USN | | | | | | | | | | |

Internal Assessment Test 2 – December 2025

Sub:	Software Quality Assurance					Sub Code:	BIS714B	Branch:	ISE
Date:	02/12/2025	Duration:	90 min's	Max Marks:	50	Sem/Sec:	VII/ A, B, C		OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1a)	Explain the objectives of software testing and discuss their importance. **Scheme:** Definition of software testing – 1M Any 4 objectives with their importance – 2M Discussion of importance – 2M **Solution**: Software testing is the process of executing a program to find defects and provide information about quality before release. Objectives: i) Detect defects in code, design, and requirements before deployment. ii) Verify conformance of the software to specified requirements and design. iii) Validate user needs, ensuring the system does what the customer actually wants. iv) Assess quality attributes such as reliability, performance, usability, and security. Importance: i) Reduces the cost of failure by catching problems before they appear in the field, where correction is very expensive. ii) Prevents financial and reputational damage caused by failures at the customer site. iii) Supports compliance with contracts, standards, and regulations. iv) Improves maintainability, because defects and design weaknesses are removed early, leading to cleaner code and documentation.	5M	CO3	L3
1b)	Categorize different software testing strategies with suitable examples. **Scheme:** Any 4 strategies, each with a short explanation + example- 5 **Solution:** i) Unit testing: Tests individual modules/functions in isolation (white-box or black-box). *Example:* Testing a calculateInterest() method with different principal and rate values. ii) Integration testing: Tests interactions between integrated modules; strategies: top-down, bottom-up, sandwich. *Example:* Integrating "Login UI" with "Authentication service" and testing data flow.	5M	CO3	L3

	iii) System testing: Tests the complete integrated system against its system requirements. *Example:* Testing an entire online banking system, including fund transfer, balance enquiry, and statement generation with realistic data. iv) Acceptance testing: Performed by or on behalf of the customer to decide whether to accept the system. Types: alpha (in developer's environment), beta (at customer side). *Example:* Client's UAT of a payroll application before going live.			
2a)	How do CASE tools influence software quality? Explain with examples. **Scheme:** Definition of CASE tools – 1M Any 3–4 quality influences with examples – 4M **Solution:** CASE (Computer-Aided Software Engineering) tools are automated tools that support activities across the software life cycle (analysis, design, coding, testing, maintenance). Influence on quality (with examples): i) Improved requirements and design consistency: Modeling tools (e.g., UML tools) detect inconsistencies, missing relationships, and maintain traceability from requirements to design → fewer design defects. ii) Error reduction through automation: Code generators and refactoring tools avoid manual coding mistakes and enforce standard patterns, improving reliability and maintainability. iii) Better testing effectiveness: Test management tools, automated test frameworks, and coverage analyzers help derive systematic test cases, automate regression tests, and measure coverage → more defects caught before release. iv) Configuration and change control: Version control and configuration management tools track changes and maintain baselines, preventing wrong versions and lost fixes, which directly improves product stability. v) Metrics and reporting: Project dashboards, defect-tracking tools, and metric tools provide quantitative feedback to manage quality (defect trends, effort, schedule adherence).	5M	CO3	L4
2b)	Analyze the cost-effectiveness of SQA defect removal during different phases of S **Scheme:** Explain the idea that defect cost varies by phase – 2M Use relative cost figures/filtering effectiveness – 2M Conclude with analysis of cost-effective SQA policy – 1M **Solution:** Studies summarized by Boehm and others show that the cost of removing a defect increases steeply in later SDLC phases. A typical relative cost pattern is: requirements, design, unit test, integration test, system/acceptance test, and operation/maintenance.	5M	CO3	L4

