MASTERING SOFTWARE QUALITY ASSURANCE

Best Practices, Tools and Techniques for Software Developers

MURALI CHEMUTURI

2

FOUR DIMENSIONS OF QUALITY

CHAPTER OVERVIEW

- ★ Four dimensions essential to achieve quality: specifications, design, construction, and conformance
- ★ How to build in quality in each of the four dimensions
- ★ How to ensure quality in each of the four dimensions

BACKGROUND

Quality has four dimensions (as depicted in Figure 2.1):

1. Specification quality
2. Design quality
3. Development (software construction) quality
4. Conformance quality

Specifications are the starting point in the journey of providing a product or service, followed by design and then development. Conformance quality is ensuring how well that quality is built into the deliverable at every stage. These dimensions are discussed in detail in this chapter.

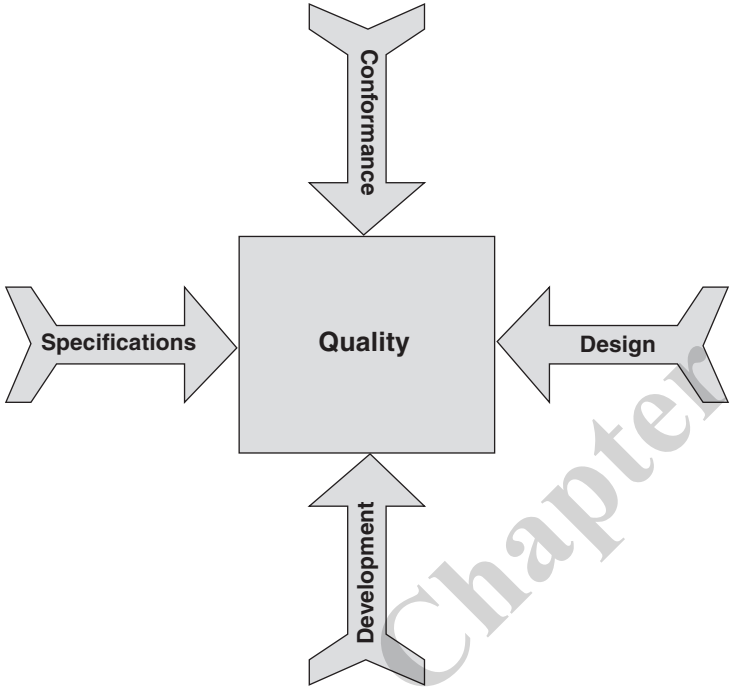

Figure 2.1. Four dimensions of quality

SPECIFICATION QUALITY

Specification quality refers to how well the specifications are defined for the product or service being provided. Specifications have no predecessor activity, and all other activities succeed specifications. Thus, if the specifications are weak, design will be weak, resulting in the development and manufacture of an inferior or incorrect product, and the effort spent on ensuring that quality is built in will have been wasted. Therefore, it is of paramount importance that specifications are comprehensive and well defined and that they take into account all possible aspects that have a bearing on the quality of the product.

Specifications normally should include the following six aspects:

1. **Functionality aspects**—Specify what functions are to be achieved by the product or service.
2. **Capacity aspects**—Specify the load the product can carry (such as 250 passengers on a plane or 100 concurrent users for a Web application) or the number of persons to whom a service can cater.

3. **Intended use aspects**—Specify the need or needs the product or service satisfies.
4. **Reliability aspects**—Specify how long the product can be enjoyed before it needs maintenance, or the surety of delivering the service and the conformance to the user requirements.
5. **Safety aspects**—Specify the threshold levels for ensuring safety to persons and property from use of the product or service.
6. **Security aspects**—Specify any threats for which the product or service needs to be prepared.

How do we make sure that we have comprehensive and correct specifications? The first aspect of ensuring that specifications are drawn up right is to engage qualified persons, such as business analysts or systems analysts, to carry out the job. These professionals must be properly trained to carry out requirements engineering. The second aspect is to either develop in-house standards or adopt the standards of a professional association or a standards body that the analysts are to follow. These standards set minimum levels in drawing up specifications.

