Testing

Software Testing

Quality

Quality Assurance

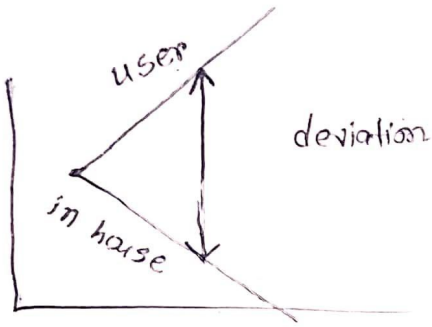
Quality Control

Software quality and its dimension

Software Verification and validation

Bug, error, failure

Debugging



deviation যত কমবে, quality তত বাড়বে

scalability

Correctness

trustme

---

process which is being followed is verified

product assess করে valid, invalid

Verified product

set of task যা দিয়ে process কে assess করা

Bug, error, failure, fault,

environment এ কোন problem হলে fault

debugging

কোন fail করার কারণ টা বের করা

Testing

ভুল বের করা intention

Software Testing Goals

Post implementation

- Reduce Maintenance Cost
- Improve Testing Process

Short term
- Bug Discovery
- Bug Prevention

long Term
- Reliability
- Quality
- User Satisfaction
- Risk Management

team আলোচনা না করলে planning আসা হয় না

Risk — time
feature যদি last time এ feature management এ ধরা পড়ে

Continuously test করতে হবে।

Post Implementation

Maintenance cost বেশি হয়

Short team ensure করতে
পারলে long term এর অনেক লাভ

Requirement Analysis
Design
Coding
testing
Mai...

testing যে একটা phase, এটা myth.
এই শুধু শেষেই test করতে হবে, তা কিন্তু
আগে শুরু হবে, তা বোঝা

একটা software completely test করা possible না

psychology — ভুল ধরানো হবে
ভুল ধরানো হবে
tester কে কাজে না লাগানো

tester হতে হলে অনেক coding knowledge থাকতে হবে - myth
domain knowledge থাকতে হবে

Complete testing ——— অসম্ভব

- Infinite Input Domain
  - valid Input
  - Invalid Input
  - edited Input
- Resource Constraints
  - Hardware
  - Human
  - Time
  - Cost

Software testing — principles & —

Myers

effective Testing

Infinite input set থেকে sample নিয়ে test

testing 2 প্রকার

White box testing — implementation, internal mechanism দেখে test করা

black box testing — ... না দেখে test করা

acceptance testing

performance

beta version

beta testing — subset of user এর কাছে publish release করে আগের থেকে feedback নেওয়া

alpha testing — company এর মধ্যে testing.

unit testing — একটা function/method normally একটা task perform করে একটা task/unit test করা

Regression testing — আগের individual সবগুলা একসাথে কাজ করছে এখনও ওইভাবে কাজ করছে কিনা

Integration testing

Scalability

Sanity testing

Scaling { Horizontal — আরও PC add করা
Vertical — একটা PC এর কার্যক্ষমতা resource বাড়ানো

backward compatibility

Wifi — wireless fidelity

Lifi — light এর speed এ

testing vs debugging

গাড়ি চলে না — তেল নাই

Effective testing vs Exhaustive testing

* Effectively test করবে, exhaustively না
* Sob phase এই testing চলে
* destructive ~~[illegible]~~ approach follow করে constructive মত করা

যেখানে ভুল হচ্ছে পাচ্ছি, সেখানে আরও ভুল পাওয়ার সম্ভাবনা কম
যেখানে ভুল বেশি পাচ্ছি, সেখানে আরও ভুল পাওয়ার সম্ভাবনা বেশি

testing — Expensive

dynamic testing 2 ধরণের type এর

White Box, Black Box

Black Box → implementation না জানলে

seperate team এর tester নিতে হবে

Documented রাখতে হবে

steps, input, output, time, platform etc সব লিখে রাখতে হবে

Unit testing

103.221.253.35/software

Unit মানে ~~একটা~~ কিছু, যেটা defined না

Project

1. test case + report
2. test smell

Verification → Quality assurance

validation → Quality control

কয়েক ধরনের test case,

but সব case কে cover করে

5 P 2

Robustness

B

A max+

A min−

5 C 2

$\frac{5 \times 4}{2}$

BVC

Boundary Value Checking

A min min+

max max−

normal

boundary value checking এ 4 টা state, (except normal) so test case $(4n+1)$

BVC

normal = 51

max = 100

min = 1

min^+ = 2

max^- = 99

Robust

max^+ = 101

min^- = 0

assignment

number of variables, range of each variables

Equivalent test case

A, B, C [1, 100]

