Software Quality Assurance(SQA)

ضمان جودة البرمجيات

ITSE5302

Dr. Moamar Elyazgi

Selected Portions in Software Quality Assurance

- Chapter 1: All the sections
- Chapter 2: All the sections
- Chapter 3: All the sections
- Chapter 6: All the sections
- Chapter 7: All the sections
- Chapter 8: All the sections

Chapter 1

Introduction

Problems of defining software quality 1

- One approach is to that the software will be used for a specified purpose and that therefore it must have *certain features and characteristics* that enable it to be used for that purpose.
- This approach leads to the well-known, if rather casual definition that 'quality means that the software is fit for its purpose'.

Problems of defining software quality 2

- ***Example:***
- Let us assume that the minimum set of characteristics necessary to ensure that the software was fit for the purpose of word processing have been defined and that two different designs have been produced both of which will, if implemented, possess this minimum set of characteristics.
- Two different types of quality can now be distinguished:

➢ The quality of conformance reflects the extent to which the developer succeeded in producing software which implemented the design.

➢ The quality of design reflects the extent to which a given word processor meets a customer's expectations.

Problems of defining software quality 3

- Differences in the quality of design can explain a user's preferences for a particular supplier's product, even when both possess the minimum set of characteristics necessary to make them fit for the task of word processing.
- The quality of conformance of the software to its design is dependent upon the production process and the quality of the resources used to produce it.

Problems of defining software quality 4

- The introduction of the user's expectations into the definition of quality and the resultant need to meet them have far-reaching implications.

- This leads to the conclusion that there are at least three components which must be considered when trying to define the quality of a software product:

- An objectively measurable component.
- A subjectively assessable component.
- A non-assessable component.

Problems of defining software quality 5

1. The objectively measurable component is the extent to which the software meets its specification.

✓This specification expresses requirements which may originate from both the use to which the customer intends to put the product lind the customer's personal preferences.

2. The subjectively assessable component relates to the customer's perception of the extent to which the software meets their preferences and expectations.

Problems of defining software quality 8

3. The non-assessable component refers to the ability of the software to continue to meet the customer's expectations in circumstances which were not envisaged when the specification was agreed.

- Whether it is reasonable to expect a given software product to work in circumstances not covered by the specification is not at issue here, the point is simply that if the software did continue to meet the customer's expectations in these unforeseen circumstances, then the customer would rightly consider it to be of higher quality than if it did not.

Problems of defining software quality 9

These attributes are known as quality factors in this quality model by Boehm et al. (1980). Such a list of attributes might include:

- Efficiency
- Reliability
- Usability
- Extendability
- Portability
- Testability
- Understandability
- Reusability
- Maintainability
- Interoperability
- Integrity
- Survivability

Problems of defining software quality 10

- The problem of defining quality however is a general one; it is not peculiar to software.
- A good analysis of the problem is given by Garvin (1984), who argues that there are a number of different approaches to defining quality, arising from different disciplines, all of which may be relevant.

Garvin's five approaches - 1

- Garvin's five approaches to defining quality are as follows:

1. Transcendent Approach

In this approach, which originates from philosophy, the quality of a piece of software is viewed as its innate excellence. Quality is an un-analysable property.

Garvin's five approaches - 2

2. Product-Based Approach

- The quality of a piece of software is related to the presence of some attributes or characteristics.
- This approach to defining quality seems implicitly to depend upon these attributes or characteristics making the software more or less suitable for use in a particular situation.

Garvin's five approaches - 3

3. User-Based Approach

- The quality of a software product is partly related to its fitness for use in a particular application.
- In this approach the quality is positively related to (or equated with) the software user's satisfaction with the software product in any given application.

Garvin's five approaches - 4

4. Value-Based Approach

- This appears to combine quality, which is a measure of excellence with value, which is a measure of worth, by defining a quality product as one which provides performance at an acceptable price or conformance at an acceptable cost.

Garvin's five approaches - 5

5. Manufacturing Approach - 1

- Traditionally this approach has been concerned with engineering and manufacturing practice and is summarised in the expression 'make it right first time'.
- Quality is equated with conformance to stated requirements.
- For instance, the design of the software system would have been checked to ensure that if implemented it would meet the stated requirements; any subsequent deviation from a formally approved design would be seen as a reduction in quality.

Garvin's Dimensions - 1

The Basic Elements of a product quality

❖Garvin suggests that there are a number of different dimensions of quality which may be of importance when considering the quality of a software product.

1. Performance and 2. Features

- The distinction between these two seems a little strained, but *performance relates to the primary operating characteristics of the software* and *features refer to the secondary characteristics that supplement the software's basic functions*.

Garvin's Dimensions - 2

3. Reliability

- There is the probability of a software product failing within a specified period of time. This is a very difficult concept to define for software since it does not physically deteriorate.

