

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 both 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 complete description of cognitive levels is provided at the end of this document.

I. General Knowledge, Conduct, and Ethics (16 Questions)

A. Quality philosophy and principles

1. Benefits of software quality

Describe how software quality engineering can benefit an organization. (Comprehension)

2. Prevention vs. detection

Describe how quality engineering methodologies can reduce the length of time for testing and can influence other defect detection methods. (Comprehension)

3. Organizational and process benchmarking

Identify, analyze, and model best practices at the macro (organizational) and micro (process and project) levels. Identify and develop business objectives, use metrics to monitor their achievement, and provide feedback to close the process improvement loop. (Analysis)

B. Standards, specifications, and models

Identify and use software process and assessment models, including ISO 9001, ISO 15504, IEEE software standards, IEEE/EIA 12207, SEI Capability Maturity Model Integrated (CMMI), etc., in a variety of situations. (Application)

C. Leadership tools and skills

1. Organizational leadership

Define, describe, and apply leadership tools and techniques, including analyzing current situations, proposing, justifying, implementing, and managing change (using change-agent tools), developing and implementing quality initiatives, obtaining cross-functional commitment and collaboration, ensuring knowledge transfer, motivating personnel, etc. (Application)

2. Team management

Define and use various team management techniques, including identifying and assigning roles and responsibilities (e.g., champion, sponsor, facilitator, leader, coach), identifying and assessing team member skills, interpreting team dynamics and stages of team development, handling dominant or disruptive team members, recognizing how diversity in teams strengthens the creative process, etc. (Application)

3. Team tools

Define, describe, and use tools such as brainstorming, nominal group technique (NGT), joint application development (JAD), rapid application development (RAD), etc. (Application)

4. Facilitation Skills

Use various tools to manage and resolve conflict. Use negotiation techniques to produce win-win outcomes. Identify and use time and meeting management tools to maximize performance. (Application)

5. Communication skills

Define, describe, and apply various communication elements used in verbal, written, and presentation formats, including interviewing and listening skills. Apply communication elements to create effective process and procedural documents, including identifying roles and responsibilities. (Application)

D. Ethical conduct and professional development

1. ASQ Code of Ethics

Determine appropriate behavior in situations requiring ethical decisions, including identifying conflicts of interest and recognizing/resolving ethical issues related to software licensing and use. (Evaluation)

2. Software liability and safety issues

Identify legal issues related to software product liability and safety, including negligence, customer notification requirements, and other legal or regulatory issues. (Application) [NOTE: Other aspects of product safety and hazard analysis are covered in IV.C.4.]

3. Professional training and development

Define, describe, and apply training needs analysis methods for software quality professionals, and manage training resources and materials. (Application)

II. Software Quality Management (30 Questions)

A. Goals and objectives

1. Quality goals and objectives

Describe, analyze, and evaluate quality goals and objectives for programs, projects, and products. (Evaluation)

2. Outsourced services

Define, analyze, and evaluate the impact of acquisitions, subcontractor services, and other external resources on the organization's goals and objectives. (Evaluation)

3. Planning

Identify, apply, and evaluate scheduling and resource requirements necessary to achieve quality goals and objectives. (Evaluation)

4. Software quality management (SQM) systems documentation

Identify and describe various elements related to SQM system documentation. (Comprehension)

5. Customer requirements

Analyze and evaluate customer requirements and their effect on programs, projects, and products. (Evaluation) [NOTE: *Changes* in requirements are covered in III.B.3. The focus in this section is to ensure that customer requirements are evaluated properly.]

B. Methodologies

1. Review, inspection, and testing

Define, describe, evaluate, and differentiate between these defect detection methods. (Evaluation)

2. Change management methods

Identify and apply various methods appropriate for responding to changes in technology, organizations, environment, human performance, etc. (Evaluation) [NOTE: Change-agent tools are covered in I.C.1.]

3. Cost of quality (COQ)

Define, differentiate, and analyze COQ categories (prevention, appraisal, internal failure, external failure) and their impact on products and processes. (Analysis) [NOTE: Interpreting and reporting COQ data are covered in IV.B.2.]