Valid test, Invalid test

$I_1 = \{ \langle A, B, C \rangle : A > 100 \}$

$I_2 = \{ \langle A, B, C \rangle : A < 1 \}$

valid

$I_1 = \{ \langle dd, MM, yyyy \rangle : 1 \le dd \le 31 \ \& \ MM = \{1,3,5,7,8,10,12\} \ \& \ 1901 \le yyyy \le 2022 \}$

$I_2 = \{ \langle dd, MM, yy \rangle : 1 \le dd \le 30 \ \& \ MM = \{4,6,9,11\} \}$

$I_3 = \{ \langle dd, MM, yy \rangle : 1 \le dd \le 29 \ ; \ MM = 2 \ , \ yy \% 4 = 0 \}$

$I_4 = \{ \langle dd, MM, yy \rangle : 1 \le dd \le 28 \ , \ MM = 2 \quad yy \% 4 != 0 \}$

$I_5 = \{ \langle dd, MM, yy \rangle : yy < 1901 \}$

$I_6 = \{ \ldots \quad yy > 2022 \}$

$I_7 = \{ \ldots : mm < 1 \}$

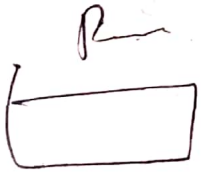
$I_8 = \{ \ldots : mm > 12 \}$

$I_9 = \{ \ldots : dd < 1 \}$

$I_{10} = \{ \ldots dd > 31 \}$

~~$I_{11} = \{ \ldots : MM = 2 \ \& \ dd >$~~

I_1 input ✓ exped ✓



output

Scanned by CamScanner

Testing

Test smell
↘
test code এর বোঝা smell

11 test smell.

