Date 15 Nov 2022
Page

Coding & Testing

Book : Fundamentals of software engineering by Rajib Mall
(IIT Kharagpur prof.)

Coding

(A)

→ ① Standard style of coding — req. analysis

req. specific

→ ② Coding standard & Guidelines. — design.

Individual modules } → ~~integra~~ ~~interra~~ integrate it

Testing → should be done at each & every stage. (V-model).

✓ UNIT testing → only module

✓ Integration testing →

✓ System testing

Date
Page

100% for Exam
Imp Question

Verification	Validation

Verification
→ design document

→ are we developing right way ??

→ Process

Validation

→ are we developing the right product?

{ Testing = verification + Validation }

(1) Consistent coding / Uniform coding format

→ Name
→ What it does
→ params
→ How to use it?
→

→

→ Easy to Read
→ integration would be easy.
→ maintainability of Software

(2)

→ Extensability
→ Reusability
→

→ Avoid use of goto statement
→ Avoid use of global vars.

Code Review

2 Process

$$\begin{tikzcd}
 & \text{2 Process} \arrow[dl, dash] \arrow[dr, dash] & \\
\text{Code walkthrough} & & \text{Code inspection}
\end{tikzcd}$$

{Statically analyze the code}

Code walkthrough

→ Just looking at code
→ X not executing it
→

Code inspection

→ splint /

[Clean room testing]
↓
(semiconductor)

Code walkthrough
+ inspection
+ formal verification.

★ error in ppt

formal ~~spec~~ $\boxed{\text{verification}}$ +
incremental development +
structured programming +

— x — Coding Ends — x —

Date ________
Page ________

Testing

{Generate test cases}

Internal Documents

External Documents

- User Manual
- Design Document
- SRS

Input.

Triplet [I, S, O] → output [A triplet]

{test case}

State of system

{Test Suite = Collection of Test case}

(2) Diff betwn Verification & Validation

(3) Inputs : ??

- test suite design : → How to efficiently design it?
- Running test cases & check results ~~for~~ to detect ~~error~~. failures
- Locate errors / debugging
- Error Correction

Outputs : ??

Eg, if (x>y)
        max = x;
    else
        max = x;

1) { (x=3, y=2),
(x=2, y=3) }

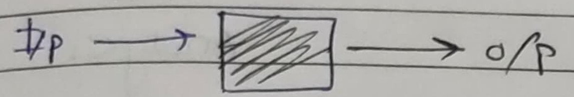
2) { (x=3, y=2),
(x=4, y=

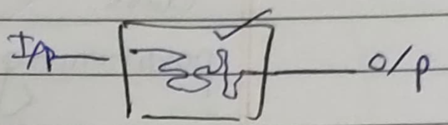
Not no-of TC → ⊗ !

way → ✓ (covering all possible cases)

testing

~~Fun~~ Black Box (functional) ⟹ White Box. (structural)





(BIG Picture thinking)

white (focus on internal details)

{make sure all the statement of code run}

What happens if module is missing?!

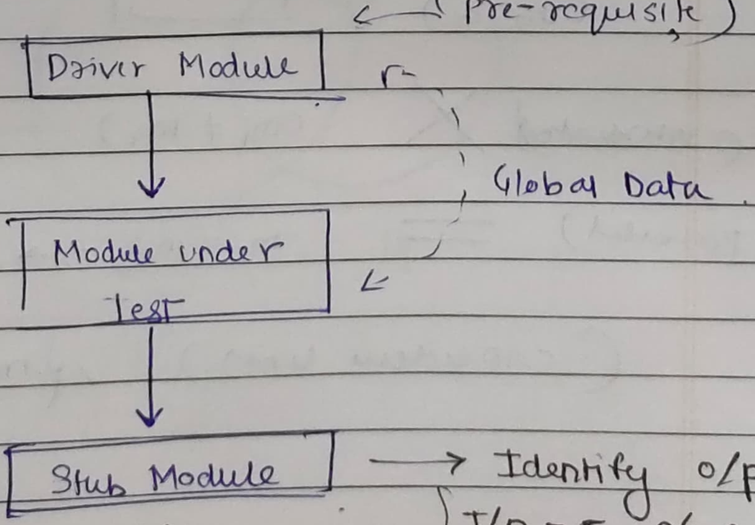
[combination]

---

testing in small ⟶ testing in large.

Unit testing — Integration & system testing

Unit Testing ⟵ (pre-requisite)



⟶ Identify O/P {I/P = 5 O/P = 120}

Highly simply way to map I/P → O/P.

