SEMESTER V - EXAMINATION QUESTION BANK

Paper XI: Software Engineering

Syllabus Scope: Units I - IV

Total Questions: *200*

Target Weightage: 1, 2, 5, & 8 Marks

Format: *structured PDF*

UNIT I: Introduction & Software Lifecycle Models

--- 1 Mark Questions ---

Q1. Define Software Engineering. *[1 Mark]*

Q2. What is meant by exploratory style of software development? *[1 Mark]*

Q3. State one main reason for the emergence of software engineering as a discipline. *[1 Mark]*

Q4. What is a software development life cycle (SDLC) model? *[1 Mark]*

Q5. Name two extensions of the classic Waterfall model. *[1 Mark]*

Q6. What does RAD stand for in software engineering? *[1 Mark]*

Q7. Define the term 'Software Crisis'. *[1 Mark]*

Q8. Mention one major limitation of the classical Waterfall model. *[1 Mark]*

Q9. What is the core philosophy of Agile development models? *[1 Mark]*

Q10. Which software lifecycle model uses a risk-driven approach? *[1 Mark]*

Q11. Define Computer Systems Engineering. *[1 Mark]*

Q12. Give one advantage of using the Prototype lifecycle model. *[1 Mark]*

Q13. What is a software development project? *[1 Mark]*

Q14. How does the evolutionary development style differ from exploratory style? *[1 Mark]*

Q15. State one change in modern software development practices compared to early practices. *[1 Mark]*

Q16. In which phase of the Waterfall model are system requirements documented? *[1 Mark]*

Q17. What is an iteration in the context of Agile models? *[1 Mark]*

Q18. Name any one Agile framework widely used in the software industry. *[1 Mark]*

Q19. True or False: The Spiral model is suitable for small, low-risk software products. *[1 Mark]*

Q20. What is the primary focus of the Rapid Application Development model? *[1 Mark]*

--- 2 Marks Questions ---

Q21. Differentiate between a software product and a software program. *[2 Mark]*

Q22. Why is the exploratory style of software development unsuited for large commercial applications? *[2 Mark]*

Q23. Explain how software engineering shifted from a craft to an organized engineering discipline. *[2 Mark]*

Q24. Briefly describe the Iterative Waterfall model and its advantages over the classical model. *[2 Mark]*

Q25. What are the main characteristics of the Rapid Application Development (RAD) model? *[2 Mark]*

Q26. Explain the concept of risk assessment in the context of the Spiral lifecycle model. *[2 Mark]*

Q27. What is Agile software development? Mention two core values from the Agile Manifesto. *[2 Mark]*

Q28. How does Computer Systems Engineering interact with Software Engineering? *[2 Mark]*

Q29. Explain the role of prototyping in modern software lifecycles. *[2 Mark]*

Q30. Identify two major disadvantages of using an Agile methodology for large-scale systems. *[2 Mark]*

Q31. What are software development practices, and why have they changed over time? *[2 Mark]*

Q32. Describe the structure of a single quadrant or loop in the Spiral model. *[2 Mark]*

Q33. Define scalability in software projects and explain its significance. *[2 Mark]*

Q34. Why is document maintenance considered a critical part of lifecycle models? *[2 Mark]*

Q35. Compare the time-boxing concept of RAD with sprint iterations in Agile. *[2 Mark]*

--- 5 Marks Questions ---

Q36. Discuss the evolution of software from an ad-hoc craft to a structured engineering discipline. *[5 Mark]*

Q37. Explain the exploratory style of software development. Analyze its limitations when applied to modern corporate projects. *[5 Mark]*

Q38. Detail the classical Waterfall model. Explain its sequential phases and why it is often considered rigid. *[5 Mark]*

Q39. Elaborate on the Rapid Application Development (RAD) model, highlighting its phases, pros, and cons. *[5 Mark]*

Q40. Provide an in-depth explanation of Agile Development Models. Discuss how they handle changing requirements dynamically. *[5 Mark]*

Q41. Analyze the Spiral Model. Draw or explain its quadrants and justify why it is called a risk-driven lifecycle model. *[5 Mark]*

Q42. Compare and contrast the Waterfall Model and the Agile Model based on flexibility, risk management, and client involvement. *[5 Mark]*

Q43. Discuss the historical changes in software development practices and how automation and tools influenced this shift. *[5 Mark]*

