Certified Software Quality Engineer (CSQE)
Body of Knowledge

The topics in this Body of Knowledge include additional detail in the form of subtext explanations and the cognitive level at which the questions will be written. This information will provide useful guidance for the Examination Development Committee and the candidates preparing to take the exam. The subtext is not intended to limit the subject matter or be all-inclusive of what might be covered in an exam. It is intended to clarify the type of content to be included in the exam. The descriptor in parentheses at the end of each entry refers to the highest cognitive level at which the topic will be tested. A more comprehensive description of cognitive levels is provided at the end of this document.

I. General Knowledge (16 questions)

A. Benefits of software quality engineering within the organization

Describe the benefits that software quality engineering can have at the organizational level. (Understand)

B. Ethical and Legal Compliance

1. ASQ Code of Ethics for professional conduct

Determine appropriate behavior in situations requiring ethical decisions such as identifying conflicts of interest and recognizing and resolving ethical issues. (Evaluate)

2. Regulatory and legal issues

Describe the importance of compliance to federal, national, international, and statutory regulations on software development (e.g., EU GDPR and China's Data Protection Laws). Determine the impact of issues such as copyright, trademark, intellectual property rights, product liability, and data privacy [e.g., Health Insurance Portability and Accountability Act (HIPPA) and US Data Protection Laws for personal identifying information (PII)]. (Understand)

C. Standards and models

Define and describe the ISO standards such as 9000 and 27000, NIST Special Publication 800-53, IEEE software standards, and the Capability Maturity Model Integration (CMMI) assessment models. (Understand)

D. Leadership skills

1. Organizational leadership

Use leadership tools and techniques (e.g. organizational change management, knowledge transfer, motivation, mentoring and coaching, and recognition). (Apply)

2. Facilitation skills

Use facilitation and conflict resolution skills and negotiation techniques to manage and resolve issues. Use meeting management tools to maximize meeting effectiveness. (Apply)

3. Communication skills

Identify and apply various communication methods in oral, written, and presentation formats. Apply various techniques for working in multi-cultural environments. Identify and describe the impact that culture, communications, and Diversity, Equity, and Inclusion (DEI) can have on an organizations. (Apply)

E. Team management

Use various team management skills including assigning roles and responsibilities, identifying the classic stages of team development (forming, storming, norming, performing, and adjourning), and monitoring and responding to group dynamics. Work with diverse groups and in distributed work environments. Use techniques for working with virtual, in-person, and hybrid teams. (Apply)

II. Software Quality Management (27 questions)

A. Quality management system

1. Quality goals and objectives

Design software quality goals and objectives that are consistent with business objectives and regulations. Incorporate software quality goals and objectives into high level program and project plans. Develop and use documents and processes necessary to support software quality management systems. (Create)

2. Quality metrics and monitoring

Determine software quality goals and objectives for measuring and monitoring software including software product and process metrics, and analysis and reporting techniques. (Analyze)

3. Customers and other stakeholders

Assess and evaluate the effect of various stakeholder group requirements (e.g., quality requirements for contractual obligations) on software projects and products. (Evaluate)

4. Outsourcing

Determine the impact that outsourced services can have on organizational goals and objectives. Identify criteria for evaluating suppliers/vendors and subcontractors including governance and quality requirements for contractual obligations. (Analyze)

5. Business continuity, data protection, and data management

Design plans for business continuity, disaster recovery, business documentation, change management, information security, and protection of sensitive and personal data. (Analyze)

B. Methodologies

1. Cost of Quality (COQ) and Return on Investment (ROI)

Analyze cost of quality (COQ) categories (e.g., prevention, appraisal, internal failure, and external failure) and return on investment (ROI) metrics in relation to products and processes. (Analyze)

2. Process improvement

Define and describe elements of benchmarking, lean processes, and the six sigma methodology. Use the Define, Measure, Act, Improve, Control (DMAIC) model and the Plan-Do-Check-Act (PDCA) model for process improvement. (Analyze)

3. Corrective and preventive action procedures

Evaluate corrective and preventive action procedures and impact assessments related to software defects, process nonconformances, and other quality system deficiencies. (Evaluate)

4. Defect prevention

Design and use defect prevention processes such as technical reviews, software tools and technology, and special training. (Evaluate)

