

# Overview of ASQ's<sup>1</sup> Certified Software Quality Engineer (CSQE) Body of Knowledge

**Douglas Hoffman**

Software Quality Methods, LLC.  
24646 Heather Heights Place  
Saratoga, California 95070-9710  
Phone 408-741-4830 Fax 408-867-4550  
<http://www.SoftwareQualityMethods.com>  
[doug.hoffman@acm.org](mailto:doug.hoffman@acm.org)

## Abstract


In 2002, the American Society for Quality (ASQ) has restructured and updated the Body of Knowledge (BOK) used for their Certification in Software Quality Engineering (CSQE). This paper describes the certification, outlines the updated BOK content, and highlights many of the changes<sup>2</sup>. It also provides a method of adapting such a BOK to describe key skills and levels of performance in a group. The following topics are covered in the paper:

- Certification Requirements
- The Subject Areas of the CSQE 2002 Body of Knowledge
  - General Knowledge, Conduct, and Ethics
  - Software Quality Management
  - Software Engineering Processes
  - Program and Project Management
  - Software Metrics, Measurement, and Analytical Methods
  - Software Verification and Validation (V&V)
  - Software Configuration Management
- Levels of Cognition (from Bloom's Taxonomy, 1956)
- Performance Skill Levels
- Mapping of Performance Levels to Job Requirements
- Describing Individual Performance Levels

---

<sup>1</sup> Much of the reference material contained in this document comes from ASQ's CSQE Certification brochure, ASQ Item B0110, Revised 5-02. Copyright © 2002 American Society for Quality. Reprinted with permission.

See the side bar [About the ASQ](#) for a brief description of ASQ. Further information on the American Society for Quality (ASQ) and certifications can be found at their web site at <http://www.asq.org> or by calling them at 800-248-1946.

<sup>2</sup> Special thanks to Bill Wortman and Wes Richardson at QCI and Phil Marriott for the mapping of the 2002 BOK with the 1996 BOK. (Quality Council of Indiana  <http://www.qualitycouncil.com>)

The purpose of this paper is to outline ASQ's CSQE BOK and describe how a mapping of such knowledge and skill areas into performance measures can be useful. By identifying relevant skill areas and performance measures, a quality engineering team can clearly understand what levels of performance are expected and how to evaluate the levels of performance being shown.

Readers should come away with a better understanding of:

- The topics contained in ASQ's CSQE BOK
- The level of understanding needed for certification
- How to map BOK topics into job requirements and individual skill sets
- A spreadsheet technique for displaying QA tasks and levels of performance
- A spreadsheet technique for mapping tasks and performance to job requirements
- A further technique for describing an individual's demonstrated performance level

The following detailed material is provided in appendices:

- I. Outline of the Subject Areas of the CSQE 2002 Body of Knowledge
- II. Mapping of the 2002 CSQE BOK with the 1996 BOK
- III. CSQE Reference Materials

## Background

*"The Certified Software Quality Engineer...*

*... is a professional who has comprehensive understanding of software quality development and implementation; has a thorough understanding of software inspection, testing, verification, and validation; and can implement software development and maintenance processes and methods."*<sup>3</sup>

I often illustrate the difference between software test and quality assurance by explaining about a course I teach to prepare engineers to take the CSQE examination. The course is four days long (28 hours), and I introduce and define the subject areas in the BOK. Software Testing is covered in about two hours. The rest of the time is spent on Software Quality Assurance.

## About the ASQ

The American Society for Quality (ASQ), a not-for-profit professional society, has been the leading quality improvement organization in the United States for more than 50 years. ASQ has more than 117,000 individual and 1,100 corporate sustaining members worldwide. Individual members belong to one of 247 local Sections located throughout the United States, Canada, Mexico, Puerto Rico, and an International Chapter. ASQ also has 22 industry and topic-specific Divisions.

ASQ was incorporated as the American Society for Quality Control in 1946 as the result of the merger of several local quality societies that had formed after wartime statistical quality control classes. The classes were held to improve and maintain the quality of defense materials during World War II. To meet the needs of a changing marketplace, the organization changed its name to the American Society for Quality in 1997.

<sup>3</sup> ASQ CSQE Certification brochure, Item B0110, Revised 5-02. Copyright © 2002 American Society for Quality. Reprinted with permission. All rights reserved.

Since 1968, nearly 80,000 certifications have been awarded to professionals through ASQ's programs. The BOK and CSQE certification was first made available in 1996. Forty countries offer ASQ's programs as their method for certification. ASQ conducts professional certification programs in eleven different areas of expertise including quality engineers, quality managers, and software quality engineers. ASQ reviews and updates the BOK for each area on a regular basis to keep up with the evolving information and technologies available in each field. Computer Science in particular is evolving and changing quickly, necessitating ongoing reevaluation and adaptation of software quality assurance concepts, tools, and techniques.

Many software quality assurance organizations have come from software development or user oriented software testing. Their principles and techniques have evolved from the emerging fields of Computer Science or Software Testing, and many have been created without benefit of research or experience in quality assurance. ASQ's programs are especially valuable for these software quality professionals because they are founded on principles and techniques from the broader, general science of quality assurance.

Although the CSQE BOK may not be universally applicable to software quality organizations, it provides an excellent starting place for detailing a BOK appropriate for a given organizational context. Performance Skill Levels can then be defined for each subject area in the BOK to describe corresponding behaviors. Job Requirements can be listed to establish performance expectations in each area for the job titles and grades used for quality engineers. (A Senior Software Quality Engineer would be expected to perform many of the same tasks as an Associate Software Engineer, but at a much higher level.) Likewise, an individual's performance can be described in each of the areas to identify strengths and weaknesses.

## Certification Requirements

There are three requirements areas a candidate must fulfill to earn ASQ's CSQE:

- Education and/or Experience
- Proof of Professionalism
- Examination covering the CSQE BOK

### Education and/or Experience

One must have eight years of on-the-job experience in one or more of the areas of the CSQE BOK. A minimum of three years of this experience must be in a decision-making position. "Decision-making" is defined as the authority to define, execute, or control projects/processes and to be responsible for the outcome. This may or may not include management or supervisory positions.

## The RAB

The Registrar Accreditation Board (RAB), a separately incorporated affiliate of ASQ, is engaged in ISO 9000 and ISO 14000 accreditation and certification activities. The RAB is a partner with the American National Standards Institute (ANSI) in the National Accreditation Program (NAP) which accredits registrars and training course providers. The NAP process ensures customers and other stakeholders that companies have implemented proper management systems as defined by the ISO 9000 and ISO 14000 standards. The RAB independently operates the U.S. certification programs for both ISO 9000 and ISO 14000 auditors. These programs provide assurance that individuals are qualified to audit management systems against the requirements of recognized standards.

If a person has been previously certified by ASQ as a Quality Engineer, Quality Auditor, Reliability Engineer, or Quality Manager, experience used to qualify for certification in these fields applies to certification as a Software Quality Engineer.

