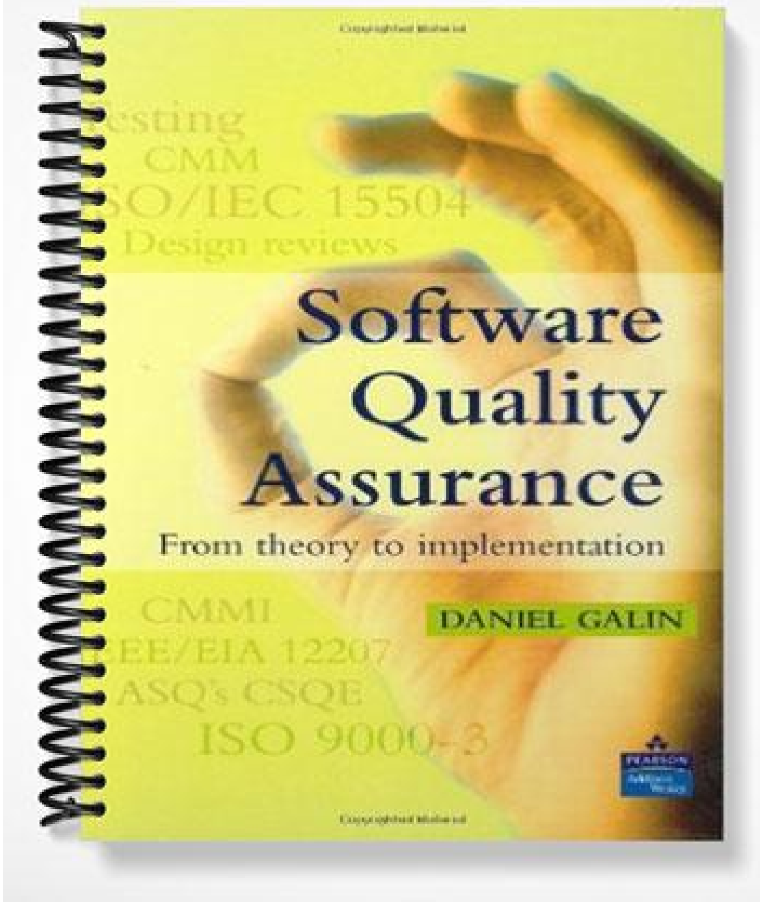
SOLUTIONS MANUAL



Solutions Manual

Software Quality Assurance

From Theory to Implementation

Daniel Galin

For further instructor material please visit:

www.booksites.net/galin

ISBN 0273705660

PEARSON
Addison Wesley

Harlow, England • London • New York • Boston • San Francisco • Toronto • Sydney • Singapore • Hong Kong
Tokyo • Seoul • Taipei • New Delhi • Cape Town • Madrid • Mexico City • Amsterdam • Munich • Paris • Milan

Pearson Education Limited
Edinburgh Gate
Harlow
Essex CM20 2JE
England

and Associated Companies around the world

Visit us on the World Wide Web at:
http://www.pearsoned.co.uk

First published 2005
© Pearson Education Limited 2005

The right of Daniel Galin to be identified as author of
this Work has been asserted by him in accordance with
the Copyright, Designs and Patents Act 1988.

ISBN: 0273705660

Contents

Part I

Introduction

CHAPTER 1

The software quality challenge

Review questions

1.1

There are three major differences between software products and other industrial products.

1. Identify and describe the differences.

2. Discuss the ways in which these differences affect SQA.

Solution

1. Software products differ from other industrial products with respect to the following characteristics:
   - **Product complexity** – a typical software product allows tens of thousands of operational options. Typical industrial products and even advanced industrial products do not reach this level of variety of options.
   - **Product visibility** – since software products are invisible, defects in the software are not visible unless testing procedures are applied. In contrast, industrial products are visible and most defects are visible to the production team by changes in color or shape. Even non-professionals can observe color changes that reveal defects in chemical and food products, or the changes in shape or structure that reveal defects in household products.
   - **Development and production process** – Defects in industrial products usually occur during each stage through which the product has to pass, namely development, product production planning, and manufacturing. At each stage the product is examined and tested independently by a different team. In contrast, software products are examined and tested during the development stage only. The product production planning and the manufacturing stages deal mainly with the duplication and packaging of the software product without contributing to the detection of software defects.

