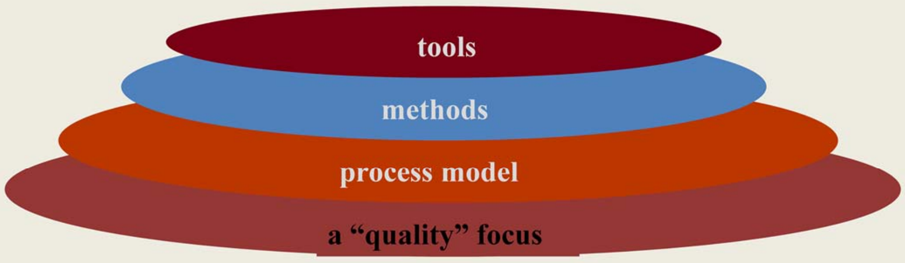
Software Quality Management
concepts, review techniques, and SQA

A layered overview of SE



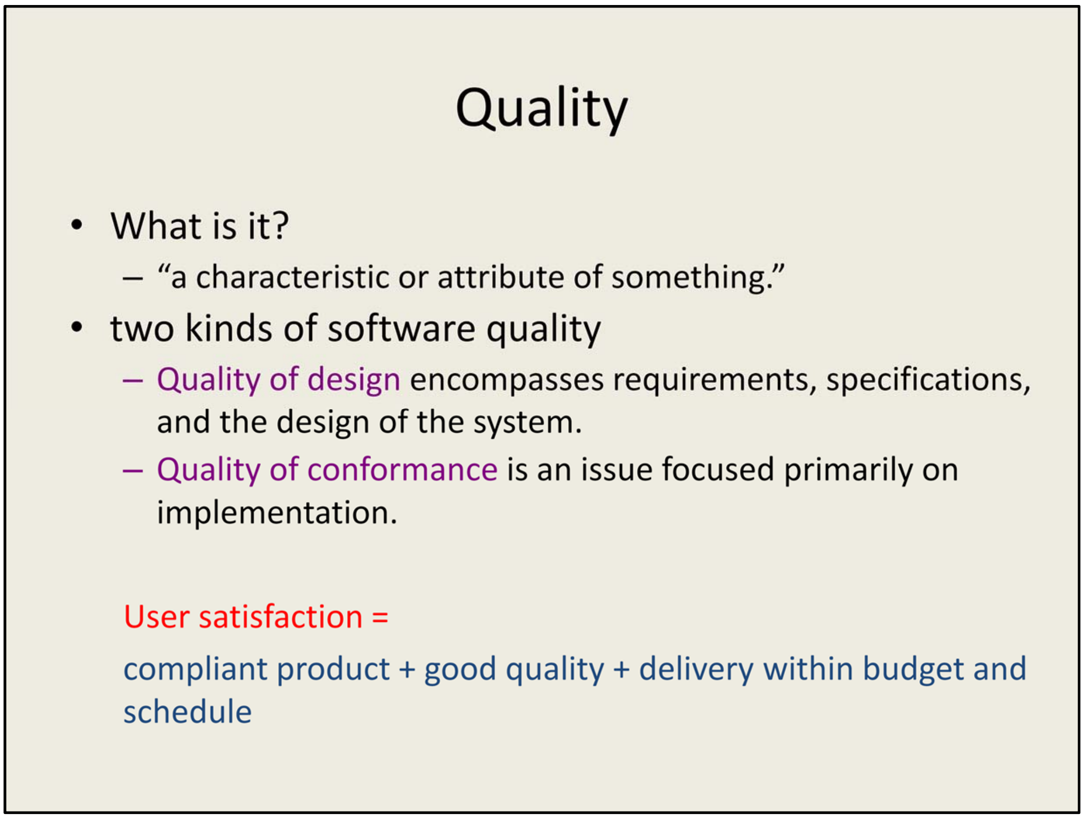
- Tools developed in Computer Science provide technologies which are necessary for fulfilling the goals of software development.

The Primary Goal of Any Software Process: *High Quality*

High quality = project timeliness

Why?

Less rework!



Quality

- What is it?
  - “a characteristic or attribute of something.”
- two kinds of software quality
  - Quality of design encompasses requirements, specifications, and the design of the system.
  - Quality of conformance is an issue focused primarily on implementation.

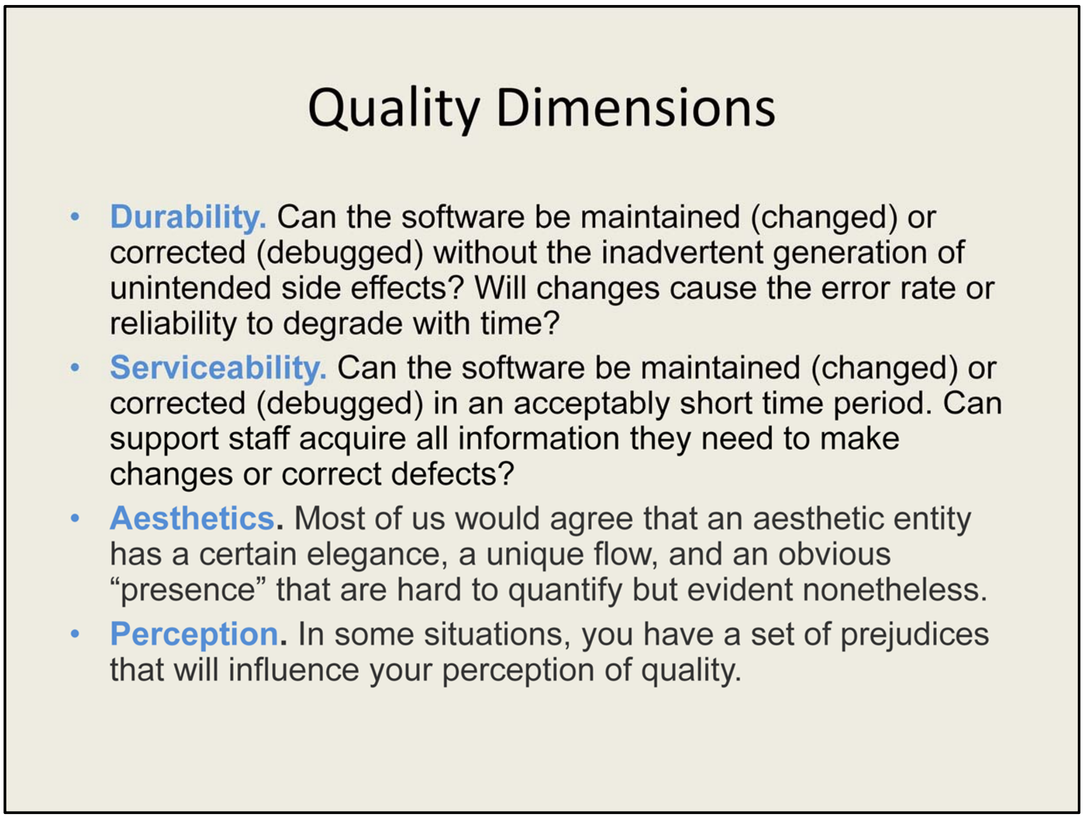
User satisfaction =

compliant product + good quality + delivery within budget and schedule

quality of design encompasses the degree to which the design meets the functions and features specified in the requirements model. *Quality of conformance* focuses on the degree to which the implementation follows the design and the resulting system meets its requirements and performance goals.