	At the same time, each SQA activity (reviews, inspections, tests) acts as a filter with a specific defect-removal effectiveness. The multiple-filter model shows that: i) Early reviews (requirements, design) remove a large proportion of defects at very low cost per defect. i) If these activities are weak or missing, many defects escape to system test or operation, where each defect costs up to 40–110 times more to remove. Iii) Adding more late testing reduces failures but is less cost-effective than strengthening early reviews and inspections. Therefore, an SQA plan is most cost-effective when it invests adequately in early-phase verification activities (reviews, inspections, walkthroughs) combined with well-designed testing later, so that the total cost of detection + correction + residual failures is minimized, not just the testing cost.			
3a)	Explain the main components of project progress control with suitable examples. **Scheme:** List 4 main components – 2M Short explanation + example for each – 3M **Solution:** i) Control of risk management activities: Follow up on identified risk items and the effectiveness of mitigation actions. Example: Periodically reviewing "key staff leaving" risk and verifying backup staffing plans. ii) Project schedule control: Monitoring completion of activities and milestones against the approved schedule. Example: Using milestone reports to detect that "module integration" is delayed by 2 weeks and taking corrective action. iii) Project resource control: Tracking use and mix of resources (mainly human, but also tools and facilities). Example: Checking whether senior analysts' effort is exceeding plan and rebalancing roles to avoid budget overrun. iv) Project budget control: Comparing actual expenditure with planned budget for items like manpower, tools, COTS, hardware, and subcontracting. Example: Detecting overspending on subcontractor testing and renegotiating scope or reallocating tasks.	5M	CO4	L2
3b)	Discuss various computerized tools used for project progress monitoring and their benefits. **Scheme:** Mention 4 tools and their benefits- 5M **Solution:** Risk management control: Maintain lists of risk items, categories, and planned solution dates; highlight overdue items that threaten the completion date. Schedule control: Gantt/PERT charts, lists of delayed activities and delayed milestones, automatic recalculation of critical paths, and updated schedules for teams and units. Resource control: Reports on planned vs. actual resource usage (man-hours, tool time), utilization graphs, and alerts when limits are exceeded. Budget control: Tracking actual expenditure vs. planned for manpower, tools, hardware, and subcontractors; early indication of overruns. Benefits: Centralized, up-to-date information; better visibility into critical delays and risks; faster decision-making and "what-if" analysis; reduced manual reporting effort and errors.	5M	CO4	L3

4a)	Classify software quality metrics into process and product metrics with examples. **Scheme:** Definition of both- 1M Classification with examples- 4M **Solution:** • Process metrics measure how well the development/maintenance process is performed. • Product metrics measure characteristics of the delivered software product in operation. Process metrics (examples from Chapter 21): i) Time Table Observance (TTO): TTO = MSOT / MS (milestones completed on time / total milestones). Indicates how well the project adheres to its schedule. ii) Average Delay of Milestone Completion (ADMC): Sum of delays for all milestones/number of milestones. Higher ADMC indicates poorer schedule performance. iii) Code Error Density (CED): CED = NCE / KLOC (number of code errors per KLOC). Indicates the quality of the development process. Product metrics (examples): i) Failure rate in operation (e.g., failures/month) – reflects product reliability experienced by users. ii) Severity-weighted failure density–weighted failures per KLOC or per function point, showing the impact of serious faults. iii) Help desk request rate – number of support calls per user/month; high values may indicate poor usability or documentation quality.	5M	CO3	L3
4b)	Analyze the challenges and limitations in using software quality metrics in organizations. **Scheme:** Any 5 challenges and limitations- 5M **Solution:** • Data collection overhead: Collecting defect, effort, and size data requires time and discipline; teams may see it as extra work. Ii) Poor data quality and consistency: Inconsistent defect classification or inaccurate effort logging makes metric values misleading. Iii) Misinterpretation and wrong incentives: Focusing on "number of defects" alone may cause developers to hide defects instead of reporting them. Iv) Context dependence: Metrics like defect density or productivity vary by project type, domain, and technology; cross-project comparisons may be unfair. V) Lagging indicators: Many metrics reveal problems only after they are serious (e.g., high post-release defects), limiting their preventive value.	5M	CO3	L4

