Open-source software licence

- allows software to be freely used, modified and shared by anyone.

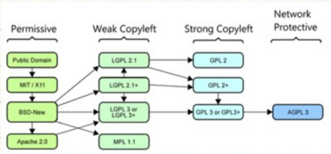
2 types 1. permissive → flexible, can use code, modify & keep your change private

2. Copyleft → must stay open-source if you use or modify it.

	Type	Comment
MIT License	permissive	allow copyright protection, commercial use, used in closed source
Apache	permissive	allow copyright protection, commercial use, patent license, used in closed source
GNU – GPL V2	copyleft	allow copyright protection, commercial use
GNU – GPL V3	copyleft	allow copyright protection, commercial use, patent licence

} must stay under GPL

Copyright → software source code   Patent → innovative methods used in code



Creative Common license

- used for license creative work (music, art, writing etc.)

BY → Attribution (credit)

NC → Non-Commercial   i.e CC BY-SA

ND → No-derivatives (No changes/remixes)   - can be used and modify but credit & same license

SA → Share Alike (Derivatives must use the same licence)

Software testing

1. verification → meet design & technical spec → code is correct & behave correctly (discover defect)
2. validation → meet user needs → useful and useable for customers

during verification, software inspection is used → no execution, just reviewing the code and design.

It is used in early development process. It is human-based review.

during verification, static analysis is also used → software tool for source text processing to discover erroneous conditions

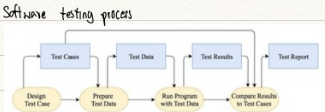
stages

1. Control flow - loops, exit point, unreachable code
2. Data usage - uninitialized variables, never-used/written twice variable
3. Interface - routine & procedure declarations
4. Information flow - dependencies of output variable
5. Path → identify paths through the program

Test Table: White box testing

Program Path/ Test Case	Test Data	Expected Result	Program Result	Note
1. When i==0	0	'yes'	'no'	Not pass
2. When i !=0	1	'no'	'no'	pass

Software testing process



Test Table: Blackbox Testing

	Test Case	Test Data	Expected Result	Program Result	Note
Valid	1.Between 4 and 10	4, 7, 10	T	T	pass
Valid	2. <4	3,0,-1	F	T	Not pass
Valid	3. >10	11,50,100	F	F	pass
Invalid	4. Characters and Symbols	A, %, +	Reject	Reject	pass
Invalid	5. null		Reject	Reject	pass

in verification, we also need to do <u>component testing</u>, then <u>system testing</u>
↳ component developer   ↳ testing team

in system testing,
- Top-down → testing from main flow and go down to components
- Bottom-up → testing each component and combine them

you can look at it like component, integration, system

You can also categorized them as how the test is conducted

1. Black box → test all possibilities
2. White Box → tester know the code & test the logic
3. Performance → speed/responsiveness (increases load steadily)
4. Stress → pushing the system's limit
5. Interface → How components talk (API/UI)

Software Integration

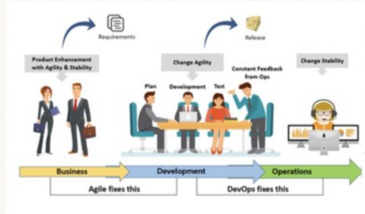
What is DevOps? practice that emphasizes collaboration between software dev and other IT professional

DevOps pipeline - practices that devs and ops team implement to build, test and deploy software faster.

Plan → code → build → test → release → deploy → operate → monitor. It's the goal
(devs: plan → test; ops: release → monitor)

Agile + DevOps (devs / ops)

tolerate changes in requirements & continuous deployment



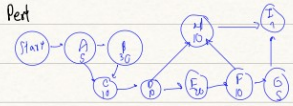
To follow the pipeline, you need tools to automate

1. Building software (Maven, Ant, Grunt) } CI → build + test
2. Testing (JUnit, JMeter, JBehave) } CI/CD → automation engine
3. Deployment (Jenkins, Spinnaker) } CD → deploy
4. Running it reliably (<u>containers</u> & <u>IaC</u>)
   ↳ packing app with its require libraries/frameworks in a package called container i.e Docker → run anywhere
   ↳ writing code to define & manage your infrastructure (servers, networks, databases) → automate infrastructure setup

Project management

	Task	Duration	Start	End	Pred
A	Identify Interviewee	5	3-Sep-2025	7-Sep-2025	
B	Conduct Interviews	30	10-Sep-2025	19-Oct-2025	A
C	Review Documentation	10	22-Oct-2025	02-Nov-2025	A,B
D	Analyze Data	10	5-Nov-2025	16-Nov-2025	C
E	Draft Preliminary Report	20	19-Nov-2025	14-Dec-2025	D
F	Obtain Feedback on Draft Report	10	17-Dec-2025	28-Dec-2025	E
G	Write Final Report	5	31-Dec-2025	4-Jan-2025	F
H	Prepare Presentation	10	31-Dec-2025	11-Jan-2025	D, F
I	Make Final Presentation	1	14-Jan-2025	14-Jan-2025	H, G