resource optimism smell

testing ~~use~~ Boundary value checking

5.2 — test case optimization

Validation Activities

Unit

Integration

System

Testing tactics

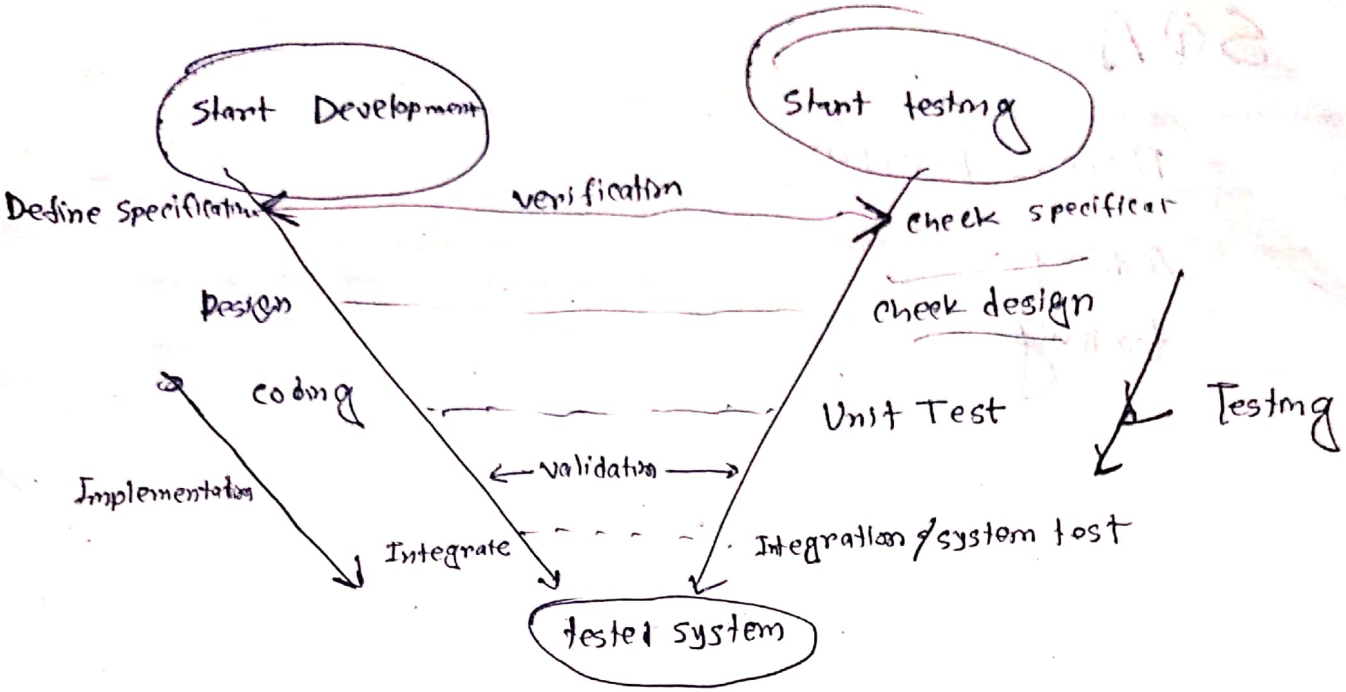
Static testing

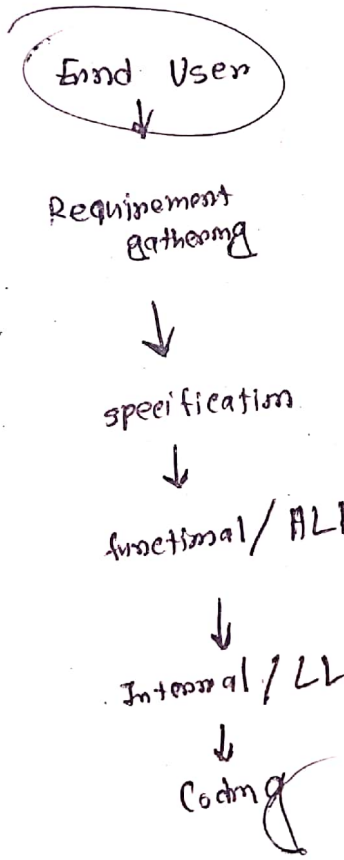
Dynamic testing

black-box / functional testing

white-box / structural ,,

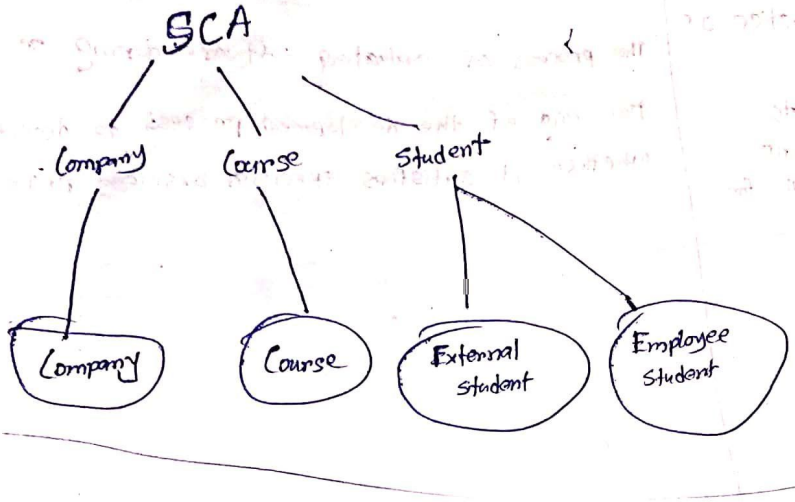
Chap 3

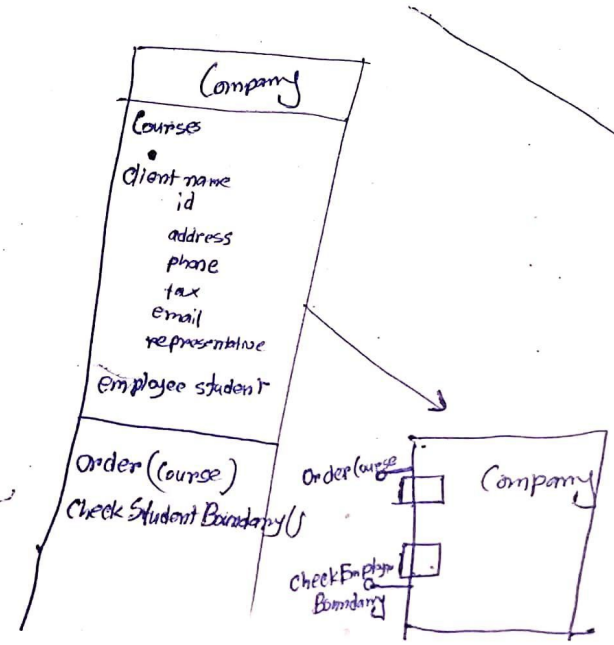


