Software Quality Engineer Certification

Body of Knowledge

The following is an outline of topics that constitute the Body of Knowledge for Software Quality Engineer. This new BOK started with the exams on December 6, 2008. Compare the old and new BOKs (PDF, 321 KB).

You have ASQ's consent to republish and redistribute this information.

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 page.

1. General Knowledge (16 questions)
    1. Quality principles
        1. Benefits of software quality

            Describe the benefits that software quality engineering can have at the organizational level. (Understand)
        2. Organizational and process benchmarking

            Use benchmarking at the organizational, process, and project levels to identify and implement best practices. (Apply)
    2. Ethical and Legal Compliance
        1. ASQ Code of Ethics

            Determine appropriate behavior in situations requiring ethical decisions, including identifying conflicts of interest, recognizing and resolving ethical issues, etc. (Evaluate)
        2. Legal and regulatory issues

            Define and describe the impact that issues such as copyright, intellectual property rights, product liability, data privacy, the Sarbanes-Oxley Act, etc., can have on software development. (Understand)
    3. Standards and models

        Define and describe the following standards and assessment models: ISO 9000 standards, IEEE software standards, and the SEI Capability Maturity Model Integrated (CMMI). (Understand)
    4. Leadership skills
        1. Organizational leadership

            Use leadership tools and techniques, such as organizational change management, knowledge-transfer, motivation, mentoring and coaching, recognition, etc. (Apply)

2. Facilitation skills

Use various approaches to manage and resolve conflict. Use negotiation techniques and identify possible outcomes. Use meeting management tools to maximize performance. (Apply)

3. Communication skills

Use various communication elements (e.g., interviewing and listening skills) 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)

5. 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, and working with diverse groups and in distributed work environments. (Apply)

2. Team tools

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

2. Software Quality Management (26 questions)

1. Quality Management System

1. Quality goals and objectives

Design quality goals and objectives for programs, projects, and products that are consistent with business objectives. Develop and use documents and processes necessary to support software quality management systems. (Create)

2. Customers and other stakeholders

Describe and distinguish between various stakeholder groups, and analyze the effect their requirements can have on software projects and products. (Analyze)

3. Planning

Design program plans that will support software quality goals and objectives. (Evaluate)

4. Outsourcing

Determine the impact that acquisitions, multi-supplier partnerships, outsourced services, and other external drivers can have on organizational goals and objectives, and design appropriate criteria for evaluating suppliers and subcontractors. (Analyze)

2. Methodologies

1. Cost of quality (COQ)

Analyze COQ categories (prevention, appraisal, internal failure, external failure) and their impact on products and processes. (Evaluate)

2. Process improvement models

Define and describe elements of lean tools and the six sigma methodology, and use 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, etc. (Evaluate)

3. Audits

1. Audit types

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

2. Audit roles and responsibilities

Identify roles and responsibilities for audit participants: 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)

3. Systems and Software Engineering Processes (27 questions)

1. Lifecycles and process models

Evaluate various software development lifecycles (iterative, waterfall, etc.) and process models (V-model, Feature Driven Development, Test Driven Development, etc.) and identify their benefits and when they should be used. (Evaluate)

2. Systems architecture

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

3. Requirements engineering

1. Requirements types

Define and describe various types of requirements, including feature, function, system, quality, security, safety, regulatory, etc. (Understand)

2. Requirements elicitation

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

3. Requirements analysis

Identify and use tools such as data flow diagrams (DFDs), entity relationship diagrams (ERDs), etc., to analyze requirements. (Apply)

4. Requirements management

1. Participants

Identify various participants who have a role in requirements planning, including customers, developers, testers, the quality function, management, etc. (Understand)

2. Requirements evaluation

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

3. Requirements change management

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

4. Bidirectional traceability

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

5. 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 such as object-oriented analysis and design (OOAD), structured analysis and design (SAD), and patterns. (Understand)
    2. Quality attributes and design
        Analyze the impact that quality-related elements (safety, security, reliability, usability, reusability, maintainability, etc.) 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
        Select the appropriate development tools to use for modeling, code analysis, etc., and analyze the impact they can have on requirements management and documentation. (Analyze)
    5. Software development methods
        Define and describe principles such as pair programming, extreme programming, cleanroom, formal methods, etc., and their impact on software quality. (Understand)
6. 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, etc., and their impact on software quality. (Understand)

4. Project Management (24 questions)
1. Planning, scheduling, and deployment
    1. Project planning
        Use forecasts, resources, schedules, task and cost estimates, etc., to develop project plans. (Apply)
    2. Project scheduling
        Use PERT charts, critical path method (CPM), work breakdown structure (WBS), Scrum, burn-down charts, and other tools to schedule and monitor projects. (Apply)
    3. Project deployment
        Use various tools, including milestones, objectives achieved, task duration, etc., to set goals and deploy the project. (Apply)
2. Tracking and controlling
    1. Phase transition control
        Use phase transition control tools and techniques such as entry/exit criteria, quality gates, Gantt charts, integrated master schedules, etc. (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)

