Software is:
1) instructions (computer programs that when executed provide desired features, function & performance
2) data structures that enable the programs to adequately manipulate info
3) documentation that describes the operation & use of programs

Stakeholder
- Users, Developers, Customer

Software Engineering
Requirement analysis & Definition
System design
Coding
Testing (unit)
Integration testing
System testing
Maintenance → System delivery

Lehman Laws
- Continuing Change
- Increasing Complexity/Entropy
- Continuing Growth
- Declining Quality

Framework Activities
- Communication, Planning, Modelling, Construction, Deployment

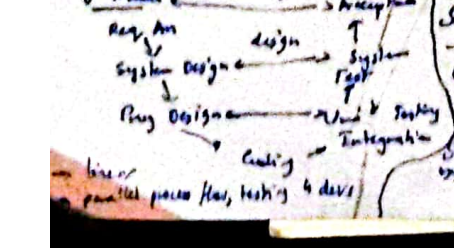
Process Flow - describes how the framework activities & the actions & tasks that occur within each framework activity are organized with sequence & time
→ Linear Process Flow
→ Iterative Process Flow : repeats
→ Evolutionary Process Flow : circular
→ Parallel Process Flow : different parts of system work in parallel

Communication Activity
- Inception, Elicitation, Elaboration, Negotiation, Specification, Validation

Waterfall Model : Classical
→ artefacts: deliverables of project
→ works well with no or minimal changes in requirements
→ can't go to back to previous area of development
→ no guidance in changes to products
→ lots of documentation
→ Long wait before product
→ suitable for large projects

Waterfall: Prototyping
→ partially developed product
→ helps devs assess alt design strats (design prototype)
→ users understand what system will be like (user interface prototype)
→ is linear, some iterative

Waterfall: V Model:
→ uses unit testing to verify procedural design
→ uses integration testing to verify overall system
→ uses acceptance testing to validate requirements



→ has parallel process flow, testing & devs

client to check if requirements of software are met

Phased Development
Cycle time is time b/w requirements doc was written to when system delivered (current)
Two systems : prod system & development system (next)
→ linear

Incremental dev: develop subsystem & add parts to it every release

Iterative dev: make full system & make changes to it every build

Prototyping
allows repeated investigation of requirements or design
→ reduces risk & uncertainty
→ used when vague ideas presented by client

Pros : (Prototyping)
→ reduced impact of requirement changes
→ customer is involved early & often
→ works well for small projects
→ reduced likelihood of product rejection

Cons : (Prots)
→ Customer involvement, delay
→ temptation to ship prots
→ Work lost in throwaway prots
→ hard to plan & manage

Spiral Model (Evolutionary Process)
→ Plans, determine goals, alt & constraints, evaluate alts & risks, develop & test
→ iterative prots + waterfall
→ another pair - combination of work products & conditions
→ Unified Process Model
→ iterative, use-case & modelled by UML

Lifecycle: Inception, Elaboration, Construction, Transition, Production
→ staggered on concurrency
beta testing: deploy a feature or build to a limited audience

RAD (Rapid Application Development)
→ linear, concise development cycle using element based construction approach

Phases of RAD : Business Modelling, Data Modelling, Process Modelling, Application Development, Testing & Turnover
→ 2-3 months, requirements are well-known & technical risk is limited

Agile : Adaptive Planning, Evolutionary Dev, Early Delivery, Continual Improvement, Responsiveness to change
→ invest time in producing software than in doc
→ Focus on customer collab instead of negotiation

XP
→ XP Planning, XP Design, XP Coding, XP Testing
Planning: user stories, prio to story, team assigns cost, Project velocity
→ KIS, CRC Cards, spike solutions, refactoring
→ unit tests, pair programming, continuous integration
→ acceptance tests

Scrum
Sprint backlog: items cannot be added to this
Product backlog: " can be " "
→ sprints, 2-4 weeks
Scrum team: Scrum Master, Product Owner, Dev Team
→ self organising & assigning
Code Increment: Sum of all product backlog items in prev sprints & backlog items to be completed in current sprints
Sprint velocity is an estimation of work a team is delivering in sprints, stories