Initially, specifications should be developed for a product in the usual way. In an internal or external customer-driven project scenario, user requirements are collected. In a commercial off-the-shelf product scenario, requirements are gathered from a market survey exercise.

Once requirements have been collected, they need to be developed. This involves separating the requirements into functional requirements, usability requirements, safety and security requirements, reliability requirements, and so on. These requirements also must be checked against organizational standards for usability, safety, and security, and any missing requirements need to be filled in. Then, each class of requirements is analyzed for comprehensiveness against either the backdrop of an existing product or a past project. If neither is available, functional experts should scrutinize and fill in the missing requirements. If access to experts is not available, then a team inside the organization is formed to carry out a brainstorming exercise to ensure that the specifications are comprehensive in all the classes. In a commercial off-the-shelf product scenario, a second market survey to tap the potential users can be conducted to improve the specifications.

DESIGN QUALITY

Design quality refers to how well the product or service to be delivered is designed. The objectives for design are to fulfill the specifications defined for the

product or service being provided. Design determines the shape and strengths of the product or service. Therefore, if the design is weak, the product or service will fail, even if the specifications are very well defined. Although design is a creative activity, it can be split into two phases: conceptual design and engineering. *Conceptual design* selects the approach to a solution from the myriad approaches available. *Engineering* uses the approach selected and works out the details to realize the solution. Conceptual design is the creative part of the process, and engineering is the details part.

Let's use the design of a bridge as an example to illustrate the difference between conceptual design and engineering. A bridge can be either a simply supported bridge or a suspension bridge. A simply supported bridge has a number of equally spaced pillars (columns) that support the bridge and the traffic that flows on it. A suspension bridge has a pillar at each end, with cables drawn from these two pillars to support the bridge. For this class of bridge, there are many alternatives for the suspension material, location of the pillars, design of the pillars, design of the suspension cables, and so on. Conceptual design decides these aspects. Engineering design works out details such as the dimensions for each component, selection of materials, methods of jointing, and so on.

In terms of software, conceptual design refers to software architecture, navigation, number of tiers, approaches to flexibility, portability, maintainability, and so on. Engineering design refers to database design, program specifications, screen design, report design, etc. Software design normally contains the following elements:

1. Functionality design
2. Software architecture
3. Navigation
4. Database design
5. Development platform
6. Deployment platform
7. User interface design
8. Report design
9. Security
10. Fault tolerance
11. Capacity
12. Reliability
13. Maintainability
14. Efficiency and concurrence
15. Coupling and cohesion

16. Program specifications
17. Test design

How do we ensure that the right designs are selected and implemented? As with specification quality, before software design is attempted, it is essential that qualified people, trained in the art and science of software design, are in place. Either software design standards are developed in-house or, alternatively, they are adopted from a professional association or a standards body. These standards assist designers to achieve the best design possible.

It is normal to conduct a brainstorming session at the beginning of a software design project, to select one optimum design alternative and to decide on the overall design aspects, such as the number of tiers, technology platform, software coupling and cohesion, etc. A brainstorming session helps designers arrive at the best possible solution for the project at hand. A prototype of the design may be created and evaluated, which is normal practice specifically in the case of commercial off-the-shelf product development.

The final design is then evaluated against the organizational standards to ensure that the design will work for the project. The design is subjected to reviews from peers, experts, and managers as required before carrying out the detailed design of the entire product.

DEVELOPMENT (SOFTWARE CONSTRUCTION) QUALITY

In certain fields, there is no way quality can be tested without destroying the product itself. For example, the thickness and adherence of paint on a surface cannot be ensured without destroying the paint itself. Various shafts used in automobiles are forged and heat treated to make them stronger, and there is practically no way to test them to ensure that the desired qualities are built in without destroying the shafts. In such cases, in-process inspection is performed to ensure that the process is adhered to diligently and a few samples are subjected to destructive testing.