4. Quality data tracking

Define, describe, select, and implement information systems and models used to track quality data in various situations. (Evaluation)

5. Problem reporting and corrective action procedures

Define, describe, analyze, and distinguish between these procedures for software defects, process nonconformances, and other quality system deficiencies. (Evaluation)

6. Quality improvement processes

Define, describe, analyze and distinguish between various defect prevention, detection, and removal processes, and evaluate process improvement opportunities in relation to these tools. (Evaluation)

C. Audits

1. Program development and administration

Identify roles and responsibilities for various audit participants, including team leader, team members, auditee, auditor, etc. (Comprehension)

2. Audit preparation and execution

Define and distinguish between various audit types, including process, compliance, supplier, system, etc. Define and describe various steps in the audit process, from scheduling the audit through the closing meeting and subsequent follow-up activities. Define and identify various tools and procedures used in conducting audits. (Comprehension)

3. Audit reporting and follow up

Identify, describe, and apply the steps of audit reporting and follow up, including the need for and verification of corrective action. (Application)

III. Software Engineering Processes (26 Questions)

A. Environmental conditions

1. Life cycles

Compare and evaluate the characteristics of spiral, waterfall, incremental, rapid prototyping, V-model, etc. Differentiate these life cycles, describe what they are designed to do, what their benefits are, and in what situations they should be used. (Evaluation)

2. Systems architecture

Identify, describe, evaluate, and distinguish between system architectures, including client server, n tier, B to B, B to C, and B to E, web (internet/intranet/extranet) and wireless development, messaging and collaboration software, etc. (Analysis)

B. Requirements management

1. Requirements prioritization and evaluation

Describe, assess, prioritize, and evaluate the requirements for verifying software correctness, consistency, completeness, and testability. Determine what should be covered in a requirements statement, how to specify a requirement, etc. (Evaluation)

2. Requirements change management

Define, describe, and evaluate various elements of managing requirements change, including what processes should be followed, when requirements need to change, what review processes to use, etc. Define the effect of changing requirements at various stages of the project life cycle. (Evaluation)

3. Bi-directional requirements traceability

Describe, select, and evaluate various traceability elements, including requirements to design, design to code, and requirements to test. Describe and apply traceability tools and mechanisms, such as system verification diagrams, traceability matrices, etc. (Evaluation) [NOTE: Traceability of configuration items is covered in VII.C.5.]

C. Requirements engineering

1. Requirement types

Define, describe, and analyze various requirement types such as security, regulatory, quality, feature and product functionality, etc., and the significant elements of each. (Analysis)

2. Requirements elicitation

Define and describe various elicitation methods, including using tools such as quality function deployment (QFD), joint application development (JAD), customer needs analysis, etc. Describe the key steps necessary for gathering product requirement details, and identify common causes of failure to comply with requirements. (Comprehension)

3. Requirements analysis and modeling

Describe, select, and use tools such as data flow diagrams (DFDs), entity relationship diagrams (ERDs), use cases, etc. Describe how they are used at different phases of development and requirements specifications. (Analysis)

4. System and software requirements specifications

Define and distinguish between these two types of specifications and their purpose, and describe their relationship to each other. (Analysis)

D. Analysis, design, and development methods and tools

1. Software design methods

Define and use various design methods, including object-oriented analysis and design (OOAD), structured analysis and design (SAD), unified modeling language (UML), etc. Identify the steps used in program design and explain their uses. (Application)

2. Types of software reuse

Define, describe, and differentiate the use of various reuse methods including reengineering, reverse engineering, plug-and-play, etc., and describe the design paradigms that address these concepts. (Application)

3. Cleanroom and other formal methods

Define and describe these methods and their benefits. (Comprehension)

4. Software development tools

Identify, describe, use, and distinguish between various tools used for modeling, code analysis, documentation, relational databases, etc. (Application)

E. Maintenance management

1. Maintenance types

Describe the characteristics of corrective, adaptive, and perfective maintenance types and their benefits and risks. (Comprehension)

2. Operational maintenance

Describe the various categories of and activities involved in providing operational services to the customer, managing application portfolios, and providing basic software maintenance. (Comprehension)

IV. Program and Project Management (24 Questions)

A. Planning

1. Project planning elements

Describe and use factors such as forecasts, resources, schedules, etc., to develop, initiate, and accomplish project goals. (Application)

2. Goal-setting and deployment

Identify and use milestones, objectives achieved, task duration, and other goal-setting and deployment methods. (Application)

3. Project planning tools

Define, apply, and analyze various methods of managing risk, estimating costs, scheduling resources, etc. using tools such as PERT charts, critical path method (CPM), work breakdown structure (WBS), etc. (Analysis) [NOTE: Gantt charts are covered in IV.B.1.]