Cards have prio, owner, type, due date
WIP (Work in Progress) Limit means max no. of cards on work column
→ pull-based system
Stages: To-do, Dev, Testing, Done
Lead Time: duration of card in ur workflow & final deploy from workflow
Cycle Time: duration btw card's arrival in working state & card is ready to release
Throughput: Performance = WIP / Cycle Time

Problem-based Estimation of size
$S = (S_{opt} + 4S_m + S_{pess}) / 6$

Effort: $\frac{LOC}{Avg\ Prod}$ = person-months
Cost per line code = $\frac{Labour\ Rate}{Avg\ Prod}$
Total Cost = LOC × $ cost per line code
$FP = count\ total \times [0.65 + 0.01 \times sum(Fi)]$
ILF : internal logical file
EIF : external interface
Cost per FP: Labour Rate / Avg Prod
Cost = $ estimated × Cost per FP

COCOMO (Cost Constructive Cost Model)

COCOMO I
Effort = $a(KLOC)^b$ person months
Development time = $cx E^d$ months
Person Estimation = $N = \frac{Effort}{Deadline}$
Productivity = P = KLOC / E

COCOMO II
New object Points (NOP) = Object Points × (100 − %reuse) / 100
Effort in PM = NOP / PROD = person months
Project Schedule: describes the software development cycle for a particular project by enumerating the phases or stages of project & breaking each into discrete tasks or activities to be done
→ project deliverables
Activity: part of project that takes place over a period of time
Milestone: completion of activity
Activity:
Is precursor: an event or set of events that must occur before activity can begin
Is duration: length of time for activity
Is due date:
Is endpoint: deliverable signifying activity has ended
WBS (Work breakdown structure)
Activity Graph
Milestones
Estimating Completion
CPM Activity Graph
CPM Bar Chart
WBS for Commercial Software

Gantt Chart
Task list is a collection of software engineering work tasks that must be accomplished to complete a project
Factors causing variation of Task list
→ Size
→ No of potential users
→ Mission criticality
→ Stability of requirements
ES ST EF
Activity
LS LF

Inception: initial stage where project is conceived & the goals & objectives are outlined. Identifying the key stakeholder, understanding their initial requirements, & setting the project's overall direction
Elicitation: The process of gathering info on requirements from stakeholders & understand their needs & wants expectations. Conducting interviews, surveys, workshops or other methods to extract relevant info from stakeholders
Elaboration: The stage where gathered requirements is analyzed & detailed to create a comprehensive understanding of project scope. Breaking down high-level requirements into more detailed specifications, creating more use cases, & developing a more intricate plan
Negotiation: The process of reconciling conflicting requirements & priorities among stakeholders. Facilitating discussions & negotiations to resolve conflicts, finding compromises, & ensuring alignment among stakeholders
Specification: The formal doc & communication of detailed project requirements & specifications
- Creating detailed doc, such as requirements doc, design specifications, & other relevant materials that serve as a reference for project development
Validation: The stage where the documented requirements are reviewed & verified to ensure they meet the stakeholders' expectations. Conducting reviews, inspections, & tests to validate that the project deliverables align with the specified requirements & meet the intended objectives

Spiral Model
Pros
- continuous customer involvement
- development risks are managed
- suitable to large, complex projects
- works well for extensible product
Cons
- Risk analysis failures can doom the project
- Hard to manage project
- requires an expert dev team

UPM
Pros
- Quality doc is emphasized
- Continuous customer involvement
- accommodate requirement changes
- Works well for maintenance projects
Cons
- Use cases are not always precise
- Overlapping phases can cause problems
- requires expert dev team

RAD
Pros
- encourages & prioritizes customer feedback
- reduced development time
- increases the reusability of features
Cons
- requires highly skilled designers
- suitable only for projects that have small development time
- Only a system that can be modularized can use RAD
- high technical risk, not suitable
- requires user involvement