5. Metrics and monitoring

Apply metrics and monitoring techniques for adherence to software quality goals and objectives, process improvement, and defect prevention. (Apply)

C. Audits

1. Audit types

Define and distinguish between various audit types (e.g., process, compliance, supplier, system, internal, and external) and methods (e.g., virtual). (Understand)

2. Audit roles and responsibilities

Identify roles and responsibilities for audit participants including the client, lead auditor, audit team members, and auditee. (Understand)

3. Audit process

Define and describe the steps in conducting an audit, developing and delivering an audit report, and determining appropriate follow-up activities. (Apply)

III. System and Software Engineering Processes (32 questions)

A. Lifecycles and process models

1. Waterfall software development lifecycle

Apply the waterfall lifecycle and related process models and identify their benefits and when they are used. (Apply)

2. Incremental / iterative software development lifecycles

Apply the incremental and iterative lifecycles and related process models (e.g., rapid application development methodology) and identify their benefits and when they are used. (Apply)

3. Agile software development lifecycle

Apply the agile lifecycle and related process models (e.g., lean agile and SCRUM methodology) and identify their benefits and when they are used. (Apply)

4. DevOps

Employ DevOps to combine software development and IT operations to continuously release software. (Apply)

B. Systems architecture

Identify and describe various architectures (e.g., embedded systems, client-server, n-tier, web, wireless, messaging, and collaboration platforms) and analyze their impact on quality. (Analyze)

C. Cloud computing models and platforms

Describe cloud computing models such as Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaS) and cloud computing platforms such as multi-tenant systems and distributed systems. (Understand)

D. Requirements engineering

1. Product requirements

Define and describe various types of product and user requirements including but not limited to system, feature, function, non-functional, interface, integration, performance, globalization, and localization. (Understand)

2. Data / information requirements

Define and describe various types of data and information requirements including data management, data integrity, and privacy by design. (Understand)

3. Quality requirements

Define and describe various types of quality requirements, including reliability, usability. (Understand)

4. Compliance requirements

Define and describe various types of regulatory, safety, and data retention requirements. (Understand)

5. Security requirements

Define and describe various types of security requirements including data security, information security, cybersecurity, data privacy, and encryption requirements for data in transit and at rest. (Understand)

6. Derived requirements

Apply derived requirements such as environmental requirements and user-defined security. (Apply)

7. Non-functional requirements

Define and describe various types of non-functional requirements such as documentation, performance, and technical maintenance**.** (Understand)

8. Requirements elicitation methods

Describe and use various requirements elicitation methods such as customer needs analysis, use cases, human factors studies, usability prototypes, Joint Application Development (JAD), and storyboards. (Apply)

9. Requirements evaluation

Assess the completeness, consistency, correctness, and testability of requirements and determine their priority. (Evaluate)

E. Requirements management

1. Requirements change management

Assess the impact that changes to requirements will have on software development processes for all types of life-cycle models. (Evaluate)

2. Bidirectional traceability

Use various tools and techniques to ensure bidirectional traceability from requirements elicitation and analysis through design and testing. (Apply)

F. Software analysis, design, and development

1. Design methods

Identify the steps used in software design and their functions. Define and distinguish between software design methods. (Understand)

2. Quality attributes and design

Analyze the impact that quality-related elements (e.g., safety, security, reliability, usability, reusability, and maintainability) can have on software design. (Analyze)

3. Software reuse

Define and distinguish between software reuse, reengineering, and reverse engineering and describe the impact these practices can have on software quality. (Understand)

4. Software development tools

Analyze and select the appropriate development tools for modeling, code analysis, requirements management, and documentation. (Analyze)

G. Maintenance management

1. Maintenance types

Describe the characteristics of corrective, adaptive, perfective, and preventive maintenance types. (Understand)

2. Maintenance strategy

Describe various factors affecting the strategy for software maintenance including service-level agreements (SLAs), short- and long-term costs, maintenance

releases, and product discontinuance and their impact on software quality. (Understand)

3. Customer feedback management

Describe the importance of customer feedback management including quality of product support and post-delivery issues analysis and resolution. (Understand)

