1 : Introduction to Software Testing

IT6206 – Software Quality Assurance

Level III - Semester 6

Overview

- In this section, you will be introduced to the fundamental concepts of Software Testing.
- As you go through this section, you will encounter the terms **Software, Software Quality, Testing, Software Quality Assurance, Software Quality Control** etc.
- You will be learning about the objectives of Software Quality Assurance and Testing, Testing principles, Software Quality Assurance models and standards, and the differences between manual and automation testings.

Intended Learning Outcomes

At the end of this lesson, you will be able to;

- Define software, software quality, and software quality assurance.
- Distinguish between quality assurance and testing.
- Distinguish between quality control and quality assurance.
- Explain the objectives of software quality assurance activities.
- Distinguish between testing and debugging.
- Distinguish between software errors, faults, and failures.
- Identify various causes for software errors.
- Identify various testing principles.
- Identify the advantages of using software quality assurance standards.

List of sub topics

1.1 Introduction to Software Quality Assurance and Testing
- 1.1.1 What is Software?
- 1.1.2 What is Software Quality?
- 1.1.3 Quality Assurance vs. Testing
- 1.1.4 What is Software Quality Assurance?
- 1.1.5 Software Quality Assurance vs. Software Quality Control

1.2 Typical Objectives of Software Quality Assurance and Testing
- 1.2.1 Objectives of Software Quality Assurance in Development
- 1.2.2 Objectives of Software Quality Assurance in Maintenance
- 1.2.3 Objectives of Software Quality Assurance in Testing

1.3 Testing and Debugging
- 1.3.1 Is Testing and Debugging the same?
- 1.3.2 Testing in different software development stages

List of sub topics

1.4 Errors, Faults, and Failure
- 1.4.1 What are software errors, faults, and failures?

1.5 Defects, Root Causes and Effects
- 1.5.1 Root cause of a defect
- 1.5.2 Causes for software errors
  - 1.5.2.1 Faulty definition of requirements
  - 1.5.2.2 Client-developer communication failures
  - 1.5.2.3 Deliberate deviations from software requirements
  - 1.5.2.4 Logical design errors
  - 1.5.2.5 Coding errors
  - 1.5.2.6 Non-compliance with documentation and coding instructions
  - 1.5.2.7 Shortcomings of the testing process
  - 1.5.2.8 Procedure errors
  - 1.5.2.9 Documentation errors

List of sub topics

- 1.6 Manual vs. Automation Testing
  - 1.6.1 Manual Testing vs. Automation Testing
  - 1.6.2 Advantages and Disadvantages of Automation Testing
- 1.7 Seven Testing principles
- 1.8 Software Quality Assurance Models and Standards
  - 1.8.1 Software Quality Assurance Standards
    - 1.8.1.1 ISO 9000-3
    - 1.8.1.2 Capability Maturity Model
    - 1.8.1.3 Bootstrap Model
  - 1.8.2 Benefits of using Software Quality Assurance Standards

1.1 Introduction to Software Quality Assurance and Testing

1.1.1 What is Software?

- Software is any set of machine-readable instructions to the computer's processor to perform specific operations.
- According to the IEEE definition(IEEE, 1991), Software is computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system.
- According to ISO definition (ISO, 1997), software includes four basic components: computer programs, procedures, documentation, and data necessary for operating the software system.
- All these four components are essential in order to assure the quality of a software development process.

1.1.2 What is Software Quality?

- According to IEEE (IEEE, 1991), software quality has two alternative definitions as follows.
- Software quality is:
  1. The degree to which a system, component, or process meets specified requirements.
  2. The degree to which a system, component, or process meets customer or user needs or expectations.

- The first alternative definition refers to the degree to which a written software meets the specification prepared by the customer and his/her professional team.
- The second alternative definition aims at achieving customer satisfaction and views the fulfilment of customers' real needs as the goal of software quality.

1.1.3 Quality Assurance vs. Testing

- Is quality assurance (QA) the same as testing? The answer is No!
- Quality assurance is one part of a larger concept called quality management. Quality assurance affects not only software development but also human resources, delivery process etc.
- Quality assurance is associated with ensuring that a company's standard ways of performing various tasks are carried out correctly. If the quality assurance process is carried out correctly, then the resulting products will be of higher quality.
- Testing is more of a quality control activity.
- Testing plays an essential supporting role in delivering quality software. However, testing by itself is not sufficient. Testing should be integrated into a complete, team-wide and development process-wide set of activities for quality assurance.
- Quality assurance supports good testing.

