Day

5) Efficiency:
↳ Design of the app must work on user's efficiency.

"WEEK # 11 & 13"

— Software Testing —

⇒ It is the process of evaluating a system or its component with the intent to find whether it satisfies the specified req or not

⇔ Software testing is done to
↳ Detect Defect
↳ Reliability estimation

→ Process of executing a program under +ve/-ve conditions by manual and automated means

→ Testing should be repeatable

- Who Should Test System?

↳ Developer
- Understand the system
- But will test gently
- Driven by deadlines

↳ Independent Testers
- Must learn sys
- But, will attempt to break it
- Driven by quality

Date _____ Day _____

→ Testing is intended to show that a program does what it is intended to do & to discover defects.

- Testing Goals
  1. To demonstrate to the developer and the customer that the software meets its requirements
  2. To discover situations in which the behaviour of the software is incorrect or doesnot conform to its specification.

- VALIDATION TESTING
  - ↳ To demonstrate the developer and customer that the software meets its reqs.
  - ↳ Sucessfull test shows that system operates as intended

- DEFECT TESTING
  - ↳ To discover faults or defects in the s/w
  - ↳ A succesful test is a test that makes system perform incorrectly and so exposes a defect in the system.

Date _____ Day _____

Verification & Validation

→ **Verification**
↳ Are we building the product right?

→ **Validation**
↳ Are we building the right product?

→ Aim of V&V → "fit for purpose"

Inspection & Testing

↳ Inspection ——→ cannot check Non functional Req

- A static analysis technique to review code, design or documents (SRS, ERD, SDS etc) without executing actual app.
- To find defects early in documents or code

↳ Testing

- A dynamic analysis technique that involves executing the software
- To find bugs or failures by running the system

Stages Of Testing

1. Development Testing
2. Release Testing
3. User Testing

Date _____ Day _____

DEVELOPMENT TESTING

↳ Testing activities done by development team.

1. Unit Testing
2. Component Testing
3. System Testing

(1) Unit Testing

↳ Process of testing individual units or functions
↳ Object classes with several attributes & methods

(1.1) Object Class Testing

Complete test coverage of a class involves
↳ Testing all operations associated with an object
↳ Exercising object in all possible states

(1.2) Automated Test Components

i- Setup Part : Setup test data or mock objects
ii- Call : Invoke method/func to be tested
iii- Assertion : Compare actual vs expected output

(1.3) Partition Testing

↳ Divides input data into equivalance partitions
↳ A few test cases are choosen from each partition

Ex:

For age input:

Date _____ Day _____

- Position 1 : age < 18 → invalid
- Position 2 : age ≥ 18 & age ≤ 50 → valid
- Position 3 : age > 50 → invalid

↳ Take one value from each 14, 28, 54

(1.4) Guideline Base Testing

↳ Uses practical experience and common coding mistakes to guide test case selection

(a) Sequence based guidelines

Test with:

- A single value (eg : [5])
- Different sizes (eg: [1,2,4], [1,2,3,8,9] -)
- First, middle & last element
- Empty sequences

(b) General Guidelines

- Trigger all possible error msgs
- Cause buffer overflows
- Repeat input to check memory leaks
- Try to generate invalid outputs
- Force extreme computations

(2) Component Testing

↳ Testing individual components as a whole

↳ Testing a complete Login Module (UI + DB check)

↳ Integrating multiple units to form component.

Date _____ Day _____

(2.1) Interface Testing

- Process of verifying that diff modules or system interact correctly thru their interfaces
- Purpose:
  - ↳ Ensure data is passed correctly b/w components
  - ↳ Check for mismatch in datatypes / formats
  - ↳ Detect communication errors, timeouts or invalid responses
- Interface misuse
- Interface misunderstanding
- Timming errors
- Use stress testing in message passing system
- In shared memory system vary the order in which components are activated.