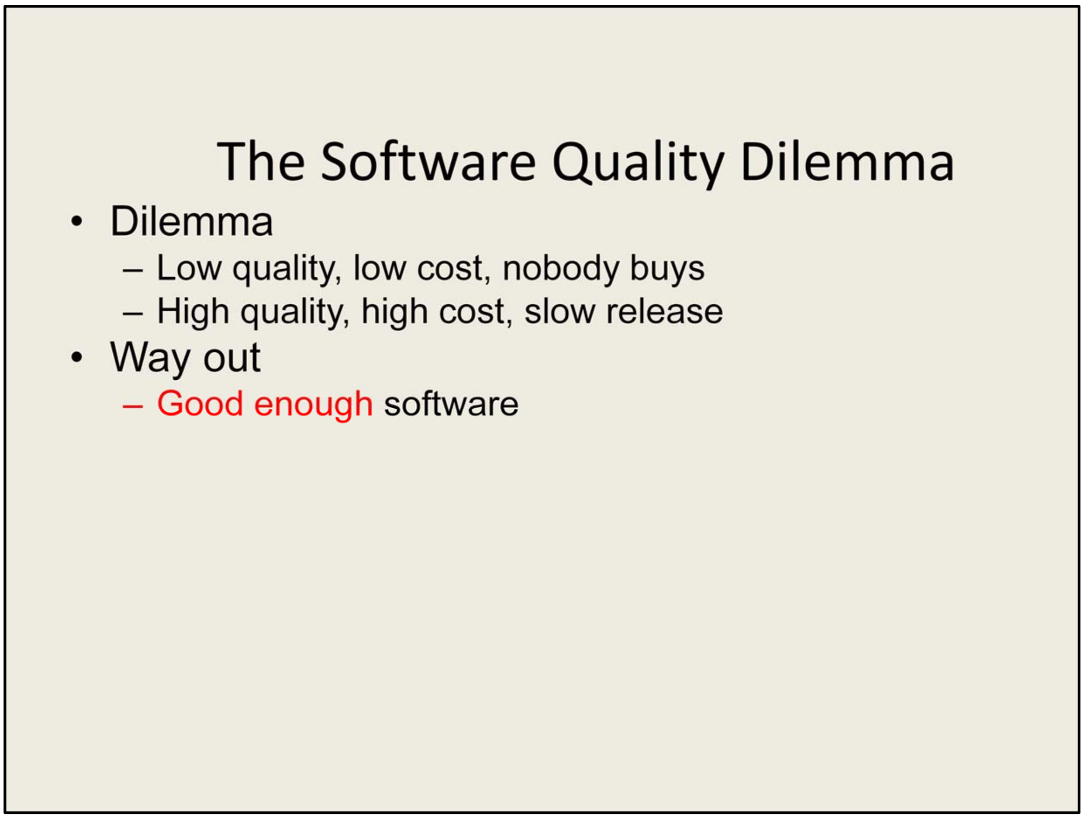
Quality Dimensions

- **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 de facto design and coding conventions? For example, does the user interface conform to accepted design rules for menu selection or data input?



Quality Dimensions

- **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.



The Software Quality Dilemma

- Dilemma
  - Low quality, low cost, nobody buys
  - High quality, high cost, slow release
- Way out
  - Good enough software

If you produce a software system that has terrible quality, you lose because no one will want to buy it.

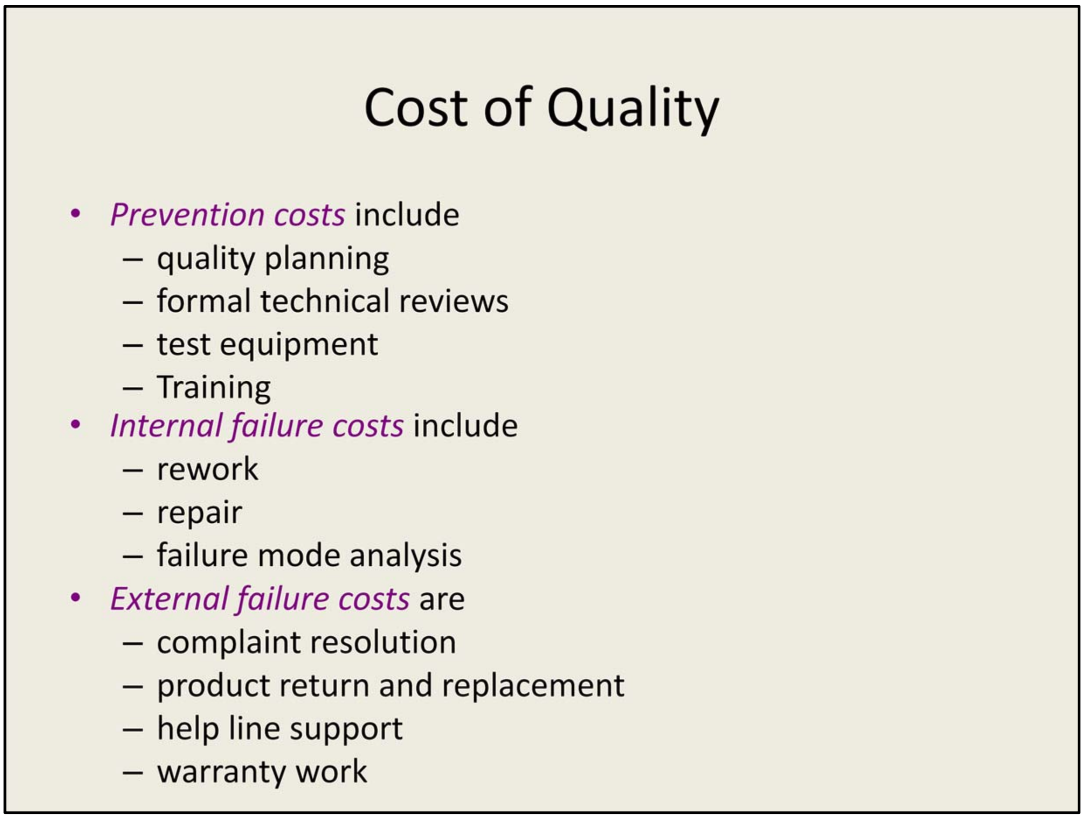
If on the other hand you spend infinite time, extremely large effort, and huge sums of money to build the absolutely perfect piece of software, then it's going to take so long to complete and it will be so expensive to produce that you'll be out of business anyway.
Either you missed the market window, or you simply exhausted all your resources.

So people in industry try to get to that magical middle ground where the product is good enough not to be rejected right away, such as during evaluation, but also not the object of so much perfectionism and so much work that it would take too long or cost too much to complete.

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.