1.1.4 What is Software Quality Assurance?

- IEEE Glossary(IEEE, 1991), provides the following definition regarding Software Quality Assurance(SQA).
- Software quality assurance is:
  1. A planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements.
  2. A set of activities designed to evaluate the process by which the products are developed or manufactured.
- According to the above definition, SQA is based on planning and the application of a variety of actions that are integrated into all the stages of the software development process.
- However, SQA should not be limited to only the development process. It should be extended to cover the long years of service and software delivery.
- Also, SQA should not be limited to technical aspects of the functional requirements, but include activities that deal with scheduling and the budget as well.

1.1.4 What is Software Quality Assurance?

- A broader definition considering all the aforementioned factors would be as follows [2].

> "Software Quality Assurance is a systematic, planned set of actions necessary to provide adequate confidence that the software development process or the maintenance process of a software system conforms to established functional requirements as well as with managerial requirements of keeping the schedule and operating within the budgetary confines."

1.1.5 Software Quality Assurance vs. Software Quality Control

- **Quality Control**: A set of activities designed to evaluate the quality of a developed or manufactured product. The main objective of quality control is the withholding of any product that does not qualify.
- **Quality Assurance**: A set of activities which detect and prevent the causes of errors, and correct errors early in the development process. The main objective is to reduce the cost of guaranteeing quality in a product.
- Quality control and quality assurance activities serve different objectives.
- Quality control activities are only a part of the total range of quality assurance activities.

1.2 Typical Objectives of Software Quality Assurance and Testing

1.2.1 Objectives of Software Quality Assurance in Development

- Assuring that the software will satisfy the functional and technical requirements.
- Assuring that the software will satisfy the managerial scheduling and budgetary requirements.
- Initiating and managing activities for the improvement and efficiency of the software development process and software quality assurance activities.

1.2.2 Objectives of Software Quality Assurance in Maintenance

- Assuring that the software maintenance activities are satisfying the functional and technical requirements of the system.
- Assuring that the software maintenance activities are satisfying the managerial scheduling and budgetary requirements.
- Initiating and managing activities to improve and increase the efficiency of software maintenance and software quality assurance activities.

1.2.3 Objectives of Testing

- To verify whether all the specified requirements have been fulfilled.
- To find failures and defects. (This is typically a primary focus for software testing).
- To provide sufficient information to stakeholders to allow them to make informed decisions, especially regarding the level of quality of a test object.
- To reduce the level of risk of inadequate software quality.
- To comply with contractual, legal or regulatory requirements of standards, and/or to verify the test objects' compliance with such requirements or standards.

1.3 Testing and Debugging

1.3.1 Is Testing and Debugging the same?

- Testing and debussing are two different concepts.
- **Debugging** is a software development activity where a developer finds a defect, analyzes it and removes it.
- **Testing** is often used to detect system defects and the failures which are caused by the defects.
- After debugging further tests have to be applied to ensure that the system is functioning without any failures.
- In some cases, software testers are conducting initial and final confirmation tests, while the developers work on debugging.
- In software development life cycles such as Agile, testers are participating in debugging as well.

1.3.2 Testing in different software development stages

The below table displays some examples where testing could contribute to a successful system in different software development phases.

Software Development phase where testing is applied	Advantage
Requirement Reviews, Refinement of user stories.	• Can detect defects before any designing or coding activities. • Reduces the risk of an incorrect or untestable software being developed.
While the system is being designed.	• Increase system stakeholders' understanding of the system design. • Reduce the risk of design defects.

1.3.2 Testing in different software development stages cont.

Software Development phase where testing is applied	Advantage
While coding the system.	• Increase each parties understanding of the code and how it should be tested. • Reduce the risks of defects in the code.
Prior to the release.	• Can detect failures that might otherwise have been missed. • Ensure that the developed software satisfies the stakeholders' requirements.

