Software Engineering

Lecture 12

Software Quality Assurance

Quality Concepts

1. Quality

- Quality as “a characteristic or attribute of something.”
- Two kinds of quality may be encountered:
  - *Quality of design* of a product increases, if the product is manufactured according to specifications.
  - *Quality of conformance* is the degree to which the design specifications are followed during manufacturing.
- In software development,
  - 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

2. Quality Control

- *Quality control* involves the series of inspections, reviews, and tests used throughout the software process.
- Quality control includes a feedback loop to the process.
- A key concept of quality control is that all work products have defined, measurable specifications to which we may compare the output of each process.
- The feedback loop is essential to minimize the defects produced.

3. Quality Assurance

- *Quality assurance* consists of the auditing and reporting functions of management.
- If the data provided through quality assurance identify problems, it is management’s responsibility to address the problems and apply the necessary resources to resolve quality issues.

4. Cost of Quality

- The *cost of quality* includes all costs incurred in the pursuit of quality or in performing quality-related activities.
- *Quality costs* may be divided 3 mode of cost:
  - Prevention
  - Appraisal
  - Failure.

- Prevention costs include
  - Quality planning
  - Formal technical reviews
  - Test equipment
  - Training
- *Appraisal costs* include activities to gain insight into product
  - In-process and Inter-process inspection
  - Equipment calibration and maintenance
  - Testing
- *Failure costs*
  - Internal Failure Cost
    - rework
    - repair
    - failure mode analysis
  - External Failure Cost
    - complaint resolution
    - product return and replacement
    - help line support
    - warranty work

Software Quality Assurance (SQA)

- Definition:
  - Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software.
- Definition serves to emphasize three important points:
  - Software requirements are the foundation from which quality is measured. Lack of conformance to requirements is lack of quality.
  - Specified standards define a set of development criteria, If the criteria are not followed, lack of quality will almost surely result.
  - A set of implicit requirements often goes unmentioned. If software conforms to its explicit requirements but fails to meet implicit requirements, software quality is suspect

SQA group activities

- SQA group made up of software engineers, project managers, customers, salespeople, and the individuals members.
- Software quality assurance is composed of two different constituencies.
  - Software engineers who do technical work
  - SQA group that has responsibility for quality assurance planning, oversight, record keeping, analysis, and reporting.
- Software engineers address quality activities by applying technical methods and measures, conducting formal technical reviews, and performing well-planned software testing.
- SQA group is to assist the software team in achieving a high quality end product.

Role of an SQA group

1. Prepares an SQA plan for a project.

- The plan is developed during project planning and is reviewed by all stakeholders.
- 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

2. 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.

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

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

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

- The SQA group reviews selected work products; identifies, documents, and tracks deviations; verifies that corrections have been made; and periodically reports the results of its work to the project manager.

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

- Deviations may be encountered in the project plan, process description, applicable standards, or technical work products.

6. **Records any noncompliance and reports to senior management.**

- Noncompliance items are tracked until they are resolved.

Software Review

- Software reviews are a "filter" for the software engineering process.
- That is, reviews are applied at various points during software development and serve to uncover errors and defects that can then be removed.
- Types of Review
  - Informal Review
    - Meeting around the coffee machine and discussing technical problems.
  - Formal Review
    - Formal presentation of software design to an audience of customers, management, and technical staff
- A FTR is the most effective filter from a quality assurance standpoint.

Cost Impact of Software Defects

- The primary objective of formal technical reviews is to find errors during the process so that they do not become defects after release of the software.
- Direct benefit is early discovery of error, do not propagate to the next step.
- Design activities introduce between 50 and 65 percent of all errors (and ultimately, all defects) during the software process.
- However, FTR have been shown to be up to 75 percent effective in uncovering design flaws.
- FTP substantially reduces the cost of subsequent steps in the development and support phases.