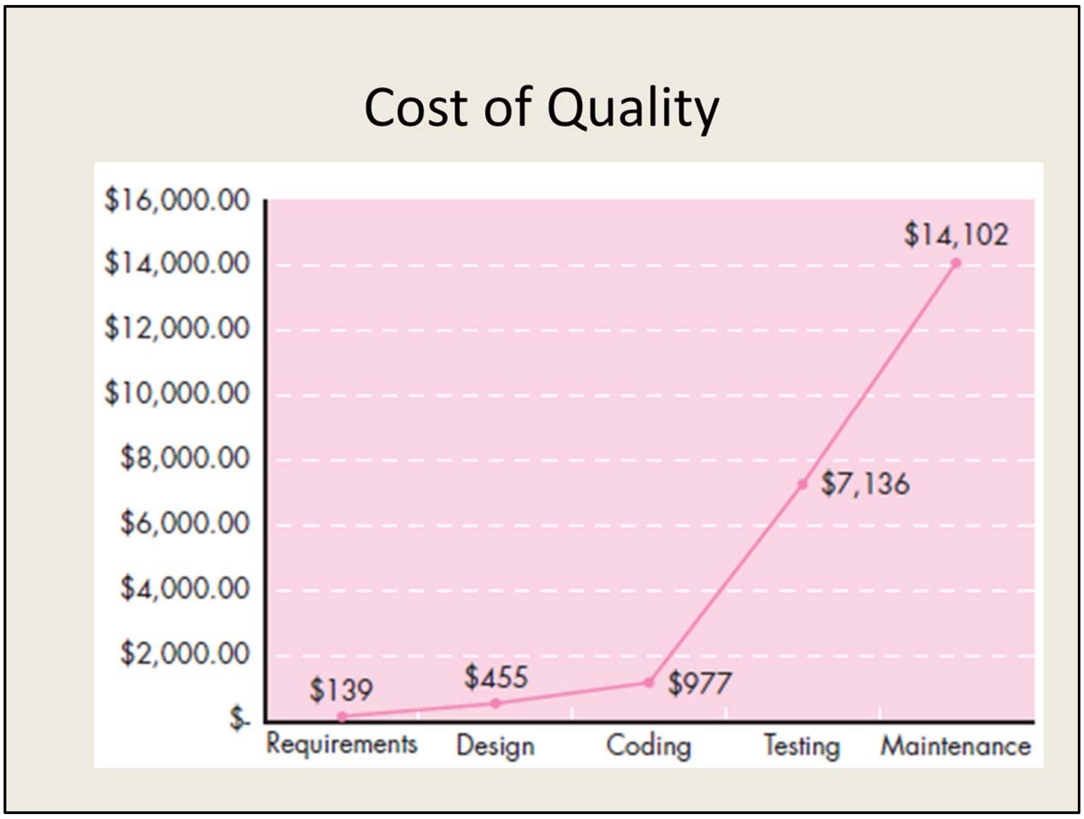
Arguments *against* “good enough.”

- It is true that “good enough” may work in some application domains and for a few major software companies. After all, 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.



Cost of Quality

- *Prevention costs* include
  - quality planning
  - formal technical reviews
  - test equipment
  - Training
- *Internal failure costs* include
  - rework
  - repair
  - failure mode analysis
- *External failure costs* are
  - complaint resolution
  - product return and replacement
  - help line support
  - warranty work

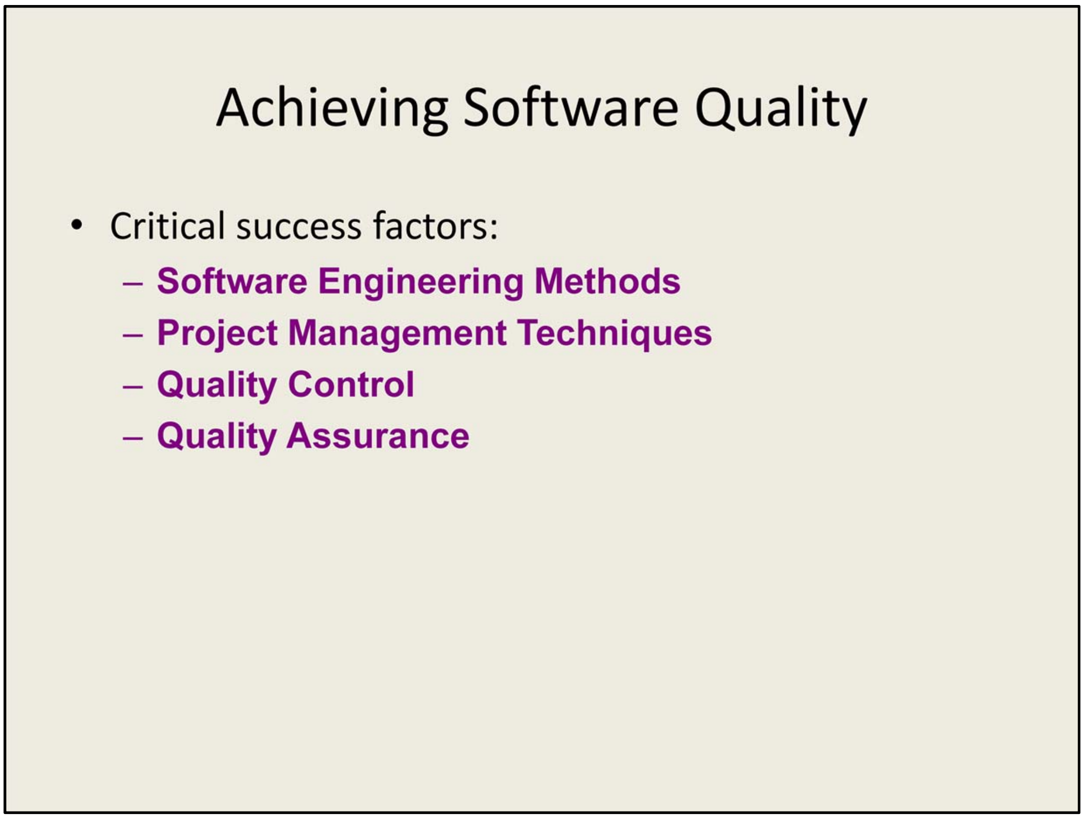


Cost of Quality

As expected, the relative costs to find and repair an error or defect increase dramatically

