DUMPSBOSS.

Certified Software Quality EngineerExam

ASQ CSQE

Version Demo

Total Demo Questions: 29

Total Premium Questions: 297

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

Topic Break Down

Topic	No. of Questions
Topic 1, General Knowledge	21
Topic 2, Software Quality Management	82
Topic 3, Software Processes	21
Topic 4, Software Project Management	36
Topic 5, Software Engineering	43
Topic 6, Software Testing	75
Topic 7, Mix Questions	19
Total	297

QUESTION NO: 1

Rank in order, from first to last, the following steps in creating an affinity diagram.

Conduct an interview, focus group, or brainstorming session

Draw a diagram

Group similar statements

Identify themes

A. 1, 3, 4, 2

B. 1, 4, 2, 3

C. 2, 1, 4, 3

D. 2, 4, 1, 3

ANSWER: A

Explanation:

The correct sequence is **1, 3, 4, 2**. An affinity diagram begins by obtaining a broad set of unstructured ideas or observations, so conducting an interview, focus group, or brainstorming session is first. This produces the statements that the team will analyze. The team then groups similar statements according to their natural relationships. Affinity analysis is intentionally data-driven: the relationships among the individual statements are allowed to emerge before labels are imposed. After the clusters have formed, the team identifies themes that accurately summarize the common idea represented by each cluster. These themes become the headings or labels for the groups and help convert a large collection of raw input into meaningful information. Finally, the resulting groups and their themes are drawn or documented as an affinity diagram so that the outcome can be communicated, reviewed, and used in subsequent quality analysis, planning, or problem solving. ASQ describes an affinity diagram as a tool for organizing a large number of ideas into groups based on their natural relationships: ASQ Affinity Diagram resource. This sequence also reflects the approach described in ASQ's quality-tool guidance: ASQ Quality Resources.

QUESTION NO: 2

What type of audit is performed to verify that the components of a system match the documented deliverables?

A. Second-party

B. Third-party

C. Physical configuration

D. Functional configuration

ANSWER: C

Explanation:

Physical configuration is correct because a physical configuration audit verifies that the configuration item as built, released, or delivered agrees with its approved configuration documentation. In a software context, this means confirming that the required deliverables—such as source and object files, executables, installation media, release notes, user documentation, licenses, test artifacts, and version identifiers—are present, correctly identified, and consistent with the documented product baseline. The audit provides objective evidence that the contents of a release package accurately represent what the organization says it is delivering. This verification is an important configuration-management control before formal acceptance, release, or transfer to a customer because it establishes the completeness and traceability of the delivered product and associated records. ISO 10007 describes configuration management as including configuration-status accounting and verification activities to establish that a product conforms to its configuration information; see ISO 10007:2017 Configuration management—Guidelines for configuration management. The U.S. Department of Defense's

configuration-management guidance likewise characterizes physical configuration auditing as verifying that the product conforms to its product configuration documentation; see DAU Acquipedia: Configuration Management.

QUESTION NO: 3

Which of the following best describes a process audit?

A. The evaluation of established procedures

B. An investigation of the quality management system

C. The verification that product characteristics are met

D. An internal review of the quality policy

ANSWER: A

Explanation:

The evaluation of established procedures is the best description of a process audit. A process audit examines whether a defined process is being performed as planned and whether its documented methods, controls, inputs, activities, responsibilities, and outputs are suitable and effective. The auditor follows the process through its actual operation, gathers objective evidence such as records and observations, and compares that evidence with the organization's established procedures and applicable requirements. The purpose is not merely to confirm that documentation exists; it is to evaluate consistent implementation and identify opportunities to improve process performance and conformity. This aligns with ISO's auditing guidance, which defines audit evidence as records, statements of fact, or other information relevant to audit criteria and verifiable. In a process-based quality management system, procedures establish the intended way of working, while process auditing determines whether that intended method is effectively executed in practice. ASQ's audit resources also emphasize the use of objective evidence and evaluation against defined criteria. See ISO 19011 auditing guidance and ASQ's auditing resource overview.

QUESTION NO: 4

Which of the following approaches best facilitates the analysis of software product and process metrics to enhance monitoring and reporting techniques?

A. Conducting regular audits to ensure compliance with industry standards

B. Integrating software development tools for better workflow visualization

C. Implementing automated testing tools to increase the frequency of data collection

D. Using statistical methods to determine correlations between process efficiencies and product defects

ANSWER: D

Explanation:

Using statistical methods to determine correlations between process efficiencies and product defects best facilitates meaningful analysis of software product and process metrics. Software quality metrics are most useful when they do more than count events: they must reveal patterns, variation, relationships, and trends that support informed quality-management decisions. Statistical techniques can be used to examine whether changes in cycle time, review effectiveness, test coverage, requirements volatility, or defect-removal efficiency are associated with defect arrival rates, defect density, escaped defects, or reliability outcomes. This analysis helps a software quality engineer distinguish normal process variation from potentially significant changes and identify where corrective or preventive action is most likely to improve results.