Fortunately, when it comes to software, nothing needs to be destroyed during testing, and corrections can be made without any material loss, but testing takes much longer to perform in the software development field than it does in manufacturing. Inspection and testing take only a fraction of the time it takes to fabricate a part or a product in manufacturing, but software testing can sometimes take more time and effort than it takes to develop the software. It is commonly agreed that 100% testing is not practical in the software develop-

ment field. Therefore, the way in which software is developed assumes greater importance.

Normally, the following activities form part of developing software:

1. Create the database and table structures
2. Develop dynamically linked libraries for common routines
3. Develop screens
4. Develop reports
5. Develop unit test plans
6. Develop associated process routines for all other aspects, such as security, efficiency, fault tolerance, etc.

Good-quality construction is achieved by adhering to the coding guidelines of the programming language being used. Normally there is a separate coding guideline for every programming language used in an organization. It is customary to define the coding guidelines before beginning to write programs in a language. Coding guidelines contain naming conventions, code formatting, efficiency guidelines, and defect prevention guidelines that help developers write reliable and defect-free code. Of course, it is very important to have qualified people trained in software development. Construction follows software design, and it should always conform to the design document. In this way, good quality in construction can be achieved. Sample coding guidelines are given in Appendix I.

CONFORMANCE QUALITY

Conformance quality deals with how well an organization ensures that quality is built into a product through the above three dimensions. It is one thing to do a quality job, but it is quite another to unearth any defects lurking in the work product and ensure that a good-quality product is indeed built. Essentially, conformance quality examines how well quality control is carried out in the organization.

How do we determine how well an organization conducts the activities that ensure that quality is indeed built into a deliverable? One way to ascertain the efficacy of quality assurance activities is to use a set of quality metrics. These metrics include the defect removal efficiency of each of the quality control activities, product quality, and the defect density. Another way to ascertain the

efficacy of quality assurance activities is to compare industry benchmark data for quality metrics with the organizational metrics. Appendix G covers quality metrics and measurements in greater detail.

This book discusses how to build quality into a deliverable and ensure that quality is built into the first three dimensions mentioned (software specifications, design, and construction), as well as ensure the quality of conformance itself through quality measurement and metrics.

ENSURING QUALITY IN SPECIFICATIONS

This is the first activity in building either a product or a service. Needless to say, it is a creative activity. In the software industry, specifications are referred to as user requirements. That is, end users of a software product perceive them as the requirements for the proposed product. The following are possible scenarios for obtaining user requirements:

1. A business analyst conducts a feasibility study, writes up a report, and draws up the user requirements. The analyst:
   a. Meets with all the end users and notes their requirements and concerns
   b. Meets with the function heads and notes their requirements and concerns
   c. Meets with management personnel and notes their requirements and concerns
   d. Consolidates the requirements and presents them to select end users, function heads, and management personnel and receives their feedback, if any
   e. Implements the feedback and finalizes specifications
2. A ready set of user requirements is presented as part of a request for proposal
3. A request for proposal points to a similar product and requests replication with client-specific customization

Regardless of the scenario, once the specifications are ready, quality assurance steps in. The role of quality assurance in this area is to ensure that the specifications are exhaustive and cover all areas, including functionality, capacity, reliability, safety, security, intended use, etc.

The tools for building quality into specifications are as follows:

1. **Process documentation**—Details the methodology for gathering, developing, analyzing, and finalizing the specifications
2. **Standards and guidelines, formats, and templates**—Specify the minimum set of specifications that needs to be built in
3. **Checklists**—Help analysts to ensure comprehensiveness of the specifications

Using these tools, analysts can develop specifications that are comprehensive and are clear in order to carry out the next activity (which is design) and that ensure quality is built into specifications.