(3) System Testing

- The process of testing the interaction b/w components.
- It is a collective process.
- Performed after integration testing

- Use case Testing

→ Each usecase usually involves several system components so testing the use case force them to interact.

Date _____ Day _____

Example #03

LOC = 50,000

Developer = 5

Duration = 12 months

1- Total Effort req = Total LOC / LOC per pm

= 50000 / 600 × 5

= 16.67 months

Effort in pm = 5 dev × 16.67 months

= 83.35 person months

2- Total cost / developer per month = 160 × 50

= 8000 $

3- Total cost of project = $50000 \times \frac{8000}{600}$

= 666,666.67 $

However project is expected to complete in 12 months,

So total cost of project = 8000 × 5 × 12

= 480,000 $

Date _____ Day _____

- Exhaustive System Testing
  - ↳ The process of testing all possible combinations of input, paths and scenarios
  - ↳ Impractical & impossible

- Regression Testing
  - ↳ All tests are re-run whenever a change is made
  - ↳ It can be done thru automated testing

- Path Testing
  - ↳ To test each path of a program at least once. (See slide pg # 68)
  - ↳ A control graph is used for this purpose

- Release Testing
  - ↳ To test a particular release of sys intended to use outside dev team.
  - ↳ The goal is to convice ~~customer~~ supplier of system that product is good enough for users
  - ↳ black box testing [tester don't know internal code or structure]
  - ↳ Form of system testing
  - ↳ System ~~testing~~ done by dev team for ~~checking~~ discovering bugs whereas release testing is done to check whether sys

Date _____ Day _____

based

- Requirement Testing
  - ↳ Involves examining each req & developing a test or tests for it. (See pg #76)

- Performance Testing
  - ↳ Checking performance and reliability of a system
  - Load Testing → load level upto expected user load
    - ↳ Testing done by applying maximum load in terms of software accessing and manipulate large input data.
    - ↳ Identifies maximum capacity of a software & it's behaviour at peak time
  - Stress Testing
    - ↳ Testing the software under abnormal conditions
      - ↳ Turning database on/off
      - ↳ Running diff process that consume resources
    - ↳ Sees how system fails & recover
    - ↳ Load level higher than sys's limit
    - ↳ Focus stability & err handling under extrem conditions.

Date _____ Day _____

- User Testing
  ↳ Users / customers provide input & advice on system testing

① Alpha Testing
- Conducted at the developer's site
- Real users work closely with the development team
- Helps catch easy bugs

② Beta Testing
- Software is released to a limited group of external users
- Users try it in real-world conditions

③ Acceptance Testing
- Conducted by customer to decide if the S/W req are meet
- Happens before final deployment

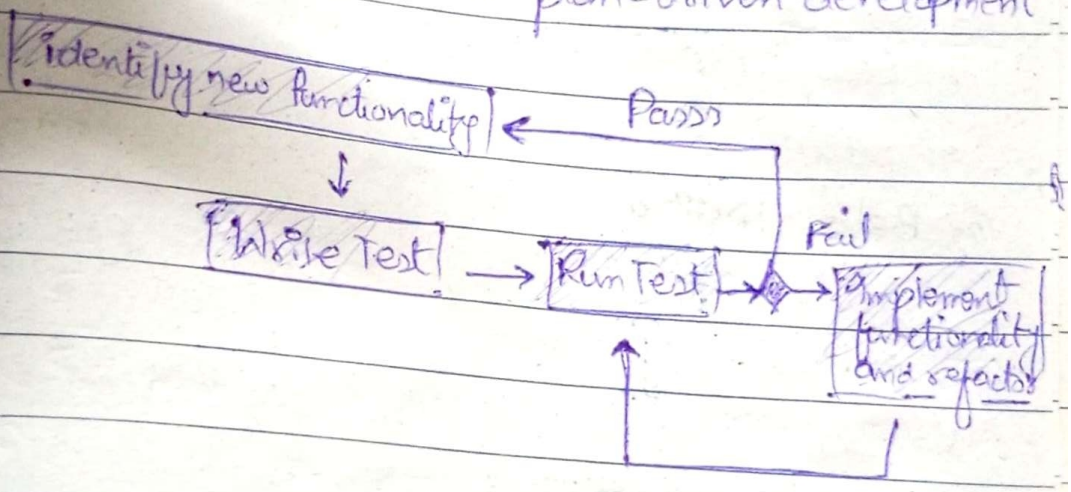
Stages:
i- Define acceptance criteria
ii- Plan acceptance testing
iii- Derive " test
iv- Run " tests
v- Negotiate test results
vi- Reject / Accept system