Correlation analysis also supports more informative monitoring and reporting because it turns separate operational measures into evidence about how process performance affects product quality. Reports can then communicate trends, relevant relationships, and quantified risk rather than presenting disconnected metric totals. This is consistent with the ASQ CSQE Body of Knowledge emphasis on selecting, collecting, analyzing, and reporting software quality metrics to assess

process and product performance. For statistical quality-control concepts applicable to interpreting variation and process data, see the ASQ Statistical Process Control resource. The CSQE examination scope is available in the ASQ Certified Software Quality Engineer overview.

QUESTION NO: 5

Which of the following techniques will improve the accuracy of data collection the most?

A. Automating the data evaluation process

B. Training staff m the data gathering process

C. Defining the data gathering time frame

D. Analyzing post-project data

ANSWER: A

Explanation:

Automating the data evaluation process is the best choice because a properly designed automated workflow applies the same validation, calculation, formatting, and acceptance rules every time data are processed. This substantially reduces errors introduced by manual transcription, inconsistent calculations, omitted checks, and subjective interpretation. Automation can also enforce required fields, flag values outside defined ranges, preserve timestamps and source information, and create an auditable record of how raw observations were transformed into usable quality data. These controls make collected and reported measures more repeatable and reliable, which is essential when the data will be used for process monitoring, trend analysis, defect decisions, or management reporting. The automation must itself be verified and controlled; once validated, however, it provides consistent execution at a scale that manual evaluation cannot reliably match. NIST's guidance on quality measurement emphasizes controlling the measurement process and evaluating data quality characteristics, including accuracy and consistency. See the NIST/SEMATECH e-Handbook of Statistical Methods and FDA guidance on maintaining trustworthy electronic records through appropriate controls in Part 11 electronic-record guidance.

QUESTION NO: 6

While conducting a unit test, a developer would use a

A. black-box test case

B. driver or stub

C. top-down or bottom-up design

D. scenario-based test case

ANSWER: B

Explanation:

During unit testing, a developer commonly uses a *driver or stub* to isolate the individual software unit from components that are not available, not yet integrated, or outside the intended scope of the test. A driver is a small test program that invokes the unit under test and supplies inputs, simulating the behavior of a calling component. A stub is a controlled replacement for a component or service that the unit under test calls; it returns predefined responses so the developer can verify the unit's behavior under known conditions. These test doubles make the test repeatable and allow the developer to focus on the unit's logic, interfaces, boundary conditions, and error handling without requiring a fully assembled system. This approach is especially useful when dependencies such as databases, external services, hardware interfaces, or adjacent modules would otherwise make the test slow, nondeterministic, or impossible to execute early in development. Unit testing is therefore supported directly by drivers and stubs as practical scaffolding for exercising a module in isolation. The ISTQB glossary defines both a driver and a stub in this testing context; see the ISTQB driver definition and the ISTQB stub definition.

QUESTION NO: 7

The executive summary of a project management report should emphasize details related to

A. completed tasks

B. testing deliverables

C. risks to the project

D. stage phase containment status

ANSWER: C

Explanation:

risks to the project is correct because an executive summary is intended for senior decision-makers who need a concise view of matters requiring attention, decisions, resources, or escalation. Project risks are uncertain events or conditions that could affect objectives such as scope, schedule, cost, quality, or customer satisfaction. Presenting the most significant risks, their likely impact, current exposure, and the planned response enables executives to understand whether the project remains viable and where leadership support may be needed.

This focus supports timely governance: executives can approve contingency actions, remove organizational barriers, accept or transfer major exposures, and ensure that risk owners have adequate authority and funding. A meaningful executive-level report therefore communicates the project's overall health through forward-looking information, rather than attempting to reproduce operational-level detail. The report should make clear which risks are material, how they may affect business outcomes, and whether escalation or a management decision is required. This approach aligns with project-management guidance that treats risk management as a continuing process for identifying, analyzing, responding to, and monitoring uncertainty throughout the project life cycle. See the Project Management Institute's risk management guidance and ASQ's overview of risk management.

QUESTION NO: 8

In order for a risk management plan to be effective., it should be

A. updated regularly to address new risks

B. monitored continuously to determine risk frequency

C. based on maturing goals and sub-goals

D. reviewed for contributions to the risk database

ANSWER: A

Explanation:

An effective risk management plan must be **updated regularly to address new risks**. Risk management is an ongoing lifecycle activity rather than a document created once and then left unchanged. As software projects progress, requirements, technologies, suppliers, threat conditions, schedules, and organizational priorities can change. Each change can introduce risks, alter the likelihood or impact of known risks, or make previously selected responses unsuitable.