5a)	List and explain major software quality management standards (ISO, CMMI, IEEE). **Scheme:** List 3 families of standards – 1M Brief explanation of each – 4M **Solution:** ISO 9001 / ISO 9000-3 (now ISO/IEC 90003): International quality management standard specifying requirements for a quality management system (QMS). ISO 9000-3 provides guidelines for applying ISO 9001 to software development and maintenance (processes, documentation, management responsibility, resource management, product realization, and improvement). Capability Maturity Model Integration (CMMI): Framework for assessing and improving organizational process maturity. Defines maturity levels (e.g., Initial, Managed, Defined, Quantitatively Managed, Optimizing) and process areas like Project Planning, Configuration Management, Verification, Validation, etc. IEEE software engineering standards (e.g., IEEE 1012, 1028, 12207): IEEE 1012: Standard for Verification and Validation (V&V), defining V&V processes and tasks across the life cycle. IEEE 1028: Standard for software reviews and audits, describing how to conduct inspections, technical reviews, walkthroughs, and audits. IEEE/EIA 12207: Standard for software life-cycle processes (acquisition, supply, development, operation, maintenance).	5M	CO2	L2
5b)	Differentiate between CMM and CMMI models in terms of structure and applicability. **Scheme:** Five differences with example- 5 **Solution:** i) CMM: Originally focused mainly on *software* development processes. CMMI: Integrates software, systems engineering, and other disciplines into single framework. ii) CMM: Single-staged model with 5 maturity levels and Key Process Areas (KPAs). CMMI: Provides both staged representation (maturity levels) and continuou representation (capability levels per process area). iii) CMM: Separate models existed for software, systems engineering, people, acquisition, etc., leading to overlaps and inconsistencies. CMMI: Integrates these into *one coherent model*, reducing redundancy and conflicts. iv) CMM: KPAs grouped by maturity levels, each with goals and key practices CMMI: Uses process areas with specific and generic goals/practices; same generic practices applied across process areas, improving consistency. v) CMM: Mostly used by software development organizations. CMMI: Used widely across domains (software, systems, services, acquisiti and often required in large government/defence projects. vi) CMM: Emphasis on basic software project management discipline. CMMI: Stronger emphasis on quantitative management, continuous improvement, and integration with business objectives.	5M	CO2	L3
6a)	Describe the steps involved in ISO 9000-3 certification for software organizations	5M	CO2	L3

	Scheme: Entire Steps- 5M **Solution:** Steps are as follows: i) Planning the process leading to certification: Decide scope (projects/units covered), select accreditation/certification body, define schedule, and allocate responsibilities and resources. ii) Development of the organization's SQA / quality management system: Define processes, procedures, and templates that satisfy ISO 9000-3 requirements (management responsibility, documentation, design, purchasing, verification, etc.). iii) Implementation of the QMS: Put procedures into practice on real projects; train staff; produce records (plans, reviews, audits, corrective actions) demonstrating that the system is actually used. iv) Internal audits and corrective actions: Conduct internal audits to verify compliance, detect non-conformities, and implement corrective and preventive actions before external audit. v) Undergoing certification audits: External auditors (from a certification body) perform stage-wise audits; if requirements are met, the organization receives the ISO 9000-3 / ISO 9001 certificate. iv) Maintaining certification: Periodic surveillance audits and re-certification every few years; continuous improvement and updating of the QMS.			
6b)	Analyze how IEEE standards (1012 & 1028) support verification, validation, and review activities **Scheme:** Brief explanation of IEEE 1012 and its role – 2M Brief explanation of IEEE 1028 and its role – 2M Combined analysis/benefits – 1M **Solution:** IEEE Std 1012 – Verification and Validation (V&V) i) Defines V&V processes across the entire life cycle: management, acquisitic development, operation, and maintenance. ii) Specifies V&V activities for each phase (requirements, design, implementa installation, operation), and the number of tasks depending on software inte iii) Provides the Software V&V Plan (SVVP) template (purpose, organization, resources, V&V processes, reporting, documentation), ensuring systematic execution of V&V. IEEE Std 1028 – Reviews i) Standardizes *how* to perform technical reviews and audits, including inspec walkthroughs, technical reviews, management reviews, and process/produc ii) Prescribes roles (moderator, author, reviewers, recorder), entry and exit crit steps, and documentation. Combined support/analysis: IEEE 1012 ensures that V&V is planned and integrate cycle with appropriate tasks and resources. IEEE 1028 provides detailed procedure V&V techniques (reviews and inspections) to be effective and repeatable. Togethe increase consistency, completeness, and effectiveness of verification, validation, ar activities, thereby reducing defects and improving confidence in the software produ	5M	CO5	L4