If you have completed a degree\* (and have the diploma) from a college, university, or technical school with accreditation accepted by ASQ, part of the eight-year experience requirement will be satisfied, as follows (only one degree may be claimed):

Diploma From	Experience
Technical or trade school	One year
Associate degree	Two years
Bachelor's degree	Four years
Master's or doctorate	Five years

\*Degrees or diplomas from educational institutions outside the United States must be equivalent to degrees from U.S. educational institutions.

## Proof of Professionalism

Proof of professionalism may be demonstrated in one of three ways:

1. Membership in ASQ, an international affiliate society of ASQ, or another society that is a member of the American Association of Engineering Societies or the Accreditation Board for Engineering and Technology
2. Registration as a Professional Engineer
3. The signatures of two persons who are ASQ members, members of an international affiliate society, or members of another recognized professional society, verifying that you are a qualified practitioner of the quality sciences

## Examination Covering the CSQE BOK

Each certification candidate is required to pass a written examination that consists of multiple choice questions that measure comprehension of the BOK. The CSQE examination is a one-part, 160-question, four-hour exam and is offered in the English language only. Because the BOK for certification is affected by new technologies, policies, and the changing dynamics of manufacturing and service industries, changed versions of the examination based on the current BOK are used at each offering.

Examinations are conducted twice a year, in June and December, by local ASQ sections and international organizations. Some special examinations may be added in conjunction with ASQ sponsored conferences for attendees of the conference. All examinations are open-book. Each participant must bring his or her own reference materials. Use of reference materials and calculators is explained in the detailed guidance provided to applicants.

## Topics in the Body of Knowledge

The CSQE 2002 BOK is divided into seven areas, as outlined below. (The complete, ten page outline of the BOK as provided by ASQ is attached in Appendix I.) Each subject area covers important aspects of the field of software quality engineering that a professional software quality engineer should be familiar with.

Although a CSQE is not expected to have mastered all of the BOK (or all of any subject area), they must demonstrate their understanding of the field by getting passing marks on an intensive examination covering all of the subjects. There are four hours allotted to answer 160 multiple-choice questions in a proctored, open book, open notes examination. This gives an average of 90 seconds per question, which leaves very little time to refer to books or notes during the exam.

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

These topics relate to quality philosophies, principles, standards, specifications, and models in general, and particularly as they relate to software quality assurance. This knowledge area includes general quality tools and skills useful for effective teamwork and leadership.

### II. Software Quality Management (30 Questions)

These topics focus on the quality management systems for software. They include setting goals and objectives, techniques for evaluating and managing software quality, process audits, and quality improvement processes.

### III. Software Engineering Processes (26 Questions)

Software engineering processes include development processes, system architectures, software tools, and methods for software requirements, analysis, design, and development. The area is covered from project concept and development, through maintenance and obsolescence.

### IV. Program and Project Management (24 Questions)

Topics include software development planning, project tracking, project controls, and risk management. Techniques and concepts are applied to both software projects (the development and release of a software product) and software programs (ongoing operation and maintenance of software systems).

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

These topics focus on software product and process metrics, measurement theory, technology, psychological aspects of software metrics, common software metrics, measurement and analytical techniques, and quality tools.

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

Topics covered in this area include more than formalized software testing, reviews, and inspections. They cover techniques for verification (checking to see that each phase of work is done right) and validation (checking to see that the right work is done and requirements are met) including project planning, and technical analysis. Subject matter for software testing techniques encompasses test planning, test types, test tools, strategies, test design, environmental factors, documentation, implementation, reviews, execution, and evaluation.

## VII. Software Configuration Management (16 Questions)

Configuration management topics, critically important contributors to software quality, include the infrastructure components, configuration identification, component control, traceability, status reporting, document control, audits, and release issues.

## **Bloom's Six Levels of Cognition<sup>4</sup>**

A software quality professional is not expected to master all subject areas to the same extent. In addition to content specifics, the subtext detail in the outline for the BOK indicates the intended complexity level of test questions for each topic (and thus, the depth of understanding expected). The six levels of comprehension are based on Bloom's "Levels of Cognition," and are presented below from least complex to most complex.

### **Knowledge**

(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. Activities at this level are often described using verbs like define, list, label, name, state, or write.

### **Comprehension**

Being able to read and understand descriptions, communications, reports, tables, diagrams, directions, regulations, etc. Activities at this level are often described using verbs like describe, explain, illustrate, paraphrase, or summarize.

### **Application**

Being able to apply ideas, procedures, methods, formulas, principles, theories, etc., in job-related situations. Activities at this level are often described using verbs like apply, compute, construct, demonstrate, solve, or use.

### **Analysis**

Being 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. Activities at this level are often described using verbs like analyze, categorize, compare, contrast, or separate.

### **Synthesis**

Being 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. Activities at this level are often described using verbs create, design, develop, hypothesize, or invent.

### **Evaluation**

Being 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. Activities at this level are often described using verbs like critique, judge, justify, or recommend.

---

<sup>4</sup> Bloom, B. S., Engelhart, M. D., Furst, E. J., Hill, W. H., & Krathwohl, D. R. (1956). Taxonomy of educational objectives handbook 1: Cognitive domain. New York: McKay.

## Examples of Performance Skill Levels<sup>5</sup>

By listing and defining each subject area expected to be understood by a software quality professional, the BOK also defines the scope of activities for quality engineers. A similar list or BOK can be created to identify the context specific subject areas applicable to any particular software quality organization. The subject areas, levels of understanding, specific job functions, and performance activities are unique to each organization and evolve over time. We can define performance expectations and develop a guide for gaging individual performance by listing the organization and job specific subject areas, and describing typical activity exemplars for each performance level. The incomplete table below provides an example of subject areas and activity descriptions. (**Table 1: Example Performance Skill Levels**)

The first column of the table lists the subject areas for the organization's BOK. The subsequent columns each represent successive levels of understanding and performance. Each row of the table describes a subject area and the various levels of understanding demonstrated by the types of behaviors listed. The columns are shaded uniquely to simplify identification of job requirements and individual performance levels.

The example shows six Performance Skill Levels corresponding to Bloom's Levels of Cognition. There can be more or fewer Performance Skill Levels, and they can be defined as appropriate for the organization. For example, three levels of performance could have columns titled "Familiarity," "Proven Skill," and "Creative Expertise," with descriptions of behaviors in the cells for each skill area.

## Mapping of Performance Levels to Job Requirements

Various jobs within an organization encompass different task activities, and various levels of performance are expected for different grades or levels of experience. We can define the expectations of performance for various job titles in a software quality organization using the BOK and the table depicting Performance Skill Levels appropriate for each grade. The result can be represented in a tabular form, much the same as for Performance Skill Levels. (**Table 2: Example Job Performance Requirements**)

The first column of the table lists the subject areas appropriate for the performance of duties for the job functions. This can include the entire BOK or a subset of it, depending on the tasks required for the job functions. Subsequent columns are created for each job grade. The cells each describe the types of activities a person with this job title is expected to perform for this subject area. The appropriate color is shown if the description is identical to the Performance Skill Levels. Text is added to clarify or describe the particular performance required. No color is shown if the subject area is not applicable or not required for the job function and grade. A typical table covers all the grades for a particular job title and the subset of BOK subjects applicable to the job.