2. The first characteristic, product complexity, makes the task of quality assurance for software much more difficult, as the product must function correctly for all of the defined options, even those that are highly complicated and rarely used. The other two characteristics make software defects more difficult to detect. The combined effect of these software characteristics creates a situation in which software developers cannot be certain that the software product they supply is defect-free.

1.2

It is claimed that no significant SQA activities are expected to take place during the phase of production planning for software products.

1. Discuss this claim.

2. Compare the required production planning for a new automobile model with the production planning efforts required for the new release of a software product.

Solution

This review question refers mainly to the issues discussed in review question 1.1, but whereas question 1.1 compares a software product to an industrial product, this review question focuses on a comparison between the software product development process and the development process of an industrial product.

1. This claim is correct. The product production planning stage and the manufacturing stage in the case of software products deal mainly with duplication and packaging of the product, with no activities of review or testing of the software code. Therefore, it is not expected that software defects will be detected at these stages.

2. There is considerable similarity between the development of a new automobile model and a new software project. Both processes are characterized by creativity and require specific reviewing and testing procedures. It should be noted that many of the complicated operational options of automobiles are controlled by software (operated by the car's computer or computers).

1.3

Seven issues characterize the professional software development and maintenance environment.

1. Identify and describe these characteristics.

2. Which of these environmental characteristics mainly affect the professional efforts required for carrying software development and maintenance projects. List the characteristics and explain why a professional effort is needed.

3. Which of these environmental characteristics mainly affect the managerial efforts required for carrying out software development and maintenance projects. List the characteristics and explain why such efforts are needed.

Solution

1. The characteristics of the software development and maintenance environment are:
   – Contract conditions and commitments defining content and timetable.

- Conditions of the customer–supplier relationship, as exemplified by the need for consultation with customers and securing of their approval.
- Teamwork requirements.
- Need for cooperation and coordination with other software and hardware development teams both internally and externally.
- Need for interfaces with other software systems.
- Need for continuity in carrying out a project when team members change.
- Need for ongoing maintenance of the software system over several years.

2. Each of the abovementioned characteristics affects both the professional and the managerial efforts, though usually to differing extents. The following characteristics affect mainly the professional side:
   - **Contract conditions defining content** – the need to prepare a document listing functional and other requirements of the project.
   - **Conditions of the customer–supplier relationship** – the need to maintain ongoing contacts with the customer's professionals for presentations of development products, consultations with the customer, and securing customer approval of the development products.
   - **Teamwork requirements** – the need for the team leader to take responsibility for professional instruction of the team members and checking of their products.
   - **Cooperation and coordination with other software and hardware development teams both internally and externally** – the need to understand the tasks performed by other teams to the extent that enables proper professional communication.
   - **Interfaces with other software systems** – the need to acquire familiarity with the interfacing standards or interfacing design of equipment units and/or software packages.

3. The following characteristics affect mainly the managerial side:
   - **Contract conditions defining timetable** – the commitments relating to completion of a project and usually also to completion of each stage of the development process.
   - **Teamwork requirements** – the need to recruit a team and appoint a team leader to manage and to supervise the work of each of the team members.
   - **Continuity in carrying out a project when team members change** – the need to recruit, sometimes at short notice, a replacement team member having the professional knowledge and experience similar to those of the departing member.
   - **Ongoing maintenance of the software system over several years** – the need to ensure the constant availability of updated documentation on the software system, and to maintain a professional team well acquainted with the software system and capable of providing support at short notice.

Topics for discussion

1.1

Educational systems are assumed to prepare the students to cope with real life conditions. Examine the procedural requirements of a software development project or final software project, and determine what of the requirements could be considered as preparatory to professional life situations as discussed above.

Solution

Only part of the software development environment can be practiced within the framework of educational systems.

Let us examine this issue with reference to each of the seven environmental characteristics:

- Contract conditions and commitments defining content and timetable. Students' projects simulate contract conditions to some extent. A typical students' project includes definitions of the required functions as well as the time schedule for completion. Budget commitments are naturally not applicable.
- Conditions of the customer−supplier relationship, as exemplified by the need for consultation with customers and securing of their approval. The instructor–student relationship simulates to some extent the relationship between customer and supplier.
- Teamwork requirements. Projects done by student teams incorporate some aspects of team work, but are usually carried out without an appointed team leader.
- Need for cooperation and coordination with other software and hardware development teams both internally and externally. This is usually not applicable in students' projects.
- Need for interfaces with other software systems. This is applicable in some cases.
- Need for continuity in carrying out a project when team members change. This is usually not applicable in students' projects.
- Need for ongoing maintenance of the software system over several years. This is usually not applicable in students' projects.

