投票 最多可选1项

设置

Are you ready ?

A Yes

B No

提交

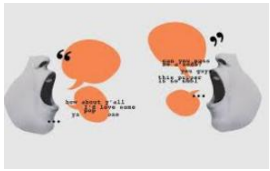
Part 3
Quality Management

Chapter 19
Quality Concepts

Contents

19.1 Software Quality

- Software quality remains an issue, but whom to blame?
  - Customers blame developers, arguing that sloppy practices lead to low-quality software.
  - Developers blame customers (and other stakeholders), arguing that irrational delivery dates and a continuing stream of **changes** force them to deliver software **before** it has been **fully validated**.



blame each other

19.1 Quality

- *Quality* (The *American Heritage Dictionary*)
  - "a characteristic or attribute of something."

- For software, two kinds of quality may be encountered:
  - Quality of design encompasses requirements, specifications, and the design of the system.
  - Quality of conformance is an issue focused primarily on implementation.

User satisfaction = compliant product + good quality
+ delivery within budget and schedule

19.1 Quality—A Pragmatic View

- The *transcendental view* argues that quality is something that you immediately recognize, but **cannot explicitly define**.
- The *user view* sees quality in terms of an end-user's **specific goals**. If a product meets those goals, it exhibits quality.
- The *manufacturer's view* defines quality in terms of the **original specification** of the product. If the product conforms to the spec, it exhibits quality.
- The *product view* suggests that quality can be tied to inherent **characteristics** (e.g., functions and features) of a product.
- The *value-based view* measures quality based on **how much** a customer is willing to pay for a product. In reality, quality encompasses all of these views and more.

19.1 Software Quality

- Software quality can be defined as:

  – *An **effective software process** applied in a manner that creates a **useful product** that provides **measurable value** for those who produce it and those who use it.*
  (three aspects)

19.1 Effective Software Process

- Establishes the **infrastructure** that supports any effort at building a high quality software product.
- The management aspects of process create the checks and balances that help **avoid** project **chaos**(a key contributor to poor quality).
- Software engineering practices allow the developer to analyze the problem and design a solid solution(both critical to building high quality software).
- Umbrella activities such as ***change management*** and ***technical reviews*** have as much to do with quality as any other part of software engineering practice.

19.1 Useful Product

- A *useful product* delivers the content, functions, and features that the end-user desires
- But delivers these assets in a reliable, error free way.
- Always satisfies those requirements that have been explicitly stated by stakeholders.
- Satisfies a set of implicit requirements (e.g., ease of use) that are expected of all high quality software.

19.1 Adding Value

By *adding value for both the producer and user* :

- The software organization gains added value because high quality software requires less maintenance effort, fewer bug fixes, and reduced customer support.
- The user community gains added value because the application provides a useful capability in a way that expedites some business process.

1. greater software product revenue
2. better profitability when an application supports a business processd
3. improved availability of information that is crucial for the business.