---

<sup>5</sup> The tabular method of describing and mapping of a BOK to job descriptions and performance is based on work done by James Bach ([www.satisfice.com](http://www.satisfice.com)) and made available by STLabs.

## Describing Individual Performance Levels

A table can also be used to describe an individual's performance by using the first column for the BOK subject areas and a second column for examples describing the actual behaviors demonstrated. This information can be compared with the Job Performance Requirements for the person's job title and grade to identify the individual's strengths and weaknesses relative to the job requirements. (**Table 3: Example Individual Performance Levels**)

The color code in the second column identifies the Performance Skill Level observed. Text in the cells can amplify or explain the observations. The third column is a direct copy of the applicable Job Requirements for the individual. Remarks may be color coded to indicate areas of strength or where improvement is needed.

Area \ Level	Knowledge	Comprehension	Application	Analysis	Synthesis	Evaluation
General Knowledge, Conduct, and Ethics						
Quality philosophy and principles	Defines and lists terminology, philosophies, and principles.	Summarizes and explains descriptions and methods related to quality principles.	Applies quality philosophies, principles, and methods in job-related situations.	Compares and contrasts quality philosophies and principles.	Creates quality philosophies and principles as required.	Makes judgments and recommendations regarding quality philosophies and principles.
Standards, specifications, and models	Identifies, lists, and defines terminology, standards, and models.	Describes and illustrates relevant standards and specifications.	Applies ideas, procedures, methods, formulas, principles, theories, etc., in job-related situations.	Compares standards, specifications, and models for applicability to given situation.	Develops standards, specifications, and models as required.	Critiques, recommends, and justifies quality philosophies and principles.
Leadership tools and skills	Lists leadership terminology, definitions, and principles.	Explains and illustrates leadership tools and techniques.	Applies appropriate leadership principles, tools, and skills.	Compares and contrasts appropriate leadership tools and techniques.	Develops leadership tools and techniques appropriate for job situations.	Critiques, recommends, and justifies leadership tools and approaches.
Ethical conduct and professional development	Defines a professional code of ethics and can identify and locate methods of professional development.	Explains a professional code of ethics and options for professional development.	Abides by a professional code of ethics and develops professionally.	Analyzes professional development requirements and options.	Designs and develops professional development programs.	Critiques, recommends, and justifies professional development approaches.

**Table 1: Example Performance Skill Levels**

Area \ Job Title	Associate QE	Quality Engineer	Senior QE	QE Fellow
General Knowledge, Conduct, and Ethics				
Quality philosophy and principles	In assigned work area.			
Standards, specifications, and models	N/A	In assigned work area.		
Leadership tools and skills	N/A			
Ethical conduct and professional development				

\* Colors correspond to columns in Table 1: Example Performance Skill Levels

**Table 2: Example Job Performance Requirements**

Area	<name>	Quality Engineer	Remarks
General Knowledge, Conduct, and Ethics			
Quality philosophy and principles	Knows quality principles in assigned work area.		
Standards, specifications, and models	Does not use available standards	In assigned work area.	Send to ISO 9000 overview class
Leadership tools and skills	Excellent leadership skills.		Get into mentor program
Ethical conduct and professional development	High ethics. Took leadership class.		

\* The Quality Engineer column is included as reference.

**Table 3: Example Individual Performance Levels**

## Conclusion

ASQ's CSQE BOK covers a very broad range of subjects applicable to software quality engineering. The BOK is generic in the sense that it may be applied to any software quality organization, even though the particular subjects and task emphasis relevant to specific organizations are different. The subjects are broken into seven general areas, and the CSQE BOK provides a detailed outline describing the concepts, tools, and techniques in each area. Along with the description of the knowledge area, the outline also lists the expected level of understanding a CSQE is expected to have.

A similar BOK can be developed that applies to a specific software quality organization. This BOK can also be used to identify the tasks associated with various job functions and the levels of performance of the tasks required at different job grades. Individual performance can then be described in terms of the subject areas and tasks appropriate for their job. The individual performance can then be compared to the job performance requirements to identify the individual's strengths and weaknesses.

# Software Quality Engineer Certification (CSQE) Body of Knowledge<sup>6</sup>

The following is an outline of topics that constitute the ASQ's 2002 Body of Knowledge for Software Quality Engineer.

- 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

---

<sup>6</sup> Copyright © 2002 American Society for Quality. All rights reserved.

- 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 analyze 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. Clean room 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. Quality tools  
Define, select, and use quality analysis and problem-solving tools such as flow charts, Pareto charts, cause and effect diagrams, check sheets, scatter diagrams, control (run) charts, histograms, root cause analysis, affinity diagrams, tree diagrams, process decision program charts (PDPCs), matrix diagrams, interrelationship digraphs, prioritization matrices, activity network diagrams. (Analysis)
3. Sampling theory and techniques  
Describe, differentiate, and analyze various sampling techniques for use in auditing, testing, product acceptance, 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 [6B1]  
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)

## Mapping of 2002 CSQE BOK with the 1996 BOK<sup>7</sup>

CSQE BOK 2002	CSQE BOK 1996
<b>I. General Knowledge, Conduct and Ethics (16 Questions)</b>	<b>I. General Knowledge, Conduct and Ethics (24 Questions)</b>
I.A. Quality philosophy and principles	I.B. Quality Philosophies and Principles
I.A.1. Benefits of software quality [No similar topic found in BOK 2002]	I.B.1. Benefits of software quality
I.A.2. Prevention vs. detection [No similar topic found in BOK 2002]	I.B.2. Quality philosophies (e.g., Juran, Deming, Crosby)
I.A.3. Organizational and process benchmarking	I.B.3. Prevention vs. Detection philosophies
I.B. Standards, specifications, and models	I.B.4. Software Total Quality Management principles and applications
I.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.	I.B.5. Organization and process benchmarking (i.e., identifying, analyzing, and modeling best practices)
I.C. Leadership tools and skills	I.A. Standards
I.C.1 Organizational leadership I.C.2. Team management I.C.4. Facilitation skills	I.A.1. Domestic and international standards and specifications (e.g., ISO 9000, IEEE, Human Factors and Ergonomics Society, graphical user interface guidelines) I.A.2. Software quality and process initiatives, ventures, and consortia (e.g., SEI, SPICE, bootstrap, ESPRIT)
I.C.1 Organizational leadership I.C.2. Team management I.C.3 Team Tools I.C.4. Facilitation skills	I.C. Organizational and Interpersonal Techniques
I.C.1 Organizational leadership I.C.2. Team management I.C.4. Facilitation skills	I.C.5. Facilitation (e.g., team management, customer-supplier relationships)
I.C.1 Organizational leadership I.C.2. Team management I.C.4. Facilitation skills	I.C.6. Principles of team leadership and facilitation
I.C.1 Organizational leadership I.C.2. Team management I.C.4. Facilitation skills	I.C.7. Meeting management
I.C.1 Organizational leadership I.C.2. Team management I.C.4. Facilitation skills	I.C.8. Conflict resolution
I.C.1 Organizational leadership I.C.2. Team management	I.C.9. Organization and implementation of various types of quality teams
I.C.5. Communication skills	I.C.1. Verbal communication and presentation I.C.2. Written communication I.C.3. Effective listening I.C.4. Interviewing

<sup>7</sup> Copyright © 2002 Quality Council of Indiana  Based on work by Phil Marriott for QCI's CSQE Primer text. Reprinted with permission. All rights reserved.