1.2

Referring to the seven environmental characteristics of software development and maintenance, consider the characteristics of future software products, discussing whether the professional and managerial burden of coping with these characteristics in future is expected to be higher or lower when compared with the current performance of these activities.

Solution

- **Contract conditions and commitments defining content and timetable.** Customers can be expected to be much more demanding with respect to full implementation of functional and other requirements. Typical time schedules for similar development projects are expected to be substantially shorter than those currently allowed.

- **Conditions of the customer–supplier relationship.** The nature of future projects is likely to demand a much closer relationship between client and supplier.

- **Teamwork requirements.** Teamwork will continue to be required, though it is expected that new technologies will be implemented to support teamwork.

- **Need for cooperation and coordination with other software and hardware development teams both internally and externally.** These characteristics will become increasingly critical for the successful handling of projects. More comprehensive standardization will facilitate more effective coordination and cooperation.

- **Need for interfaces with other software systems.** The number of interfaces and the intensity of their use can be expected to increase. More comprehensive standardization of interfaces between software systems and between software and hardware will facilitate more effective development of interfaces.

- **Need for continuity in carrying out a project when team members change**. No significant change is expected.

- **Need for ongoing maintenance of the software system over several years.** No significant change is expected.

1.3

The interfaces of a salary processing system are exhibited in Frame 1.2.

1. Suggest what are the main benefits of applying computerized interfaces instead of transferring printouts.

2. Give two additional examples where input interfaces are applied.

3. Give two additional examples where output interfaces are applied.

4. Suggest additional situations where the use of input and output interfaces is not applied and should be recommended.

5. Would you advise all information transfers from one organization to another be performed by computerized interface? Discuss the reasons behind your answer.

Solution

1. The main benefits are:
   – Reducing the time period required to handle the input and update the system's database, and contributing to better up-to-dateness of the information provided by the system.
   – Drastically reducing the percentage of input errors, thus significantly improving the accuracy and completeness of the system's outputs.
   – Drastically reducing the human resources required to handle the input, both for keying in the input data and for correcting identified errors. This reduction will significantly reduce the costs of handling the input.

2., 3. A software interface serves two software systems, since it serves as an output interface for one and as an input interface for the other. Let us examine the following two examples:

Example 1: A monthly procedure of money transfers from the bank account of an employer to the employees' bank accounts. The required interface is between the software system for calculation of salaries and the bank's system for transferring money to customers' accounts.

Example 2: A centralized price update procedure for a network of stores. This procedure distributes the price updates and sale prices fixed by the network's head office, thus ensuring that all stores apply a uniform pricelist. Efficient operation of this procedure requires an interface between the price and sales management system of the head office and the point-of-sale system of the store.

The following table summarizes the two examples presented above:

Procedure carried out by the interface	**Software system for which it serves as an output interface**	**Software system for which it serves as an input interface**
Procedure for money transfers from the employer's bank account to employees' bank accounts	Employer's software system for salary calculation	Bank's system for transferring money to customers' accounts.
Centralized price update procedure for a network of stores	Head office price and sales management system	Store's point-of-sale system

4. Another example: Interface between a personal computerized health log system and a clinic's computerized information system. The computerized personal health system would probably be encapsulated as a "smart card", whose database includes

personal medical records. The clinic's system allows the medical staff to decide on the treatment needed by the patient, and to record diagnoses, medical treatments, medications, etc. A standard interface between personal health log systems and the clinic's information system, stationed in clinics, hospitals, first aid centers, etc., is required in order to benefit from the medical information it provides. The service provided by this two-way interface is summarized in the following table:

Procedure carried out by the interface	**Software system for which it serves as an output interface**	**Software system for which it serves as an input interface**
Transfer of patient's medical information to the clinic system	Personal computerized health log system	Clinic's computerized information system
Transfer of records of current medical treatment to update the patient's health log system	Clinic's computerized information system	Personal computerized health log system

5. In some situations an automatic computerized transfer of data may be not desirable.
   – Where an interface is hardly used, the expense is not economically justifiable.
   – If the data sources are of low quality, the transfer of information will be inaccurate and incomplete. In such cases the data and any necessary corrections should be checked manually before any input into the database of the recipient can be considered reliable.