Regularly updating the plan keeps the risk register, assessment criteria, ownership assignments, mitigation actions, contingency triggers, and monitoring approach aligned with the project's current conditions. It also ensures that management decisions are based on current risk exposure and that resources are directed to the most significant risks. This supports timely action before a risk event becomes an issue affecting quality, cost, delivery, security, or customer satisfaction.

NIST describes risk management as a continuous process that includes ongoing assessment and monitoring, and its guidance emphasizes maintaining risk information as conditions evolve. See NIST SP 800-30 Rev. 1 and NIST SP 800-37 Rev. 2.

QUESTION NO: 9

An auditor is reviewing compliance with a software change-control procedure. The procedure requires that each production change have documented impact analysis, approval, implementation evidence, and post-implementation verification. Which of the following would provide the strongest objective evidence that the procedure was followed for a sampled change?

A. A manager's statement that the team consistently follows the procedure

B. A change record linked to the approval, affected configuration items, deployment record, and verification result

C. A list of the source-code versions currently installed in production

D. A training attendance record for the personnel who implemented the change

ANSWER: B

Explanation:

A change record linked to the approval, affected configuration items, deployment record, and verification result is correct because it provides traceable evidence of each required step for the specific change. The auditor can evaluate whether the documented impact analysis was completed, whether proper authority approved the change, whether the approved configuration was deployed, and whether the result was verified after implementation.

A manager's statement is testimonial evidence and is weaker than records that can be independently examined. A list of current source-code versions may help establish the current configuration but does not demonstrate approval or post-implementation verification. A training attendance record shows that personnel received instruction, not that the procedure was followed for the sampled change.

QUESTION NO: 10

An internal auditor is assigned to audit the software release process. The auditor wrote the current release procedure and is responsible for approving exceptions to it. Which of the following actions is most appropriate?

A. Assign an auditor who is independent of the release process being audited

B. Retain the auditor after documenting the relationship in the audit plan

C. Limit the audit to release records prepared by personnel other than the auditor

D. Allow the release-process owner to approve the auditor’s findings before issue

ANSWER: A

Explanation:

Assign an auditor who is independent of the release process being audited is most appropriate. Audit conclusions must be objective and impartial. Because the assigned auditor designed the procedure and approves exceptions, the auditor would be evaluating work for which the auditor has direct responsibility. This creates an independence concern and can undermine confidence in the audit findings.

Having the auditor disclose the relationship does not remove the conflict. Limiting the audit to records prepared by other employees does not resolve the auditor’s responsibility for the procedure and its exceptions. Allowing the process owner to approve the findings would further compromise audit objectivity.

QUESTION NO: 11

Which of the following baselines defines the list of prioritized, known stories at the beginning of each iteration?

A. Release

B. Allocated

C. Developmental

D. Product backlog

ANSWER: D

Explanation:

Product backlog is correct because it is the ordered, evolving source of known work for a product, commonly expressed as user stories. At the start of an iteration, the team uses the prioritized Product Backlog to understand the available work and select items that best support the current product goal. Its ordering reflects factors such as value, risk, dependencies, and urgency, giving the team a clear basis for iteration planning.

The backlog is not a fixed requirements specification; it is continually refined as more is learned about customer needs, solution details, and delivery constraints. However, at any given iteration boundary, it provides the known and prioritized set of candidate stories from which the team plans. The Scrum Guide defines the Product Backlog as an emergent, ordered list of what is needed to improve the product and identifies it as the single source of work undertaken by the Scrum Team. See the Scrum Guide. The Agile Alliance also describes a product backlog as a prioritized list of work for the development team, supporting its use as the planning baseline for iterative delivery: Agile Alliance Product Backlog glossary.

QUESTION NO: 12

Which of the following items are examined regularly at a program review meeting?

A. Project dependencies

B. Change request approvals

C. Quality system updates

D. Problem reports

ANSWER: A

Explanation:

Project dependencies are examined regularly in a program review because a program coordinates related projects whose combined management produces benefits that could not be achieved by managing each project independently. Dependencies reveal where one project's deliverable, decision, resource, interface, schedule milestone, or technical outcome constrains another project. Reviewing them at the program level enables the program manager and governance stakeholders to evaluate cross-project effects, prioritize shared resources, resolve ownership and sequencing issues, assess program-level risks, and make timely decisions to protect planned benefits and strategic objectives.

Regular attention is particularly important because a change or delay in a dependent project can propagate through the program and affect integrated releases, benefit realization, funding, and overall program timing. Dependency information therefore provides the program-level visibility needed to coordinate work across component projects rather than treating their status as isolated results. This aligns with the Project Management Institute's description of program management as coordinated management of related projects and other work to obtain benefits not available when managed separately. See PMI's The Standard for Program Management and its overview of project, program, and portfolio management.

QUESTION NO: 13

When performing top-down testing, which of the following tools is used in place of a module that is not yet developed?