→ In Agile user / customer is part of dev team & responsible for making decision on acceptibility of system.

Date

Day

- Test Driven Development
  - ↳ Test are written before code
  - ↳ You don't move onto next increment until the perivous code has been tested.
  - ↳ It is part of XP.
  - ↳ Can also be used in plan-driven development



Date _____ Day _____

WEEK # 14

Software Project Management

→ Model the project activities and their relationships as a network.
→ In network time flows from left → right
→ CPM (Critical Path Method) or CPA (... Analysis)
→ PERT (Programme Evaluation Review Technique)

→ Some definitions

- Early Start represents the earliest date an activity can possibly begin, based on all its predecessors and successors

- Early Finish represents the earliest date an activity can possibly finish
- Late Start, the latest an activity can start without affecting the planned project finish date
- Late Finish, the latest an activity can finish, without delaying the finish of project
- Total Float / Float / Slack, an amount of time activity can be delayed without delaying the overall project duration
- Free Float, amount of time an activity can be delayed without delaying early start of an immediate successor activity.

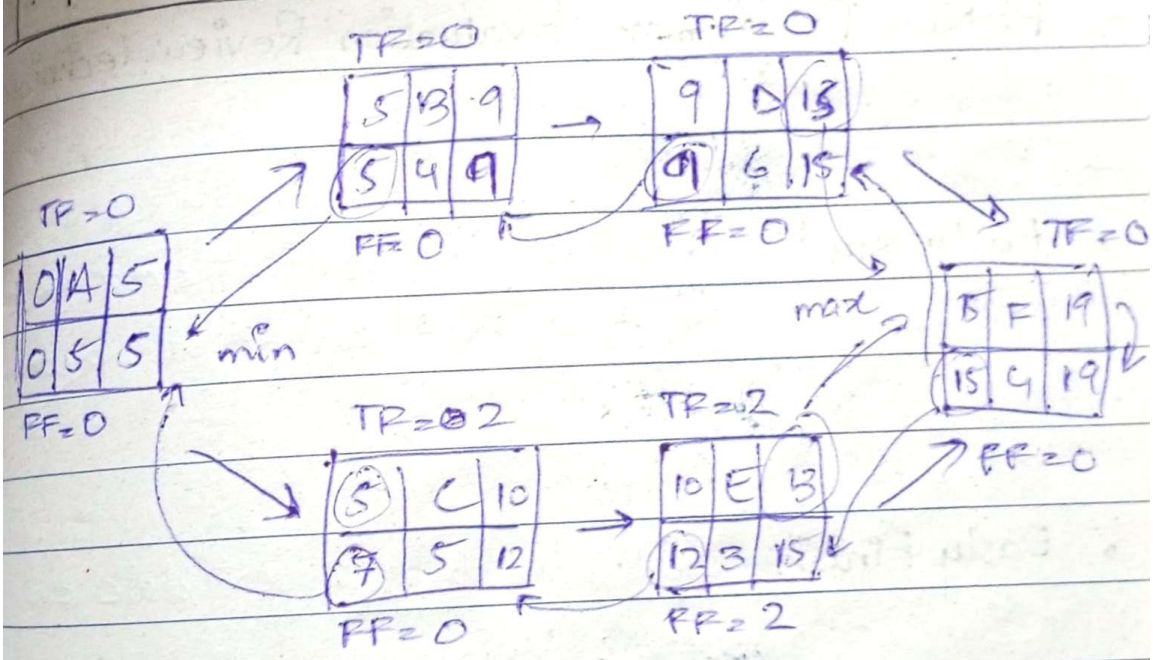
Free Float matlab k ek activity ko kitna delay kr sakte hai k uska successor late na ho. early start change na ho.