1.4 Errors, Faults, and Failure

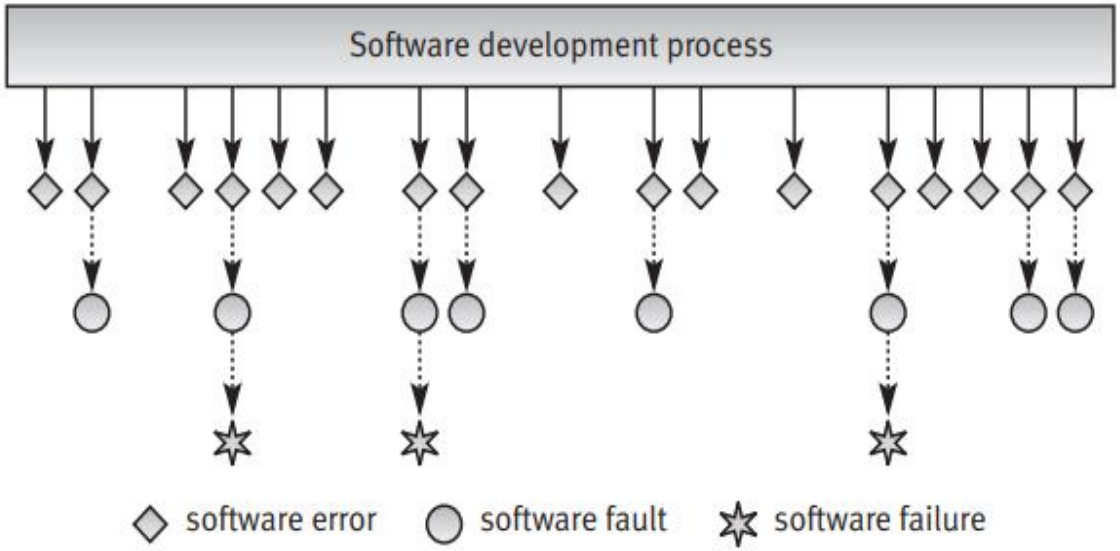
1.4.1 What are software errors, faults, and failures?

- Software engineering is a human-intensive activity. Sometimes software developers make **errors**. An error can be a grammatical error in one or more code lines or a logical error in carrying out one or more of the clients' requirements.
- Some software errors become software **faults** (Software defects) causing improper functioning in the system or a part of the system. Some software errors will be corrected or neutralized by subsequent code lines.
- Not all software faults become software failures. A software fault becomes a software **failure** when a software user tries to apply the specific faulty application. When a program is executed, if the right conditions exist, a software fault may result in unexpected behaviour. This is known as a software failure.

1.4.1 Errors, faults, and failures cont.

- Only a portion of software errors become software faults (defects).
- Only a portion of software faults become software failures.

The following figure displays the relationship between software errors, faults, and failures. Here, the development process created 17 errors, but only 8 of those became software faults. Out of the faults, only three became software failures.



Ref [2] : page 18

Software Quality Assurance; From theory to implementation, Daniel Galin, Pearson Education Limited, 2004, ISBN 0201 70945 7

1.5 Defects, Root Causes, and Effects

1.5.1 Root cause of a defect

- Root cause is a source of a defect such that if it is removed, the occurrence of the defect type is decreased or removed.
- Root causes are the earliest actions or conditions that cause the defects.
- Once a defect is found on an object, a root cause analysis should be conducted to avoid having similar defects in future systems.
- Generally, a root cause is an organizational issue.

1.5.2 Causes for software errors

- It is important to identify the causes of software errors in order to avoid poor-quality software.
- A software error could be found in any phase of the software development life cycle and could be made by any stakeholder associated with the system.
- Software errors may occur due to many reasons such as,
  - Time pressure
  - Human fallibility
  - Inexperienced or insufficiently skilled project participants
  - Miscommunication between stakeholders
  - Complexity of the code, design, underlying problem etc.
  - Misunderstandings of intra-system and inter-system interfaces.
  - New or unfamiliar technologies, or tools to be used.

1.5.2 Causes for software errors cont.

- A software error could be a Code error, Procedure error, Documentation error, or Software data error.
- D. Galin[2] classifies software errors into nine categories according to the stages of the software development process in which they occur. They are as follows.