A. Stubs

B. Driver

C. Hamess

D. Test script

ANSWER: A

Explanation:

Stubs are used during top-down integration testing to stand in for subordinate, lower-level modules that have not yet been developed, completed, or integrated. Top-down integration begins with the highest-level control module and progressively adds the modules it calls. Because the higher-level module may depend on services from lower-level components that are unavailable at that point, a stub provides a controlled temporary implementation of the called component's interface. It can return predefined values, simulate expected outputs, record calls, or provide only enough behavior to let the calling module's control flow and interfaces be tested.

This approach enables the team to test high-level logic, navigation, and interactions early, while incrementally replacing each stub with the actual module as it becomes available. The temporary nature of a stub is important: it is not intended to reproduce the full production behavior, but to support meaningful integration tests before the real dependency exists. This terminology is widely used in software-testing practice and aligns with the distinction between stubs for called dependencies and test drivers for calling lower-level units. See the ISTQB glossary definition of "stub" and the IBM overview of integration testing.

QUESTION NO: 14

Which of the following methods is an example of dynamic analysis?

A. Piloting

B. Peer reviews

C. Quality gates

D. Mathematical proofs

ANSWER: A

Explanation:

Piloting is an example of dynamic analysis because it involves operating the software in a realistic, limited-use environment and observing its actual behavior. Dynamic analysis obtains evidence by executing a system, component, or code and evaluating runtime results, such as functional behavior, performance, reliability, usability, error handling, and interactions with users or external systems. A pilot commonly deploys a near-production version to a controlled group of users or sites before broader release. This provides practical execution-based evidence about whether the solution performs as intended under representative conditions and workflows. Findings from the pilot can then guide corrective actions, readiness decisions, and release planning. This aligns with the distinction used in software testing practice: dynamic testing requires execution of the test item, whereas static approaches examine work products without running the software. The ISTQB glossary defines dynamic testing as testing that involves execution of the test item; see the ISTQB Glossary entry for dynamic testing. For further context on testing activities and evaluating software through execution, see ISO/IEC/IEEE 29119-1 software testing concepts.

QUESTION NO: 15

Which of the following statements is true about component reuse?

A. The more a developer knows about the implementation, the more reusable a component will be.

B. Information-hiding impedes the development of reusable components.

C. Development costs are improved by reusable components.

D. Modifying a previously used module is an example of module reuse.

ANSWER: C

Explanation:

Development costs are improved by reusable components. A component that has already been designed, implemented, tested, documented, and used in an appropriate context can reduce the effort needed to deliver later systems. Instead of creating equivalent functionality from scratch, a team can select and integrate a proven component, concentrating its engineering effort on requirements, interfaces, integration, and risk-specific verification. Reuse can also lower the cost of quality because prior use may have exposed defects and strengthened the component's test assets, operational knowledge, and documentation. These benefits are not automatic: effective reuse depends on suitable architecture, clear interfaces, configuration control, adequate documentation, licensing awareness, and validation that the component satisfies the new system's requirements. There can be upfront investment to create reusable assets and a repository or governance process, but the economic benefit is realized as those assets are used across products or projects. This relationship between reusable software assets, productivity, and lifecycle quality is consistent with the software engineering practices covered in the IEEE Computer Society Guide to the Software Engineering Body of Knowledge and with the reuse-related quality-engineering knowledge expected for ASQ's Certified Software Quality Engineer certification.

QUESTION NO: 16

Which of the following terms refers to a set of test procedures and cases that are executed on software code?

A. Static testing

B. Dynamic testing

C. Code walkthrough

D. Peer review

ANSWER: B

Explanation:

Dynamic testing is the correct term because it assesses software by running the executable code with defined test cases, data, and procedures. The test team supplies inputs, observes the system's actual behavior and outputs, and compares the observed results with expected results and specified requirements. This execution-based approach can reveal failures in functional behavior, interfaces, data processing, timing, resource use, security behavior, and error handling. Dynamic testing may occur at unit, integration, system, or acceptance test levels, and it can be performed manually or through automated test execution.

The defining characteristic is that the software is operated during the test. A set of procedures and cases provides the repeatable instructions for exercising the code, documenting expected outcomes, and recording results. This supports objective evaluation of whether the implemented product performs as required in its runtime environment. The International Software Testing Qualifications Board describes dynamic testing as testing that requires execution of the software under test, while its glossary distinguishes it from static testing activities. See the ISTQB definition of dynamic testing and the ISO/IEC/IEEE 29119 software testing standard overview.

QUESTION NO: 17

A software engineer is asked to analyze the relationship between the number of defects found and the complexity of various software applications. Which of the following is the appropriate tool to use for this analysis?