Day

Activity	Predecessor	Duration
A	–	5
B	A	4
C	A	5
D	B	6
E	C	3
F	D, E	4

ES	Node	EF
LS	D	LF

LS = LF - D
TF = LF - EF



→ The nodes where TF=0 is critical paths.

A → B → D → F. (critical pathvaali kisi bhi activity ko agr ek din bhi aage badhaya to poora project ek din aage hoga)

Free Float = Minimum(ES successor) - ESnode - Duration

FF(A) = Min(5,5) - 0 - 5 = 0

→ Free Float matlab hum us activity ko utna din delay kr sakte hai without affecting whole project. Like "E" ko 2 din tak delay kiya jaa sakta hai.

Date _____ Day _____

- Gantt Chart (Pg # 25)

"WEEK # 15"

Risk Management

- Risk is an uncertainity
- It might happen and it might not
- Risk exposure = Size (loss) * probability (loss)
- Problem vs Risk
  - ↳ Problem is some event which has already occured
  - ↳ Risk is something that is unpredectable

- There are no 100% risk - instead they are called constraints

▷ Risk Categorization

- Project Risk
  - ↳ They threaten the project plan
  - ↳ If they become real, it is likely that the project schedul will slip & cost increase

Day

- Technical Risk
  - ↳ They threaten the [quality] & [timeliness]
  - ↳ If become real, [implementation] become difficult or impossible

- Business Risks
  - ↳ They threat the success of project
  - ↳ They put the project in danger.
  - ↳ Sub categories
    - Market Risks - building an excellent product that no one really wants
    - Strategic Risks - building a product that no longer fits into the business strategy
    - Sales Risks - building a product that sales team don't know how to sell
    - Management Risks - loosing support of senior management
    - Budget Risks - loosing budget or personnel commitment.

- Sub categories of all types of risks
  - (1) Known Risks
    - ↳ Risks that can be identified in advance through careful analysis

Date _____ Day _____

↳ Often found by reviewing the project plan, reqs, timeline or env.

↳ Ex: Unrealistic delivery time, lack of required tools or technology, incomplete reqs, Tight budget

(2) Predictable Risks

↳ Risks that are expected based on past experience

↳ Can be forecasted using data from previous similar projects

↳ Ex: High developer turnover, inaccurate effort estimation, ~~Tight budget~~

(3) Unpredictable Risks

↳ Unexpected event

↳ Can't be identified during planning

↳ Ex: Natural disaster, Sudden Legal / political change, Critical staff illness / emergency, Cyber Attack

Day

7 Principles of Risk Management

1. Maintain a global perspective
   ↳ Understand how software risks affect entire solution

2. Take a forward-looking view
   ↳ Focus on future risks not just current ones and create contingency plan

3. Encourage Open communication
   ↳ Allow all stake holders to freely raise risk concerns

4. Integerate Risk Management
   ↳ Make risk manag. built in part of ~~SDP~~ Softw- dev process

5. Emphasize continuous risk management
   ↳ Update & reasses risk as the project evolves

6. All stake holder should have common understanding of the product

7. Encourage team work

Date _____ Day _____

▷ Reactive Vs Proactive Risk Manag. Strategies

- **Proactive :**
  - ↳ Prevention focused : Anticipate risks before they occur
  - ↳ Minimize risk impact or prevent it completely
  - ↳ Applied early in project life cycle
  - ↳ Ex : Planning for staff turnover, budgeting buffer time