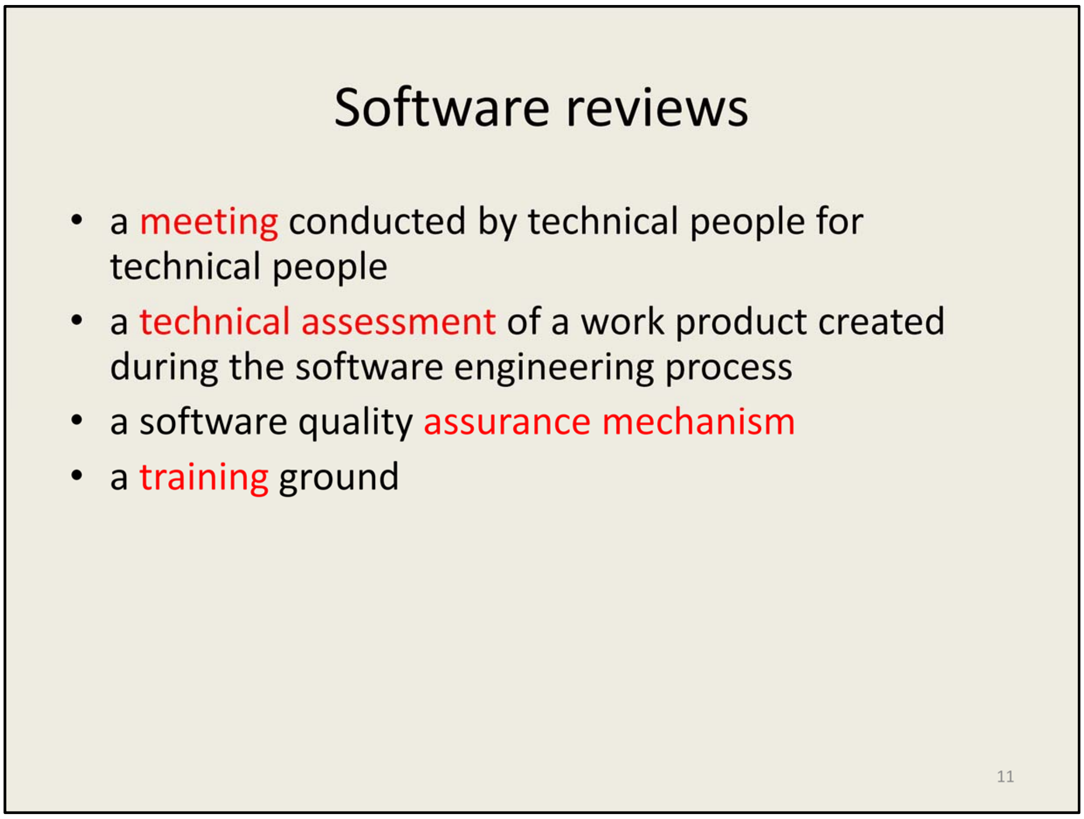
as we go from prevention to detection to internal failure to external failure costs.

Basically, the later the defect is discovered, the higher cost it takes to repair it.



Achieving Software Quality

- Critical success factors:
  - **Software Engineering Methods**
  - **Project Management Techniques**
  - **Quality Control**
  - **Quality Assurance**



Software reviews

- a meeting conducted by technical people for technical people
- a technical assessment of a work product created during the software engineering process
- a software quality assurance mechanism
- a training ground

11

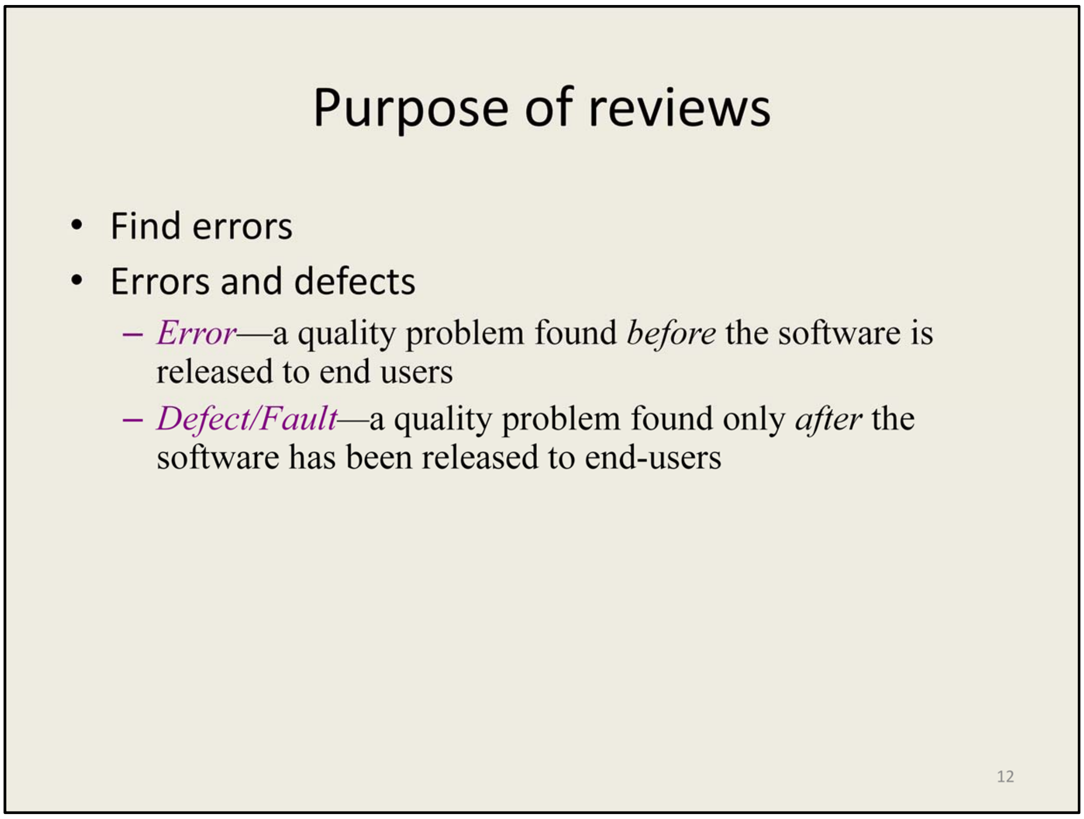
reviews are applied at various points during software engineering and serve to uncover

errors and defects that can then be removed

A project summary or progress assessment

A meeting intended solely to impart information

A mechanism for political or personal reprisal!



Purpose of reviews

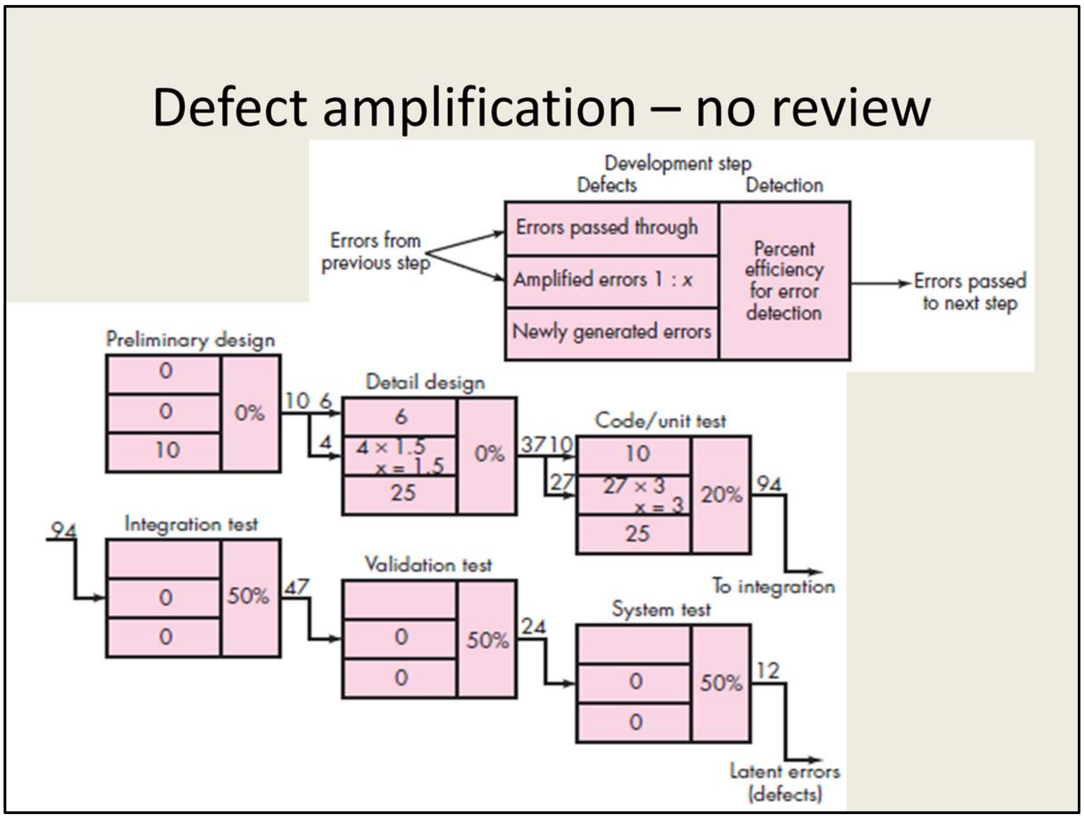
- Find errors
- Errors and defects
  - *Error*—a quality problem found *before* the software is released to end users
  - *Defect/Fault*—a quality problem found only *after* the software has been released to end-users