3. Risk management

1. Risk management methods

Use risk management techniques (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, and determine 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)

5. Software Metrics and Analysis (24 questions)

1. Metrics and measurement theory

1. Terminology

Define and describe metrics and measurement terms including reliability, internal and external validity, explicit and derived measures, etc. (Understand)

2. Basic measurement theory and statistics

Define the central limit theorem, and describe and use mean, median, mode, standard deviation, variance, and range. Apply appropriate measurement scales (nominal, ordinal, ratio, interval) in various situations. (Apply)

3. Psychology of metrics

Describe how metrics and measuring affect the people whose work is being measured and how people affect the ways in which metrics are used and data are gathered. (Understand)

2. Process and product measurement

1. Software metrics

Use metrics to assess various software attributes such as size, complexity, number of defects, the amount of test coverage needed, requirements volatility, and overall system performance. (Apply)

2. Process metrics

Measure the effectiveness and efficiency of software using functional verification tests (FVT), cost, yield, customer impact, defect detection, defect containment, total defect containment effectiveness (TDCE), defect removal efficiency (DRE), process capability and efficiency, etc. (Apply)

3. Metrics reporting tools

Use various metric representation tools, including dashboards, stoplight charts, etc., to report results efficiently. (Apply)

3. Analytical techniques

1. Sampling

Define and distinguish between sampling methods (e.g., random, stratified, cluster) as used in auditing, testing, product acceptance, etc. (Understand)

2. Data collection and integrity

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

3. Quality analysis tools

Describe and use classic quality tools (flowcharts, Pareto charts, cause and effect diagrams, control charts, histograms, etc.) and problem-solving tools (affinity and tree diagrams, matrix and activity network diagrams, root cause analysis, etc.) in a variety of situations. (Apply)

6. Software Verification and Validation (V&V) (27 questions)

1. Theory

1. V&V methods

Select and use V&V methods, including static analysis, structural analysis, mathematical proof, simulation, etc., and analyze which tasks should be iterated as a result of modifications. (Analyze)

2. Software product evaluation

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

2. Test planning and design

1. Test strategies

Select and analyze test strategies (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. (Create)

3. Test designs

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

5. Tests of supplier components and products

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

6. Test coverage specifications

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

7. Code coverage techniques

Identify and use techniques such as branch-to-branch, condition, domain, McCabe's cyclomatic complexity, boundary, etc. (Apply)

8. Test environments

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

9. Test tools

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

3. Reviews and inspections

Identify and use desk-checks, peer reviews, walk-throughs, Fagan and Gilb inspections, etc. (Apply)

4. Test execution documentation

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

5. Customer deliverables

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

7. Software Configuration Management (16 questions)

1. 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 these tools as they are used for managing libraries, build systems, defect tracking systems, etc. (Understand)

3. Library processes

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

2. Configuration identification

1. Configuration items

Describe configuration items (documentation, software code, equipment, etc.), identification methods (naming conventions, versioning schemes, etc.), and when baselines are created and used. (Understand)

2. Software builds

Describe the relationship between software builds and configuration management functions, and describe methods for controlling builds (automation, new versions, etc.). (Understand)

3. Configuration control and status accounting

1. Item, baseline, and version control

Describe processes for documentation control, tracking item changes, version control, etc., 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 and responsibilities of the CCB and its members and the procedures they use. (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. (Understand)

4. Configuration audits

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

5. Product release and distribution

1. Product release

Review product release processes (planning, scheduling, defining hardware and software dependencies, etc.) and assess their effectiveness. (Evaluate)

2. Archival processes

Review the source and release archival processes (backup planning and scheduling, data retrieval, archival of build environments, retention of historical records, offsite storage, etc.) and assess their effectiveness. (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 levelof 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, etc.

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

Apply

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

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, etc., 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.

Software Quality Engineer Certification

References

These references support the topic areas in the CSQE Body of Knowledge. They are used to write and review the questions in the test question pool. To determine which references you need, review the CSQE Body of Knowledge. Determine which topic areas you need to study further. Then review this reference list to identify those that support your needs.

Note: ASQ Certification Board does not endorse any one particular reference source.

* ANSI/ISO/ASQ Q9001-2000: Quality Management Systems: Requirements.
* ANSI/ISO/IEETICKIT Guidelines.
* Arter, Dennis, Quality Audits for Improved Performance, Third Edition, Milwaukee: ASQC Quality Press, 2003.
* Beizer, Boris, Black-Box Testing: Techniques for Functional Testing of Software and Systems, New York: John Wiley & Sons, 1995. ISBN 0471120944.
* Booch, Grady, James Rumbaugh, and Ivar Jacobson, The Unified Modeling Language User Guide, Addison-Wesley, 1999. ISBN 0201571684.