主观题 10分

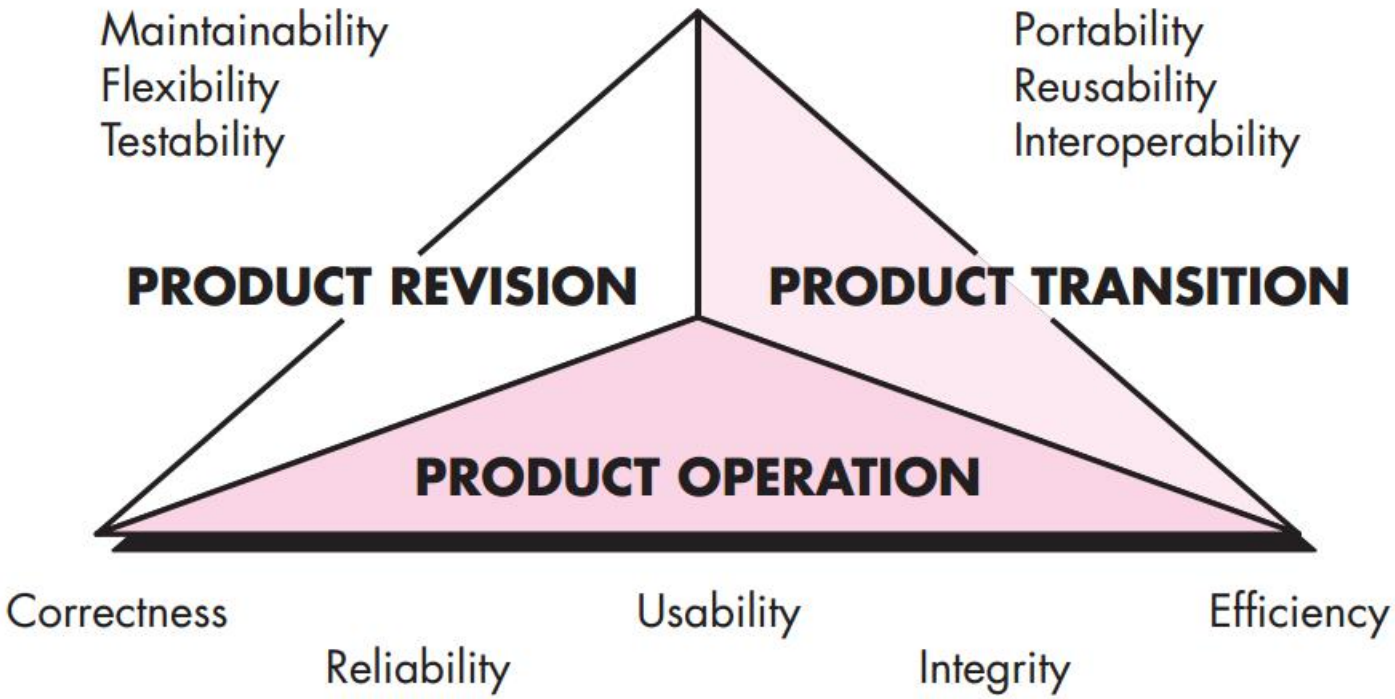
设置

Do you think it’s possible to create a useful product that provides measurable value without using an effective process? Explain your answer.

