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 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, including identifying conflicts of interest, recognizing and resolving ethical issues, etc. (Evaluate)

2. Regulatory and legal issues

Describe the importance of compliance to federal, national, and statutory regulations on software development. Determine the impact that issues such as copyright, intellectual property rights, product liability, and data privacy. (Understand)

C. Standards and models

Define and describe the ISO 9000 and IEEE software standards, and the SEI Capability Maturity Model Integration (CMMI) for Development, Services, and Acquisition 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, recognition). (Apply)

2. Facilitation skills

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

3. Communication skills

Use various communication methods in oral, written, and presentation formats. Use various techniques for working in multi-cultural environments, and identify and describe the impact that culture and communications can have on quality. (Apply)

E. Team Skills

1. Team management

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

2. Team tools

Use decision-making and creativity tools, such as brainstorming, nominal group technique (NGT), multi-voting. (Apply)

II. Software Quality Management (22 questions)

A. Quality Management System

1. Quality goals and objectives

Design software quality goals and objectives that are consistent with business objectives. 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. Customers and other stakeholders

Describe and analyze the effect of various stakeholder group requirements on software projects and products. (Analyze)

3. Outsourcing

Determine the impact that outsourced services can have on organizational goals and objectives, and identify criteria for evaluating suppliers/vendors and subcontractors. (Analyze)

4. Business continuity, data protection and data management

Design plans for business continuity, disaster recovery, business documentation and 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 COQ categories (prevention, appraisal, internal failure, 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, the six sigma methodology, and use Define, Measure, Act, Improve, Control (DMAIC) model and the plan-do-check-act (PDCA) model for process improvement. (Apply)

3. *Corrective action procedures*

Evaluate corrective action procedures 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, special training. (Evaluate)

C. Audits

1. *Audit types*

Define and distinguish between various audit types, including process, compliance, supplier, system. (Understand)

2. *Audit roles and responsibilities*

Identify roles and responsibilities for audit participants including 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 and identify their benefits and when they are used. (Apply)

3. *Agile Software Development Lifecycle*

Apply the Agile Lifecycle and related Process Models and identify their benefits and when they are used. (Apply)

B. Systems architecture

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

C. Requirements engineering

1. *Product requirements*

Define and describe various types of product requirements, including system, feature, function, interface, integration, performance, globalization, localization. (Understand)

2. ***Data/Information requirements***

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

5. ***Security Requirements***

   Define and describe various types of security requirements including data security, information security, cybersecurity, data privacy. (Understand)

6. ***Requirements elicitation methods***

   Describe and use various requirements elicitation methods, including customer needs analysis, use cases, human factors studies, usability prototypes, joint application development (JAD), storyboards, etc. (Apply)

7. ***Requirements evaluation***

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

D. Requirements management

1. ***Requirements change management***

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

2. ***Bidirectional traceability***

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

E. Software analysis, design, and development

1. ***Design methods***

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

2. ***Quality attributes and design***

   Analyze the impact that quality-related elements (safety, security, reliability, usability, reusability, 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)

F. 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, 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 (22 questions)

A. Planning, scheduling, and deployment

1. *Project planning*

Use forecasts, resources, schedules, task and 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, task duration to set goals and deploy the project. (Apply)

B. Tracking and controlling

1. *Phase transition control*

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

2. *Tracking methods*

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

3. *Project reviews*

Use various types of project reviews such as phase-end, management, and retrospectives or 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 their performance, technical accomplishments, resource utilization, etc. (Understand)

C. Risk management

1. *Risk management methods*

Use risk management techniques (e.g. assess, prevent, mitigate, transfer) to evaluate project risks. (Evaluate)

2. *Software security risks*

Evaluate risks specific to software security, including deliberate attacks (hacking, sabotage, etc.), inherent defects that allow unauthorized access to data, and other security breaches. Plan appropriate responses to minimize their impact. (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 (19 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), process capability). (Apply)

4. *Data Integrity*

Describe the importance of data integrity from planning through collection and analysis and 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, including dashboards, stoplight charts, etc. 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 and tree diagrams, matrix and activity network diagrams, root cause analysis and data flow diagrams (DFD)). (Apply)

VI. Software Verification and Validation (29 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 as a result 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, etc.) for various situations. (Analyze)

2. Test Plans

Develop and evaluate test plans and procedures, including system, acceptance, validation, etc., 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

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

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

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

VII. Software Configuration Management (20 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, defect tracking systems. (Understand)

3. *Library processes*

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

B. Configuration identification

1. *Configuration items*

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

2. *Software builds and baselines*

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

C. Configuration control and status accounting

1. *Item change and version control*

Describe processes for documentation control, item change tracking, version control that are used to manage various configurations, and 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 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*

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

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, defining hardware and software dependencies.) (Evaluate)

2. ***Customer Deliverables***

   Assess the completeness of customer deliverables including packaged and hosted or downloadable products, license keys and user documentation, 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, offsite storage.) (Evaluate)