A. Cause and effect diagram

B. Control chains

C. Histogram

D. Scatter diagram

ANSWER: D

Explanation:

Scatter diagram is the appropriate tool because the analysis concerns the possible relationship between two quantitative variables: software complexity and the number of defects found. Each application can be represented as a single plotted point, with its complexity measurement on one axis and its defect count on the other. Viewing all applications together makes it possible to identify an overall association, such as a tendency for applications with greater complexity to have more reported defects. The plot can also reveal the direction, strength, and form of the relationship, as well as unusual applications that merit further investigation. This is a foundational quality-analysis technique: the ASQ describes a scatter diagram as a graph used to investigate relationships between two variables. It supports analysis and hypothesis generation, but the observed pattern should not by itself be treated as proof that complexity causes defects. The team should ensure that complexity is measured consistently across the applications and interpret defect counts in context, including comparable test effort, application size, and operational exposure. See ASQ's Scatter Diagram resource and the Seven Basic Quality Tools overview.

QUESTION NO: 18

One advantage of outsourcing is that it allows the primary company to

A. focus resources on its core competencies

B. redirect its resources on new product development

C. reduce the cost of ongoing training

D. eliminate the need for a skills-based workforce

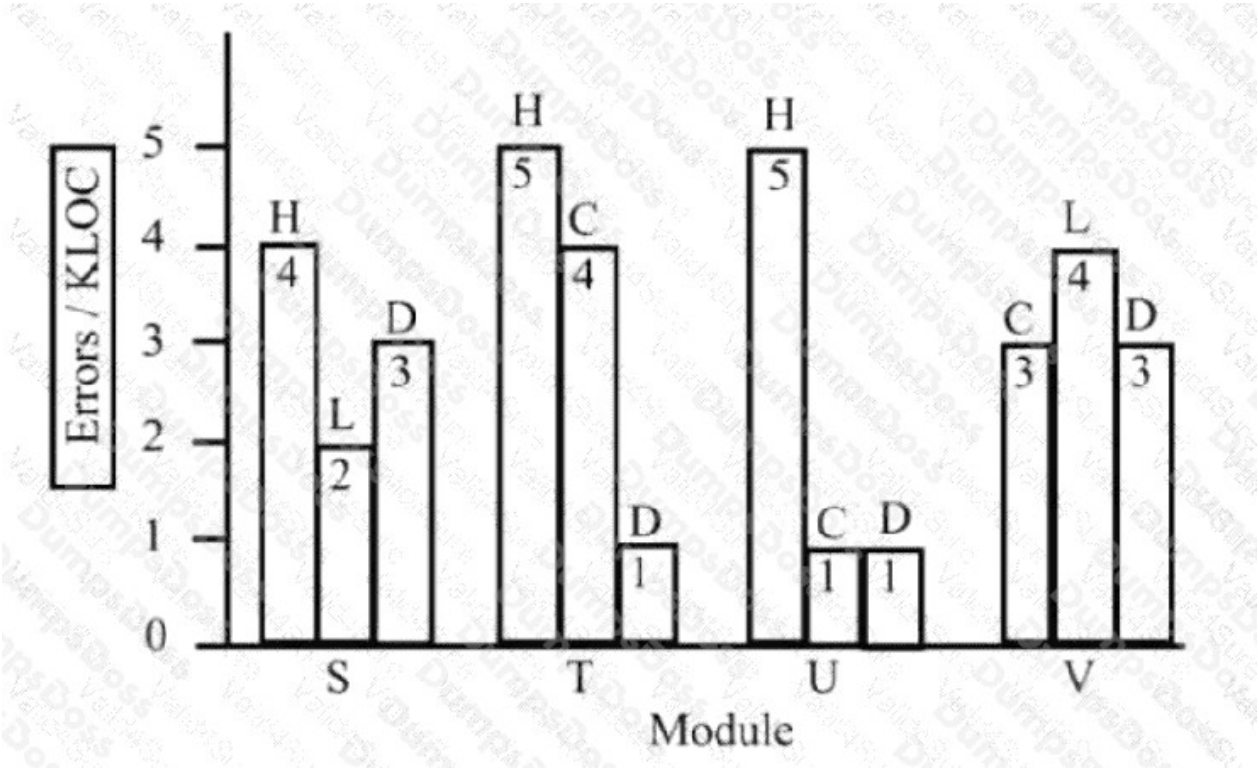
ANSWER: A

Explanation:

focus resources on its core competencies is correct. Outsourcing assigns selected activities, services, or processes to an external provider that has the needed specialized capability. This enables the primary organization to devote more of its internal people, management attention, budget, and technical capacity to the activities that most directly create customer value and distinguish the organization in its market. In a software-quality context, a company may obtain external support for specialized or noncore functions while retaining accountability for the product and concentrating internal expertise on strategic engineering, quality planning, customer requirements, and business-critical delivery. The benefit is not merely shifting work elsewhere; it is the deliberate alignment of organizational resources with the company's central mission and distinctive strengths. Effective outsourcing therefore requires clear requirements, supplier evaluation, defined service expectations, and ongoing oversight, but its strategic rationale is allowing the organization to concentrate on what it does best. This aligns with the ASQ Certified Software Quality Engineer body of knowledge, which includes supplier management and quality-system responsibilities: ASQ Certified Software Quality Engineer. The strategic role of concentrating on core competencies is also consistent with the U.S. Small Business Administration's guidance on using outsourcing to obtain expertise and free internal capacity: U.S. Small Business Administration.

