Unit I – Software Crisis & Evolution (Q1)

1960s–70s me jab hardware cheap aur powerful hone laga, organizations ne large, complex software systems demand karna shuru kiya, lekin development approach abhi immature thi. Result tha "software crisis": projects bar-bar late deliver hote the, budget bahut exceed hota tha, systems unreliable, crash-prone aur maintain karna mushkil hote the. Symptoms me schedule slippage, cost overrun, poor quality, documentation ki kami, user ki real needs se mismatch, aur changes handle na kar pana include the. Root causes: ad-hoc coding, proper design/analysis na karna, weak project management, unrealistic estimates, tools/techniques ka lack, aur rapidly changing hardware/environment.

Is crisis ne "software engineering" ko janam diya—idea ki software ko bhi engineering discipline jaise treat kiya jaye, jahan systematic process, documentation, standards, metrics, reviews aur testing follow ho. Life-cycle models (waterfall, incremental, spiral) ne phases clear kiye: feasibility, requirements, design, implementation, testing, deployment, maintenance. Structured programming, modular design, data abstraction, information hiding ne complexity ko manageable parts me todna easier banaya. Quality assurance, code reviews, configuration management, version control jaise practices se defects early catch hone lage.

Tools side par higher-level languages, IDEs, debuggers, test frameworks, CASE tools, SCM systems aaye, jisse productivity aur control dono improve hua. Later agile methods, DevOps, continuous integration/delivery ne fast feedback aur continuous improvement culture promote kiya, jisse rework aur big-bang failures kam hue. Aaj bhi large distributed, safety-critical systems me risk hai, lekin disciplined software engineering ne crisis ko kaafi had tak address kar diya: zyada predictable schedules, better quality aur manageable maintenance possible ho payi.

Unit II – Requirements & Product Qualities (Q2/6 combined)

Requirements analysis SDLC ka wo phase hai jahan "system kya karega" clearly define kiya jata hai. Analyst stakeholders se baat karke business goals, problems, constraints samajhta hai aur functional requirements (system ke services) plus non-functional requirements (performance, security, usability) identify karta hai. Techniques jaise interviews, workshops, questionnaires, observation, prototyping use hote hain, phir unhe models (use cases, DFD, ERD) aur SRS document me structure kiya jata hai. Good requirements clear, unambiguous, complete, consistent, testable, traceable aur modifiable honi chahiye; warna baad me costly rework hota hai.

Software product/process ke important qualities directly isi phase se influence hote hain. Correctness ka matlab software documented requirements ko sahi tarah satisfy kare. Reliability defect rate aur failure frequency se related hai—system stable rahe aur predictable output de. Robustness unexpected inputs ya environment changes par bhi reasonable behavior maintain kare. User-friendliness me simple UI, clear messages, help aur consistency aati hai. Maintainability ka matlab code ko easily modify/fix/extend kar pana; modular design, good documentation aur coding standards isko support karte hain. Reusability reusable components/libraries se future projects fast banane me help karti hai. Portability different hardware/OS platforms par kam changes me run karne ki capability hai. Data abstraction aur

modularity complexity hide karke sirf essential interface expose karte hain, jisse large systems ko logically organize karna easy hota hai. Principles of software engineering (modularity, abstraction, separation of concerns, incremental development, verification) inhi qualities ko achieve karne ki guidelines hain.

Unit III – 4GL, End-User Computing & System Investigation (Q8/9)

Fourth Generation Languages (4GLs) high-level tools/languages hain jo “what” specify karne par focus karte hain, “how” ka detail tool handle karta hai. Examples: SQL, report generators, form builders, query-by-example tools. Inka aim productivity boost karna hai—kam code, zyada kaam. End-user computing ka matlab hai ki non-programmer business users khud chhote reports, queries, macros ya apps bana sakte hain, mostly 4GL tools ka use karke. Advantages: development fast, user ke control me, IT department ka load kam, rapid prototyping possible.

Lekin limitations bhi hain: 4GLs usually performance-critical ya very complex logic handle nahi kar pate, generated code less optimized hota hai, aur vendor-specific tools lock-in ka risk create kar sakte hain. Non-technical users kabhi-kabhi undocumented, insecure, inconsistent “shadow systems” bana dete hain jisse data integrity aur maintenance issues ho sakte hain. Isliye governance, standards aur integration planning zaruri hai.

Systems investigation information system life cycle ka starting phase hai jahan problem ko properly samjha jata hai aur decide kiya jata hai ki naya/modified system justified hai ya nahi. Steps: problem recognition, preliminary study, feasibility analysis (technical, economic, operational), risk assessment, high-level requirement identification. Is stage me users, managers, system analysts, domain experts involve hote hain. Objectives: correct problem define karna, goals clarify karna, rough cost-benefit dekhna, aur decide karna ki project aage badhna chahiye ya nahi. Poor system investigation se wrong solution ya scope creep hota hai, isliye clear objectives, stakeholder involvement aur documented findings bohot important hain.

Unit IV – SDLC, PERT & CASE Tools (Q10/11/12)

System Development Life Cycle (SDLC) ek structured framework hai jo software project ko phases me divide karta hai: requirements, analysis/design, coding, testing, implementation, maintenance, review. Requirements phase me “what” define hota hai; analysis/design me architecture, data structures, interfaces, algorithms plan hote hain. Coding me actual implementation hota hai, jise coding standards follow karke likha jata hai. Testing levels—unit, integration, system, acceptance—defects detect karte hain aur verify karte hain ki system requirements meet kar raha hai. Implementation me conversion, training, deployment hota hai; maintenance me bug fixes, enhancements, environment changes handle kiye jate hain. System review periodically check karta hai ki system still business needs ke sath aligned hai ya nahi.

PERT (Program Evaluation Review Technique) project management tool hai jo activities ko network diagram ke roop me represent karta hai, dependencies aur critical path identify karta

hai. Har activity ke liye optimistic, most likely, pessimistic estimates se expected duration compute hoti hai; isse schedule risk samajhna easy ho jata hai. Software projects me PERT milestones (requirements complete, design freeze, code complete, test cycles) plan karne, slippage predict karne aur resources adjust karne me help karta hai.

CASE (Computer-Aided Software Engineering) tools software development activities ko automate/support karte hain—requirements management, modeling (DFD/UML), code generation, testing, configuration management, documentation, etc. Upper CASE tools earlier phases (analysis/design), lower CASE implementation/testing, integrated CASE both cover karte hain. Benefits: productivity, consistency, auto-documentation, error reduction, standardization. Selection me factors: project type, existing process fit, integration capability, training effort, performance, cost, vendor support. Engineering approach + SDLC + CASE tools milkar cost control, quality improvement aur repeatable process establish karte hain.