IV. Project Management (16 questions)

A. Planning, scheduling, and deployment

1. Project planning

Use product acceptance, configurating acceptance, forecasts, resources, schedules, task, cost estimates, etc. to develop project plans. (Apply)

2. Work breakdown structure (WBS)

Use Work Breakdown Structure (WBS) in scheduling and monitoring projects. (Apply)

3. Project deployment

Use various tools including milestones, objectives achieved, and task duration to set goals and deploy projects. (Apply)

B. Tracking and controlling

1. Phase transition control

Use various tools and techniques such as entry/exit criteria, quality gates, Gantt charts, and integrated master schedules to control phase transitions. (Apply)

2. Tracking methods

Calculate project-related costs such as earned value, deliverables, and productivity and track the results against project baselines. (Apply)

3. Project reviews

Use various types of project reviews such as phase-end, management, and retrospectives and post-project reviews to assess project performance and status, to review issues and risks, and to discover and capture lessons learned from the project. (Apply)

4. Program reviews

Define and describe various methods for reviewing and assessing programs in terms of performance, technical accomplishments, and resource utilization. (Understand)

C. Risk management

1. Risk management methods

Use risk management techniques (e.g. risk impact assessment, prevention, mitigation, and transfer) to evaluate project risks. (Evaluate)

2. Software security risks

Evaluate risks specific to software security including deliberate attacks (e.g., hacking and sabotage), inherent defects that allow unauthorized access to data, and other security breaches. Plan appropriate responses to minimize their impact. Use threat analysis and modeling tools to identify and prevent potential security risks. (Evaluate)

3. Safety and hazard analysis

Evaluate safety risks and hazards related to software development and implementation and determine appropriate steps to minimize their impact. (Evaluate)

V. Software Metrics and Analysis (21 questions)

A. Process and product measurement

1. Terminology

Define and describe metric and measurement terms such as reliability, internal and external validity, explicit and derived measures, and variation. (Understand)

2. Software product metrics

Choose appropriate metrics to assess various software attributes (e.g., size, complexity, the amount of test coverage needed, requirements volatility, and overall system performance). (Apply)

3. Software process metrics

Measure the effectiveness and efficiency of software processes (e.g., Functional Verification Tests (FVT), cost, yield, customer impact, defect detection, defect containment, Total Defect Containment Effectiveness (TDCE), Defect Removal Efficiency (DRE), and process capability). (Apply)

4. Data integrity

Describe the importance of data integrity from planning through collection and analysis. Apply various techniques to ensure data quality, accuracy, completeness, and timeliness. (Apply)

B. Analysis and Reporting Techniques

1. Metric reporting tools

Using various metric representation tools such as dashboards and stoplight charts to report results. (Apply)

2. Classic quality tools

Describe the appropriate use of classic quality tools (e.g., flowcharts, Pareto charts, cause and effect diagrams, control charts, and histograms). (Apply)

3. Problem-solving tools

Describe the appropriate use of problem-solving tools (e.g., affinity, tree,, matrix, activity network, root cause analysis, and Data Flow Diagrams (DFD)). (Apply)

VI. Software Verification and Validation (32 questions)

A. Theory

1. V&V methods

Use software verification and validation methods (e.g., static analysis, structural analysis, mathematical proof, simulation, and automation) and determine which tasks should be iterated because of modifications. (Apply)

2. Software product evaluation

Use various evaluation methods on documentation, source code, etc. to determine whether user needs and project objectives have been satisfied. (Analyze)

B. Test planning and design

1. Test Strategies

Select and analyze test strategies (e.g., test-driven design, good-enough, risk-based, time-box, top-down, bottom-up, black-box, white-box, simulation, automation, and continuous) for various situations such as Software as a Service (SaaS) and new product applications. (Analyze)

2. Test plans

Develop and evaluate test plans and procedures such as system, acceptance, and validation to determine whether project objectives are being met and risks are appropriately mitigated. (Create)

3. Test designs

Select and evaluate various test designs, including fault insertion, fault-error handling, equivalence class partitioning, boundary value. (Evaluate)

4. Software tests

Evaluate and execute various tests including unit, functional, performance, integration, regression, usability, acceptance, certification, environmental load, stress, worst-case, perfective, exploratory, and system. (Evaluate)