QUESTION NO: 19

The following graph depicts errors per 1,000 lines of code (KLOC) for modules S, T, U, and V.



H: Header Comments Incomplete
C: Commented Out Code
L: Logic Errors
D: Data Initialization Errors

The program manager should be most concerned about which module?

A. S

B. T

C. U

D. V

ANSWER: C

Explanation:

U is the appropriate module of greatest concern because the graph shows that it has the largest overall error density, measured as errors per thousand lines of code. Error density normalizes the observed errors by module size, enabling a manager to compare the relative quality of modules without allowing a larger codebase alone to make its error count appear disproportionately high. The elevated combined level for U, including its substantial contributions across the displayed error categories, signals a concentration of defects that merits prompt investigation.

For a program manager, this finding supports risk-based prioritization: assign additional analysis, testing, defect root-cause investigation, and corrective action to the area with the highest defect density. Doing so addresses the component with the clearest evidence of elevated quality risk and helps focus limited project resources where they are most likely to improve reliability and reduce rework. Defect density is a standard software-quality measurement used to monitor product quality and to identify areas requiring improvement. ASQ includes software quality engineering measures and metrics within the scope

of the Certified Software Quality Engineer body of knowledge; see ASQ Certified Software Quality Engineer. For practical guidance on using defect metrics in software measurement, see the NIST Software Quality Group.

QUESTION NO: 20

When the number of failures found during acceptance testing is compared to the number of failures found after release, the result is a measure of

A. test effectiveness

B. test coverage

C. product maintainability

D. measurement efficiency

ANSWER: A

Explanation:

test effectiveness is correct because comparing failures detected during acceptance testing with failures that escape into production indicates how successfully the testing activity identifies failures before customers use the product. This is commonly expressed through defect detection effectiveness (DDE), which uses defects found before release relative to the total defects found both before and after release. A higher result means acceptance testing contained a greater proportion of the defects ultimately discovered, providing evidence that the test process, test design, environment, data, and acceptance criteria are effective at detecting relevant product failures.

This comparison is especially useful for software-quality improvement because post-release failures represent escaped defects: issues not identified by the release decision point. Teams can trend the measure across releases, investigate changes in effectiveness, and use the findings to improve risk-based testing and defect-prevention activities. The measure concerns the outcome of testing—its ability to find failures before release—rather than merely the amount of code, requirements, or scenarios exercised. ASQ identifies defect containment and removal effectiveness as meaningful quality measures, while ISO/IEC/IEEE 29119 defines a framework for managing and measuring software testing activities. See ASQ's software quality resources and ISO/IEC/IEEE 29119-1 software testing overview.

QUESTION NO: 21

What type of questions should developers ask customers when discussing requirements at the inception of the project?

A. context-free

B. detail-driven

C. multiple-choice

D. yes/no

ANSWER: A

Explanation:

“

context-free

” is correct because initial requirements conversations should encourage customers and other stakeholders to describe their work, objectives, problems, constraints, and measures of success without being steered toward a presumed solution. Context-free questions are deliberately broad questions that do not depend on detailed prior knowledge of the application domain or embed assumptions about what the system should do. They help the development team establish a shared understanding of the business situation before discussing features, interfaces, or design choices.

At project inception, this approach supports effective elicitation by revealing stakeholder goals, terminology, operational context, pain points, decision criteria, and important external influences. Questions such as “What problem are you trying to solve?”, “What outcomes would make this project successful?”, and “Who is affected by this process?” allow customers to frame needs in their own language. The information gained becomes a foundation for later, more focused requirements analysis, clarification, prioritization, and validation. This technique is associated with early requirements exploration practices described in Gause and Weinberg’s *Exploring Requirements: Quality Before Design*; see its bibliographic record at Google Books. It also aligns with the Business Analysis Body of Knowledge’s emphasis on eliciting information from stakeholders to understand needs and recommend appropriate solutions: IIBA BABOK Guide.

QUESTION NO: 22

Which of the following corrective actions is the first action to take when a project experiences a significant deviation on baselined budgets, baselined schedules, or required quality levels?