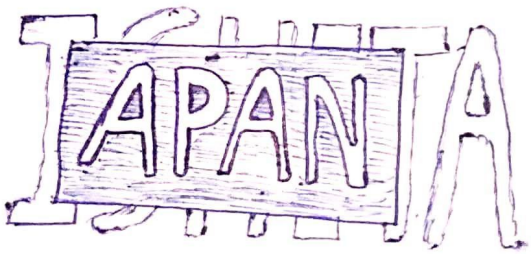


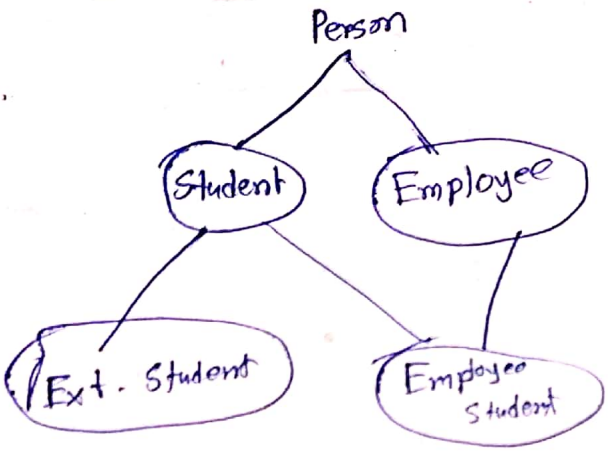
Verification	Validation
✾ Verification is a static practice of verifying The process of evaluating work-products of a development phase to determine whether they meet the specified requirements for that phase	The process of evaluating software during or at the end of the development process to determine whether it satisfies specified busmess requirem