<b>CSQE BOK 2002</b>	<b>CSQE BOK 1996</b>
I.D. Ethical conduct and professional development	I.E. Professional Conduct and Ethics II.C. Organizational and Professional Software Quality Training
I.D.1. ASQ Code of Ethics	I.E.1. ASQC Code of Ethics I.E.2. Conflict of interest issues for a software quality engineer I.E.3. Ethical issues involving software product licensing
I.D.2. Software liability and safety issues	I.E.4. Legal issues involving software product liability and safety (e.g., negligence, customer notification, recall, regulations)
I.D.3. Professional training and development	II.C.1. Quality training subject areas (e.g., inspection, testing, configuration management, project management) II.C.2. Available training resources, materials, and providers II.C.3. Professional societies, technical associations, and organizations for software quality engineers
<b>II. Software Quality Management (30 Questions)</b>	<b>II. Software Quality Management (16 Questions)</b>
II.A. Goals and Objectives	II.A. Planning
II.A.1. Quality goals and objectives II.A.3. Planning	II.A.1. Product and project software quality goals and objectives
II.A.2. Outsourced Services	IV.A.6. Supplier management methodologies
II.A.3. Planning	II.A.3. Quality and customer support activities
II.A.4. Software quality management (SQM) systems documentation	[New topic in BOK 2002]
II.A.5. Customer requirements	II.A.2. Customer requirements for quality
II.B. Methodologies	II.B. Tracking
II.B. Change methodologies	III.B. Process and Technology Change Management
II.B.1. Review, inspection, and testing	III.A.3. Defect prevention, detection, and removal methods
II.B.2. Change management methods	III.B.1. Software process and technology change management theory and methods
[No similar topic found in BOK 2002]	III.B.2. Process maturity models
[No similar topic found in BOK 2002]	III.B.3. Software process assessment and evaluation techniques
[No similar topic found in BOK 2002]	III.B.4. Software process modeling (e.g., entry and exit criteria, task definition, feedback loops)
II.B.3. Cost of Quality (COQ)	IV.B.3. Cost of Quality categories (e.g., prevention, appraisal, internal failure, external failure)

CSQE BOK 2002	CSQE BOK 1996
II.B.4. Quality data tracking	II.B.1. Scope and objectives of quality information systems II.B.2. Categories of quality data and their uses II.B.4. Techniques for implementing information systems to track quality-related data II.B.5. Records and data collection, storage, maintenance, and retention
II.B.5. Problem reporting and corrective action procedures	II.B.3. Problem reporting and corrective action procedures (e.g., software defects, process nonconformances)
II.B.6. Quality improvement processes	III.A.3. Defect prevention, detection, and removal methods I.B.3. Prevention vs. Detection philosophies III.B.6. Barriers to the implementation or success of quality improvement efforts and quality systems
II.C. Audits	<a href="#">VII. Software Audits (16 Questions)</a>
II.C.1. Program development and administration - Identify roles and responsibilities for various audit participants, including team leader, team members, auditee, auditor, etc.	VII.C. Audit Planning VII.C.1. Audit team member responsibilities VII.C.2. Management (auditee and auditor) responsibilities concerning audits VII.C.3. Hosting external audits
II.C.1. Program development and administration - Identify roles and responsibilities for various audit participants, including team leader, team members, auditee, auditor, etc.	VII.C.4. Audit program development and administration VII.B.4. Audit process (e.g., objectives, criteria, techniques and methods, participant roles)
[No similar topic found in BOK 2002]	VII.C.5. Auditing requirements (e.g., industry and government standards)
II.C.2 Audit preparation and execution	VII.A. Audit Types VII.B. Audit Methodology
II.C.2 Audit preparation and execution – (a) Define and distinguish between various audit types, including process, compliance, supplier, system, etc.	VII.A.1. Performing internal audits (e.g., quality system, product, process, project, customer) VII.A.2. Performing external audits (e.g., supplier qualifications, certification of supplier systems, auditing testing done by independent agencies) VII.B.1. Purpose, objectives, frequency, and criteria of the overall audit program and individual software audits
II.C.2 Audit preparation and execution – (b) Define and describe various steps in the audit process, from scheduling the audit through the closing meeting and subsequent follow-up activities.	VII.B.3. Audit steps (planning, preparation, execution reporting, corrective action, verification, follow-up)

<b>CSQE BOK 2002</b>	<b>CSQE BOK 1996</b>
II.C.2 Audit preparation and execution – (c) Define and identify various tools and procedures used in conducting audits.	VII.B.2. Procedures, tools, and issues related to conducting audits in specific areas (e.g., software development, project management, configuration management)
II.C.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.	VII.B.3. Audit steps (planning, preparation, execution reporting, corrective action, verification, follow-up)
<b>III. Software Engineering Processes (26 Questions)</b>	<b>III. Software Processes (24 Questions)</b>
III.A. Environmental Conditions	III.A. Development and Maintenance Methods
	III.A.1. Software development procedures
III.A.1. Life cycles	III.A.2. Life cycle or process models, e.g. waterfall, spiral, etc.
III.A.2. Systems Architecture	[New topic in BOK 2002]
III.B. Requirements Management	[New topic in BOK 2002]
III.B.1. Requirements prioritization and evaluation	VI.C.5 Methods for evaluating requirements for correctness, consistency, completeness, and testability
III.B.2. Requirements change management	VI.C.9 Methods for assessing all proposed modifications, enhancements, or additions to determine the effect each change will have on the system
III.B.3. Bi-directional requirements traceability	VI.B.13 Traceability mechanisms (e.g., system verification diagrams) VI.C.3 Methods for evaluating software life cycle products and processes (e.g., physical traces, documentation, source code, plans, test and audit results ) to determine if user needs and project objectives are satisfied VI.C.4 Methods for performing requirements traceability (e.g., requirements to design, design to code)
III.C. Requirements Engineering	III.A. Development and Maintenance Methods
III.C.1. Requirement types III.C.3. Requirements analysis and modeling	III.A.4. Requirement analysis and specification methods (e.g., data flow diagram, entity-relationship diagram)
III.C.2. Requirements elicitation	III.A.5. Requirements elicitation methods and techniques (e.g., Quality Function Deployment, Joint Application Development, context-free questioning, needs analysis, focus groups)
III.C.4. System and Software Requirements Specifications	[New topic in BOK 2002]