Date
Page

Black Box Testing
two
approached

(*) tendency

Equivalent class partitioning	Boundary Value Analysis

Atleast 2 classes
① valid ② invalid

-1 0 5000 5001

{Integers only}
[0, 5000]

{1 element from each equivalent class}

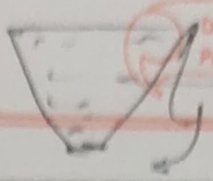
[-∞, 1] ✓ [5001, ∞) ✓

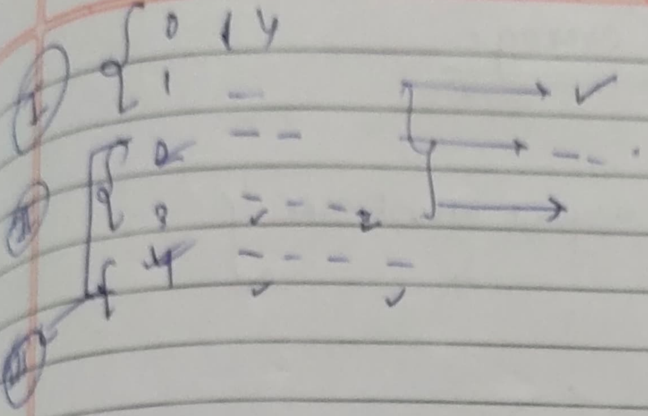
(Q2) $(m_1, c_1) \rightarrow$ BB $\rightarrow$ point of inter
$(m_2, c_2) \rightarrow$

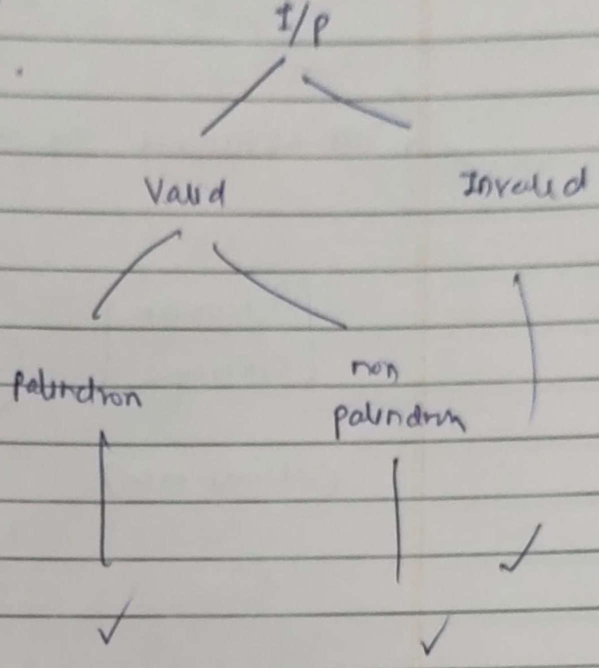
(intersected) X $(m_1 \neq m_2)$

(parallel) = $m_1 = m_2 \rightarrow$ No point of intersect

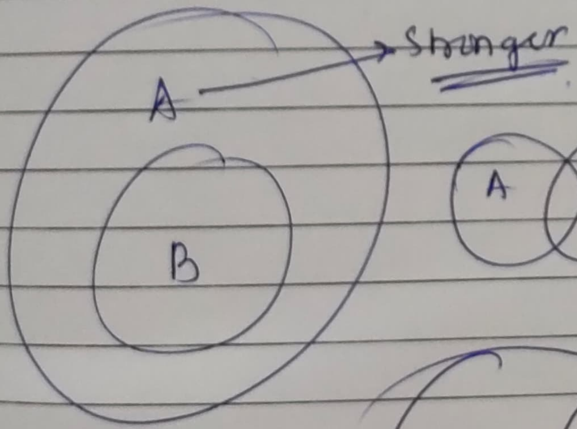
(coincident lines) $\{m_1 = m_2, c_1 = c_2\}$

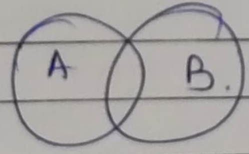






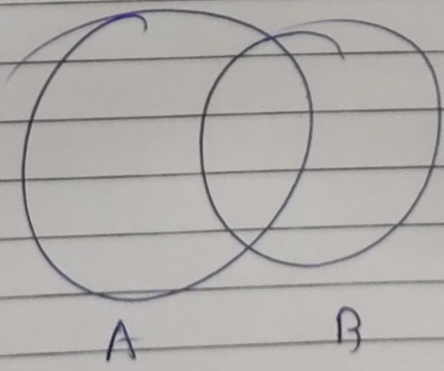
[White Box Testing]





Better criteria

✓ strategy to cover/ identify all errors



Coverage

✓ { all statements are executed atleast once }

[Statement Coverage]

(atleast once)

Branch/Decision Coverage

✓✓