*

Booch, Grady, Objected-Oriented Analysis and Design with Applications Second Edition, Addison-Wesley Publishing Co., Inc., 1994. ISBN 0805353402.
*

Brassard, Michael and Diane Ritter, The Memory Jogger II: A Pocket Guide of Tools for Continuous Improvement and Effective Planning, Goal/QPC, 1994.
*

Brooks, Frederick P. Jr., The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition, Reading, Mass: Addison-Wesley Publishing Co., 1995. ISBN 0201835959.
*

Capability Maturity Model® Integration (CMMI), Version 1.1. CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing, March 2002. Available here.
*

Daughtrey, Taz, Edition, Fundamental Concepts for the Software Quality Engineer, ASQ Quality Press, 2002.
*

Dunn, Robert H., and Richard S Ullman, TQM for Computer Software (System Design and Implementation) Second Edition, McGraw Hill, 1994. ISBN 0070183147.
*

Fewster, Mark, and Dorothy Graham, Software Test Automation: Effective Use of Test Execution Tools, Great Britain: Addison-Wesley, 1999. ISBN 0201331403.
*

Freedman, Daniel, and Gerald M Weinberg, Handbook of Walkthroughs, Inspections, and Technical Reviews: Evaluating Programs, Projects, and Products, Third Edition, New York: Dorset House, 1990. ISBN 0932633196.
*

Futrell, Robert T., Donald F. Shafer, and Linda I. Shafer, Quality Software Project Management, Upper Saddle River, NJ: Prentice Hall, 2002. ISBN 0130912972.
*

Gilb, Tom, and Dorothy Graham, Software Inspection, London: Addison-Wesley Professional, 1993. ISBN 0201631814.
*

Grady, Robert B, Practical Software Metrics for Project Management and Process Improvement, Englewood Cliffs, New Jersey: Prentice Hall, 1992. ISBN 0137203845.
*

Gryna, Frank M., Quality Planning and Analysis: From Product Development through Use, Fourth Edition, New York: McGraw-Hill, 2001.
*

Hetzel, Bill, Complete Guide to Software Testing, Second Edition, Wellesley, Mass: QED Information Sciences, 1988. ISBN 0894352423.
*

Hetzel, Bill, Making Software Measurement Work, Boston, MA, QED Publishing Group, 1993. ISBN 0894354655.
*

Humphrey, Watts, A Discipline for Software Engineering, Reading, Mass: Addison-Wesley, 1995. ISBN 0201546108.
*

Humphrey, Watts, Managing the Software Process, Reading, Mass: Addison-Wesley, 1989. ISBN 0201180952.
*

IEEE Standard 12207.0-1996: IEEE Standard for Industry Implementation of ISO/IEC 12207:1995.
*

ISO/IEC TR 15504-1998: Parts 1-9 Information Technology—Software Process Assessment. Available here.
*

Juran, Joseph M., Juran's Quality Handbook, Fifth Edition, New York: McGraw-Hill, 1999.
*

Kan, Stephen H., Metrics and Models in Software Quality Engineering, Second Edition, Boston: Addison-Wesley, 2003. ISBN 0201729156.
*

Kaner, Cem, Jack Falk, and Hung Quoc Nguyen, Testing Computer Software, Wiley Computer Publishing, 1999. ISBN 0471358460.
*

Kerzner, Harold, PhD, Project Management: A Systems Approach to Planning, Scheduling, and Controlling, Eighth Edition, Hoboken, New Jersey: John Wiley & Sons, Inc, 2003. ISBN 0471225770.
*

Kit, Edward, Software Testing in the Real World: Improving the Process, Addison-Wesley Publishing Co., 1995. ISBN 0201877562.
*

Lyu, Michael R., Handbook of Software Reliability Engineering, Los Alamitos: IEEE Computer Society Press, New York: McGraw-Hill, 1996. ISBN 0070394008. Complete book available online.
*

McConnell, Steve, Rapid Development: Taming Wild Software Schedules, Redmond, Wash: Microsoft Press, 1996. ISBN 1556159005.
*

Myers, Glenford J., The Art of Software Testing, New York: John Wiley & Sons, 1979. ISBN 0471043281.
*

Nguyen, Hung, Testing Applications on the Web: Test Planning for Internet-Based Systems, New York: John Wiley & Sons, Inc, 2001. ISBN 047139470X.
*

Paulk, Mark C., et al. The Capability Maturity Model—Guidelines for Improving the Software Process, Addison-Wesley Publishing Co., 1995. ISBN 0201546647.
*

Pressman, Roger S., Software Engineering: A Practitioner's Approach, Fifth Edition, New York: McGraw-Hill, 2000. ISBN 0072496681.
*

Rakitin, Steven R., Software Verification and Validation for Practitioners and Managers, Second Edition, Boston: Artech House, 2001. ISBN 1580532969.
*

Russell, J.P., ed., ASQ Quality Audit Division, The Quality Audit Handbook, Second Edition, Milwaukee: ASQ Quality Press, 2000.
*

Scholtes, Peter R., Brian L. Joiner, and Barbara J. Streibel, The Team Handbook, Third Edition, Oreil Inc., 2003.
*

Schulmeyer, G. Gordon, and James I. McManus, Handbook of Software Quality Assurance, Third Edition, Upper Saddle River, NJ: Prentice Hall, 1999.

Books with links are available from ASQ Quality Press.