- **Reactive :**
  - ↳ Response Focused : Deal with risks after they occur
  - ↳ Triggered after a problem arises
  - ↳ Can be costly due to emergency actions and delays
  - ↳ Ex : Fixing bugs found late, replacing a team member suddenly

▷ RMMM Plan

① Risk ~~Migit~~ Mitigation → problem avoidance activity
  - ↳ Reduce the prob of impact of risk before it occur
  - ↳ Actions :
    - ↳ Develop contingency plan

Date

Day

↳ Allocate buffers (cost + time)
↳ Assign experience staff

(2) Risk Monitoring
↳ Track identified risks continuously throughout the project
↳ Ensure that risk aversion steps are being properly applied
↳ To collect info that can be used for future risk analysis

→ If Risk Exposure (RE = Probability × Cost) is less than the mitigation cost, don't try to mitigate (avoid) the risk but continue to monitor it

see slide #30

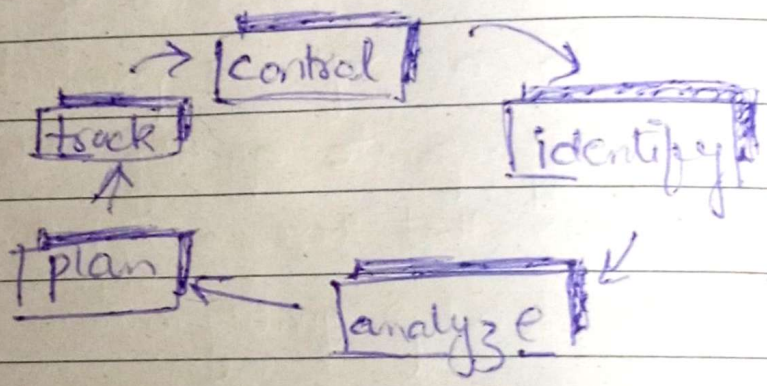
See slide #27

- Known & Predictable Risk Categories
- Product Size
- Business Impact - constraints imposed by management or market
- Customer Characteristics - customer communication & Knowledge of technicality
- Process Definition - Software process kitna well define hai
- Development Env - availability & quality of tools

Date _____ Day _____

- Technology to be built - complexity of system to be built
- Staff size & experience -

▷ Risk Management Paradigm



Date

Day

"QUALITY MANAGEMENT"

→ Quality means that a product should meet it's specifications

- Software Quality Management
  - → Ensures that required level of quality is achieved in a software product
  - → Defining appropriate quality std. and procedures and ensures that those are followed
  - → Aims to develop quality culture

- Two principal concerns
  1. Organizational Level
     - ↳ Focuses on creating a framework of processes and ~~std~~ stds accross the org.
     - ↳ Ensure consistent, high quality software accross all projects.
  2. Project Level
     - ↳ Applies the org.'s quality process within a specific project
     - ↳ Involves creating and following a quality plan which defines
       - ↳ Quality Goal
       - ↳ Std & process
       - ↳ Monitoring & Verificati activities

Date _____ Day

- Quality Management Activities

(1) Quality management monitors the sq software development process independently to ensure it meets standards.

(2) Verification of ~~Deliverables~~ Check that all project outputs (docs, codes, etc) align with org. std and goals

(3) The QA team must be seperate from dev team to maintain unbiasness.

Quality Assurance → Establish org. al procedure and std for quality

Quality Planning → Select applicable procedures and std for a particular project and modify these as req.

Quality Control → Ensures that procedure and std are followed by dev team.

Quality Management should be seperated for project management to ensure independence.

Day

Quality Planning

→ The quality plan should define the quality assessment process.

→ It should set out which organisational stds should be applied and define new if necessary

Scope Of Quality Management

→ Important for large, complex systems.

→ For smaller systems, quality management needs less documentation.