Q44. Explain the concept and scope of Computer Systems Engineering, describing how hardware, software, and human factors integrate. *[5 Mark]*

Q45. Describe the Prototype model extension of the Waterfall process. Under what circumstances is prototyping highly recommended? *[5 Mark]*

--- 8 Marks Questions ---

Q46. Provide a comprehensive analysis of the evolution of Software Engineering. Discuss the prominent factors, software crises, and shifts in development practices that mandated engineering rigors. *[8 Mark]*

Q47. Critically evaluate the Spiral Model. Explain its architectural framework, the specific activities performed in each quadrant, and analyze its strengths and weaknesses for large-scale enterprise deployments. *[8 Mark]*

Q48. Compare and contrast the Waterfall Model, RAD Model, and Agile Development frameworks. Provide a detailed comparative matrix based on project size, risk tolerance, client collaboration, delivery velocity, and handling of fluid requirements. *[8 Mark]*

Q49. Discuss Agile methodologies in detail. Explain Scrum or extreme programming (XP) lifecycle principles, ceremonies, artifacts, and evaluate how Agile effectively remedies the traditional deficiencies of predictive frameworks. *[8 Mark]*

Q50. Explain the full spectrum of Software Lifecycle Models. Define what guides a project manager to select a specific lifecycle approach (Waterfall, RAD, Spiral, or Agile) given a set of unique project constraints and operational environments. *[8 Mark]*

UNIT II: Software Project Management

--- 1 Mark Questions ---

Q51. Define Software Project Management. *[1 Mark]*

Q52. State one complexity unique to managing software projects. *[1 Mark]*

Q53. What is the primary responsibility of a Software Project Manager? *[1 Mark]*

Q54. What is project planning in software management? *[1 Mark]*

Q55. Define a software metric. *[1 Mark]*

Q56. Name a popular metric used for software project size estimation. *[1 Mark]*

Q57. What does LOC stand for in software metrics? *[1 Mark]*

Q58. What is a Function Point (FP)? *[1 Mark]*

Q59. Name one empirical estimation technique. *[1 Mark]*

Q60. What does COCOMO stand for? *[1 Mark]*

Q61. Who proposed the COCOMO model? *[1 Mark]*

Q62. What are the three development modes defined in basic COCOMO? *[1 Mark]*

Q63. Name one metric proposed in Halstead's Software Science. *[1 Mark]*

Q64. Define staffing level estimation. *[1 Mark]*

Q65. What is a Gantt chart used for in software project scheduling? *[1 Mark]*

Q66. Define software risk management. *[1 Mark]*

Q67. What does SCM stand for in software project management? *[1 Mark]*

Q68. What is a baseline in Software Configuration Management? *[1 Mark]*

Q69. Name one common software project team structure. *[1 Mark]*

Q70. What is the purpose of a software project schedule? *[1 Mark]*

--- 2 Marks Questions ---

Q71. Why are software projects inherently more complex to manage than traditional hardware construction projects? *[2 Mark]*

Q72. Summarize the key responsibilities of a Software Project Manager during the planning phase. *[2 Mark]*

Q73. Differentiate between product metrics and process metrics in software engineering. *[2 Mark]*

Q74. Explain how Lines of Code (LOC) is used as a size metric, noting one major disadvantage. *[2 Mark]*

Q75. Explain the concept behind Function Point (FP) metrics and why it is language-independent. *[2 Mark]*

Q76. What is the difference between algorithmic models and empirical techniques for project estimation? *[2 Mark]*

Q77. Briefly describe the basic COCOMO model and its underlying formula format. *[2 Mark]*

Q78. What are Halstead’s Software Science metrics? List any two parameters measured. *[2 Mark]*

Q79. Explain how staffing levels vary over the timeline of a software development lifecycle. *[2 Mark]*

Q80. Define software scheduling. How do milestones help track progress? *[2 Mark]*

Q81. Describe the democratic team structure and its ideal application environment. *[2 Mark]*

Q82. What is risk identification, and why must it be performed continuously? *[2 Mark]*

Q83. Explain the role of configuration control in Software Configuration Management. *[2 Mark]*

Q84. What is Halstead's Program Volume, and how does it relate to program length? *[2 Mark]*

Q85. Briefly explain the Chief Programmer team structure. *[2 Mark]*

--- 5 Marks Questions ---

Q86. Analyze the unique complexities involved in Software Project Management and outline how they impact overall project delivery. *[5 Mark]*