not involved execut



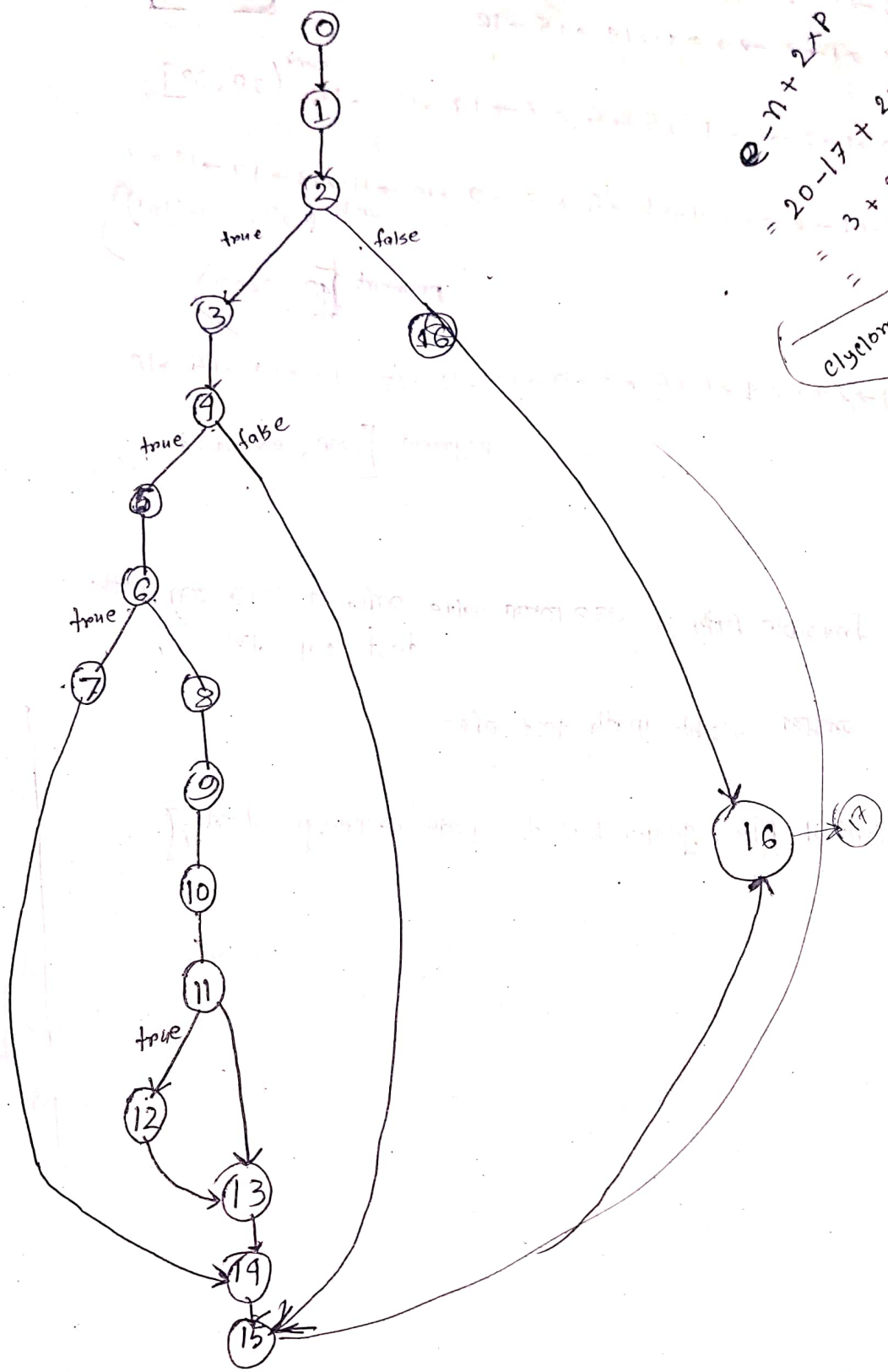






```
main() {
0    int work;
     double Payment=0;
1    scanf("%d", &work);
2    if (work>0) {
3        payment=40;
         if(work>20) {
4            if(work<=30)
5
6                payment = payment + (work-25) * 0.5;
7
8            else {
9                payment = payment + 50 + (work-30)*0.1;
10
11           if(payment >=3000)
12               payment = payment * 0.9;
13       }
14   }
15
16   printf("Find Payment %d", payment);
17 }
```

Control Flow Graph



e - n + 2*p
= 20 - 17 + 2*1
= 3 + 2
= 5

Cyclometric Complexity

work $(-max(int), 0]$

$0 \to 1 \to 2 \to 15 \to 16$

work $[1, 20]$

$0 \to 1 \to 2 \to 3 \to 4 \to 19 \to 15 \to 16$

work $(20, 30]$

$0 \to 1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 19 \to 15 \to 16$

work $(30, max(int))$

$0 \to 1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 8 \to 9 \to 10 \to 11 \to 13 \to 14 \to 16 \to 16$

payment $[40, 3000)$

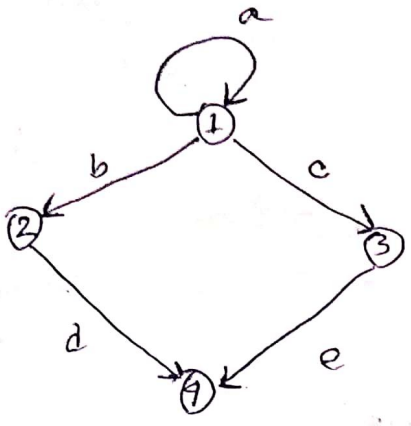
$0 \to 1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 8 \to 9 \to 10 \to 11 \to 12 \to 13 \to 14 \to 15 \to 16$

payment $[3000, max(double)]$

Invisible Path – এমন কোনো value থাকছে না, তাই তখন path test করা যায়

তাহলে visible path test করি –

Control Flow graph based path coverage testing.



