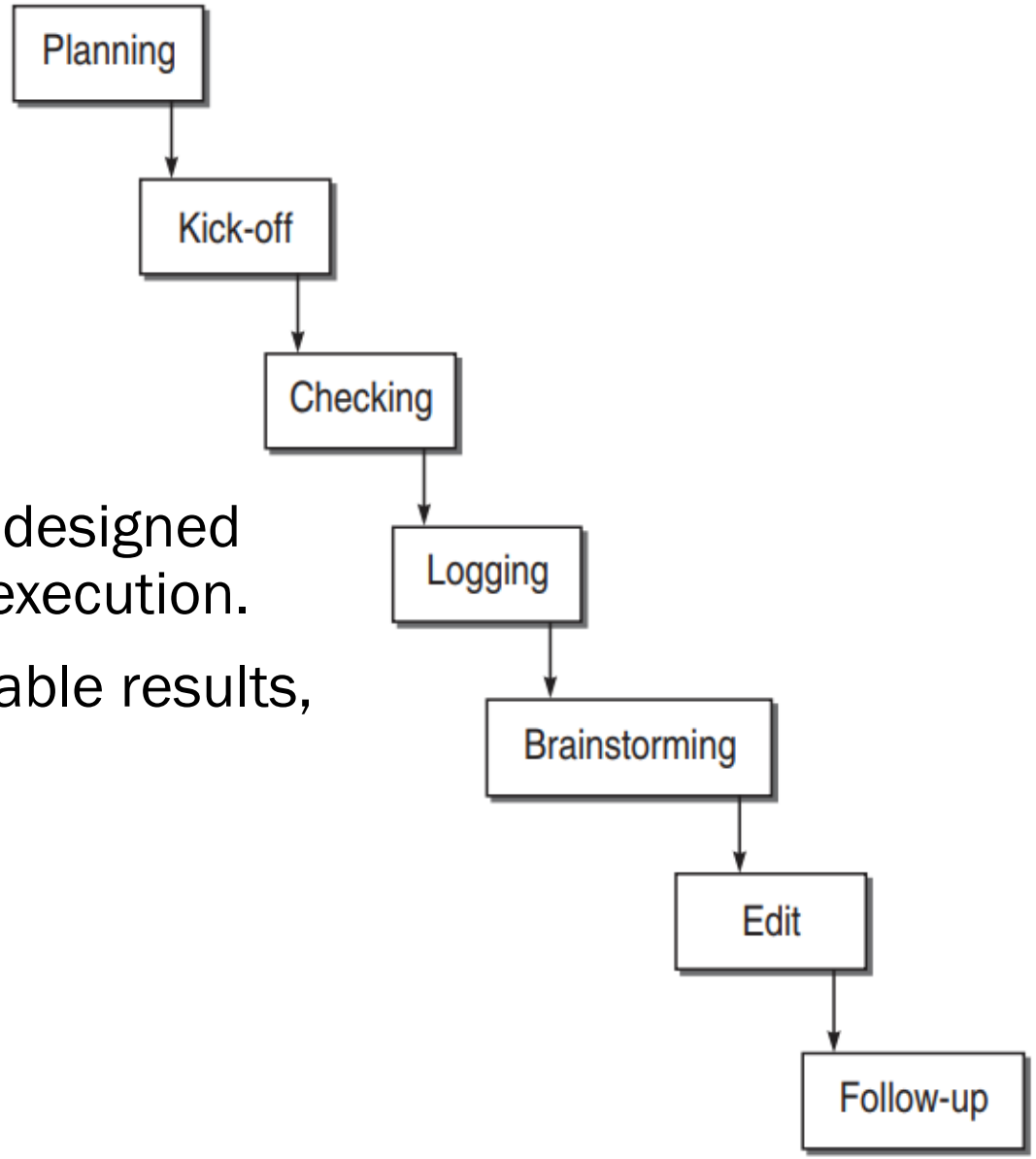
- [ ] Are code written by following standard coding style like PEP-8 for Python?
- [ ] Are variable names meaningful and consistent?
- [ ] Are functions and blocks logically organized and modular?
- [ ] Is indentation and formatting neat and according to style guidelines?
- [ ] Are comments clear, concise, and helpful not redundant?

Finding most error-prone modules

- Through the inspection process, the modules can be analysed based on error-density of the individual module
- From the example modules, Module C is more error prone. This information can be used and some decision should be taken as to:
  (i) Redesign the module
  (ii) Check and rework on the code of Module C
  (iii) Take extra precautions to test the module

Module Name	Error density (Error/ KLoC)
A	23
B	13
C	45

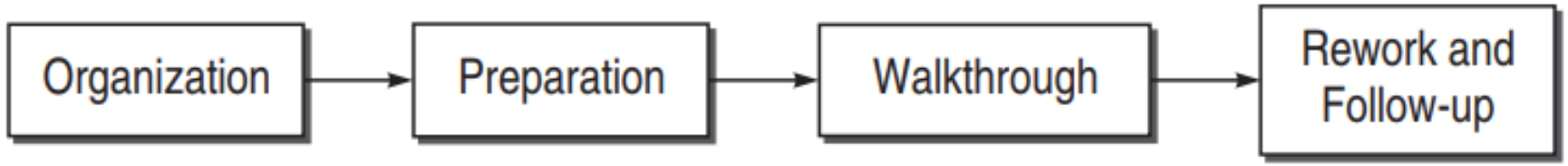
Variants of Inspection (Gilb Inspection)



- A formal team-based static testing technique designed to detect defects in software artifacts *before* execution.
- It emphasizes early defect detection, quantifiable results, and continuous process improvement

Structured Walkthrough

- A static quality assurance method where a team of peers systematically reviews a software artifact such as requirements, design, code, or test plans to identify defects and improve overall quality.
- Unlike inspections, walkthroughs are typically less formal, but still follow a defined structure
- A walkthrough is less formal, has fewer steps and does not use a checklist to guide to document the team's work



Technical Reviews in Static Testing

- to evaluate the software in the light of development standards, guidelines, and specifications
- to provide the management with evidence that the development process is being carried out
- although it is similar to an inspection or walkthrough, it is considered a higher-level formal techniques.
- a technical review team is generally comprised of management-level representatives and project management.

Sample Review Finding

- **Issue:** Data flow between modules lacks clarity
  Suggestion: Add sequence diagrams to illustrate interactions
- **Issue:** Naming conventions inconsistent across modules
  Suggestion: Align with organization’s coding standards
- **Issue:** No fallback mechanism for failed API calls
  Suggestion: Include retry logic and error handling strategy

Difference between inspection, walkthrough, and technical reviews

Aspect	Inspection	Walkthrough	Technical Review
Formality	Most formal	Less formal, semi-structured	Varies, often less formal than inspection
Leader	Trained Moderator (not author)	Author of the work product	Technical expert or trained moderator
Process	Highly structured, specific roles, phases	Author-driven presentation, discussion	Content-focused examination by experts
Primary Goal	Find as many defects as possible	Achieve common understanding, gather feedback	Ensure technical accuracy and consistency
Preparation	Extensive for all participants	Little to none for participants	Moderate for participants
Output	Formal defect log, action items	Feedback, suggestions, shared understanding	Technical recommendations, identified issues

Thank you