正常使用主观题需2.0以上版本雨课堂

作答

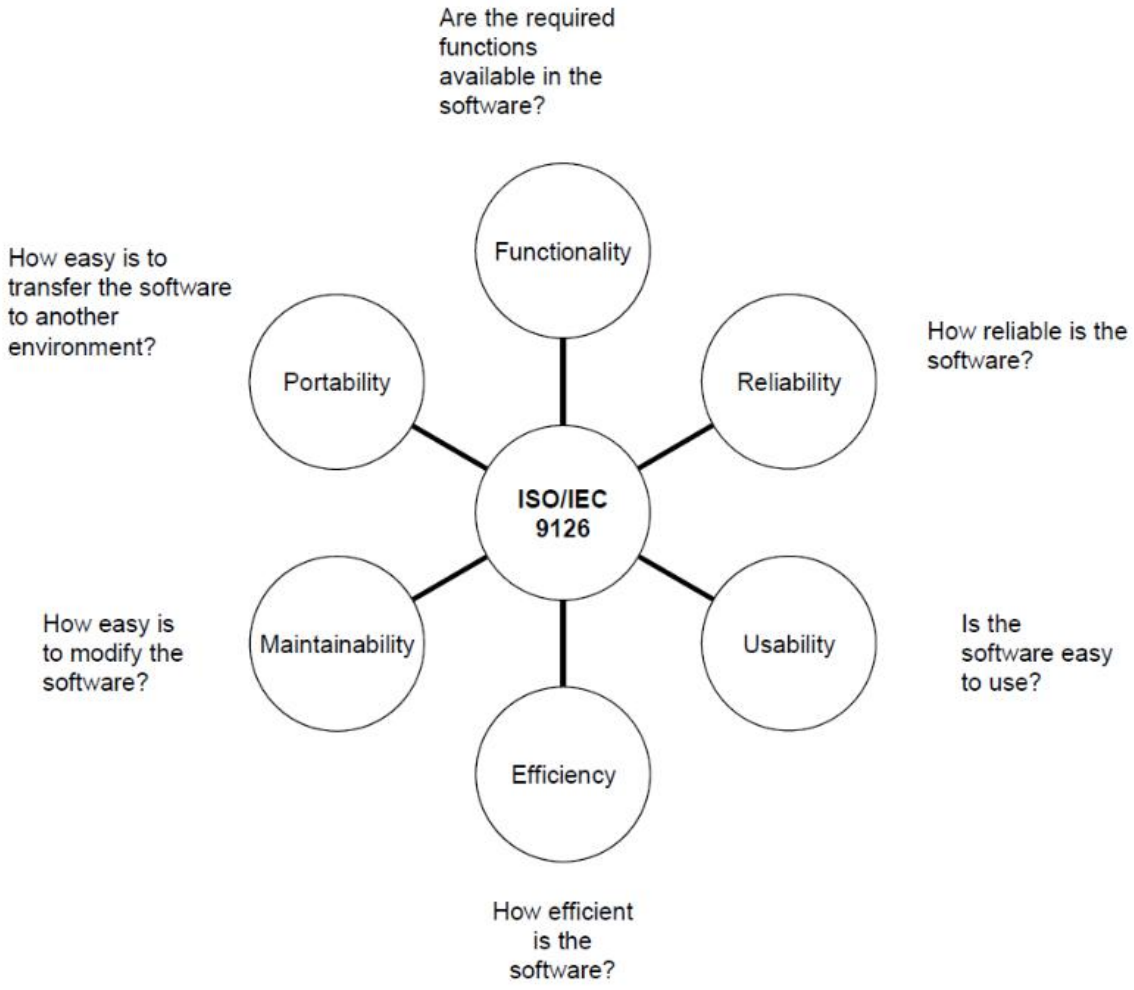
19.2.2 McCall's Quality Factors



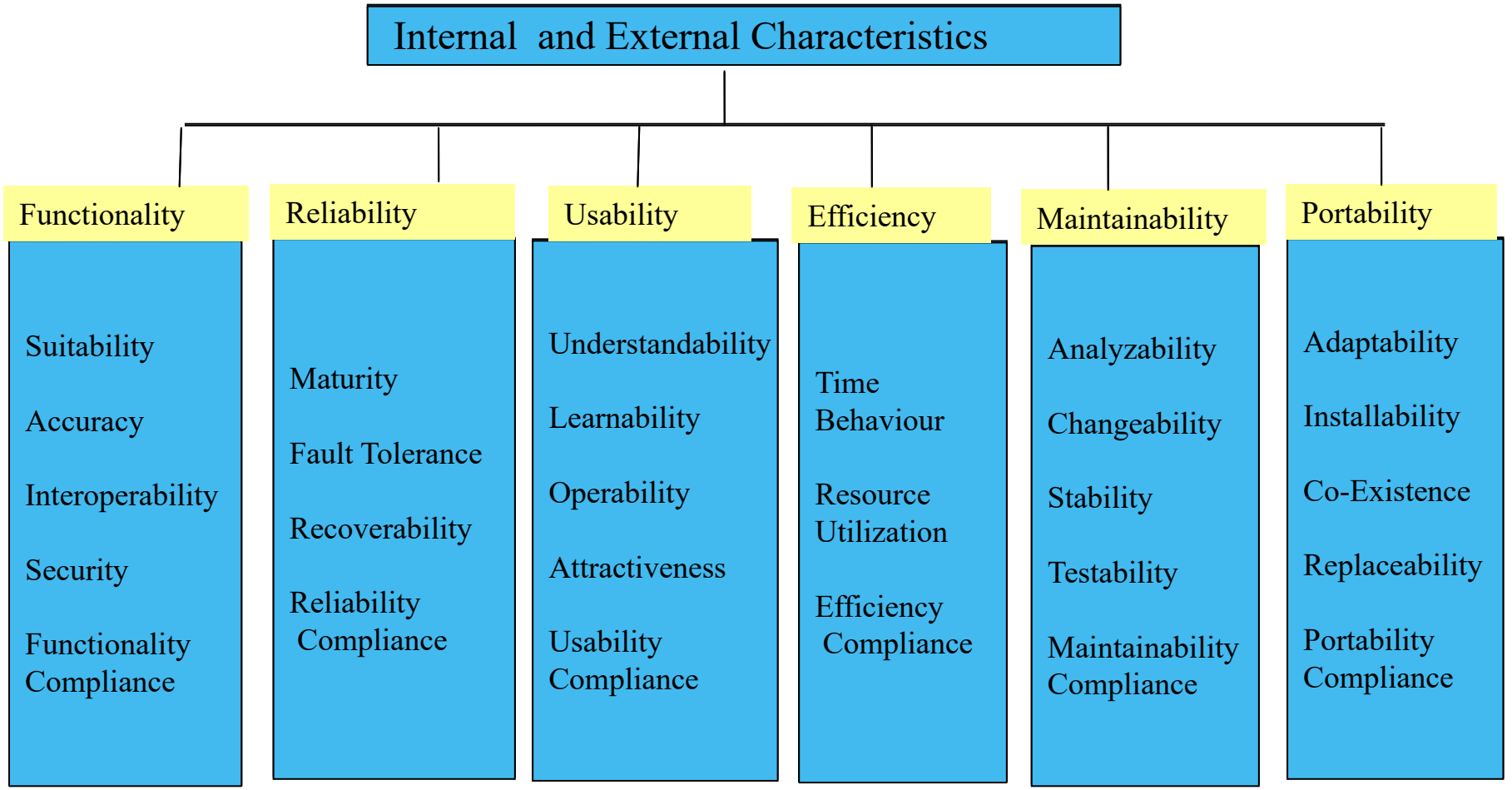
McCall, Richards, and Walters [McC77]