Q87. Explain the key responsibilities and skill sets required for a successful Software Project Manager. *[5 Mark]*

Q88. Discuss project size estimation techniques. Focus on the comparative advantages and disadvantages of LOC vs. Function Point (FP). *[5 Mark]*

Q89. Detail the Intermediate COCOMO model. Explain how Cost Drivers and Effort Multipliers adjust the initial effort estimation. *[5 Mark]*

Q90. Explain Halstead's Software Science. Discuss how it derives program length, vocabulary, volume, and effort from basic operators and operands. *[5 Mark]*

Q91. Discuss staffing level estimation and explain the Rayleigh curve model's application to software project human resource allocation. *[5 Mark]*

Q92. Elaborate on project scheduling techniques, detailing the importance of Work Breakdown Structures (WBS) and Critical Path Methods (CPM). *[5 Mark]*

Q93. Describe different organization and team structures (Democratic, Chief Programmer, Hierarchical) and analyze their operational effectiveness. *[5 Mark]*

Q94. Explain the risk management process in detail, highlighting risk identification, projection, mitigation, and monitoring. *[5 Mark]*

Q95. Discuss Software Configuration Management (SCM). Explain its major activities including configuration identification, version control, and auditing. *[5 Mark]*

--- 8 Marks Questions ---

Q96. Provide an exhaustive guide on the COCOMO model family. Detail the evolution from Basic to Intermediate and Detailed COCOMO, write down the essential mathematical formulas, define the development modes, and show how it calculates effort and schedule. *[8 Mark]*

Q97. Critically examine Software Configuration Management (SCM). Define why it is essential for multi-developer corporate projects, and detail its main pillars: identification, version control, change management, status accounting, and audits. *[8 Mark]*

Q98. Develop an integrated Software Project Plan framework. Discuss how a project manager orchestrates size estimation, effort calculation, scheduling constraints (using PERT/CPM), staffing distribution, and proactive risk management. *[8 Mark]*

Q99. Explain Halstead's Software Science framework completely. Define operators, operands, program length, vocabulary, volume, difficulty, and effort. Provide a full critique of its theoretical basis vs. modern programming languages. *[8 Mark]*

Q100. Discuss risk management within software engineering. Detail the taxonomy of software risks, the steps of risk analysis, quantification strategies, and outline comprehensive mitigation and contingency plans for common project failures. *[8 Mark]*

UNIT III: Requirement Analysis, Specification & Design

--- 1 Mark Questions ---

Q101. What is requirement gathering? *[1 Mark]*

Q102. What does SRS stand for? *[1 Mark]*

Q103. Who is the primary reader of an SRS document? *[1 Mark]*

Q104. Define a functional requirement. *[1 Mark]*

Q105. Define a non-functional requirement. *[1 Mark]*

Q106. What is formal system specification? *[1 Mark]*

Q107. Name one approach to formal specification. *[1 Mark]*

Q108. What is an axiomatic specification? *[1 Mark]*

Q109. What is an algebraic specification? *[1 Mark]*

Q110. What is an executable specification? *[1 Mark]*

Q111. What does 4GL stand for? *[1 Mark]*

Q112. Define Software Design. *[1 Mark]*

Q113. What is a good software design characteristic? *[1 Mark]*

Q114. Define modularity in software design. *[1 Mark]*

Q115. What is cohesion? *[1 Mark]*

Q116. What is coupling? *[1 Mark]*

Q117. What is a layered arrangement of modules? *[1 Mark]*

Q118. Name two approaches to software design. *[1 Mark]*

Q119. What is function-oriented software design? *[1 Mark]*

Q120. What is object-oriented software design? *[1 Mark]*

--- 2 Marks Questions ---

Q121. What are the main techniques used during the requirements gathering and analysis phase? *[2 Mark]*

Q122. Explain the significance of an SRS document in the software development lifecycle. *[2 Mark]*

Q123. Differentiate between functional and non-functional requirements with examples. *[2 Mark]*

Q124. What is formal specification, and why is it used in safety-critical systems? *[2 Mark]*

Q125. Explain the underlying concept of algebraic specification. *[2 Mark]*

Q126. How do executable specifications differ from traditional textual specifications? *[2 Mark]*

Q127. What is the role of Fourth Generation Languages (4GL) in software development? *[2 Mark]*