12

The primary objective of technical reviews is to find errors during the process so that they do not become defects after release of the software.

We make this distinction because errors and defects have very different economic, business, psychological, and human impact

However, the temporal distinction made between errors and defects in this book is *not* mainstream thinking



Defect amplification – no review

In some cases, errors passed through from previous steps are amplified (amplification factor, *x*) by current

work. The box subdivisions represent each of these characteristics and the percent of

efficiency for detecting errors, a function of the thoroughness of the review.

a software process that does NOT include reviews,

yields 94 errors at the beginning of testing and

Releases 12 latent defects to the field

Defect amplification - review

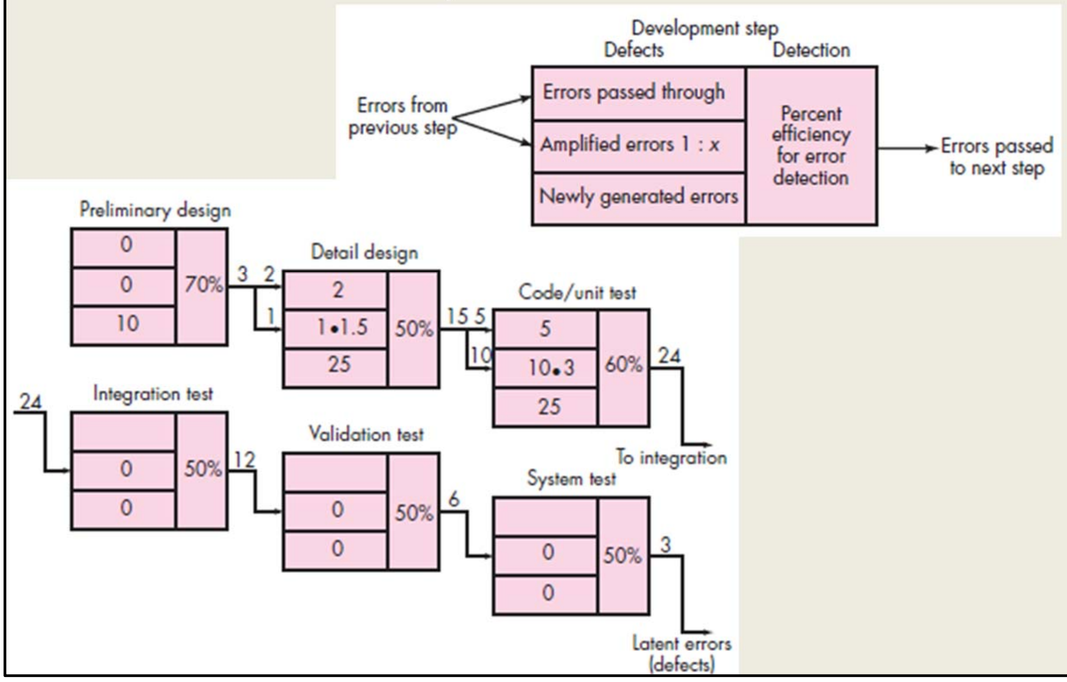


Figure 15.3 considers the same conditions except that design and

code reviews are conducted as part of each software engineering action. In this case,

10 initial preliminary (architectural) design errors are amplified to 24 errors before

testing commences. Only three latent errors exist. The relative costs associated with

the discovery and correction of errors, overall cost (with and without review for our

hypothetical example) can be established.

a software process that does include reviews,

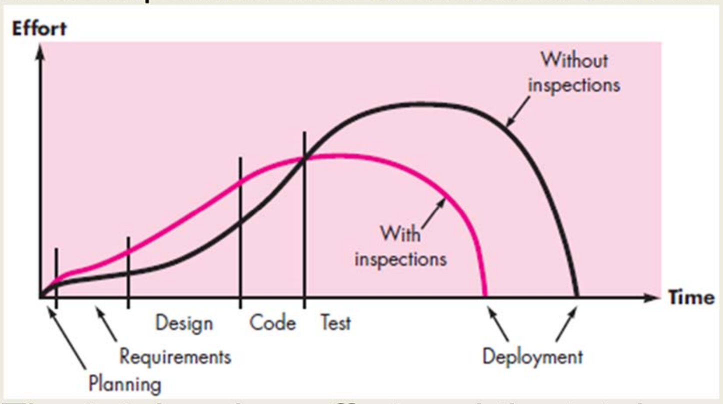
yields 24 errors at the beginning of testing and

releases 3 latent defects to the field

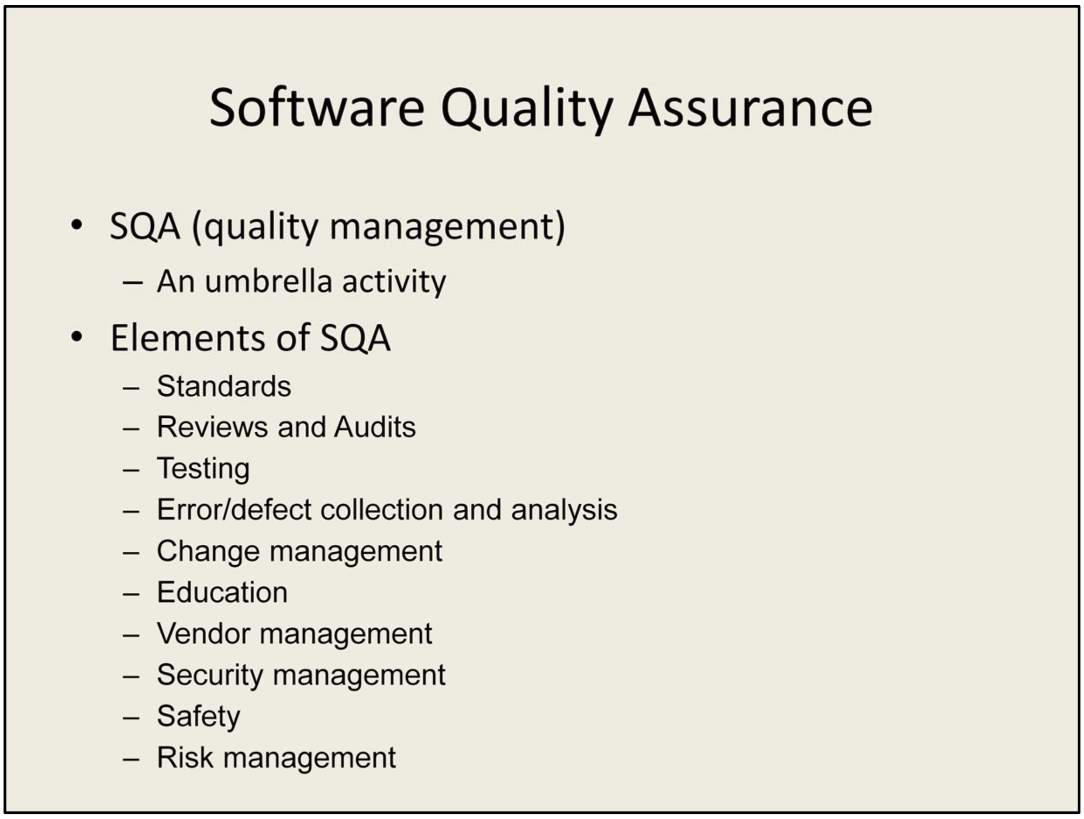
A cost analysis indicates that the process with NO reviews costs approximately 3 times more than the process with reviews, taking the cost of correcting the latent defects into account