4. Cost and value data

Identify and use various methods for calculating project-related data such as earned value, development investment costs, etc. (Application)

B. Tracking and controlling

1. Phase transition control techniques

Develop and use various control techniques for tracking projects, including entry/exit criteria, phase gate reviews, Gantt charts, etc. (Analysis)

2. Interpreting and reporting cost of quality (COQ) data

Review, interpret, and report COQ data and evaluate how each category is affected by continuous improvement strategies. (Evaluation) [NOTE: The definitions and distinctions between these categories are covered in II.B.3.]

3. Tracking elements and methods

Describe, assess, and apply different tracking methods, including establishing metrics for costs, deliverables, productivity, etc., creating and evaluating status reports and life-cycle phase reports, measuring changes in earned value, evaluating changes in business conditions, etc. (Evaluation) [NOTE: Calculating earned value is covered in IV. A. 4.]

4. Project reviews

Define, use, and differentiate various types of reviews, including post-project, senior management, team, etc., and use closed-loop methodologies to improve projects as a result of lessons learned. (Analysis)

C. Risk management

1. Risk management planning methods

Define, integrate, and analyze various risk management methods, including assessing, preventing, and mitigating risk with respect to critical aspects of a project and its supporting strategies. (Synthesis)

2. Risk probability

Describe and evaluate various risk warning signs, assess risk probability and impact, and develop contingency plans. (Evaluation)

3. Product release decisions

Identify situations and factors that require trade-offs on product release decisions. Develop and analyze various ways of bringing a project back on track when problems occur that affect quality, scheduling, customer requirements, product functionality, etc. (Evaluation)

4. Software security, safety, and hazard analysis issues

Identify, review, and evaluate various factors related to software security, safety-critical software, and hazard analyses. Identify and describe rationales for developing safety plans and for

implementing hazard analyses. (Analysis) [NOTE: The legal aspects of product safety are covered in I.D.2.]

V. Software Metrics, Measurement, and Analytical Methods (24 Questions)

A. Metrics and measurement theory

1. Definitions

Define, describe, and explain various terms related to metrics and measurement, including error, reliability, internal vs. external validity, explicit vs. derived measures, etc. (Comprehension)

2. Basic measurement theory and techniques

Define, describe, and use basic measurement scales (nominal, ordinal, ratio, interval), the central limit theorem and related terms, including mean, median, mode, standard deviation, variance, etc. (Application)

3. Psychology of metrics

Define and describe various uses of metrics. Compare and contrast how metrics affect people and how people affect metrics. (Comprehension)

B. Process and product measurement

1. Process, product, and resource metrics

Describe and use various metrics to assess processes, products, and resources. (Application)

2. Commonly used metrics

Define and use metrics to measure various aspects of software, including software complexity, lines of code (LOC), non-commented lines of code (NCLOC), design defects, requirements volatility, system performance, etc. (Application) [NOTE: Code coverage metrics are covered in VI.D.4.]

3. Software quality attributes

Identify and describe various criteria for measuring attributes such as maintainability, verifiability, reliability, usability, reusability, testability, expandability, etc. (Comprehension)

4. Defect detection effectiveness measures

Define, describe, and use defect detection measures such as cost, yield, customer impact, etc., and track their effectiveness. (Application)

5. Program performance and process effectiveness

Identify and use various methods of examining performance and effectiveness. (Analysis)

C. Analytical techniques

1. Data integrity

Define, use, and interpret various techniques to ensure the quality of metrics data, its accuracy, completeness, timeliness, etc. (Synthesis)

2. Sampling theory and techniques

Describe, differentiate, and analyze various sampling techniques for use in auditing, testing, product acceptance, etc. (Analysis)