19.2.3 ISO 9126 Quality Factors

- ISO/IEC 9126（1993）



19.2.3 ISO 9126 Quality Factors



19.2.3 ISO 9126 Quality Factors

- ISO/IEC 25010 （2011）

(Sub)Characteristic
Functional suitability
Functional completeness
Functional correctness
Functional appropriateness
Performance efficiency
Time behaviour
Resource utilization
Capacity
Compatibility
Co-existence
Interoperability
Usability
Appropriateness recognizability
Learnability
Operability
User error protection
User interface aesthetics
Accessibility

Reliability
Maturity
Availability
Fault tolerance
Recoverability
Security
Confidentiality
Integrity
Non-repudiation
Accountability
Authenticity
Maintainability
Modularity
Reusability
Analysability
Modifiability
Testability
Portability
Adaptability
Installability
Replaceability

19.2.1 Quality Dimensions

- David Garvin [Gar87]:
  - **Performance Quality.** Does the software deliver all content, functions, and features that are specified as part of the requirements model in a way that provides value to the end-user?
  - **Feature quality.** Does the software provide features that surprise and delight first-time end-users?
  - **Reliability.** Does the software deliver all features and capability without failure? Is it available when it is needed? Does it deliver functionality that is error free?
  - **Conformance.** Does the software conform to local and external software standards that are relevant to the application? Does it conform to design and coding conventions?

19.2.1 Quality Dimensions

David Garvin [Gar87] (continue):

- **Durability.** Can the software be maintained (changed) or corrected (debugged) without the inadvertent generation of unintended side effects? Will changes cause the error rate or reliability to degrade with time?
- **Serviceability.** Can the software be maintained (changed) or corrected (debugged) in an acceptably short time period. Can support staff acquire all information they need to make changes or correct defects?
- **Aesthetics.** Most of us would agree that an aesthetic entity has a certain elegance, a unique flow, and an obvious "presence" that are hard to quantify but evident nonetheless.
- **Perception.** In some situations, you have a set of prejudices that will influence your perception of quality.

19.2.3 ISO 9126 Quality Factors

- **Example: Usage of** ISO/IEC 9126

1.Quality attributes selection

2.Quality score define

Response time(second)	Quality score
<2	5
2-3	4
4-5	3
6-7	2
8-9	1
>9	0

3.Overall quality evaluation

Product quality	Importance rating(a)	Product A		Product B	
		Quality score(b)	Weighted score(a*b)	Quality score(b)	Weighted score(a*b)
Usability	3	1	3	3	9
Efficiency	4	2	8	2	8
Maintainability	2	3	6	1	2
overall			17		19

19.3 The Software Quality Dilemma

- If you produce a software system that has terrible quality, you lose because **no one** will want to **buy** it.
- If on the other hand you spend infinite time, **extremely large effort**, and huge sums of money to build the absolutely perfect piece of software, then it's going to take so long to complete and it will be **so expensive** to produce that you'll be out of business anyway.

good enough?

19.3 “Good Enough” Software

- Good enough software delivers ***high quality*** functions and features that end-users desire, but at the same time it delivers other more obscure or specialized functions and features that contain known bugs.
- Arguments *against* “good enough.”
  - If a company has a large marketing budget and can convince enough people to buy version 1.0, it has succeeded in locking them in.
  - If you work for a small company be wary of this philosophy. If you deliver a “good enough” (buggy) product, you risk permanent damage to your company’s reputation.
  - You may never get a chance to deliver version 2.0 because bad buzz may cause your sales to plummet and your company to fold.
  - If you work in certain application domains (e.g., real time embedded software), application software that is integrated with hardware can be negligent and open your company to expensive litigation.

https://www.g2intelligence.com/dont-crash-burn-why-cutting-your-quality-departments-budget-is-not-a-good-idea/