1. Faulty requirements definition
2. Client-developer communication failures
3. Deliberate derivations from software requirements
4. Logical design errors
5. Coding errors
6. Non-compliance with documentation and coding instructions
7. Shortcomings of the testing process
8. Procedure errors
9. Documentation errors

1.5.2.1 Faulty Requirements Definition

- Most common errors that belong to this type are:
  - Erroneous definitions of requirements
  - Absence of key requirements
  - Incomplete requirement definitions
  - Including unnecessary requirements

1.5.2.2 Client-developer communication failures

- Client-developer communication failures cause,
  - Misunderstandings of client's instructions
  - Misunderstandings of client's requirement changes
  - Misunderstandings of client's responses to design problems
  - Lack of attention to client messages regarding requirement changes
- Errors that occur due to this cause usually prevail in the early stages of the software development process. But it can affect the whole development process.

1.5.2.3 Deliberate deviations from software requirements

- Sometimes the developers deliberately deviate from the documented requirements. This could cause software errors. Some common situations of deliberate deviations are:
  - Developers reusing modules that are taken from different projects without sufficient analysis of the requirements.
  - Developers decide to omit some parts of required functionalities due to time or budget pressures.
  - Developers introducing software improvements without the client's approval.

1.5.2.4 Logical Design Errors

- Software errors can occur in the designing phase as well. Typical errors include:
  - Software requirement definitions with erroneous algorithms.
  - Software process definitions with sequencing errors.
  - Erroneous definitions of the software process.
  - Omission of required system states.

1.5.2.5 Coding Errors

- Coding errors can occur due to reasons such as:
  - Misunderstandings in software design.
  - Linguistic errors in programming languages.
  - Errors in application of Computer-Aided Software Engineering(CASE) tools, and other development tools.
  - Errors in data selection.

1.5.2.4 Logical Design Errors

- Software errors can occur in the designing phase as well. Typical errors include:
  - Software requirement definitions with erroneous algorithms.
  - Software process definitions with sequencing errors.
  - Erroneous definitions of the software process.
  - Omission of required system states.

1.5.2.5 Coding Errors

- Coding errors can occur due to reasons such as:
  - Misunderstandings in software design.
  - Linguistic errors in programming languages.
  - Errors in application of Computer-Aided Software Engineering(CASE) tools, and other development tools.
  - Errors in data selection.

1.5.2.6 Non-compliance with documentation and coding instructions

- Almost every development unit has its own documentation and coding standards with regard to the codes and the format of the documents that are created as a part of the software development process.
- Non-compliance with the coding and documentation instructions can cause issues in designing, implementation, testing, and even in the maintenance stages of software development.

1.5.2.7 Shortcomings of the testing process

- Shortcomings of the testing process result from causes such as:
  - Incomplete test plans
  - Failures in documenting and reporting errors and faults.
  - Failures in correcting detected errors.
  - Incomplete correction of detected errors due to negligence or time pressures.

1.5.2.8 Procedure Errors

- Procedures direct the users with respect to the activities required to be completed at each step of a process.
- Procedures are especially important in complex software systems where processes are conducted as a series of steps.
- Procedure errors could cause misinterpreted system requirements.

1.5.2.9 Documentation Errors

- Documentation errors are found in design documents and in documentation integrated into the body of the software.
- These errors can cause additional errors in the implementation and maintenance stages.
- Documentation errors in user manuals affect the system users in identifying system functionalities properly and obtaining the optimal user experience.

1.6 Manual testing vs. Automation Testing

1.6.1 Manual testing vs. Automation Testing

- **Manual testing** is a process with human intervention. Here a QA engineer or a QA analyst executes tests as a step-by-step process without using test scripts.
- **Automation testing** is a process where the system testers utilize special tools and scripts to automate the testing tasks. In automation testing, computerized tools are integrated into the process of software development. Automation tests result in cost savings, shortened test durations, improvements in test accuracy, and improvements in testing efficiency.
- The main types of automated tests are code auditing, coverage monitoring, functional tests, load tests, and test management.
- More details related to Automation testing will be discussed in the latter part of the course.

1.6.2 Advantages and Disadvantages of Automation Tests

The below table displays the advantages and disadvantages of applying automation testing in the software development process.

Advantages	Disadvantages
• Accuracy and completeness of performance • Higher accuracy in documentation • Comprehensive information • Lesser amount of manpower resources in test executions • Shorted testing periods	• High investments required in the package purchasing and training • High package development investment costs • High manpower resources for test preparation • Some testing areas can be left uncovered

1.7 Seven Testing Principles

1.7 Seven Testing principles

The following seven testing principles provide general guidelines common for all testing.