The tools that can be used to carry out quality assurance to ensure quality of specifications are expert reviews and peer reviews. The methodology for carrying out an expert review is detailed in Chapter 5.

ENSURING QUALITY IN DESIGN

Put simply, the process of design is converting product specifications (or user requirements) into design documents that can be used by programmers to develop the source code required for the product being built. Normally, software design is a two-step process:

1. **Conceptual design**—Referred to as high-level design, functional design specification, software requirements specification, and software architecture design. In this step, the overall architecture of the software product, including the number of tiers, modules, approaches to achieving the functionality, database design, robustness, reliability, and security, are determined and documented. This document is used by the designers to carry out the engineering design.
2. **Engineering design**—Referred to as low-level design, detailed design specification, software design description, and software program design. In this step, detailed specifications are drawn up for each program unit, screen, report, table, etc., and programmers use this document to develop source code.

The tools for building quality into design include the following:

1. **Process documentation**—Details the methodology for design alternatives to be considered, criteria for selecting the alternative for the project, and finalizing the conceptual design.

2. **Standards and guidelines, formats, and templates**—Specify the possible software architectures along with their attendant advantages and disadvantages, the methodology for short-listing of design alternatives, and so on.
3. **Checklists**—Help designers to ensure that design is carried out comprehensively and appropriately.

Using these tools, designers can develop designs that are comprehensive, are clear in order to carry out the next activity (which is software construction), and ensure that quality is built into designs.

The tools that are available for ensuring quality of design are expert reviews, peer reviews of each design specification, and managerial reviews of the overall design. The methodology for conducting reviews is detailed in Chapter 5.

ENSURING QUALITY IN DEVELOPMENT (SOFTWARE CONSTRUCTION)

Development is the act of building the software product in conformance with the design. In this stage, the source code is developed and is linked with the preexisting code libraries, to complete the code required for the product. This code is converted into executable code that can run on the hardware selected. This is also the stage in which the database is designed and built, so that data can be loaded and used by the software.

How is quality built into a product during the development stage? It is built in by adhering to the organizational standards for code quality as well as the coding guidelines for the development language being used. Uncontrolled changes can wreak havoc with code quality. Therefore, change management and configuration management assume importance for ensuring code quality.

There are two techniques to ensure that quality is built into a product: reviews (walkthroughs) and testing. These are detailed in Chapters 5 and 6.

ENSURING CONFORMANCE QUALITY

Ensuring that conformance quality is at desirable levels in the organization is achieved through quality measurements and metrics. Defect removal efficiency of verification and validation activities, defect injection rate, and defect density are all used for this purpose. In addition, projects are benchmarked at the organizational level and trend analysis is performed. These methods are detailed

in Appendix G. Audits also are conducted to ensure that projects conform to various applicable standards for building quality into all activities, including specifications and design. In addition, organizational data is benchmarked against industry benchmarks, and corrective or preventive actions are taken to ensure that organizational conformance is indeed on a par with the industry.

Conformance quality is built in through process definition and continuous improvement for all software development activities as well as quality assurance. Conformance quality is ensured through metrics and measurement.

Table 2.1 summarizes the techniques available for ensuring quality in each of the four dimensions of quality.

Table 2.1. Conformance techniques for four dimensions of quality

Quality dimension	How to build in quality	Techniques for ensuring quality
Quality of specifications	Specification development process documentation; standards and guidelines, formats, and templates for defining specifications; and checklists	Expert reviews, peer reviews, and brainstorming
Quality of design	Software design process documentation; standards and guidelines, formats, and templates for software design; and checklists	Expert reviews, peer reviews, managerial reviews, and brainstorming
Quality of development	Coding guidelines, configuration management, and change management	Peer reviews and software testing
Conformance quality	Diligent application of all quality assurance activities in the organization, process definition, and improvement	Audits, measurement and metrics for quality assurance activities, and benchmarking of organizational metrics against industry metrics