19.3 “Good Enough” Software

Source: https://www.isixsigma.com/implementation/financial-analysis/cost-quality-not-only-failure-costs

Figure 3: Finding the “sweet spot”

19.3 Cost of Quality

Prevention costs

- quality planning
- testing planning
- training

Appraisal costs

- formal technical reviews
- testing and debugging
- data collection and metrics evaluation

Internal failure costs

- rework
- repair
- failure mode analysis

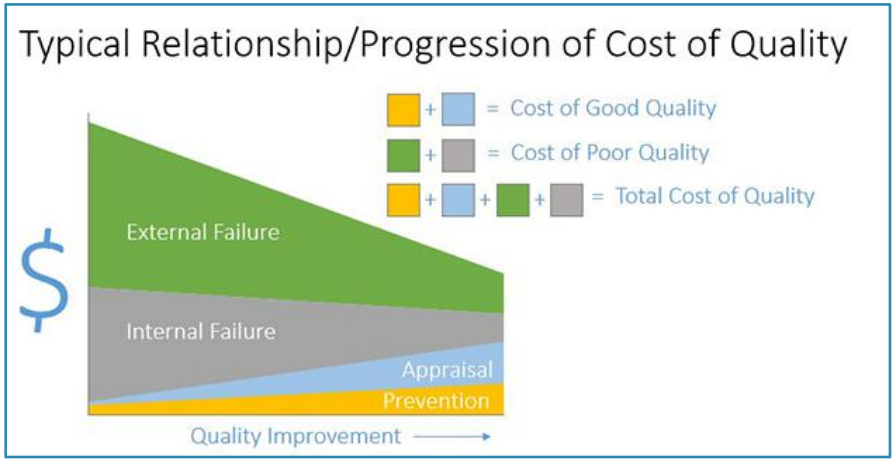
External failure costs

- complaint resolution
- product return and replacement
- help line support
- warranty work

主观题 10分

设置

Considering each of the four aspects of the cost of quality, which is the most expensive in your opinion and why?



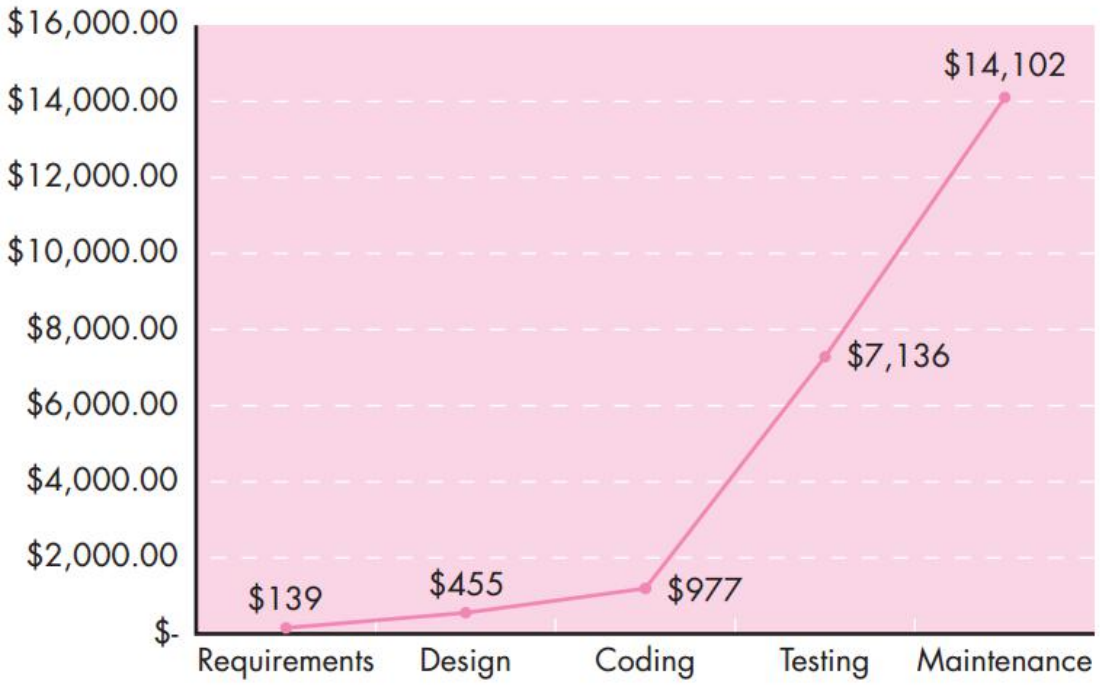
https://www.g2intelligence.com/dont-crash-burn-why-cutting-your-quality-departments-budget-is-not-a-good-idea/

正常使用主观题需2.0以上版本雨课堂

作答

19.3 Cost

- The relative costs to find and repair an error or defect increase dramatically as we go from prevention to detection to internal failure to external failure costs.