<b>CSQE BOK 2002</b>	<b>CSQE BOK 1996</b>
III.D. Analysis, Design, and Development Methods and Tools	III.A. Development and Maintenance Methods
III.D.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.	III.A.6. Software design methods (e.g., structured analyses and design, Jackson Design method, Warnier-Orr method, object-oriented) - Information Domain - Structured Analysis - Warnier-Orr method - Jackson Design method - Object Oriented
III.D.2. Types of software reuse	III.A.7. Issues related to reuse, re-engineering, and reverse engineering
III.D.3. Clean room and other formal methods	III.A.3. Defect Prevention Methods – Clean room and defect prevention
III.D.3. Clean room and other formal methods III.D.4. Software development tools	III.B.5 Software environments (e.g., development methodologies, tools, data, infrastructure)
III.E. Maintenance Management	III.A. Development and Maintenance Methods
III.E.1. Maintenance types III.E.2. Operational maintenance	III.A.8. Maintenance processes (e.g., re-engineering, reverse engineering, change management, retirement)
III.E.1. Maintenance types	IV.A.4. Maintenance types (e.g., corrective, adaptive, perfective)
III.E.2. Operational maintenance	IV.A.5. Software maintenance and adaptability program planning
<b>IV. Program and Project Management (24 Questions)</b>	<b>IV. Software Project Management 16 Questions)</b>
IV.A. Planning	IV.A. Planning
IV.A.1. Project planning elements	IV.A.1. Project planning factors (e.g., quality, costs, resources, deliverables, schedules)
IV.A.2. Goal-setting and deployment	IV.A.3. Goal-setting and deployment methodologies
IV.A.3. Project planning tools	IV.A.2. Project planning methods and tools (e.g., work breakdown structures, documentation, forecasting, estimation) Estimating, WBS, Sizing V.C.3. Commonly used metrics (e.g. complexity, reliability, defect density, phase containment, size) - Boehm – Construction Cost Model (COCOMO) - Albrecht Function Points
IV.A.3. Project planning tools (e.g. planning, cost estimating) IV.B.3. Tracking elements and methods (e.g. tracking, reporting)	IV.C.1. Project management tools (e.g., planning, tracking, cost estimating, reporting)
IV.A.4. Cost and value data	[New topic in BOK 2002]

CSQE BOK 2002	CSQE BOK 1996
IV.B. Tracking and Controlling	IV.B. Tracking
IV.B.1. Phase transition control techniques IV.B.4. Project reviews	IV.B.1. Phase transitioning control techniques (e.g., reviews and audits, Gantt Charts, PERT, budgets)
IV.B.2. Interpreting and reporting cost of quality (COQ) data	IV.B.2. Methods of collecting Cost of Quality data IV.C.2. Methods of reporting Cost of Quality data
IV.B.3. Tracking elements and methods	IV.B.4. Cost, progress, and deliverable tracking (e.g., status reports, life cycle phase reports)
IV.C. Risk Management	
IV.C.1. Risk management planning methods	I.D.3. Risk management (e.g., project, product, process)
[No similar topic found in BOK 2002]	I.D.4. Problem-solving processes
IV.C.2. Risk probability	[New topic in BOK 2002]
IV.C.3. Product release decisions	IV.C.3. Trade-off involved in product release decisions (e.g., cost, quality, schedule, customer, test sufficiency, stability)
IV.C.4. Software security, safety, and hazard analysis issues	II.A.4. Issues related to software security, safety, and hazard analysis
<b>V. Software Metrics, Measurement, and Analytical Methods (24 Questions)</b>	<b>V. Software Metrics, Measurement and Analytical Methods (24 Questions)</b>
V.A. Metrics and measurement theory [No similar topic found in BOK 2002]	V.A. Measurement Theory
V.A.2. Basic measurement theory and techniques	V.A.1. Goal, question, metric paradigm for selecting metrics
V.A.1. Definitions [No similar topic found in BOK 2002]	V.A.2. Basic measurement theory and techniques
V.A.3. Psychology of metrics	V.A.3. Definitions of metrics and measures
V.B. Process and product measurement [No similar topic found in BOK 2002]	V.A.4. Designing measures
V.B.1. Process, product, and resource metrics	V.A.5. Psychology of metrics (e.g., how metrics affect people and how people affect metrics)
V.B.2. Commonly used metrics	V.C. Software Measurement
V.B.3. Software quality attributes	V.C.1. Prediction techniques of future maintainability
V.B.4. Defect detection effectiveness measures	V.C.2. Applications of measurements to process, product, and resources
	V.C.3. Commonly used metrics (e.g. complexity, reliability, defect density, phase containment, size) - Overview - Methodology - Halstead SW Science - Size – Lines of Code - DeMarco Bang
	V.C.4. Software quality attributes (e.g., reliability, maintainability, usability, testability)
	V.C.5. Defect detection effectiveness (e.g., cost yield, escapes, customer impact)

<b>CSQE BOK 2002</b>	<b>CSQE BOK 1996</b>
V.B.5. Program performance and process effectiveness	[New topic in BOK 2002]
V.C. Analytical techniques	V.B. Analytical Techniques
V.C. Analytical Techniques	I.D. Problem-Solving Tools and Processes
V.C.1. Data integrity	V.B.1. Issues involving data integrity, completeness, accuracy, and timeliness
V.C.2. Quality tools	V.B.2. Basic statistical concepts and graphical techniques for analysis and presentation of software data (e.g., distributions, confidence intervals, statistical inference) V.B.3. Quality analysis tools (Pareto chart, Flow charts, Control charts, Check sheets, Scatter diagrams, histograms)
V.C.2. Quality tools	I.D.1. Root cause analysis I.D.2. Tools (e.g., affinity diagram, tree diagram, matrix diagram, interrelationship digraph, prioritization matrix, activity network diagram)
V.C.3. Sampling theory and techniques	V.B.4. Sampling theory and techniques as applied to audits, testing, and product acceptance
<b>VI. Software Verification and Validation (V&amp;V) (24 Questions)</b>	<b>VI. Software Inspection, Testing, Verification and Validation (24 Questions)</b>
VI.A. Theory	VI.C. Verification and Validation (V & V)
VI.A.1. V&V planning procedures and tasks – (a) Identify and select various methods for verification and validation, including static analysis, structural analysis, mathematical proof, simulation, etc.	VI.C.1 V & V planning procedures
VI.A.1. V&V planning procedures and tasks – (b) Identify and analyze which tasks should be iterated as a result of proposed or completed modifications.	VI.C.10 Methods for determining which V&V tasks should be iterated based upon proposed modifications and enhancements
VI.A.2. V&V program	VI.C.2 Methods for reviewing V & V program (e.g., technical accomplishments, resource utilization, future planning, risk management, impact analysis of proposed changes)
VI.A.3. Evaluating software products and processes	VI.C.3 Methods for evaluating software life cycle products and processes (e.g., physical traces, documentation, source code, plans, test and audit results ) to determine if user needs and project objectives are satisfied
VI.A.4. Interfaces	VI.C.6 Methods for evaluating interfaces with hardware, user, operator, and other software applications
VI.B. Reviews and inspections	VI.A. Inspection