Metrics

- Effort expended with and without reviews



- The total review effort and the total number of errors discovered are defined as:
  - $E_{review} = E_p + E_a + E_r$
  - $Err_{tot} = Err_{minor} + Err_{major}$

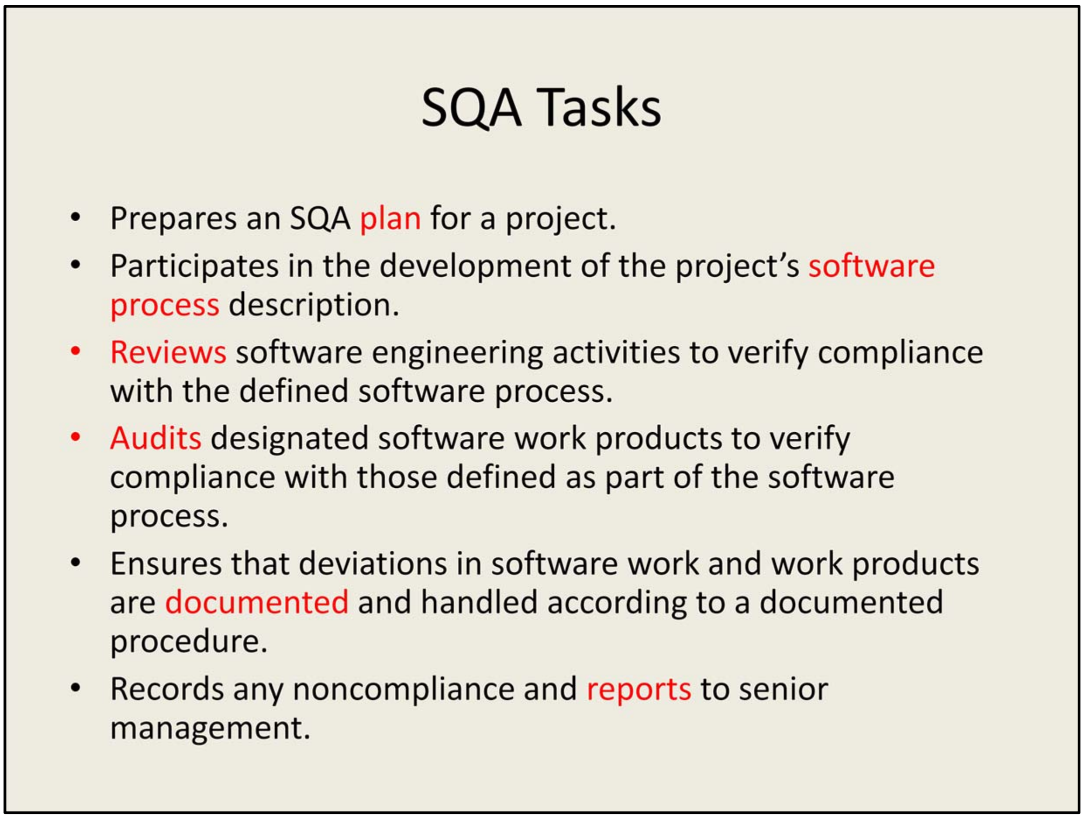


Software Quality Assurance

- SQA (quality management)
  - An umbrella activity
- Elements of SQA
  - Standards
  - Reviews and Audits
  - Testing
  - Error/defect collection and analysis
  - Change management
  - Education
  - Vendor management
  - Security management
  - Safety
  - Risk management

Software quality assurance (often called *quality management*) is an umbrella activity (Chapter 2) that is applied throughout the software process.

Software quality assurance encompasses a broad range of concerns and activities that focus on the management of software quality. These can be summarized as these elements.



SQA Tasks

- Prepares an SQA plan for a project.
- Participates in the development of the project's software process description.
- Reviews software engineering activities to verify compliance with the defined software process.
- Audits designated software work products to verify compliance with those defined as part of the software process.
- Ensures that deviations in software work and work products are documented and handled according to a documented procedure.
- Records any noncompliance and reports to senior management.

The plan identifies

evaluations to be performed

audits and reviews to be performed

standards that are applicable to the project

procedures for error reporting and tracking

documents to be produced by the SQA group

amount of feedback provided to the software project team

Participates in the development of the project's software process description.

The SQA group reviews the process description for compliance with organizational policy, internal software standards, externally imposed standards (e.g., ISO-9001), and other parts of the software project plan.

Reviews software engineering activities to verify compliance with the defined software process.

identifies, documents, and tracks deviations from the process and verifies that corrections have been made.

Audits designated software work products to verify compliance with those defined as part of the software process.