	1	2	3	4
1	a	b	c	0
2	0	0	0	d
3	0	0	0	e
4	0	0	0	0

$$\begin{bmatrix} a & b & c & 0 \\ 0 & 0 & 0 & d \\ 0 & 0 & 0 & e \\ 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} a & b & c & 0 \\ 0 & 0 & 0 & d \\ 0 & 0 & 0 & e \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} d^2 & ab & ac & (bd+ce) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

1. Simple loop
2. nested loop

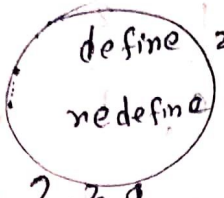
loop testing criteria

loop address করা
negative iteration check করতে, loop variable negative value দিয়ে check করা
loop এর সব statement check করতে
number of iteration fix রেখে নির্দিষ্ট সংখ্যক বার ঘুরানো
boundary value মূল্য check করা (loop parameter এর)

define

usage

kill




```
1  int a, b;
2  a = 10;
3  b = 20;
4  a = b+a;
5  if (a>10) {
       b++;
   }
```

2, 3, 4

usage of a : 4

predicate:

statement যেটা true/false boolean return করে।

kill: কোনো var এর scope কোথায় শেষ

computational ⟵ compute করা

mkdir kafka

cd kafka

sbt

define usage kill

dUK

dd → potential bugs

du → normal

dK → X

ud → ✓

uu → ✓

uk → ✓

kd → ✓

kU → bug (critical)

kK → X

Software Metrics

Understanding
Control
Improvement

Classification →
i) Product vs Process
ii) Objective vs Subjective
iii) Primitive vs Computed
iv) Private vs public

Entities
Process, Product, Resource

Attributes
internal, external
internal → e.g. lines of code
external → security

→ Size Metrics
line of Code (LoC) : KLoC, MLoC etc
Token Count (Halstead Product Metric)

Program Vocabulary: $n = n_1 + n_2$
(n_1 = # unique operators, n_2 = # unique operands)
all operators appearing in the implementation

Program length: $N = N_1 + N_2$
all operands appearing in the implementation

Program Volume
$V = N \log_2 n$

Function Point Analysis

Testing Object Oriented Software

OOT - Object, Class, encapsulation, Abstraction, inheritence, Polymorphism

Object Oriented Modeling

→ User view (use case)
→ Structural View (class diagram)
→ Behavioral " (CRC activity)
→ Environmental - - (Deployment)

Object Oriented Testing (OOT)

→ Understanding Problem
→ Dependency Problem

Strategy

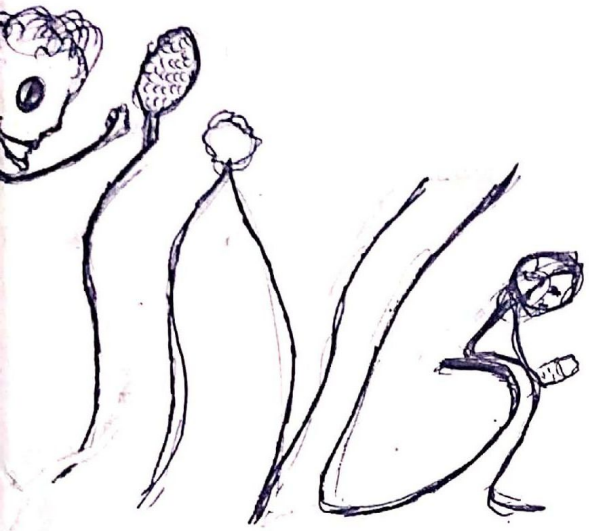
→ Method level
→ Class level
→ Cluster level
→ System level

binary
↓
assembly
↓
Structured
↓
Object oriented
↓
aspect oriented

Class এর method গুলো test → class level

Class এর যে জিনিসগুলো সরাসরি চোখে দেখা যায় না, অন্য class থেকে inherit হয়ে আসে, → Cluster level