100 times

19.3 Negligence and Liability

- The story is all too common. A governmental or corporate entity hires a major software developer or consulting company to analyze requirements and then design and construct a software-based "system" to support some major activity.
  - The system might support a major corporate function (e.g., pension management) or some governmental function (e.g., healthcare administration security).

- Work begins with the best of intentions on both sides, but by the time the system is delivered, things have gone bad.

- The system is late, fails to deliver desired features and functions, is error-prone, and does not meet with customer approval.

19.3 Quality and Security

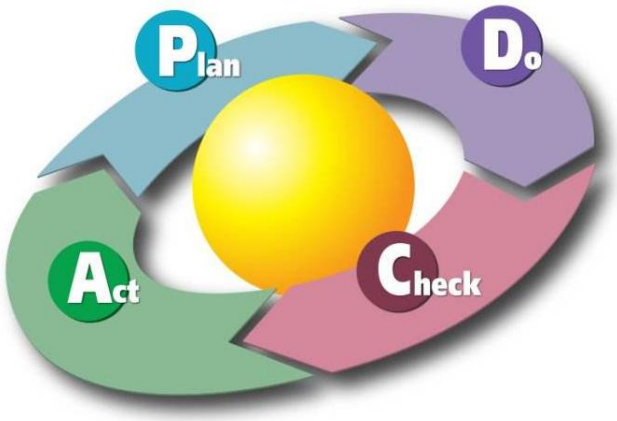
- Gary McGraw comments [Wil05]:

"Software security relates entirely and completely to quality. You must think about security, reliability, availability, dependability—at the beginning, in the design, architecture, test, and coding phases, all through the software life cycle [process]. Even people aware of the software security problem have focused on late life-cycle stuff. The earlier you find the software problem, the better. And there are two kinds of software problems. One is bugs, which are implementation problems. The other is software flaws—architectural problems in the design. People pay too much attention to bugs and not enough on flaws."

19.3 Achieving Software Quality

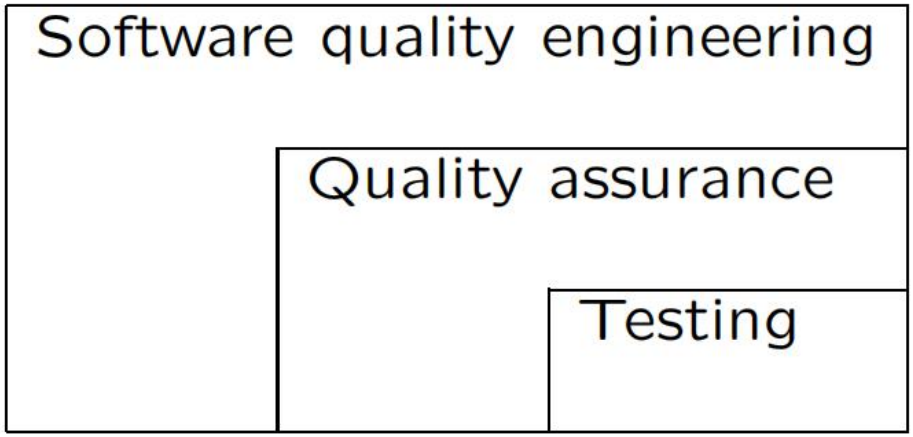
- Critical success factors:
  - **Software Engineering Methods**
  - **Project Management Techniques**
  - **Quality Control: A part of quality management focused on fulfilling quality requirements. (activity to control)**

Deming circle



19.3 Achieving Software Quality

- Critical success factors:
  - **Quality Assurance:** **A part of quality management focused on providing confidence that quality requirements will be fulfilled. (activity to check)**



Summary

- Definition: effective software process, useful product, measurable value
- Quality Model: McCall's Quality Factors,
- ISO 9126 Quality model:Functionality, Reliability, Usability, Efficiency ,Maintainability , Portability
- Cost of Quality: Prevention costs , Appraisal costs , Internal failure costs, External failure costs
- Quality Control:
  - ✓ method: checklist, Pareto principle, histogram, fishbone graph, etc.

THE END