5. Tests of external products

Determine appropriate levels of testing for integrating supplier, third-party, and subcontractor components and products. (Apply)

6. Test coverage specifications

Evaluate the adequacy of test specifications such as functions, states, data and time domains, interfaces, security, and configurations that include internationalization and platform variances. (Evaluate)

7. Code coverage techniques

Use and identify various tools and techniques to facilitate code coverage analysis techniques such as branch coverage, condition, domain, and boundary. (Apply)

8. Test environments

Select and use simulations, test libraries, drivers, stubs, harnesses, etc. and identify parameters to establish a controlled test environment. (Analyze)

9. Test tools

Identify and use test utilities, diagnostics, automation, and test management tools. (Apply)

10. Test data management

Ensure the integrity and security of test data through the use of configuration controls. (Apply)

C. Reviews and inspections

Use desk-checks, peer reviews, walk-throughs, inspections, etc. to identify defects. (Apply)

D. Test execution documents

Evaluate and manage test execution documents such as test results, defect reporting and tracking records, test completion metrics, trouble reports, and input/output specifications. (Evaluate)

VII. Software Configuration Management (16 questions)

A. Configuration infrastructure

1. Configuration management team

Describe the roles and responsibilities of a configuration management group. (Understand) [NOTE: The roles and responsibilities of the configuration control board (CCB) are covered in area VII.C.2.]

2. Configuration management tools

Describe configuration management tools as they are used for managing libraries, build systems, and defect tracking systems. (Understand)

3. Library processes

Describe dynamic, static, and controlled library processes and related procedures such as check-in/check-out and merge changes. (Understand)

B. Configuration identification

1. Configuration items

Describe software configuration items (e.g., baselines, documentation, software code, and equipment) and identification methods (e.g., naming conventions and versioning schemes). (Understand)

2. Software builds and baselines

Describe the relationship between software builds and baselines and describe methods for controlling builds and baselines (e.g., automation and new versions). (Understand)

C. Configuration control and status accounting

1. Item change and version control

Describe processes for documentation control, item change tracking, and version control that are used to manage various configurations. Describe processes used to manage configuration item dependencies in software builds and versioning. (Understand)

2. Configuration Control Board (CCB)

Describe the roles, responsibilities and processes of the Configuration Control Board (CCB). (Understand) [NOTE: The roles and responsibilities of the configuration management team are covered in area VII.A.1.]

3. Concurrent development

Describe the use of configuration management control principles in concurrent development processes. (Understand)

4. Status accounting

Apply various processes for establishing, maintaining, and reporting the status of configuration items such as baselines, builds, and tools. (Apply)

D. Configuration Audits

Define and distinguish between functional and physical configuration audits and how they are used in relation to product specification. (Understand)

E. Product release and distribution

1. Product release

Assess the effectiveness of product release processes: planning, scheduling, and defining hardware and software dependencies. (Evaluate)

2. Customer deliverables

Assess the completeness of customer deliverables including packaged, hosted, and downloadable products; license keys and user documentation; and marketing and training materials. (Evaluate)

3. Archival processes

Assess the effectiveness of source and release archival processes: backup planning and scheduling, data retrieval, archival of build environments, retention of historical records, and offsite storage. (Evaluate)

Levels of Cognition
Based on Bloom's Taxonomy – Revised (2001)

In addition to **content** specifics, the subtext for each topic in this BOK also indicates the intended **complexity level** of the test questions for that topic. These levels are based on "Levels of Cognition" (from Bloom's Taxonomy – Revised, 2001) and are presented below in rank order, from least complex to most complex.

Remember
Recall or recognize terms, definitions, facts, ideas, materials, patterns, sequences, methods, principles,

Understand
Read and understand descriptions, communications, reports, tables, diagrams, directions, regulations,

Apply
Know when and how to use ideas, procedures, methods, formulas, principles, theories,

Analyze
Break down information into its constituent parts and recognize their relationship to one another and how they are organized; identify sublevel factors or salient data from a complex scenario.

Evaluate
Make judgments about the value of proposed ideas, solutions, , by comparing the proposal to specific criteria or standards.

Create
Put parts or elements together in such a way as to reveal a pattern or structure not clearly there before; identify which data or information from a complex set is appropriate to examine further or from which supported conclusions can be drawn.