Chapter 7.
Software Quality Assurance

7.1 Introduction

- One definition of quality is that ‘‘quality is the totality of features and characteristics of a product or service which bear on its ability to satisfy a given need’’ (British Standards Institution).
- Another definition is that quality software is software that does what it is supposed to do.

- The main technique for achieving quality is the software review or walkthrough.
- The goal of inspections is to find errors. The metric used most often to evaluate inspections is errors-found/KLOC. The efficiency may be measured in terms of errors-found/hour-spent.

- Software Quality
  - Product Quality
  - Process Quality

Software Product Quality

- Most recently the international standard ISO-9126 Software Product Evaluation Characteristics (1991) has been put forward as a high-level framework for characterizing software product quality.

- The ISO 9126-1 software quality model identifies 6 main quality characteristics, namely:
  Functionality
  Reliability
  Usability
  Efficiency
  Maintainability
  Portability

Software Product Quality

- Functionality : A set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs.
  - Suitability
  - Accuracy
  - Interoperability
  - Security
  - Functionality compliance

- Reliability : A set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time.
  - Maturity,
  - Fault tolerance
  - Recoverability
  - Reliability compliance

Software Product Quality

- Usability : A set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.
  - Understandability
  - Learnability
  - Operability
  - Attractiveness
  - Usability compliance

- Efficiency : A set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions.
  - Time behavior
  - Resource utilization
  - Efficiency compliance

Software Product Quality

- Maintainability : A set of attributes that bear on the effort needed to make specified modifications.
  - Analyzability
  - Changeability
  - Stability
  - Testability
  - Maintainability compliance

- Portability : A set of attributes that bear on the ability of software to be transferred from one environment to another.
  - Adaptability
  - Installability
  - Replaceability
  - Coexistence
  - Portability compliance

Capability Maturity Model

- The Software Engineering Institute (www.sei.cmu.edu) has developed the Capability Maturity Models. The Software Engineering Capability Maturity Model (SE-CMM) is used to rate an organization's software development process. An assessment of an organization's practices, processes, and organization is used to classify an organization at one of the following levels:
  - Level 1: Initial—This is the lowest level and usually characterized as chaotic.
  - Level 2: Repeatable—This level of development capability includes project tracking of costs, schedule, and functionality. The capability exists to repeat earlier successes.
  - Level 3: Defined—This level has a defined software process that is documented and standardized. All development is accomplished using the standard processes.
  - Level 4: Managed—This level quantitatively manages both the process and the products.
  - Level 5: Optimizing—This level uses the quantitative information to continuously improve and manage the software process.

7.2 Formal Inspections and Technical Reviews

- A formal inspection is a formal, scheduled activity where a designer presents material about a design and a selected group of peers evaluates the technical aspects of the design.
- The details of how a formal inspection or technical review is done can vary widely.
- The following aspects are usually accepted as what distinguishes a formal inspection from other reviews:
    - Knowledgeable peers are used.
    - The producer is an active participant.
    - An explicit, completed product is inspected.
    - The primary purpose is to find defects.
    - A formal inspection is used routinely in software development.
    - Specific roles are assigned.
    - The inspection uses the specific steps of formal inspections.
    - At least three people are involved in the inspection.

7.2.1 INSPECTION ROLES

- Although there are variations, the following are the basic roles that most inspections use:

  Moderator—The moderator selects the team, conducts the inspection, and reports the results.

  Reader—The reader is often not the producer of the product; however, the reader will guide the team through the work product during the inspection meeting.

  Recorder—The recorder maintains the records of the inspection and accurately reports each defect.

  Producer—The producer is the one who originally produced the product. His or her role is to answer questions during the inspection. The producer is also responsible for correcting any problems identified in the inspection. He or she then reports the corrections to the moderator.

7.2.2 INSPECTION STEPS

- Following are the basic steps in an inspection:
  1. **Overview**—When the producer satisfies the entrance criteria, the inspection is scheduled. The producer then conducts an overview. It acquaints the rest of the inspection team with the product to be inspected.
  2. **Preparation**—The inspection team members study the product. The time spent in preparing is controlled based on the size of the product in KLOC. The members may use a checklist to focus on significant issues.
  3. **Inspection meeting**—The moderator supervises the inspection meeting. Some approaches use a reader other than the producer to actually conduct the inspection. The recorder makes a complete record of issues raised. All members of the inspection team sign the report. Any team member may produce a minority report if there is a disagreement.
  4. **Rework**—The producer reviews the report and corrects the product.
  5. **Follow-up**—The moderator reviews the report and the correction. If it satisfies the exit criteria, the inspection is completed. If not, the moderator can either have the producer rework the product or reinspection can be scheduled.