A. Add resources and continue with the plan.

B. Cancel the project because of the deviation.

C. Create a new baseline and continue with the plan.

D. Realign project actuals and continue with the plan.

ANSWER: D

Explanation:

Realign project actuals and continue with the plan. is the appropriate initial corrective-action direction because an approved baseline is the authorized reference for measuring project cost, schedule, and quality performance. A significant variance signals that actual project execution is no longer delivering the planned result. The project team should therefore bring execution back into alignment with the approved plan by analyzing the variance, identifying its cause and impact, and implementing suitable corrective work. This may include adjusting work sequencing, resolving defects, improving process capability, or managing approved changes, while continuing to track performance against the baseline.

Corrective action is intended to address the cause of an observed deviation and restore expected performance, rather than merely record the variance. Effective control also requires verifying that the action produces the intended result and communicating material status and decisions to relevant stakeholders. If analysis establishes that the authorized plan itself must change, that change should be evaluated and approved through the project’s change-control process; the baseline remains the control reference until such approval occurs. This approach is consistent with PMI’s description of monitoring and controlling work through performance measurement and corrective action, and with ISO guidance on corrective action for nonconformities. See PMI’s project monitoring and controlling guidance and ISO 9001 clause 10.2.

QUESTION NO: 23

In which of the following ways do metadata requirements guarantee everyone is interpreting data consistently?

A. It ensures that everyone who has access to the data has a consistent understanding of its meaning and use.

B. It ensures that data is managed, maintained, and used appropriately through its lifecycle by creating traceability

C. it ensures that the data structure remains consistent by making certain that data are correctly classified and stored.

D. It ensures that quality standards are met by helping organizations better assess potential risks related to data security.

ANSWER: A

Explanation:

It ensures that everyone who has access to the data has a consistent understanding of its meaning and use. Metadata is structured information that describes data, such as a data element’s business definition, permitted values, format, source, owner, calculation rules, and intended purpose. Establishing requirements for this information creates a shared, governed

vocabulary for data across users, teams, applications, and reports. For example, a documented definition for a metric makes clear what is included, how it is calculated, and the context in which it is valid. This reduces the risk that different stakeholders will apply the same label to different concepts or draw inconsistent conclusions from the same data set. In software quality and data governance practice, metadata therefore supports semantic consistency, reliable communication, traceability of meaning, and correct use of data throughout its lifecycle. The U.S. National Institute of Standards and Technology describes metadata as structured information about data that can support understanding and management, while ISO's metadata registry standard provides principles for specifying and registering data-element definitions. See NIST's metadata definition and ISO/IEC 11179-3 metadata registry principles.

QUESTION NO: 24

Which of the following is most often the focus of testing in an organization with a mature testing strategy?

A. Complete testing to identify product defects

B. Functional testing to prove a product is correct

C. Testing the products throughout the development cycle

D. Extensive strategic beta testing with a selected group of customers

ANSWER: C

Explanation:

Testing the products throughout the development cycle is correct because a mature testing strategy treats quality assurance and testing as continuous, lifecycle-wide activities rather than as a final phase before release. Testing begins early, with activities such as reviewing requirements for clarity, consistency, testability, and risk. It continues through architectural and design reviews, component-level testing, integration testing, system-level evaluation, acceptance activities, deployment, and maintenance. This approach supports early feedback and defect prevention, when correcting problems is generally less costly and disruptive. A mature organization also aligns test planning and test levels with project risks, establishes traceability between requirements and tests, measures results to improve the process, and uses automation selectively to provide rapid, repeatable feedback. The essential focus is therefore verifying and improving the product continuously as it evolves, while providing stakeholders with timely information about quality and risk. This lifecycle perspective is consistent with software-testing best practices described in the ISO/IEC/IEEE 29119 software testing standard and the quality-management principle that process activities should be planned and controlled throughout product realization, as outlined by ASQ's quality management system resources.

QUESTION NO: 25

The following chart was developed to compare defects in two projects. Which of the following conclusions can be drawn from this chart?

A. The projects have the same defect rates.

B. The projects have different defect profiles.

C. More defects were removed per phase from Project 1 than Project 2.

D. More defects were removed per phase from Project 2 than Project 1.

ANSWER: B

Explanation:

The projects have different defect profiles. A defect profile is the pattern in which reported defects are distributed across meaningful categories, such as defect type, origin, severity, component, or cause. Because the chart compares defect counts for the two projects by defect category, it provides a direct visual basis for comparing those distributions. When the relative counts across categories differ between projects, the projects have different defect profiles. This conclusion is useful to a software quality engineer because the profile can indicate that the projects may have differing technical risks, process

weaknesses, review effectiveness, or test-focus needs. Defect classification and analysis are central quality practices: categorizing defects consistently makes trends visible and supports targeted process-improvement actions rather than treating all defects as an undifferentiated total. ASQ describes a defect as a product or service characteristic that does not meet requirements, and its quality resources emphasize measurement and analysis for improvement. See ASQ's definition of defect and ASQ's data collection and analysis resources. The chart therefore supports a conclusion about the comparative distribution of defect categories—namely, different defect profiles.

