**3. Static Testing (Human Testing)**

Basic Ideas,

Pros/Cons;  
Variants: Fagan Inspection, Walkthrough, Desktop Testing, Reviews;  
Typical Examples Of Test Criteria (Checklists);  
Demarcation From Static Analysis;

3.1-Basic Ideas,

3.1.1- A initial of History:

[Tagern 76]: Design and local **Inspection** to reduce error in program development.

[Weinbug 71]: The psychology of computer programming:

Program should be written by peoples, too.

Program are written solely for machine execution.

Why? :

* People make mistakes.
* Group together can identify additional defects in respect of found by individuals inspector.

3.2-Pros/Cons;

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| * Group compensate the **human weaknesses**   + High diversity in the group   + Peer programming * Faster   + Avoid preparation step/immediate start   + All kind of artifacts * Ability   + Readability, understandability, etc. * Allows to incorporate customer very early * More detailed feedback   + Bunch of mistakes discovered at once. * Breaking of isolated programming * Learning from each other * Quality ensures by entire team * Phantom inspector | * Human weakness * Long term investment,   + Cost, money, etc. * Results might be misused for staff appraisal. |

3.2.2 - Human Weaknesses:

* Spot light principal: If we are working on small subset on all the problem in hand.
* Economy principal: Typically good enough make decision on small set of information
* Conciseness principal: Contradict if you want to get robust and reliable software
* Linear causality thinking: Not good enough for program
* Overrating of confirmed info: Our expectation can be wrong with few test.
* Wrong associations: We believe the association is right but it is actually wrong
* General attitudes: Every human will going to behave with his past experience
* Learning ability: Which this acquires knowledge is correct or not.
* Identification without own work: I design this program which means it is everything correct
* Limited abilities to concentrate and relocated: If we sleepy and make error which finding will not be ever happen

3.3-Static & Dynamic Testing,

Design

Formal Specification

Program

High Level Design

Requirement Specification

Static testing = Human Testing:

* without executing a code by computer,
* Early discovery, cost effective, software quality criteria.

Static testing effectiveness:

* More than 60% of program error.
* Mathematical error 90%

3.4-Variants: Fagan Inspection, Walkthrough, Desktop Testing, Reviews;

3.4.1-Program Inspection: process how to preform inspection reading of the group **only discover of misbehavior** but not for the correction and find solution to remove the **defect**, such as logical error, anomalies code, or non-compliance with coding standards.

3.4.2-Inspection of precondition:

What our expectation:

* precise specification
* Familiar organization standards
* Syntactically correct code
* Error checklist guideline
* Increase cost
* Find/discover many bug use inspection.
* Management should not directly involved in the inspection term and not use the inspection result of staff appraisal.

3.4.2.1 Inspection process:

Follow-up

Remarks

Inspection meeting

Overview

Planning

Individual preparation

3.4.2.2 Inspection Procedure: depend / evolve from overtime from **checklist**.

3.4.2.3 Inspection Team:

* 4 member, Author of the code,
* Reader who read code but not the author,
* Attention (concentrate) who discovering the defect error,
* Moderator – discovering errors
* No supervisor

3.4.2.4 Inspection Rate:

* 500 statement/hours
* Expensive extremely
* 500 lines cost 40 man/hours = $2800, extremely expensive.

3.4.2.5 Inspection Checklists:

Complement static semantic checks by only compiling not by executing.

Inspection coding guideline/programming guidelines. Uniform coding convention within the team; like who to start line, logic, etc.

* Data Fault:
  + Variable initialized
  + Have all constants been named?
  + Logical mistake which go over for loop; upper & lower bound is always 0,1 other input.
* Control fault:
  + Addition condition is missing/condition works correctly?
  + Each loops terminate?
  + Compound statements correctly bracketed?
    - If(C1)

then { S1}

endif

* + Check all possibility.
* Input/output fault?
* Interface faults: call function and use wrong parameter use/swap in function.
* Storage management fault: use pointer to store management.
* Exception management fault: to detect the fault in exception.

3.4.3 Program style:

1. Break the code/subtask in module.
2. Keep code in such a way written in one page
3. What are use function, procedure, etc.; function has reached
4. No assumption about how the modular could be called
5. How to use the module, telling how module supposed to be used for.
6. One should give comment what the program would give to me?
7. What the program should in given order.
8. No direct value should use in program like, 10000 but use a name of the value.
9. For comment you need to trade-off between code and comment.

3.4.4 Design and code inspection Fagan:

(2) Walk through: 2-3 persons. “Play Computer”

* Execute a program manually
  + symbolically
* Use specific test cases
* Document state of the program (variable value)
  + Implemented logic correct or not
* Used to start discussion about the software
* Applied to complex algorithm
  + Algorithm used extremely often
  + Algorithm with high dependability demands.
* Algorithm where are experts may mistakes

(3) Desk Checking:

* 1 – Person walk through.

(4) Review:

* 1 day
* 10-20 people involve, managers, customers involves.
* Get miles stones accepted.

Reference:

# Class Lecture Notes, Prof. Dr.-Ing. M. Heiner

# [Source] Software Testing Tutorials - ISTQB Certification: Chapter 3.1 - Static Testing, <https://www.youtube.com/watch?v=lObaWlwHefc>

# [Source] Software Testing: - Chapter Static testing, https://www.youtube.com/watch?v=Y27ylzxrWFM