Garvin's Dimensions - 3

4. Conformance

- This aspect is concerned with the extent to which the output of each phase of the software development process meets the specification for that phase.
- The implications for the software developer are that the software should be 'tested' at each phase of its development, not only after coding has started.
- This dimension is of significance, both before and after acceptance of the software by the customer.
- Deviations may become apparent only after the software system has gone into service.

Garvin's Dimensions - 4

5. Durability

- This is intended to be a measure of the length of time that the software can be used before replacement.
- There are many reasons for wishing to replace software. Some are related to the cost of changing it to meet new circumstances.
- For instance, it may be necessary to port it to new hardware to continue using it and replacement may seem preferable, or the complexity of the software may have increased so much as a result of changes to fix bugs or incorporate new features that further maintenance is felt to be more costly than purchasing a new product.

Garvin's Dimensions - 5

6. Serviceability

This aspect encompasses such things as the responsibility for rectifying defects, the length of time that this takes and the ease with which the supplier of the software accepts responsibility.

Garvin's Dimensions - 6

7. Aesthetics

- Software can be beautiful, but what counts as beauty at any moment in time and how it is perceived is a matter for the individual. This is a matter of personal judgment

Garvin's Dimensions - 7

8. Perceived quality

The individual whose opinion is sought about the quality of the software may not have full information about it and his opinion of its quality may be biased by, amongst other things, its price or the reputation of its supplier.

Garvin's Dimensions - 8

- Even if one does not fully accept Garvin's analysis, it seems clear that quality is not easily defined, except arbitrarily, and also that there are a number of dimensions to it.
- For software, probably the most common definition of quality is the user-based, closely followed by the manufacturing-based definition.

Standard Definition of Quality

- **The now standard definition of quality is:**

The totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs. (ISO 1986)

- **Alternatively:**

The degree to which the attributes of the software enable it to perform its specified end item use. (DOD 1985)

An Overview of Software Quality Assurance

- The definition of software quality assurance is arbitrary, as one might expect, given the lack of a definition of quality encompassing all the associations of the word. The generally accepted definition is that given by ANSI/IEEE (1981):

"A planned and systematic pattern of all actions necessary to provide adequate confidence that the software conforms to established technical requirements."

This definition form is also widely used, particularly by the procuring agencies.

Quality Systems - 1

A quality system is defined by ISO 8402 as

"The organisational structure, responsibilities, procedures, processes and resources for implementing quality management."

Quality Systems - 2

- The purpose of the quality system is to ensure that the software as it is developed is of the required quality.
- It should ensure that the quality is built into the software and that the quality is monitored as the software is developed. There are three parts to a quality system:

❑ Development procedures

❑ Quality control

❑ Quality assurance

Quality Control - 1

- Quality control is defined in ISO 8402 as

"The operational techniques and activities that are used to fulfil requirements for quality. "

- This definition is intended to cover activities which are aimed at monitoring a process and also those which are aimed at eliminating unsatisfactory performance.

Quality Control - 2

- **Quality control activities include:**

- Planning and tracking the development plan
- All aspects of configuration management
- Various reviews, inspections and walkthroughs
- All error reporting and corrective action systems
- Various system level tests including acceptance tests

Software Quality Costs - 1

- The conventional wisdom is that increasing quality reduces total costs up to some point.
- Quality costs refer to the cost of correction of defects and the addition of new features found to be necessary during production.
- It is expenditure on software development and maintenance in excess of that which would have been incurred if the product had been built exactly right in the first place.