1.4

The need to carry out work by a team demands additional investment in coordination of the team members. Discuss whether these managerial efforts could be saved if the work were performed as a "one-man job".

Solution

Teamwork obviously requires a team leader, who needs to spend much time on coordination among the team members, so that the work done by each of them can be assembled into a unified software system. The costs of these activities are overheads to software development costs. Obviously these extra costs are negligible by operating via a "one-man show". It should be emphasized, however, that a substantial part of a team leader's time is invested in checking the work of the team members. If the project is "a one-man show" the task of checking still needs to be performed by another member of the staff, possibly the head of department.

1.5

It is clear that a software development project carried out by a software house for a specific customer is carried out under content and timetable obligations, and is subject to the customer–supplier relationship.

1. Discuss whether a customer–supplier relationship is expected when the software developed is to be sold to the public as a software package.
2. Discuss whether a customer–supplier relationship is expected when software is developed for "in-house" use, as in the case where a software development department develops an inventory program for the company's warehouses.
3. Some managers claim that the closer relationships are to a formal pattern, the greater the prospects are for the project's success. Discuss whether implementing customer-supplier relationships in the situations mentioned in (1) and (2) are a benefit for the company (referring to the internal customer and supplier) or an unnecessary burden to the development team.

Solution

1. In the development of COTS (commercial off the shelf) software packages, the customer is the marketing department that initiates and approves the development project.
2. In the development of an internal project, the customer is the initiating department, the finance department, or the logistics department.
3. A formal relationship between an internal supplier (systems development department) and an internal customer can be expected to be beneficial for both parties by:
   - Providing more realistic project planning for scheduling and budgeting.
   - Providing more comprehensive and realistic plans with respect to functionality of the software products.
   - Contributing to better scheduling and budgetary control of the project.

   It might be claimed that a formal atmosphere among the internal parties to the project will reduce the creativity of the development team.

1.6

It has been claimed that environmental characteristics create the need for intensive and continuous managerial efforts parallel to the professional efforts that have to be invested in order to ensure the project's quality or, in other words, to assure the project's success. Discuss the reasons behind this claim, including an analysis of the managerial effort created by each of the SQA environmental characteristics.

Solution

The characteristics of the managerial environment in software development and maintenance are discussed in Review question 1.3, section (3).

The following characteristics require effort mainly on the part of management:

– **Contract conditions defining timetable.** The possibility of schedule failures can be minimized only by continuous and intensive follow-up of a project's progress at the management level, especially in problematic cases where additional team members are needed or in a situation that needs to be resolved by negotiation with the customer. Rigorous follow-up procedures will ensure earlier detection by management of deviations from schedule and their easier correction.

– **Teamwork requirements.** Teamwork requires managerial abilities in addition to professional qualities on the part of the team leader. One of the managerial aspects of teamwork is the need to recruit and instruct team members.

– **Need for continuity in carrying out a project when team members change.** Management needs to recruit, sometimes at short notice, a replacement team member having the professional knowledge and experience similar to those of a departing team member.

– **Need for ongoing maintenance of the software system over several years.** Whereas professionals engaged in a development process may resign from the company without being obliged to complete their work, management personnel are usually committed to these projects over relatively long periods. It is the duty of management to ensure the constant availability of updated documentation on the software system during this period, and to keep the professional team well acquainted with the software system and capable of carrying out maintenance tasks at short notice.

Proper managerial support, as discussed above, allows the professional teams to focus on the functional requirements of the project.

CHAPTER 2

What is software quality?

Review questions

2.1

A software system comprises of four main components.

1. List the four components of a software system.
2. How does the quality of each component contribute to the quality of the developed software?
3. How does the quality of each component contribute to the quality of the software maintenance?