A → B → C



Testing

Testing Web-based Systems

Challenges
- → Diversity & Complexity (language, tool)
- → Dynamic Environment (linux, windows)
- → Very Short Development time
- → Continuous Evolution (requirement change)
- → Compatibility and interoperability
  - chrome এ চলে, firefox এ চলে না

Quality aspects

- → Reliability
- → Performance
- → Security
- → Usability
- → Scalability
- → Availability
- → Maintainability

Testing

- → Interface
- → Usability
- → Content
- → Navigation
- → Configuration/ compatibility
- → Security

Testing

* Test case
  - Test ID
  - Purpose
  - Preconditions
  - Input
  - Expected Output

* Testware — Documentation
* Test Oracle — software metrics
* Incidents → এগুলো fail করে না, কিন্তু in the long run fail করতে পারে

Bug — life cycle in SDLC

⇒ Bugs-in — যেসব phase Bug ঢুকতে পারে, Bugging phase

⇒ Bugs-out —
- ⇒ classification — আগে Bug শনাক্ত out করবো,
- ⇒ Isolation — then locate করা
- ⇒ Resolution

Bug যে কোনো phase এ occur করতে পারে
- requirement এ
- design এ
- development এ

এক phase থেকে অন্য phase এ Bug propagate করতে পারে

Bug Occurance — why?

Environment
psychology

একটা phase এ error, অন্য phase এ Bug হতে পারে.

Bug propagate করতে পারে এক phase থেকে অন্য phase এ

ambiguous requirement } এ clear করে নিতে হবে
user friendly
secure } আগেভাগে define করা লাগবে

Impact of Bug — Economically

প্রতি phase এ Bug test করতে হবে

last এ গিয়ে taste করলে cost অনেক বেশি পড়বে

States of a Bug

1. New
2. Open
3. Assign
4. Test
5. Deferred
6. Reject
7. Validated / Done / solved / closed
8. Re-open

Bug Classification

A. Criticality

- severe / Critical — চলতেছেই না, software টি চলছে না
- Major — output টা ঠিকভাবে আসতেছে না / Incorrect output
- Medium — output ঠিক ই দিবে, কিন্তু কোথাও কোথাও সমস্যা আছে দিবেনা, convention follow করছে
- Minor — ~~font problem~~ font এ problem

deprecated হলে medium ও হতে পারে minor ও হতে পারে
কিন্তু Major বা severe হওয়ার chance খুব কম

Bug Classification — SDLC → Software Development Life Cycle

i) Requirement Phase — Bug — ambiguous requirement যদি বুঝতে ভুল হয়

ii) Design phase — Bug
- ⇒ Control flow Bug
- ⇒ Data flow Bug
- ⇒ Logic
- ⇒ Boundary Value
- ⇒ Race Condition
- ⇒ User interface
- ⇒ User expectation

Control Flow, logic, Processing, Data flow, Error handling, Race Cond., Boundary, UI

iii) Coding Phase - Bug
iv) Testing Phase - Bug
v) Post Release Phase - Bug

Control Flow : eg. if else দিয়ে control করা হচ্ছে
program থেকে control flow generate করা হলে
- missing path
- path এ infinite loop এ পড়ে যায়
- ওই path এ যায় ই না, unused/unnecessary

Data flow - এক statement থেকে অন্যটিতে যাওয়ার সময় Data flow ঠিক না থাকলে
Data related Bug
conversion, Double এর বদলে integer

logic

Boundary Value
- extreme case miss করে ফেললে

Race Condition
resource limited হলে Deadlock handle করা না হলে
severe bug হতে পারে

User interface
sign up এর বদলে sign in এ click হচ্ছে

User Expectation

Code smell - error / bug না, but future এ Bug induce হওয়ার chance বাড়িয়ে দেয়

Coding phase এ Bug

Testing phase
test method এ problem

Post Release phase

~~Bug release pb~~

Bugzilla

STLC