Quality Conflicts

→ It is not possible for any system to be optimized for all of these attributes

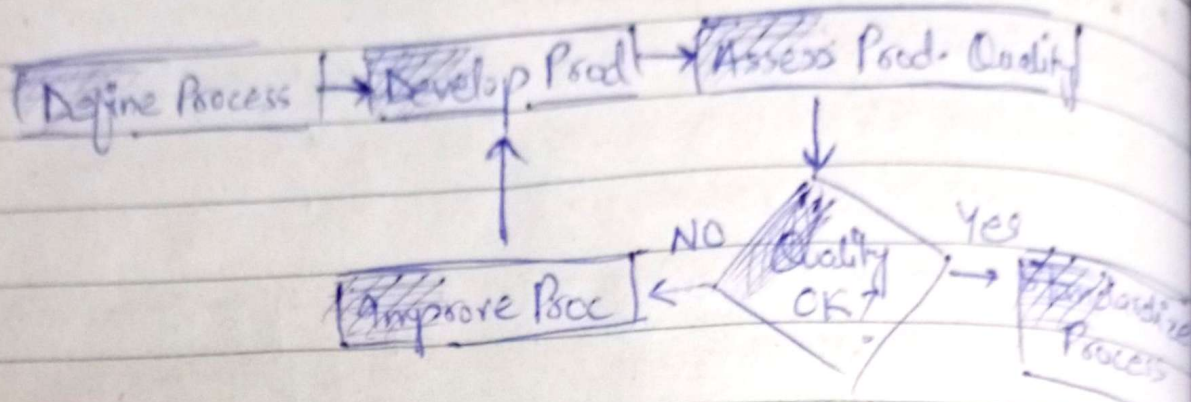
↳ Like improving security can lead to loss of performance

→ Therefore it is important to consider most imp quality attributes.

Process & Product Quality

→ The quality of a developed product is influenced by the quality of the production process.

Date _____ Day _____



- Product & Process Standards

↳ Product Standards
  ↳ Apply to final software product
  ↳ Ensures consistency and quality in software deliverables
  ↳ Examples:
    ↳ Document Std: Structure for req or design document
    ↳ Coding Std: Naming conventions, indentation
    ↳

↳ Process Standards
  ↳ Apply to the software dev process itself
  ↳ Ensures best practices are followed
  ↳ Examples:
    ↳ use of version control
    ↳ " " modeling tools
    ↳ define steps of req gathering, design & testing

Reviews & Inspections

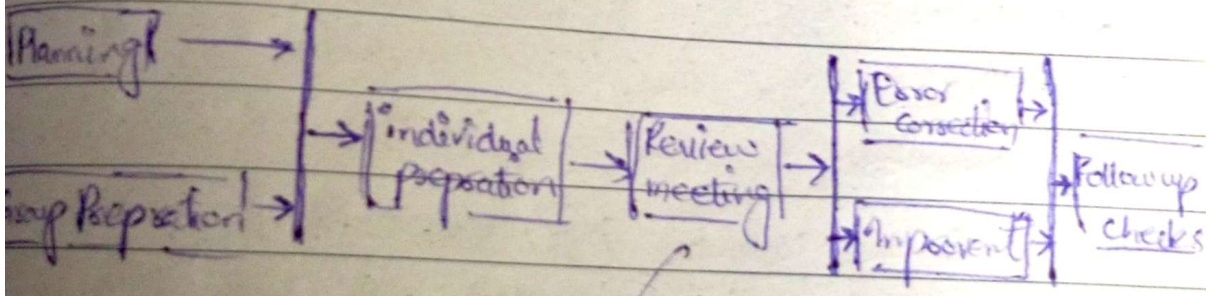
Day _____

→ ~~Reviews~~ involves checking the software, its documentation, and records of process to discover errors

→ ~~They~~ There are different types of reviews with different objectives

- ↳ Inspections for defect removal (product)
- ↳ Reviews for progress assessment
- ↳ Quality reviews.

• Software Review Process



← Pre-review activities →

← Post-review activities →

an author of the doc / program being reviewed should 'walk through' the doc with review team.