7.2.3 CHECKLISTS

- A checklist is a list of items that should be checked during the review. Sometimes the items are expressed as questions to be answered.
- The value of a checklist is that it focuses the attention of the reviewer on potential problems. Every fault that is found should be analyzed to see if it warrants a checklist item to focus on that problem.
- Checklist items that are not effective in finding faults during inspections should be considered for removal. Too many checklist items will lessen the effectiveness of the inspection.

7.3 Software Reliability

- Reliability is the probability of not failing in a specified length of time. This is usually denoted by R(n), where n is the number of time units. If the time unit is days, then R(1) is the probability of not failing in1 day.
- The probability of failing in a specified length of time is 1 minus the reliability for that length of time

  (F(n) = 1-R(n)).
- Software reliability is a measure of how often the software encounters a data input or other condition that it does not process correctly to produce the right answer. Software reliability is not concerned with software wearing out.

7.3.1 ERROR RATES

- If an error happens every 2 days, then the instantaneous error rate would be 0.5 errors per day. The error rate is the inverse of the time between errors (inter-error time). The error rate can be used as an estimate of the probability of failure, F(1). Unless we know some trend, the best estimate of the short-term future behavior is the current behavior.

EXAMPLE 7.1

If an error happens after 2 days, what is the probability that the system will not fail in 1, 2, 3, and 4 days?

If an error happens every 2 days, we can use 0.5 as the instantaneous error rate. It can also be used to estimate the failure probability for 1 day. Thus, F(1) = 0.5. Then, R(1)= 1 – F(1) = 0.5. R(2) = 0.25. R(3) = 0.125. R(4) = 0.0625.

7.3 Software Reliability

- If we can see a trend in the error rates, then we can estimate the error rate better. Instead of using equations to fit the data, plots of the failure rate can be used to visualize the behavior.
- If x is the inter-failure time, 1/ x is the instantaneous failure rate. Plot the instantaneous failure rate versus either failure number or the elapsed time of the failure. Try to fit a straight line to the points. The value of the line at the current time can be used for the error rate.
- The intersection of this line with the horizontal axis indicates either the fault number where the failure rate goes to zero, or the amount of time necessary to remove all the faults. When the x -axis is the elapsed time, then the area under the straight line (units are time*failure/time) represents the number of faults.
- Thus, empirical data about how often the software fails during testing or observation is used to estimate the current rate.

7.3.2 PROBABILITY THEORY

- F(1) is the probability of failing on the next execution. It is equal to theta, the percentage of test cases that fail. The failure probability can be estimated by the current instantaneous error rate or by the estimated error rate from the error rate plots.
- If we know R(1), then the probability that we can execute n test cases without failure is $R(n) = R(1)^n$.
- Note that $F(n)$ is not $F(1)^n$. $F(n) = 1 - (1 - F(1))^n$.

7.4 Statistical Quality Assurance

- Statistical quality assurance(SQA) is the use of statistics to estimate the quality of software. Executing the code with a small set of randomly chosen test cases will give results that can be used for estimating the quality. This is sometimes called a software probe. The error rate on the randomly chosen sample can be used as an estimate of the error rate in the finished project.
- If the percentage of correct executions is high, then the development is going well. If the percentage of correct executions is low, then remedial action may be appropriate for the development process.

7.5 IEEE Standards for SQA Plan

- An important part of achieving quality is to plan for quality, that is, to plan those activities that will help to achieve quality. The IEEE Standards Association has developed a standard (Std 730-1989) for software quality assurance plans.
- The following is part of the sections specified in IEEE Std 730-1989:

1. Purpose—This section shall list the software covered and the portions of software life cycle covered.

2. Reference Documents—This section shall list all the documents referenced in the plan.

3. Management

3.1 Organization—This section shall describe the structure of organization and the responsibilities, and usually includes an organizational chart.

3.2 Tasks—This section shall list all of the tasks to be performed, the relationship between tasks and check points, and the sequence of the tasks.

3.3 Responsibilities—This section shall list the responsibilities of each organizational unit.

7.5 IEEE Standards for SQA Plan

4. Documentation

4.1 Purpose—This section shall list all required documents and state how documents will be evaluated.

4.2 Minimum documents—This section shall describe the minimum required documentation, usually including the following:

SRS—Software Requirements Specification

SDD—Software Design Description

SVVP—Software Verification and Validation Plan

SVVR—Software Verification and Validation Report

User documentation—Manual, guide

SCMP—Software Configuration Management Plan

5. Standards, Practices, Conventions, and Metrics

This section shall identify the S, P, C, and M to be applied and how compliance is to be monitored and assured. The minimal contents should include documentation standards, logic structure standards, coding standards, testing standards, selected SQA product, and process metrics.

6. Reviews and Audits—This section shall define what reviews/audits will be done, how they will be accomplished, and what further actions are required.