Principle 1: Testing shows the presence of defects, not their absence.

- Testing can not prove that there are no defects. It can only show that defects are present. Testing reduces the probability of undiscovered defects in software.

Principle 2: Exhaustive testing is impossible

- Testing all combinations of inputs and preconditions is not feasible. Instead of testing exhaustively, risk analysis, test techniques, and test priorities should be considered further.

1.7 Seven Testing principles cont.

Principle 3: Early testing saves time and money

- Testing early in the software development life cycle helps reduce or eliminate costly changes.

Principle 4: Defects cluster together

- Usually, a small number of modules contain the majority of the defects discovered. These defect clusters are responsible for most of the operational failures.

Principle 5: Beware of the pesticide paradox

- If the same tests are repeated again and again, eventually these tests lose the ability to find new errors. Therefore, in order to find new errors, tests and test data need to keep changing and new tests should be written.

1.7 Seven Testing principles cont.

Principle 6: Testing is context dependent

- Testing should be done differently in different contexts by considering the properties of the underlying system.

Principle 7: Absence of errors is a fallacy

- Since exhaustive testing is impossible(principle 2), and testing shows the presence of defects, not their absence(principle 1), running all the tests and finding all the defects is a fallacy. Also, just finding and fixing a large number of defects does not ensure a successful system.

1.8 Software Quality Assurance models and standards

1.8.1 Software Quality Assurance Standards

1.8.1.1 ISO 9000-3

- ISO 9000-3 guidelines are offered by the International organization for Standardization (ISO). It mainly represents the implementation of the special case of software development and maintenance.
- ISO 9000-3 standard emphasizes eight principles related to quality management. They are,
  - Customer Focus
  - Leadership
  - Involvement of people
  - Process approach
  - System approach to management
  - Continual improvement
  - Factual approach to decision making
  - Mutually supportive supplier relationships.

1.8.1 Software Quality Assurance Standards cont.

- Organizations wishing to obtain ISO 9000-3 certification are required to complete the following.
  1. Develop the organization's SQA system
     - By developing quality manuals and a comprehensive set of SQA procedures.
     - By staff training, good documentation and quality records etc.
  2. Implement the organization's SQA system
     - By setting up staff instruction programs and support services for efficient problem-solving in implementing SQA tools.
  3. Undergo certification audits
     - By reviewing the quality manuals and SQA procedures developed by the organization.
     - By verifying the requirements specified by the organizations in their quality manuals and SQA procedures.

1.8.1 Software Quality Assurance Standards cont.

- The following figure displays the key steps in ISO 9000-3 certification process.