Q128. Describe the core objective of the software design process. *[2 Mark]*

Q129. What makes a software design high quality? Mention two primary attributes. *[2 Mark]*

Q130. Explain the relationship between cohesion and coupling in a modular design. *[2 Mark]*

Q131. Why is high cohesion desirable within a software module? *[2 Mark]*

Q132. Why is loose coupling preferred between independent software modules? *[2 Mark]*

Q133. Explain the layered arrangement architectural concept of software modules. *[2 Mark]*

Q134. Briefly describe the function-oriented approach to software design. *[2 Mark]*

Q135. How does the object-oriented design approach represent real-world entities? *[2 Mark]*

--- 5 Marks Questions ---

Q136. Detail the processes involved in Requirements Gathering and Analysis. Discuss the challenges faced by engineers when interacting with clients. *[5 Mark]*

Q137. Explain the structure, characteristics, and essential components of a high-quality Software Requirement Specification (SRS) document. *[5 Mark]*

Q138. Discuss Formal System Specifications, contrasting Axiomatic and Algebraic approaches with practical contextual examples. *[5 Mark]*

Q139. Explain Executable Specifications and examine the role and impact of 4GL in automated code generation and rapid prototyping. *[5 Mark]*

Q140. Describe the Software Design process. Detail how abstract requirements are transformed into concrete technical blueprints. *[5 Mark]*

Q141. Explain Cohesion in software design. Identify and describe at least four distinct levels of cohesion from weakest to strongest. *[5 Mark]*

Q142. Explain Coupling in software design. Identify and describe at least four distinct types of coupling from most desirable to least desirable. *[5 Mark]*

Q143. Discuss the Layered Arrangement of Modules. Explain how architectural layering enhances system abstraction, reuse, and maintenance. *[5 Mark]*

Q144. Elaborate on the Function-Oriented Design approach, detailing concepts like Data Flow Diagrams (DFD) and Structure Charts. *[5 Mark]*

Q145. Elaborate on the Object-Oriented Design approach, highlighting concepts like classes, inheritance, encapsulation, and polymorphism. *[5 Mark]*

--- 8 Marks Questions ---

Q146. Provide an in-depth treatise on Cohesion and Coupling. Define every level of cohesion and type of coupling, provide concrete code or structural scenarios for each, and mathematically/logically justify why high cohesion and loose coupling ensure robust software design. *[8 Mark]*

Q147. Critically analyze the Requirements Engineering phase. Detail requirement elicitation, analysis, negotiation, modeling, documentation (SRS criteria according to IEEE standards), and validation processes to minimize downstream defects. *[8 Mark]*

Q148. Compare and contrast Function-Oriented Design (FOD) and Object-Oriented Design (OOD). Discuss structural differences, modeling abstractions, maintainability, scalability, and map how each approach transforms requirements into design artifacts. *[8 Mark]*

Q149. Explain the concept of Formal Methods in Software Specification. Detail the mathematical principles behind Axiomatic and Algebraic specifications, evaluate their application in high-integrity systems, and discuss their industry adoption bottlenecks. *[8 Mark]*

Q150. Analyze the Software Architecture and Design principles. Discuss how modularity, information hiding, abstraction, step-wise refinement, and layered arrangements of modules prevent structural decay and simplify maintenance. *[8 Mark]*

UNIT IV: Coding and Testing

--- 1 Mark Questions ---

Q151. What is the primary goal of the coding phase? *[1 Mark]*

Q152. Define code review. *[1 Mark]*

Q153. What is software documentation? *[1 Mark]*

Q154. Define software testing. *[1 Mark]*

Q155. What is unit testing? *[1 Mark]*

Q156. What is black box testing? *[1 Mark]*

Q157. What is white box testing? *[1 Mark]*

Q158. Define debugging. *[1 Mark]*

Q159. How does debugging differ from testing? *[1 Mark]*

Q160. What is a static program analysis tool? *[1 Mark]*

Q161. What is integration testing? *[1 Mark]*

Q162. What is system testing? *[1 Mark]*

Q163. Define software maintenance. *[1 Mark]*

Q164. Name one type of software maintenance. *[1 Mark]*

Q165. What is regression testing? *[1 Mark]*

Q166. What is dynamic analysis? *[1 Mark]*

Q167. What is code walkthrough? *[1 Mark]*

Q168. What is boundary value analysis? *[1 Mark]*

Q169. What is code inspection? *[1 Mark]*

Q170. What is the purpose of a test case? *[1 Mark]*

--- 2 Marks Questions ---

Q171. Why is a formal code review process important in professional software projects? *[2 Mark]*