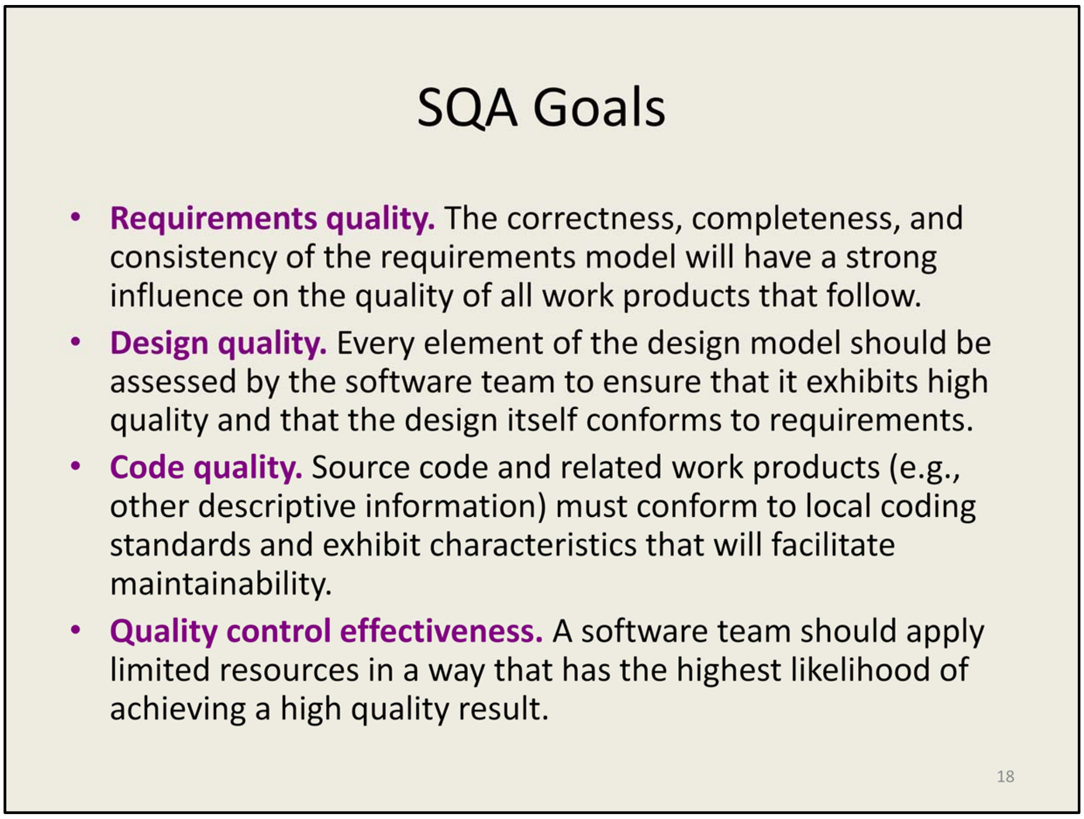
reviews selected work products; identifies, documents, and tracks deviations; verifies that corrections have been made

periodically reports the results of its work to the project manager.

Ensures that deviations in software work and work products are documented and handled according to a documented procedure.

Records any noncompliance and reports to senior management.

Noncompliance items are tracked until they are resolved.

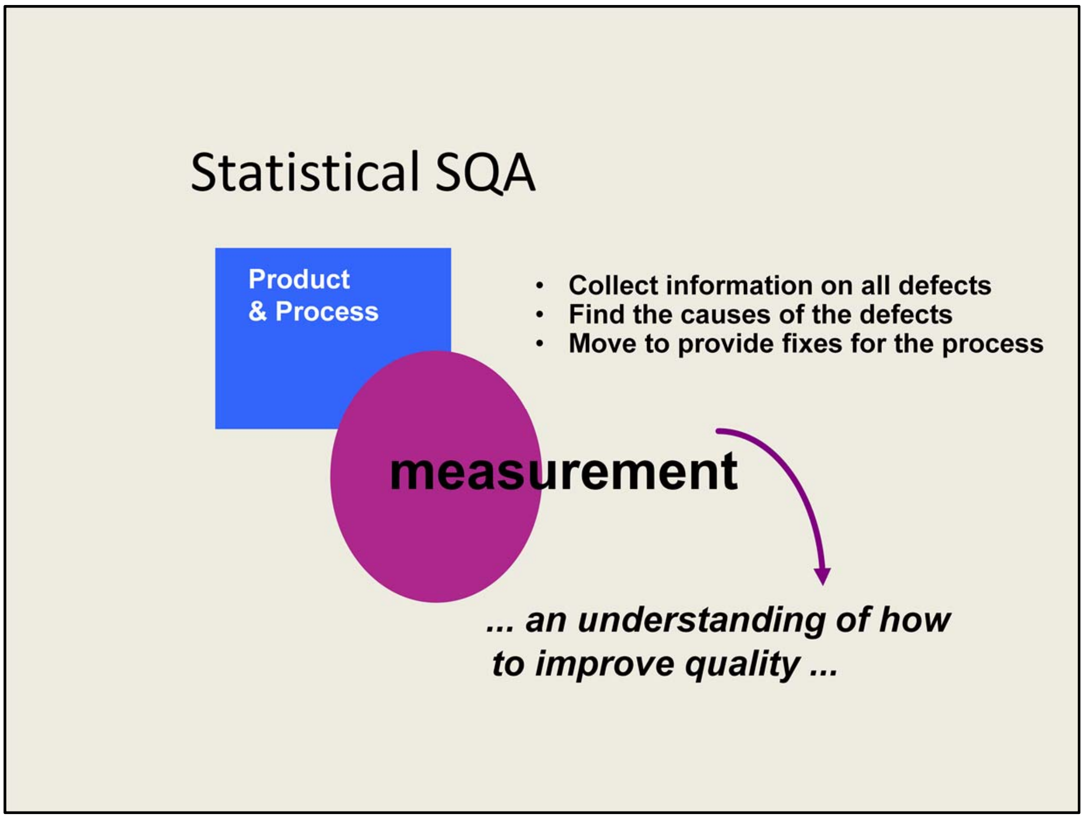


SQA Goals

- **Requirements quality.** The correctness, completeness, and consistency of the requirements model will have a strong influence on the quality of all work products that follow.
- **Design quality.** Every element of the design model should be assessed by the software team to ensure that it exhibits high quality and that the design itself conforms to requirements.
- **Code quality.** Source code and related work products (e.g., other descriptive information) must conform to local coding standards and exhibit characteristics that will facilitate maintainability.
- **Quality control effectiveness.** A software team should apply limited resources in a way that has the highest likelihood of achieving a high quality result.

18

There are detailed attributes to describe these goals, and concrete metrics for measuring each attribute.



Statistical SQA

Product & Process
measurement
... an understanding of how to improve quality ...

- Collect information on all defects
- Find the causes of the defects
- Move to provide fixes for the process

Statistical quality assurance reflects a growing trend throughout industry to become

more quantitative about quality. For software, statistical quality assurance implies

the following steps:

1. Information about software errors and defects is collected and categorized.

2. An attempt is made to trace each error and defect to its underlying cause

(e.g., nonconformance to specifications, design error, violation of standards,

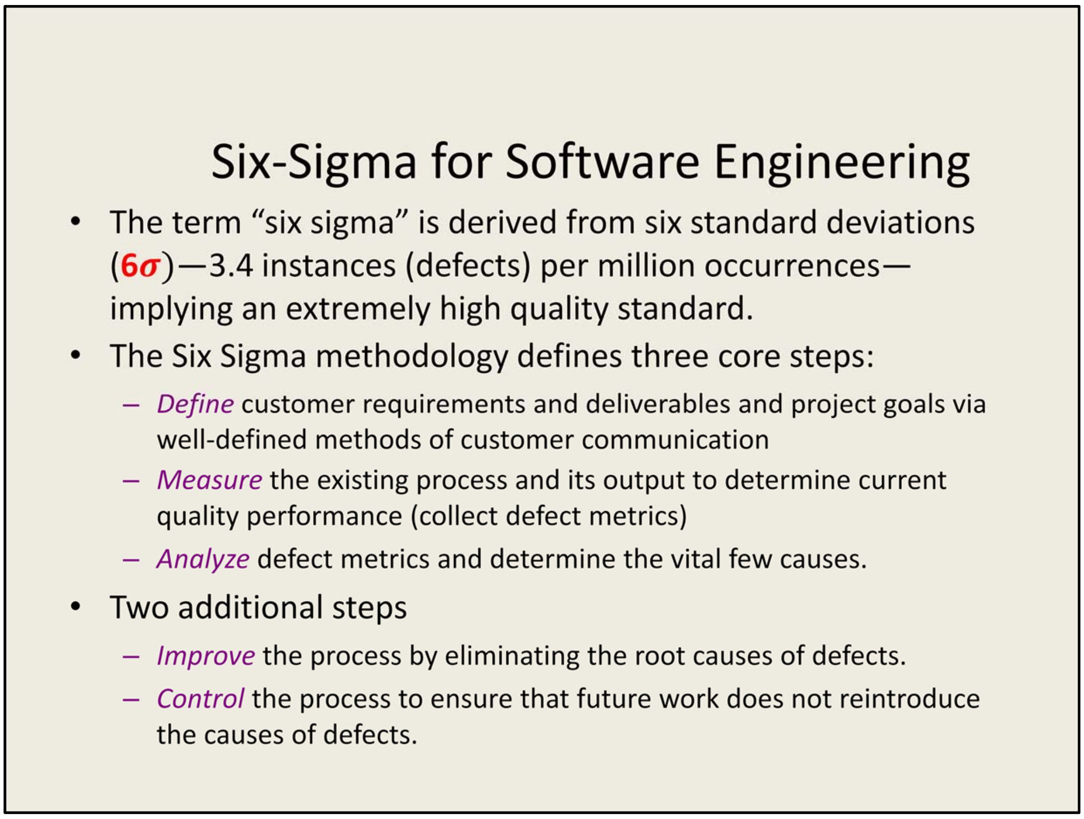
poor communication with the customer).

3. Using the Pareto principle (80 percent of the defects can be traced to 20 percent

of all possible causes), isolate the 20 percent (the *vital few*).

4. Once the vital few causes have been identified, move to correct the problems

that have caused the errors and defects.



Six-Sigma for Software Engineering

- The term “six sigma” is derived from six standard deviations (**6*σ***)—3.4 instances (defects) per million occurrences—implying an extremely high quality standard.
- The Six Sigma methodology defines three core steps:
  – *Define* customer requirements and deliverables and project goals via well-defined methods of customer communication
  – *Measure* the existing process and its output to determine current quality performance (collect defect metrics)
  – *Analyze* defect metrics and determine the vital few causes.
- Two additional steps
  – *Improve* the process by eliminating the root causes of defects.
  – *Control* the process to ensure that future work does not reintroduce the causes of defects.

Six Sigma is the most widely used strategy for statistical quality assurance in industry today.

is a rigorous and disciplined methodology that uses data and statistical analysis to

measure and improve a company’s operational performance by identifying and eliminating

defects’ in manufacturing and service-related processes.

standard deviations between the mean and the *nearest specification limit.*

Software Reliability

- A simple measure of reliability is *mean-time-between-failure* (MTBF), where

  MTBF = MTTF + MTTR

- The acronyms MTTF and MTTR are *mean-time-to-failure* and *mean-time-to-repair*, respectively.
- *Software availability* is the probability that a program is operating according to requirements at a given point in time and is defined as

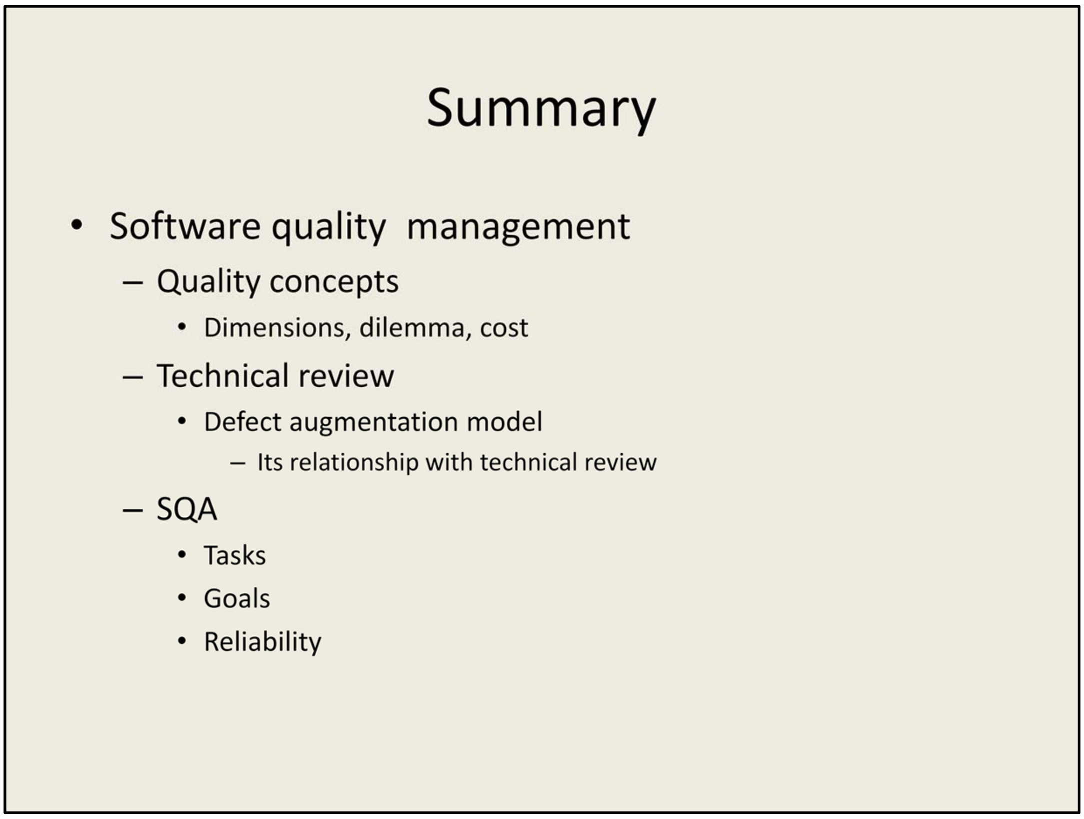
  Availability = [MTTF/(MTTF + MTTR)] x 100%

Software Safety

- *Software safety* is a software quality assurance activity that focuses on the identification and assessment of potential hazards that may affect software negatively and cause an entire system to fail.
- If hazards can be identified early in the software process, software design features can be specified that will either eliminate or control potential hazards.

ISO 9001:2000 Standard

- ISO 9001:2000 is the quality assurance standard that applies to software engineering.
- The standard contains 20 requirements that must be present for an effective quality assurance system.
- The requirements delineated by ISO 9001:2000 address topics such as
  - management responsibility, quality system, contract review, design control, document and data control, product identification and traceability, process control, inspection and testing, corrective and preventive action, control of quality records, internal quality audits, training, servicing, and statistical techniques.



Summary

- Software quality management
  - Quality concepts
    - Dimensions, dilemma, cost
  - Technical review
    - Defect augmentation model
      - Its relationship with technical review
  - SQA
    - Tasks
    - Goals
    - Reliability