Date _____ Day _____

- Distributed Reviews
  - → These are software / document reviews conducted by teams that are geographically seperated and cannot meet face-to-face.
    - ↳ Face-to-face review are impractical and too costly.
  - → Use shared documents or collabrative tools like Google docs, Github etc.

- Program Inspections
  - → These are peer reviews where engineers manually examine system artifacts (like src code, or design doc) to find defects & anomalies
  - → No execution of system required.
  - → Conducted by a group of peers to catch errors easly.

- Inspection Checklist
  - → Predefined list of common errors use to guide software reviews and ensures that frequent mistakes are not overlooked.
  - → Ex: Variable initialization, Constant naming, loop termination, Null checks etc.
  - → See fault class Pg # 34 of SPDE

Software Measurement

Day

Software Metric

→ Any type of measurement that relates to software system, process or related doc.
  ↳ Like Lines of code, no. of person-days to develop a component

Types of Process Metrics

① Time-based Metric

↳ Measures the duration of a process / activity
↳ Includes:
  ↳ Total time to complete the process
  ↳ Calendar time
  ↳ Time spent by individual Engineer
↳ Ex: Time taken to complete unit testing

② Resource-based Metric

↳ Track resources consumed by a process
↳ Includes:
  ↳ Person-days or person-hours
  ↳ Cost of travel / tools
  ↳ Computer / technical resources used
↳ Ex: 30 person days spent on system integration

10 ph → ... or ↔ 5000

Day

Date

③ Event-based Metrics

↳ Count how often a specific event occurs during the process

↳ Includes:

↳ No. of defects found in code reviews

↳ No. of requirements changes

↳ Bug reports filed

↳ Lines of code modified

↳ Ex: 15 defects found during code inspection

• Product Metrics

→ A quality metric should be a predictor of product quality.

① Dynamic Metric

• Collected during program execution

• Used to assess:

• Efficiency (eg: response time, memory usage)

• Reliability (eg: no. of failures, uptime)

• Ex: CPU usage during stress testing

② Static Metrics

• Collected by analysing software code or document without executing it.

• Measure internal structure & complexity.

• Used to assess:

• Complexity (eg: cyclomatic complexity)

• Understandability (eg: code readability)

• No. of func in a module

Page No.

Date _____ Day _____

- Static Software Product Metrics

(1) Fan-in / Fan-out

- Fan-In → No. of functions/methods that call a given function X.
- Fan-Out → No. of functions that are called by function X.

→ High Fan-In → X is tightly coupled; changes to X can affect many others

→ High Fan-Out → X is complex; it may involve complicated control logic.

(2) Length of Code (LOC - lines)

- Measure the size of a program or component
- Larger code size is usually more complex and error prone
- LOC is strong predictor of potential defects and maintainence difficulty.

(3) Cyclomatic Complexity

- Measures the control flow complexity of a program (based on decesion point like if, while, for etc)
- Higher value indicate the code is harder to understand, test and maintain

Day

Date

④ Length of Identifiers
- Calculates the avg length of variable, class and method names
- Longer descriptive names improve code readability

⑤ Depth of Conditional Nesting
- Measures how deeply "if" or conditional blocks are nested
- Deep nesting makes code harder to follow and more error-prone.

⑥ Fog Index
- Assesses document readability by analyzing sentence and word lengths
- A higher fog index means text is more difficult to understand.

- Object Oriented Metrics

① Weighted Methods Per Class (WMC)
- Measures the total no. of methods in a class, each weighted by its complexity
- Higher WMC → More complex and harder-to-understand; less reusable.

Date Day

② Depth of Inheritance Tree (DIT)

- Measures the no. of inheritance levels from the class up to the root class
- Greater DIT → Increased design complexity, as more classes must be understood

③ Numbers of Children (NOC)

- Counts the no. of immediate subclasses derived from a class
- Higher NOC → Indicates more reuse but also more responsibility for ~~base~~ base class

Relationship Between Internal & External Software