Software Quality Costs - 2

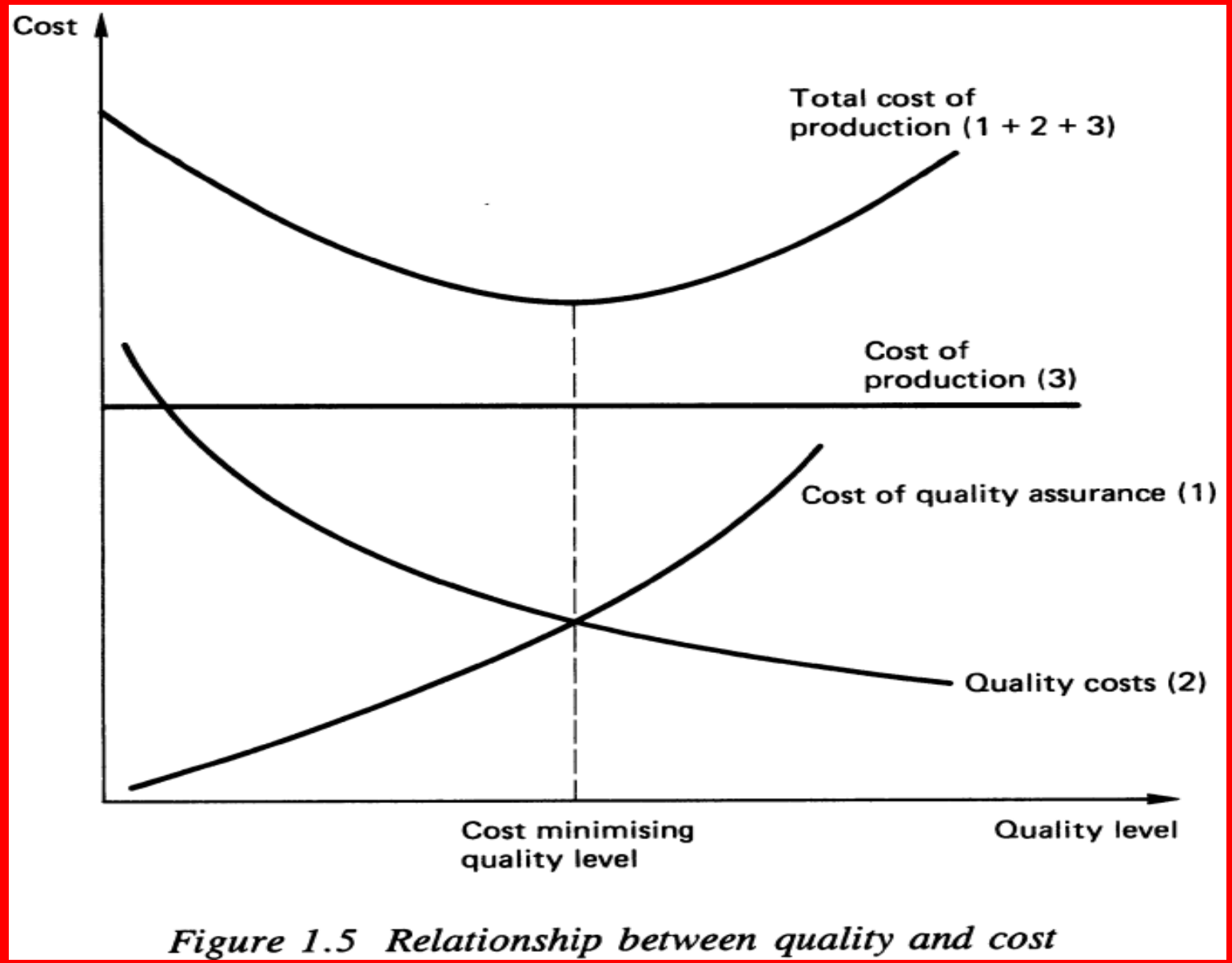

Figure 1.5 Relationship between quality and cost

Software Quality Costs - 3

- The cost of production of the software product is constant with respect to the quality level of the software.

- If differences in the quality level reflect differences in the performance, reliability, etc. of the software, there may well be a positive relationship between the cost of production and the quality level, but this should not change the relationship significantly.

Software Quality Costs - 4

- Total quality related costs (curves 1 and 2) are often subdivided into four groups:

1. Prevention costs. Quality planning, employee training, supplier education, etc.
2. Appraisal costs. Reviews, walkthroughs and other forms of testing.
3. Internal failure costs. The cost of correcting defects discovered before acceptance.
4. External failure costs. The cost of correcting defects discovered after acceptance which have to be borne by the developer.

Implications of Open Systems - 1

- The open systems approach to the development of computer-based systems has the potential to significantly improve the quality of software.
- There are a number of meanings of the term 'open systems' and the concept itself is evolving. The definition of open systems used by the DTI (1991) is:

Implications of Open Systems - 2

- ‘Open Systems' is an approach to IT planning, development and operations that uses internationally agreed standards to achieve a firm technical foundation (the 'IT infrastructure') on which flexible and responsive IT solutions to business needs can be built.
- Open systems standards define interfaces and basic infrastructure functions.
- Using these standards, IT users may develop, run and interconnect applications from a variety of sources on a variety of hardware.

The Motivation to Undertake Quality Assurance Activities -1

- There are a number of reasons why a software developer might wish to undertake quality assurance activities in addition of course to the desire to produce a good product, which can be the prime motivating factor, but usually is not.

- There is also the implication that the developer will have to preserve the documentary evidence of a satisfactory development process for the whole time that the software is in use, in case it should be needed as evidence.

The Motivation to Undertake Quality Assurance Activities -2

- The user may insist that the developer has a satisfactory software quality assurance programme. It used to be argued that this was really none of the user's concern because the software was only paid for once it had passed its acceptance test and if it did not work then the potential user had not lost anything.
- The developer may adopt a quality assurance programme because it has been shown to be cost effective.
- It has already been argued that it could reduce costs both by preventing errors and allowing them to be detected earlier than would otherwise be the case.