CSQE BOK 2002	CSQE BOK 1996
VI.B.1. Types	VI.A.1. Inspection types (e.g., peer reviews, inspections, walk-throughs) VI.A.4. Methods for reviewing inspection efforts (e.g., technical accomplishments, resource utilization, future planning)
VI.B.2. Items	[New topic in BOK 2002]
VI.B.3. Processes	VI.A.2. Inspection process (e.g., objectives, criteria, techniques and methods, participant roles)
VI.B.4. Data collection, reports, and summaries	VI.A.3. Inspection data collection, reports, and summaries
VI.C. Test planning and design	VI.B. Testing
VI.C.1. Types of tests	VI.B.1 Types of tests (e.g., functional, performance, usability, stress, regression, real-time response) VI.B.2 Test Levels (e.g., unit, integration, system, field)
VI.C.2. Test tools VI.C.6. Test environments	VI.B.7 Test environments (e.g., tools and methodologies, test libraries, drivers/stubs, equipment compatibility test laboratories)
VI.C.3. Test strategies	VI.B.3 Test strategies (e.g., top down, bottom up, automated testing, I/O first, beta testing, black box, white box)
VI.C.4. Test design	VI.B.4 Test design (e.g., test cases, fault insertion and error handling, equivalence class partitioning, usage scenarios, customer defect reports)
VI.C.5. Test coverage of specifications	VI.B.6 Test coverage of specifications (e.g., functions, states, data and time domains, localization, internationalization)
VI.C.7. Supplier components and products	VI.B.11 Methods for testing supplier components and products
VI.C.8. Test plans	VI.C.7 Methods for evaluating test plans (e.g., system acceptance, validation) to determine if software satisfies software and system objectives
VI.D. Test execution and evaluation	VI.B. Testing
VI.D.1. Test implementation	VI.B.9 Test management (e.g., scheduling, freezing, resources, dependencies, analysis of test results)
VI.D.2. Test documentation	VI.B.8. Test documentation (e.g., test plans, logs, test designs, defect recording, test reports)
VI.D.3. Test reviews	VI.B.10 Methods for reviewing testing efforts (e.g., technical accomplishments, resource utilization, future planning, risk management)

<b>CSQE BOK 2002</b>	<b>CSQE BOK 1996</b>
VI.D.4. Code coverage metrics	V.C.3. Commonly used metrics (e.g. complexity, reliability, defect density, phase containment, size) - McCabe's Cyclomatic Complexity VI.B.5 Test coverage of code (e.g., branch-to-branch, path, individual predicate, data)
VI.D.5. Customer deliverables	VI.B.12 Methods for testing the accuracy of customer deliverables including user documentation, marketing and training materials
VI.D.6. Severity of anomalies	VI.C.8 Methods for evaluating the severity of anomalies in software operation
<b>VII. Software Configuration Management (16 Questions)</b>	<b>VIII. Software Configuration Management (16 Questions)</b>
VII.A. Configuration infrastructure	[New topic in BOK 2002]
VII.A.1. Configuration management	[New topic in BOK 2002]
VII.A.2. Library/repository processes	VIII.A.3. Library control procedures
VII.A.3. Defect tracking and library tools	VIII.A.5. Configuration management tools
VII.B. Configuration identification	VIII.A. Planning and Configuration Identification
VII.B.1. Configuration items VII.B.2. Baselines	VIII.A.1. Technical and managerial factors that guide software product partitioning into configuration items and components
VII.B.3. Configuration identification methods VII.B.2. Baselines	VIII.A.4. Configuration identification methods (e.g., schemes, reidentification, naming conventions, versions and serialization, baselines)
VII.B.4 Software builds	VIII.B.2. Patching issues (e.g., testing, traceability, source updating)
VII.C. Configuration control	VIII.B. Configuration Control, Status Accounting, and Reporting
VII.C.1. Item and baseline control	[New topic in BOK 2002]
VII.C.2. Proposed modifications	VI.C.9 Methods for assessing all proposed modifications, enhancements, or additions to determine the effect each change will have on the system VIII.B.3. Trade-offs between cost, cycle time, and integrity of software product and rigor and formality of change control VIII.B.6. Techniques for assessing impacts of proposed software changes
VII.C.3. Review and configuration control boards (CCBs)	VIII.B.5. Software configuration/change control board processes
VII.C.4. Concurrent development	[New topic in BOK 2002]
VII.C.5. Traceability	[New topic in BOK 2002]
VII.C.6. Version control	VIII.B.4. Source and object code control procedures
VII.C.7. Configuration item interfaces	[New topic in BOK 2002]
VII.D. Configuration Status Accounting	VIII.B. Configuration Control, Status Accounting, and Reporting

<b>CSQE BOK 2002</b>	<b>CSQE BOK 1996</b>
VII.D.1. Status reporting	[New topic in BOK 2002]
VII.D.2. Changes to configuration items and baselines	[New topic in BOK 2002]
VII.D.3. Documentation control	VIII.B.1. Documentation control (e.g., issuing, approval, storage, retrieval, revision)
VII.E. Configuration Audits	[New topic in BOK 2002]
VII.E.1. Functional configuration audit VII.E.2. Physical configuration audit	VII.A.3. Functional and physical configuration audits - Functional - Physical
VII.F. Release and distribution issues	[New topic in BOK 2002]
VII.F.1. Product release process issues	VIII.A.2. Release process issues (e.g., supporting multiple versions, feature vs. corrective releases, hardware and software dependencies)
VII.F.2. Packaging, production, and distribution	[New topic in BOK 2002]

# Software Quality Engineer Certification (CSQE) References<sup>8</sup>

The technical content and test questions for the 2002 update to the CSQE BOK are derived from the listed reference materials. The list shows the references by subject area. References are repeated in each relevant area. A list of general reference material follows the last subject area.

## General Knowledge, Conduct, and Ethics

ANSI/ISO/IEE TICKIT Guidelines

Humphrey, Watts. *Managing the Software Process*, Addison-Wesley, 1989.

Juran, Joseph M., *Juran on Quality by Design: The New Steps for Planning Quality into Goods and Services*, New York: McGraw-Hill 1992.

Juran, Joseph M., *The Quality Control Handbook*, 4th ed., New York: McGraw-Hill, 1988.

Juran, Joseph M., *Juran's Quality Handbook*, 5th ed., New York: McGraw-Hill, 1999.

Kan, Stephen H., *Metrics and Models in Software Quality Engineering*, 3rd ed., Kansas: Addison-Wesley, 1995.

Kaner, Cem, Jack Falk, and Hung Quoc Nguyen, *Testing Computer Software*, 2nd ed., New York: Van Nostrand Reinhold, 1999.

Pressman, Roger S., *A Manager's Guide to Software Engineering*, New York: McGraw-Hill, 1992. ISBN 0070508208

Pressman, Roger S., *Software Engineering: A Practitioner's Approach*, 5th ed., New York: McGraw-Hill, 2000.

Russell, J.P., and Terry Regel, *After the Quality Audit: Closing the Loop on the Audit Process*, 2nd ed., Milwaukee: ASQ Quality Press, 2000.

Scholtes, Peter R., *The Team Handbook*, 2nd ed., Revised Madison, Wisconsin: Joiner Associates, 1996.

Schulmeyer, G. Gordon, and James I. McManus, *Handbook of Software Quality Assurance*, 3rd ed., Upper Saddle River, NJ: Prentice Hall, 1999.

---

<sup>8</sup> Copyright © 2002 American Society for Quality. All rights reserved.  
These books cover significant parts of the Body of Knowledge.  
The ASQ Certification Board does not endorse any one particular reference source.