7. Tests—This section shall include all tests that are not included in SVVP.

7.5 IEEE Standards for SQA Plan

***8. Problem Reporting*—**This section shall define practices and procedures for reporting, tracking, and resolving problems, including organizational responsibilities.

***9. Tools, Techniques, and Methodologies*—**This section shall identify the special software tools, techniques, and methodologies and describe their use.

***10. Code Control*—**This section shall define the methods and facilities to maintain controlled versions of the software.

***11. Media Control*—**This section shall define the methods and facilities to identify, store, and protect the physical media.

12. Supplier Control (for outsourcing)**—**This section shall state provisions for assuring that software provided by suppliers meets standards.

***13. Records*—**This section shall identify documentation to be retained and methods to collection, maintain, and safeguard the documentation.

***14. Training*—**This section shall identify necessary training activities.

***15. Risk Management*—**This section shall specify methods and procedures for risk management.

7.5 IEEE Standards for SQA Plan

EXAMPLE 7.2

Develop Section 3 and Section 8 of an SQA plan for a software development project. Assume a project manager named Bill; an external test team consisting of Tina, the leader, Donna, and Helen; a separate configuration management (CM) group consisting of Mike, Sam, and Joe; and a separate external quality assurance (QA) team consisting of John and James.

See Fig. 7-1.

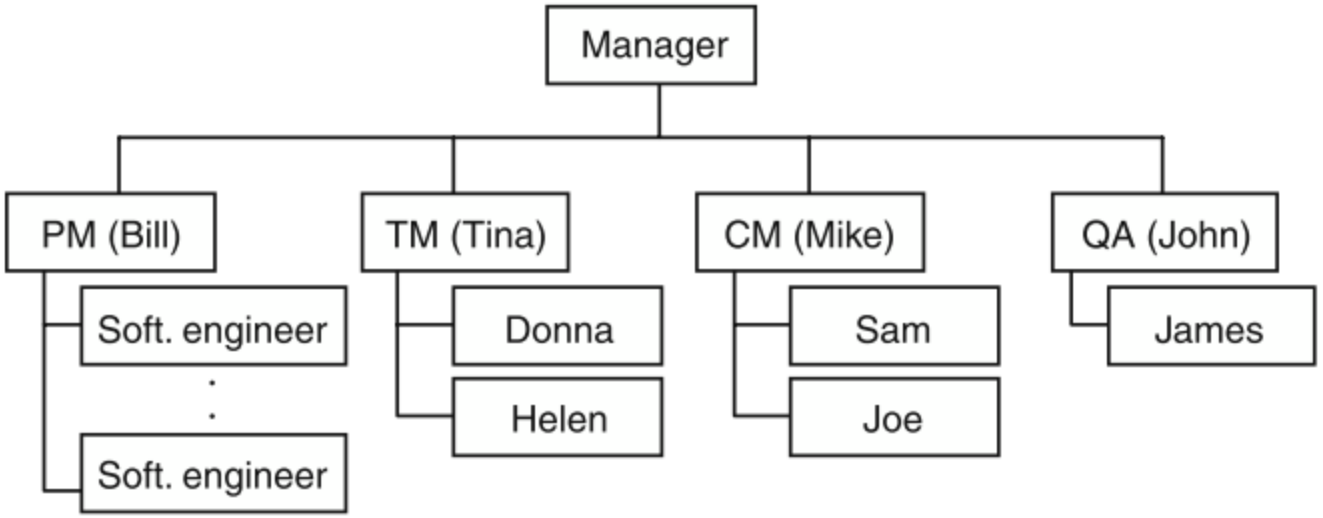

Fig. 7-1. Section 3 SQA plan.

7.5 IEEE Standards for SQA Plan

Section 3

3.1 Organization

3.2 Tasks

All documents will be reviewed. A configuration management tool will manage all documents and source code modules. All test plans will be done during the requirements phase and include an adequate number of test cases. Formal inspections will be conducted at the end of each phase.

3.3 Responsibilities

The project team is responsible for all development, including requirements, design, and implementation. The project team produces the test plan as part of the requirements. They are also responsible for all documentation, including the user manuals and training documents.

The test team is responsible for testing the base-lined version of the source code. The test team will use the test plan developed during requirements. Additional test cases will be developed to satisfy every-statement coverage of the code. Any discrepancies in the test plan and/or requirements or testing will be reported to the overall manager.

The configuration management team will be responsible for accepting software configuration items and assigning version numbers.

The quality assurance team will be responsible for overseeing all reviews, walkthroughs, and inspections. The QA team will track all problem reports.

7.5 IEEE Standards for SQA Plan

Section 8

All problems identified outside of the development unit must be reported to the QA team for assignment of a problem report number. The manager of each team will approve the corrections of problem reports assigned to that team. The QA team will be responsible for tracking all problems and weekly reporting to the overall manager.