Solution

1. The four components of a software system are:
   - Computer programs (the "code")
   - Procedures
   - Documentation
   - Data necessary for operating the software system.
2. The contributions of these components to the quality of developed software are:
   - Computer programs (the "code") – obviously, its quality is the basic component for the quality of services and functionality of the software product.
   - Procedures which define the methods of the program development process, i.e. software development planning procedure, design review procedure, software testing procedure and progress control procedure, contribute to the quality of the software product.
   - Development documentation (the requirements report, design reports, program descriptions, software testing plan, etc.) allows efficient cooperation and coordination amongst the development team members, easier replacement of any team member who leaves the team and efficient reviews and inspections of the design and programming products.
   - Data, including parameters and code lists that adapt the software to the specifications as well as test case files are necessary for testing the software before completion of the development process is possible.
3. The contributions of these components to the quality of maintenance services are:

- Computer programs (the "code") – obviously, its quality is the basic component for the quality of services and functionality of the software product.
- The procedures that accompany the software system deal with both, the regular operation of the software system and its maintenance. The regular software operational procedures define the method of program employment and the responsibilities for performing input processing, output processing and control activities. The maintenance procedures define the processes and responsibilities for the correction of "bugs". Another group of procedures deals with changes and improvements of programs, their approval and performance. The quality of these types of procedures contributes to the quality of services the software system provides.
- Documentation supports both users and maintenance professionals. The user's documentation (the "user's manual" etc.) provides a description of the available applications and the appropriate method for their use. Their quality is a major factor regarding the ability of users to successfully and efficiently apply the software applications. The maintenance documentation (the "programmer's software manual", etc.) provides the maintenance team with all the required information about the code and the structure and tasks of each software module. This information is used when trying to locate causes of software failures ("bugs") or to change or improve an existing software system.
- Data including parameters code lists that adapt the software to the needs of the specific user are necessary for operating the software. Another type of essential data is the standard test data, used to ascertain that no undesirable changes in the code or software data have occurred and to determine what kind of software malfunctioning can be expected.

2.2

1. Define software error, software fault and software failure. Explain the differences between these undesirable software statuses.
2. Suggest a situation where a new type of software failure ("bug") appears in a software package that has been serving 300 clients for the first time six years since the software package was first sold to the public.

Solution

1. A **software error** can be a grammatical or logical error in trying to comply with one or more of the client's requirements included in one or more of the code lines.

   A **software fault** is a software error which can cause improper functioning of the software in general or of a specific application.

   A **software failure** is a failure that has been "activated" and causes improper functioning of the software as a whole or of a specific, faulty application.

2. Let's refer to a meteorological application based on remote measuring stations; in this case the remote unit is required to initiate a protective closing operation that prevents

damage being caused to cold-sensitive measuring equipment each time the temperature drops below minus 20ºC. The programmer's software error defines the lower temperature limit as minus 30ºC. The meteorological information system purchased by several Spanish organizations was installed in various sites in Spain, none of which suffers from low temperatures. The fault only turned into a failure once a Russian version of the system had been developed and installed in various northern sites of Russia, causing severe damage to the measuring stations' equipment.

2.3

1. List and briefly describe the various causes of software errors.
2. Classify the causes of error according to the groups responsible for the error: the client's staff, the systems analysts, the programmers, the testing staff – or is it a shared responsibility belonging to more than one group?

Solution

1. Nine causes of software errors are listed:

   a. Faulty definition of requirements
   The faulty definition of requirements, usually prepared by the client, is one of the main causes of software errors. The most common errors of this type are: erroneous definition of requirements, absence of vital requirements, incomplete definition of requirements and inclusion of unnecessary requirements.

   b. Client-developer communication failures
   Misunderstandings resulting from defective client-developer communication are additional causes of errors. Typical situations: Misunderstanding of the client's instructions relating to the requirement document and to changes requested either written or orally by the client. Additional misunderstandings are failures to understand and to give the needed attention to the client's response to design problems raised by the development team.

   c. Deliberate deviations from software requirements
   In several circumstances, developers may deliberately deviate from the documented requirements, often causing software errors. Common situations of this type: reuse of software modules taken from an earlier project, omission of part of the required functions in an attempt to cope with time or budgetary pressures and developer-initiated improvements, introduced without the client's approval.

   d. Logical design errors
   Software errors of this type are mainly failures of systems architects, software engineers, systems analysts, etc., to formulate the software requirements into the proper algorithms, boundary conditions, omission of required system states, etc.

   e. Coding errors
   The reasons that cause programmers to make coding errors include misunderstanding the design documentation, linguistic errors, errors in the application of CASE, other development tools, and so forth.

   f. Noncompliance with documentation and coding instructions
   One may ask why noncompliance with coding instructions should cause