## Software Quality Management

- Arter, Dennis, *Quality Audits for Improved Performance*, 2nd ed., Milwaukee: ASQC Quality Press, 1994.
- Demarco, Tom, *Controlling Software Projects: Management, Measurement, and Estimation*, NY: Yourdon Press, 1982. ISBN 0917072324.
- Dobbins, James H., *Software Quality Assurance and Evaluation* ASQC Quality Press, 1990. ISBN 0873890590
- Dunn, Robert H., *Software Quality: Concepts & Plans*, Englewood Cliffs, NJ: Prentice Hall, 1990. ISBN 0138202834
- Gryna, Frank M., *Quality Planning and Analysis: From Product Development through Use*, Boston, MA: McGraw-Hill, 2001.
- Humphrey, Watts, *Managing the Software Process*, Massachusetts: Addison-Wesley, 1989.
- Humphrey, Watts, *A Discipline for Software Engineering*, Massachusetts: Addison-Wesley, 1995. ISBN 0201546108
- Juran, Joseph M., and Frank M. Gryna, *Quality Planning and Analysis*, 3rd ed., New York: McGraw-Hill Publishing Co., 1993. ISBN 0070331839
- Juran, Joseph M., *The Quality Control Handbook*, 4th ed., New York: McGraw-Hill, 1988.
- Juran, Joseph M., *Juran's Quality Handbook*, 5th ed., New York: McGraw-Hill, 1999.
- Kan, Stephen H., *Metrics and Models in Software Quality Engineering*, 3rd ed., Kansas: Addison-Wesley, 1995.
- Kaner, Cem, Jack Falk, and Hung Quoc Nguyen, *Testing Computer Software*, 2nd ed., New York: Van Nostrand Reinhold, 1999.
- Paulk, Mark C., et al. *The Capability Maturity Model—Guidelines for Improving the Software Process*, Carnegie Mellon University Software Engineering Institute, 1995.
- Pressman, Roger S., *A Manager's Guide to Software Engineering*, New York: McGraw-Hill, 1992. ISBN 0070508208
- Pressman, Roger S., *Software Engineering: A Practitioner's Approach*, 5th ed., New York: McGraw-Hill, 2000.
- Russell, J.P., ed., ASQ Quality Audit Division, *The Quality Audit Handbook*, 2<sup>nd</sup> ed., Milwaukee: ASQ Quality Press, 2000.
- Schulmeyer, G. Gordon, and James I. McManus, *Handbook of Software Quality Assurance*, 3rd ed., Upper Saddle River, NJ: Prentice Hall, 1999. P759

## Software Engineering Processes

- Arter, Dennis, *Quality Audits for Improved Performance*, 2nd ed., Milwaukee: ASQC Quality Press, 1994.
- Booch, Grady, *Objected-Oriented Analysis and Design with Applications* 2<sup>nd</sup> ed., CA: Benjamin/Cummings Publishing Co., 1994. ISBN 0805353402
- Booch, Grady, et al., *Unified Modeling Language User's Guide Ed 1* Massachusetts: Addison-Wesley, 1999. ISBN 0201571684
- Dunn, Robert H., *Software Quality: Concepts & Plans*, Englewood Cliffs, NJ: Prentice Hall, 1990. ISBN 0138202834
- Dunn, Robert H., and Richard S. Ullman, *TQM for Computer Software (System Design and Implementation)* 2nd ed.,: McGraw-Hill, 1994. ISBN 0070183147
- Humphrey, Watts, *Managing the Software Process*, Massachusetts: Addison-Wesley, 1989.
- Humphrey, Watts, *A Discipline for Software Engineering*, Massachusetts: Addison-Wesley, 1995. ISBN 0201546108
- Juran, Joseph M., *The Quality Control Handbook*, 4th ed., New York: McGraw-Hill, 1988.
- Juran, Joseph M., *Juran's Quality Handbook*, 5th ed., New York: McGraw-Hill, 1999.
- Kan, Stephen H., *Metrics and Models in Software Quality Engineering*, 3rd ed., Kansas: Addison-Wesley, 1995.
- Kaner, Cem, Jack Falk, and Hung Quoc Nguyen, *Testing Computer Software*, 2nd ed., New York: Van Nostrand Reinhold, 1999.
- McConnell, Steve, *Rapid Development*, Washington: Microsoft Press, 1996. ISBN 1556159005
- Paulk, Mark C., et al. *The Capability Maturity Model—Guidelines for Improving the Software Process*, Carnegie Mellon University Software Engineering Institute, 1995.
- Pressman, Roger S., *A Manager's Guide to Software Engineering*, New York: McGraw-Hill, 1992. ISBN 0070508208
- Pressman, Roger S., *Software Engineering: A Practitioner's Approach*, 5th ed., New York: McGraw-Hill, 2000.
- Rumbaugh, James, et al., *Object-oriented Modeling and Design*, NJ: Prentice Hall, 1991. ISBN 0136298419

Scholtes, Peter R., *The Team Handbook*, 2nd ed., revised Madison, Wisconsin: Joiner Associates, 1996.

Schulmeyer, G. Gordon, and James I. McManus, *Handbook of Software Quality Assurance*, 3rd ed., Upper Saddle River, NJ: Prentice Hall, 1999.

Tingey, Michael, *Comparing ISO 9000 Malcolm Baldrige, and SEI CMM for Software*, Prentice Hall, 1997. ISBN 0133762602

## Program and Project Management

Booch, Grady. *Object Solutions: Managing the Object-Oriented Project*, CA: Addison-Wesley Publishing Co., Inc., 1996. ISBN 0805305947

Dunn, Robert H., *Software Quality: Concepts & Plans*, Englewood Cliffs, NJ: Prentice Hall, 1990. ISBN 0138202834

Futrell, Robert T., Donald F. Shafer, and Linda I. Shafer, *Quality Software Project Management*, New Jersey: Prentice Hall, 2002. ISBN 0130912972

Gryna, Frank M., *Quality Planning and Analysis: From Product Development through Use*, Boston, MA: McGraw-Hill, 2001.

Humphrey, Watts. *Managing the Software Process*, Massachusetts: Addison-Wesley, 1989.

Juran, Joseph M., and Frank M. Gryna, *Quality Planning and Analysis*, 3rd ed., New York: McGraw-Hill Publishing Co., 1993. ISBN 0070331839

Juran, Joseph M., *The Quality Control Handbook*, 4th ed., New York: McGraw-Hill, 1988.

Juran, Joseph M., *Juran's Quality Handbook*, 5th ed., New York: McGraw-Hill, 1999.

Kan, Stephen H., *Metrics and Models in Software Quality Engineering*, 3rd ed., Kansas: Addison-Wesley, 1995.

Kaner, Cem, Jack Falk, and Hung Quoc Nguyen, *Testing Computer Software*, 2nd ed., New York: Van Nostrand Reinhold, 1999.

King, David, *Project Management Made Simple: A Guide to Successful Management of Computer Systems Projects*, New Jersey: Yourdon Press, 1992. ISBN 0137177291

Paulk, Mark C., et al. *The Capability Maturity Model—Guidelines for Improving the Software Process*, Carnegie Mellon University Software Engineering Institute, 1995.