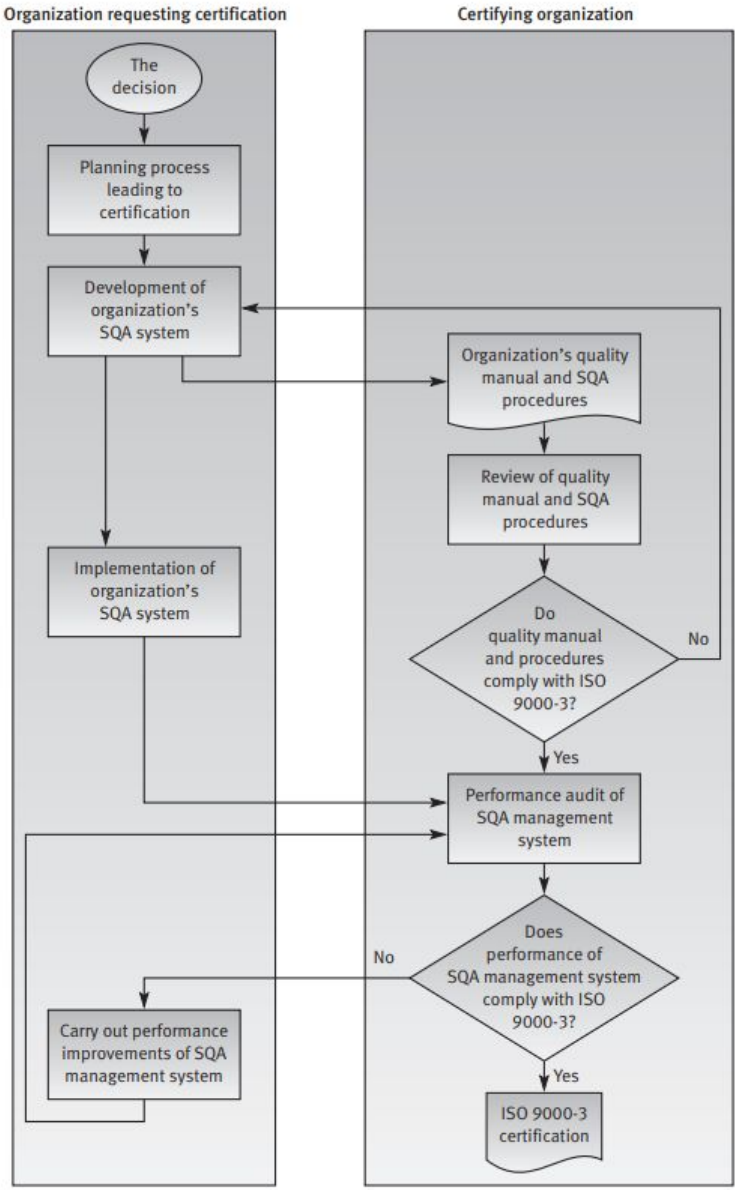


Figure 23.1: The ISO 9000-3 certification process

Ref [2] : Page 482

Software Quality Assurance; From theory to implementation, Daniel Galin, Pearson Education Limited, 2004, ISBN 0201 70945 7

1.8.1 Software Quality Assurance Standards

1.8.1.2 Capability Maturity Models

- The Capability maturity Model (CMM) is an initiative suggested by Carnegie Mellon University in 1986 which mainly focuses on developing and refining an organization's software development process.
- Capability maturity Model Integration (CMMI) is an updated model of CMM, which focuses on adding agile principles to CMM to improve development processes, software configuration management, and software quality management.

1.8.1.3 Bootstrap Model

- The Bootstrap methodology measures the maturity of an organization and its projects on the basis of 31 quality attributes categorized into three classes; process, organization, and technology.

1.8.2 Benefits of using Software Quality Assurance Standards

- Ability to make use of the most sophisticated and comprehensive methodologies and procedures in software quality assurance.
- Better understanding and cooperation between the system users such as software developers, customers, and other stakeholders
- Ability to obtain SQA certifications based on independent professional quality audits.

Summary

- **Software quality assurance** is:
  1. A planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements.
  2. A set of activities designed to evaluate the process by which the products are developed or manufactured.

- **Quality control** is a set of activities carried out to withhold products which do not qualify with pre-defined quality constraints. **Quality assurance** minimizes the costs of quality by introducing a variety of activities throughout the software development life cycle.

Summary

- Debugging and testing are two different concepts. **Debugging** is a software development activity where a developer finds a defect, analyzes it and removes it. **Testing** is often used to detect system defects and the failures which are caused by the defects.
- Software errors, faults, and failures are three different concepts. Only a portion of software **errors** become software **faults** (defects). Only a portion of software faults become software **failures.**
- D. Galin [2] classifies software errors into nine categories according to the stages of the software development process in which they occur.
- Several SQA standards have been suggested to ensure the application of the most sophisticated and comprehensive methodologies and procedures in software quality assurance and promote better understanding among system stakeholders.

References

[1] Foundations of Software Testing ISTQB Certification, Rex Black, Erik van Veenendaal, Dorothy Graham, ISBN 978-1473764798

[2] Software Quality Assurance; From theory to implementation, Daniel Galin, Pearson Education Limited, 2004, ISBN 0201 70945 7

[3] IEEE (1991) "IEEE Std 610.12-1990 – IEEE Standard Glossary of Software Engineering Terminology", Corrected Edition, February 1991, in IEEE Software Engineering Standards Collection, The Institute of Electrical and Electronics Engineers, New York.

[4] ISO (1997) ISO 9000-3:1997(E), Quality Management and Quality Assurance Standards – Part 3: Guidelines for the Application of ISO 9001:1994 to the Development, Supply, Installation and Maintenance of Computer Software, 2nd edn. International Organization for Standardization (ISO), Geneva.