QUESTION NO: 26

What type of information should a project ' s configuration status accounting communicate?

A. Budget updates for the project

B. Changes to the project plan

C. Activity on baselined items

D. Change control board minutes

ANSWER: C

Explanation:

Configuration status accounting should communicate **activity on baselined items**. In configuration management, a baseline is an approved, formally controlled version of a configuration item, such as a requirements specification, design document, source-code build, test asset, or release. Once established, the baseline is not altered informally; proposed modifications are identified, evaluated, approved or rejected through change control, implemented when authorized, and tracked to closure.

Status accounting provides the recorded and reportable evidence of this control. It identifies the current approved version and state of each configuration item and communicates the status of requested and authorized changes, including what change activity has occurred and whether implementation has been completed. Project stakeholders can therefore determine whether the delivered or working product matches its approved baseline and can trace its evolution. This reporting function is a core configuration-management discipline, alongside configuration identification, change control, and configuration verification/audit. NASA's configuration-management guidance describes status accounting as recording and reporting configuration-item and change-request status, while the SEI glossary likewise defines it as recording and reporting the status of configuration items and change requests. See NASA Configuration Management and the SEI CMMI for Development, Version 1.3.

QUESTION NO: 27

Which of the following terms is used to describe the measure of interdependence among modules in a computer program?

A. Coherence

B. Concurrency

C. Cohesion

D. Coupling

ANSWER: D

Explanation:

Coupling is the software-design term for the degree of interdependence between modules. It characterizes how much one module relies on another module's data, services, control information, or implementation details to perform its function. A system with low coupling limits these dependencies through well-defined interfaces, so a change to one module is less likely to require changes to others or introduce unintended side effects. This separation supports key software-quality objectives, including maintainability, testability, understandability, and adaptability. For example, a module that uses a stable interface offered by another component is less coupled than one that directly accesses the other component's internal data structures.

Software engineering design guidance generally seeks low coupling while maintaining appropriate functional focus within each module. The ISO/IEC/IEEE 24765 systems and software engineering vocabulary defines coupling in terms of the strength of relationships between modules, and established design guidance treats minimizing unnecessary module dependencies as a central way to control complexity. See the ISO/IEC 25010 product quality model for the maintainability context and the Software Architecture in Practice overview from SEI for architecture and modular-dependency principles.

QUESTION NO: 28

Conflict in teams can be useful when it

A. establishes decision-making patterns

B. speeds decision-making

C. produces new information

D. focuses management s attention on the team

ANSWER: C

Explanation:

Conflict in teams can be useful when it **produces new information**. Constructive task-related conflict brings differing facts, assumptions, interpretations, and alternatives into the team's discussion. This expanded information base helps team members test ideas, identify risks or overlooked constraints, and make choices using a more complete understanding of the problem. In a software-quality context, such discussion can expose ambiguous requirements, competing stakeholder needs, process weaknesses, or unexamined quality risks before they become costly defects. The benefit arises when the conflict concerns the work rather than personal attacks, and when the team uses respectful communication and evidence to evaluate the new perspectives. The Project Management Institute describes conflict as potentially beneficial when it results in improved decision-making and solutions, particularly when managed constructively. Similarly, research summarized by the American Psychological Association recognizes that task conflict can support better team outcomes under appropriate conditions. A software quality engineer should therefore facilitate an environment in which relevant disagreements can surface meaningful evidence and perspectives, enabling informed analysis and continual improvement. References: Project Management Institute: Conflict Resolution Techniques; American Psychological Association: The Benefits of Conflict.

QUESTION NO: 29

During an audit, the lead auditorâ€™s primary responsibility is to

A. decide which procedures are to be audited

B. collect data and materials presented during the audit

C. keep the audit team focused on the audit scope

D. develop and distribute the audit checklist

ANSWER: C

Explanation:

The correct answer is *keep the audit team focused on the audit scope*. The lead auditor directs and manages the audit activities, ensuring that the team carries out the audit plan effectively, uses audit time appropriately, and pursues evidence relevant to the defined objectives and scope. Maintaining this focus is essential because an audit must remain a systematic, independent, evidence-based evaluation of the agreed audit criteria rather than expand into unrelated operational issues. The lead auditor also coordinates auditors, manages communication with the auditee, resolves issues arising during the audit, and is responsible for reporting audit results. These leadership responsibilities all support completion of an orderly audit that remains aligned with its authorized boundaries. ISO 19011 describes the audit team leader's responsibilities as including managing audit-team members and ensuring the audit is conducted according to the audit plan; this necessarily includes keeping activities directed toward the established scope. ASQ's auditing body of knowledge likewise emphasizes

managing audit teams and audit activities in accordance with audit objectives, scope, and plans. See ISO 19011:2018—Guidelines for auditing management systems and ASQ Certified Software Quality Engineer.