Pert



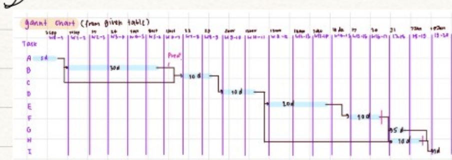
Path (5 30 10 10 20 10 10 1)	days	slack
A → B → C → D → E → F → H → I	96	
A → B → C → D → E → F → G → I	91	5
A → B → C → D → H → I	66	30
A → C → D → E → F → H → I	66	30
A → C → D → E → F → G → I	61	35
A → C → D → H → I	36	60

Software project management

goal: software is delivered on time & within budget.

Management Activities
- Proposal writing
- Project planning and scheduling < distribute tasks / finish on time & budget
  - → Bar chart (Gantt) → schedule against calendar time
  - → Activity network (Pert) → show task dependencies and critical path
- Project costing
- Project monitoring and reviews
- Personnel selection and evaluation
- Report writing and presentations – product owners

Gantt chart (from given table)



Software Cost estimation

Size measurement

1. LOC - total lines of code
2. Function points - measure functionality based on ↗
3. UFC (unadjusted function count) = sum of all (# of element × <u>weight</u>) ↑ based on
4. VAF - complexity multiplier

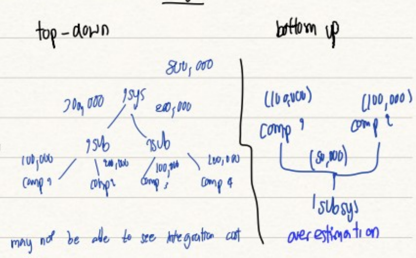
Based on a combination of program characteristics
1. External Inputs and Outputs (EI, EO)
2. User Interactions or External Queries (EQ)
3. External Interface files (EIF)
4. files used by the system or Internal Logical Files (ILF)

FP = UFC × VAF

estimated LOC = <u>AVC</u> × FP
- cost = LOC × \$/LOC
- effort = LOC / LOC/p-month

↳ average # line of code on languages

Estimation strategies
1. top-down
2. bottom up



COCOMO 2 Model for effort & cost estimation

Submodels in COCOMO 2 are:
1. Application composition model
   - Used when software is composed from existing parts
2. Early design model
   - Used when requirements are available but design has not yet started
3. Reuse model
   - Used to compute the effort of integrating reusable components
4. Post-architecture model
   - Used once the system architecture has been designed and more information about the system is available

- Formula is: PM = (NAP x (1 - %reuse/100)) / PROD
  - PM is the effort in person-months
  - NAP or NOP is the number of application points or object points
  - PROD is the productivity

PM = A × Size^B × M where
- M = PERS × RCPX × RUSE × PDIF × PREX × FCIL × SCED;
- A = 2.94 in initial calibration, constant → depends on organization
- Size in KLOC
- B varies from 1.1 to 1.24 exponent number
  - Depending on novelty of the project, development flexibility, risk management approaches and the process maturity

B = 1.01 + 0.01 x Σ SF (i) , i=1,..., 5
- SF = Scale Factor
- Each SF is rated on 6-point scale (ranging from 0 to 5) :
- very low (5), low (4), nominal (3), high (2), very high (1), extra high (0)

PM = (ASLOC * AT/100)/ATPROD   code generated
- ASLOC is the number of lines of generated code
- AT is the percentage of code automatically generated
- ATPROD is the productivity of engineers in integrating this code

ESLOC = ASLOC * (1-AT/100) * AAM   code integrated & understood
- ASLOC is the number of lines of generated code
- AT is the percentage of code automatically generated
- AAM is the adaptation adjustment multiplier computed from the costs of changing the reused code, the costs of understanding how to integrate the code and the costs of reuse decision making.
- AAM is the sum of three components (AAF+SU+AA)

- Calendar time can be estimated using a COCOMO 2 formula
  - TDEV = 3 × (PM)$^{(0.33+0.2*(B-1.01))}$ early design & post architecture model
  - PM is the effort computation and B is the exponent computed as discussed above (B is 1 for the early prototyping model)
  - This computation predicts the nominal schedule for the project

6.1 COCOMO II Effort Equation   A : 2.94

As an example, a project with all Nominal Cost Drivers and Scale Drivers would have an M of 1.00 and exponent is equal 1.0997. Assuming that the project is projected to consist of 8,000 source lines of code,

Find COCOMO II estimation? Early design model

$PM = A \times Size^B \times M = 2.94 \times \left(\frac{8000}{1000}\right)^{1.0997} \times 1 = 28.94$