Q172. Differentiate between internal documentation and external documentation in coding. *[2 Mark]*

Q173. Explain why software testing can only prove the presence of defects, not their absence. *[2 Mark]*

Q174. What are the main entry and exit criteria for unit testing? *[2 Mark]*

Q175. Explain the concept of black box testing and mention one of its techniques. *[2 Mark]*

Q176. Explain the concept of white box testing and mention one of its techniques. *[2 Mark]*

Q177. What is the primary role of Program Analysis Tools in modern software construction? *[2 Mark]*

Q178. Describe the purpose of integration testing and why it follows unit testing. *[2 Mark]*

Q179. What is system testing, and how does it differ from integration testing? *[2 Mark]*

Q180. Define software maintenance and state why it often consumes the largest share of the lifecycle budget. *[2 Mark]*

Q181. Differentiate between corrective and adaptive software maintenance. *[2 Mark]*

Q182. Explain the difference between a code walkthrough and a formal code inspection. *[2 Mark]*

Q183. What is cyclomatic complexity, and which testing strategy uses it? *[2 Mark]*

Q184. Briefly explain the integration testing approach known as Top-Down integration. *[2 Mark]*

Q185. What is the role of an automated test runner tool? *[2 Mark]*

--- 5 Marks Questions ---

Q186. Discuss the coding standard guidelines and explain why practices like code reviews and documentation are vital for long-term project survival. *[5 Mark]*

Q187. Explain software testing fundamentals. Discuss psychology of testing, test case design, and the difference between verification and validation. *[5 Mark]*

Q188. Detail Black Box Testing. Explain Equivalence Partitioning and Boundary Value Analysis with clear engineering examples. *[5 Mark]*

Q189. Detail White Box Testing. Explain Control Flow Testing, Basis Path Testing, and how Cyclomatic Complexity determines test coverage. *[5 Mark]*

Q190. Compare and contrast Testing and Debugging, outlining their objectives, processes, and typical tools utilized. *[5 Mark]*

Q191. Discuss Program Analysis Tools. Differentiate between static analysis tools and dynamic analysis tools, explaining their deployment benefits. *[5 Mark]*

Q192. Elaborate on Integration Testing. Compare Big Bang, Top-Down, and Bottom-Up integration testing methodologies. *[5 Mark]*

Q193. Describe System Testing. Detail its different forms including Functional, Performance, Stress, Security, and Acceptance Testing. *[5 Mark]*

Q194. Explain Software Maintenance. Identify and detail the four major types: Corrective, Adaptive, Perfective, and Preventive maintenance. *[5 Mark]*

Q195. Discuss the challenges of Software Maintenance. Explain how legacy code, poor design, and staff turnover compound maintenance costs. *[5 Mark]*

--- 8 Marks Questions ---

Q196. Provide a comprehensive, structural analysis of Software Testing levels. Detail the progressive sequence from Unit Testing to Integration Testing, System Testing, and Acceptance Testing. Explain the goals, methodologies, and specific stakeholders associated with each level. *[8 Mark]*

Q197. Critically evaluate Black Box and White Box testing paradigms. Provide deep technical explanations of Equivalence Class Partitioning, Boundary Value Analysis, Cause-Effect Graphing, Statement Coverage, Branch Coverage, and Path Coverage, including mathematical tracking of code metrics. *[8 Mark]*

Q198. Detail the full landscape of Software Maintenance. Explain Lehman's laws of software evolution, the key structural models of maintenance costs, the detailed processes of re-engineering and reverse engineering, and strategies to prolong the utility of legacy code bases. *[8 Mark]*

Q199. Explain the Code Quality Assurance ecosystem. Discuss structural coding standards, formal code inspection methodologies (Fagan inspections), code walkthrough structures, and analyze how static code verification tools proactively minimize system crashes. *[8 Mark]*

Q200. Develop a comprehensive Testing Strategy for an enterprise software product. Discuss test plan generation, manual vs. automated testing trade-offs, test suite selection, test execution environments, defect tracking workflows, and the integration of continuous regression testing. *[8 Mark]*

This question bank is generated based on standard academic curriculums for Software Engineering. AI responses may include mistakes.