3. Quality tools: Charts

Define, select, and use tools such as flowcharts, Pareto charts, scatter diagrams, control/run charts, histograms, process decision program charts (PDPCs), etc. (Analysis)

4. Quality tools: Diagrams

Define, select, and use tools such as cause and effect diagrams, scatter diagrams, affinity diagrams, tree diagrams, activity network diagrams, etc. (Analysis)

5. Miscellaneous quality tools

Define, select, and use various problem-solving tools such as root cause analysis, interrelationship digraphs, prioritization matrices, etc. (Analysis)

VI. Software Verification and Validation (V&V) (24 Questions)

A. Theory

1. V&V planning procedures and tasks

Identify and select various methods for verification and validation, including static analysis, structural analysis, mathematical proof, simulation, etc. Identify and analyze which tasks should be iterated as a result of proposed or completed modifications. (Synthesis)

2. V&V program

Describe and analyze methods for managing and reviewing a V&V program, including technical accomplishments, resource utilization, program status, etc. (Analysis)

3. Evaluating software products and processes

Analyze and select various ways of evaluating documentation, source code, test and audit results, etc., to determine whether user needs and project objectives have been satisfied. (Synthesis)

4. Interfaces

Identify various interfaces used with hardware, user, operator, and software applications. (Comprehension)

B. Reviews and inspections

1. Types

Define, describe, and use various types of reviews and inspections, including desk-checking, walk-throughs, Fagan and Gilb inspections, technical accomplishments, resource utilization, future planning, etc. (Application)

2. Items

Identify, describe, and use various review and inspection items, including proposals, project charters, specifications, code, tests, etc. (Application)

3. Processes

Define, describe, and use various review and inspection processes to examine objectives, criteria, techniques, methods, etc. (Application)

4. Data collection, reports, and summaries

Define, describe, and use terms related to data collection, including preparation rates, defect density yield, phase containment, etc. (Application)

C. Test planning and design

1. Types of tests

Select, apply, and develop various types of test, including functional, performance, regression, certification, environmental load, stress, worst case, perfective, exploratory, etc. (Synthesis)

2. Test tools

Define and describe the application and capabilities of commonly used test tools such as acceptance test suites, utilities (for memory, screen capture, string-finding, file viewer, file comparison, etc.), and diagnostics (for hardware, software, configuration, etc.). (Comprehension)

3. Test strategies

Identify, analyze, and apply various test strategies, including top-down, bottom-up, black-box, white-box, simulation, automation, etc. (Synthesis)

4. Test design

Identify, describe, and apply various types of test design including fault insertion, fault-error handling, equivalence class partitioning, boundary value, etc. (Application)

5. Test coverage of specifications

Identify, apply, and develop various test coverage specifications, including functions, states, data and time domains, etc. (Synthesis)

6. Test environments

Identify various environments and use tools such as test libraries, drivers, stubs, harnesses, etc., in those environments, and describe how simulations can be used in test environments. (Synthesis)

7. Supplier components and products

Identify the common risks and benefits of incorporating purchased software into other software products. Use various methods to test supplier components and products in the larger system. (Application)

8. Test plans

Identify, describe, and apply methods for creating and evaluating test plans including system, acceptance, validation, etc., to determine whether project objectives are being met. (Application)

D. Test execution and evaluation

1. Test implementation

Define, describe, and use various implementation elements, including scheduling, freezing, dependencies, V-model, error repair models, acceptance testing, etc. (Application)

2. Test documentation

Define, describe, and use various documentation procedures, including defect recording and tracking, test report completion metrics, trouble reports, input/output specifications, etc. (Application)

3. Test Reviews

Describe, develop, and analyze various methods of reviewing test efforts, including technical accomplishments, future planning, risk management, etc. (Synthesis)

4. Code coverage metrics

Define and apply various metrics including branch-to-branch, condition, domain, McCabe's cyclomatic complexity, boundary, etc. (Application) [NOTE: Other types of metrics are covered in V.B.2.]