Pressman, Roger S., *A Manager's Guide to Software Engineering*, New York: McGraw-Hill, 1992. ISBN 0070508208

Pressman, Roger S., *Software Engineering: A Practitioner's Approach*, 5th ed., New York: McGraw-Hill, 2000.

Schulmeyer, G. Gordon, and James I. McManus, *Handbook of Software Quality Assurance*, 3rd ed., Upper Saddle River, NJ: Prentice Hall, 1999.

## Software Metrics, Measurement, and Analytical Methods

ANSI/ISO/IEE TICKIT Guidelines

Dunn, Robert H., *Software Quality: Concepts & Plans*, Englewood Cliffs, NJ: Prentice Hall, 1990. ISBN 0138202834

Dunn, Robert H., *Software Defect Removal*, NY: McGraw-Hill, 1984. ISBN 0070183139

Dunn, Robert H., and Richard S Ullman, *TQM for Computer Software (System Design and Implementation)* 2nd ed., McGraw Hill, 1994. ISBN 0070183147

Humphrey, Watts. *Managing the Software Process*, Massachusetts: Addison-Wesley, 1989.

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

Juran, Joseph M., *The Quality Control Handbook*, 4th ed., New York: McGraw-Hill, 1988.

Juran, Joseph M., *Juran's Quality Handbook*, 5th ed., New York: McGraw-Hill, 1999.

Kan, Stephen H., *Metrics and Models in Software Quality Engineering*, 3rd ed., Kansas: Addison-Wesley, 1995.

Pressman, Roger S., *A Manager's Guide to Software Engineering*, New York: McGraw-Hill, 1992. ISBN 0070508208

Pressman, Roger S., *Software Engineering: A Practitioner's Approach*, 5th ed., New York: McGraw-Hill, 2000.

Scholtes, Peter R., *The Team Handbook*, 2nd ed., Revised Madison, Wisconsin: Joiner Associates, 1996.

Schulmeyer, G. Gordon, and James I. McManus, *Handbook of Software Quality Assurance*, 3rd ed., Upper Saddle River, NJ: Prentice Hall, 1999.

## Software Verification and Validation

Dobbins, James H., Software Quality Assurance and Evaluation ASQC Quality Press, 1990.  
ISBN 0873890590

Dunn, Robert H., Software Defect Removal, NY: McGraw-Hill, 1984. ISBN 0070183139

Dunn, Robert H., and Richard S Ullman, TQM for Computer Software (System Design and Implementation) 2nd ed., McGraw Hill, 1994. ISBN 0070183147

Humphrey, Watts. Managing the Software Process, Massachusetts: Addison-Wesley, 1989.

Juran, Joseph M., The Quality Control Handbook, 4th ed., New York: McGraw-Hill, 1988.

Juran, Joseph M., Juran's Quality Handbook, 5th ed., New York: McGraw-Hill, 1999.

Kan, Stephen H., Metrics and Models in Software Quality Engineering, 3rd ed., Kansas: Addison-Wesley, 1995.

Kaner, Cem, Jack Falk, and Hung Quoc Nguyen, Testing Computer Software, 2nd ed., New York: Van Nostrand Reinhold, 1999.

Myers, Glenford J., The Art of Software Testing, 1st ed., New York: John Wiley & Sons, 1979. ISBN 0471043281

Myers, Glenford J., Software Reliability: Principles and Practices, NY: Wiley, 1976. ISBN 0471627658

Pressman, Roger S., A Manager's Guide to Software Engineering, New York: McGraw-Hill, 1992. ISBN 007-0508208

Pressman, Roger S., Software Engineering: A Practitioner's Approach, 5th ed., New York: McGraw-Hill, 2000.

Schulmeyer, G. Gordon, and James I. McManus, Handbook of Software Quality Assurance, 3rd ed., Upper Saddle River, NJ: Prentice Hall, 1999.

## Software Configuration Management

### ANSI/ISO/IEE TICKIT Guidelines

Dobbins, James H., Software Quality Assurance and Evaluation ASQC Quality Press, 1990.  
ISBN 0873890590

Dunn, Robert H., Software Quality: Concepts & Plans, Englewood Cliffs, NJ: Prentice Hall,  
1990. ISBN 0138202834

Dunn, Robert H., Software Defect Removal, NY: McGraw-Hill, 1984. ISBN 0070183139

Dunn, Robert H., and Richard S Ullman, TQM for Computer Software (System Design and  
Implementation) 2nd ed., McGraw Hill, 1994. ISBN 0070183147

Humphrey, Watts. Managing the Software Process, Massachusetts: Addison-Wesley, 1989.

Juran, Joseph M., The Quality Control Handbook, 4th ed., New York: McGraw-Hill, 1988.

Juran, Joseph M., Juran's Quality Handbook, 5th ed., New York: McGraw-Hill, 1999.

Kan, Stephen H., Metrics and Models in Software Quality Engineering, 3rd ed., Kansas:  
Addison-Wesley, 1995.

Osborne, Wilma M., Software Configuration Management: An Overview, Gaithersburg, MD:  
US Dept. of Commerce, National Computer Systems Lab, NIST Special Publication,  
500-161, 1989.

Paulk, Mark C., et al. The Capability Maturity Model—Guidelines for Improving the  
Software Process, Carnegie Mellon University Software Engineering Institute, 1995.

Pressman, Roger S., A Manager's Guide to Software Engineering, New York: McGraw-Hill,  
1992. ISBN 0070508208

Pressman, Roger S., Software Engineering: A Practitioner's Approach, 5th ed., New York:  
McGraw-Hill, 2000.

Schulmeyer, G. Gordon, and James I. McManus, Handbook of Software Quality Assurance,  
3rd ed., Upper Saddle River, NJ: Prentice Hall, 1999.

## General References

Brooks, Frederick P. Jr., *The Mythical Man-Month: Essays on Software Engineering*, Massachusetts: Addison-Wesley Publishing Co., 1975. ISBN 0201006502

Daughtrey, Taz, ed., *Fundamental Concepts for the Software Quality Engineer*, ASQ Quality Press, 2002. H1115

DeMarco, Tom, and Timothy Lister, *Peopleware: Productive Projects and Teams*, NY: Dorset House Publishing Co, 1977. ISBN 0932633056

Escoe, Adrienne, *The Practical Guide to People-Friendly Documentation*, ASQ Quality Press, 2001.

Lockheed Martin Advanced Concepts Center, *Succeeding with the Booch and OMT Methods: A Practical Approach*, CA: Addison-Wesley, 1996. ISBN 0805322795

Rumbaugh, James, Ivar Jacobson, and Grady Booch, eds., *The Unified Modeling Language Reference Manual*, CA: Addison-Wesley, 1999. ISBN 020130998X

Yourdon, Edward, *Decline and Fall of the American Programmer*, New Jersey: Prentice-Hall, Inc., 1992. ISBN 0132036703

Yourdon, Edward, *Rise and Resurrection of the American Programmer*, New Jersey: Prentice-Hall, Inc., 1998. ISBN 013956160