5. Customer deliverables

Identify and select various methods for testing the accuracy of customer deliverables, including packaged or downloaded products, license keys, user documentation, marketing and training materials, etc. (Synthesis)

6. Severity of anomalies

Identify and select various methods for evaluating severity of anomalies in software operations. (Evaluation)

VII. Software Configuration Management (16 Questions)

A. Configuration infrastructure

1. Configuration management

Describe the roles and responsibilities of the configuration management group. (Comprehension)

2. Library/repository processes

Define and identify processes used in a library system including dynamic, static, controlled, etc., and their related procedures. (Comprehension)

3. Defect tracking and library tools

Define and describe configuration management tools used for defect tracking, library management tools, etc. (Comprehension)

B. Configuration identification

1. Configuration items

Define, select, and use various items, including documentation, code interfaces, training materials, customer-supplied equipment, etc. (Application)

2. Baselines

Define and identify when configuration baselines are created and used. (Comprehension)

3. Configuration identification methods

Define and describe how these methods relate to schemes, naming conventions, versions, serializations, etc. (Comprehension)

4. Software builds

Define and describe the primary purpose of software builds and their relation to configuration management functions. Describe and use various methods for controlling builds, including automation, new-version builds, etc. (Synthesis)

C. Configuration control

1. Item and baseline control

Define, describe, and apply various control processes, including version control, traceability requirements, specifications, concurrent development, verifying milestones, etc. (Application)

2. Proposed modifications

Describe how to assess proposed modifications, enhancements, or additions in terms of their impact on an existing or planned system. (Comprehension)

3. Review and configuration control boards (CCBs)

Define, describe, and differentiate the roles and responsibilities of and procedures used by these boards. (Application)

4. Concurrent development

Describe how configuration management control principles can be used in concurrent development processes. (Application)

5. Traceability

Identify and apply various tools and methods for establishing and maintaining traceability design, including backward and forward traceability, naming conventions, etc., and explain how they are related to configuration management objectives. (Application) [NOTE: Traceability through product development is covered in III.B.3. The focus for this area is on traceability and evolution of configuration items in code archives and other configuration management elements.]

6. Version control

Define, describe, and use version control methods such as source code version management and others, and how such methods can be used effectively by both small and large development teams. (Application)

7. Configuration item interfaces

Define, describe, and apply management control processes for configuration item interfaces. (Application)

D. Configuration status accounting

1. Status reporting

Describe various processes for establishing, maintaining, and reporting the status of configuration items. (Comprehension)

2. Changes to configuration items and baselines

Describe the processes that should be used when changes are proposed to configuration items and baselines. (Comprehension)

3. Documentation control

Define and describe related procedures for document distribution, approval, storage, retrieval, revision, etc. (Comprehension)

E. Configuration audits

1. Functional configuration audit

Describe the primary purpose of these types of audits in relation to product specifications and in contrast to physical configuration audits. (Comprehension)

2. Physical configuration audit

Describe the primary purpose of these types of audits in relation to product specifications and in contrast to functional configuration audits. (Comprehension)

F. Release and distribution issues

1. Product release process issues

Identify and describe product release issues such as planning, scheduling, hardware and software dependencies, etc. (Comprehension)

2. Packaging, production, and distribution

Define and describe these components in relation to product release requirements and related issues. (Knowledge)

Six Levels of Cognition based on Bloom's Taxonomy (1956)

In addition to **content** specifics, the subtext detail 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, 1956) and are presented below in rank order, from least complex to most complex.

Knowledge Level

(Also commonly referred to as recognition, recall, or rote knowledge) Being able to remember or recognize terminology, definitions, facts, ideas, materials, patterns, sequences, methodologies, principles, etc.

Comprehension Level

Be able to read and understand descriptions, communications, reports, tables, diagrams, directions, regulations, etc.

Application Level

Be able to apply ideas, procedures, methods, formulas, principles, theories, etc., in job-related situations.

Analysis

Be able to break down information into its constituent parts and recognize the parts' relationship to one another and how they are organized; identify sublevel factors or salient data from a complex scenario.

Synthesis

Be able to put parts or elements together in such a way as to show 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.

Evaluation

Be able to make judgments regarding the value of proposed ideas, solutions, methodologies, etc., by using appropriate criteria or standards to estimate accuracy, effectiveness, economic benefits, etc.