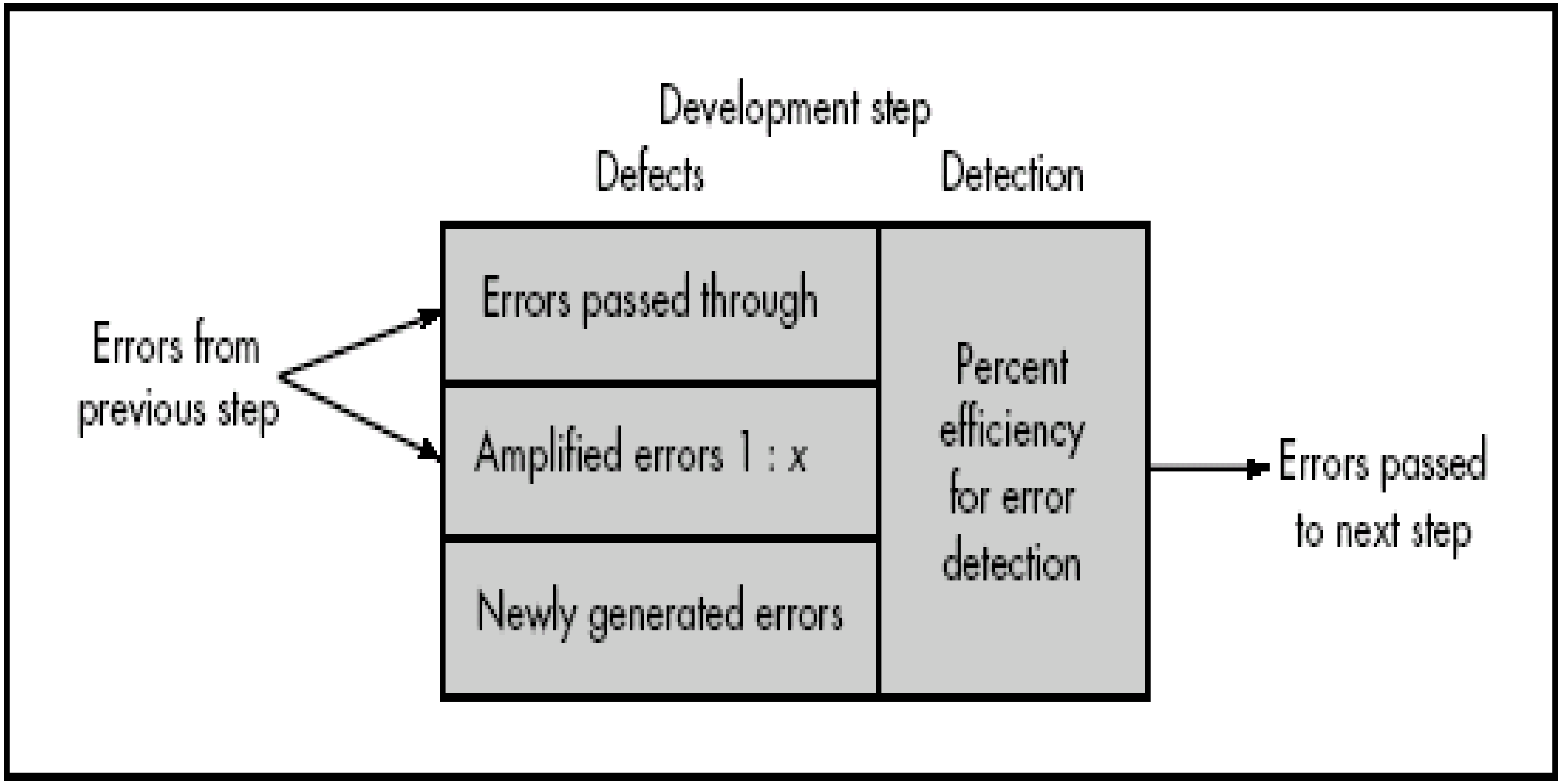
Contd.

- Assume that an error uncovered during design will cost 1.0 monetary unit to correct.
- Relative to this cost, the same error uncovered just before testing commences will cost 6.5 units; during testing, 15 units; and after release, between 60 and 100 units.

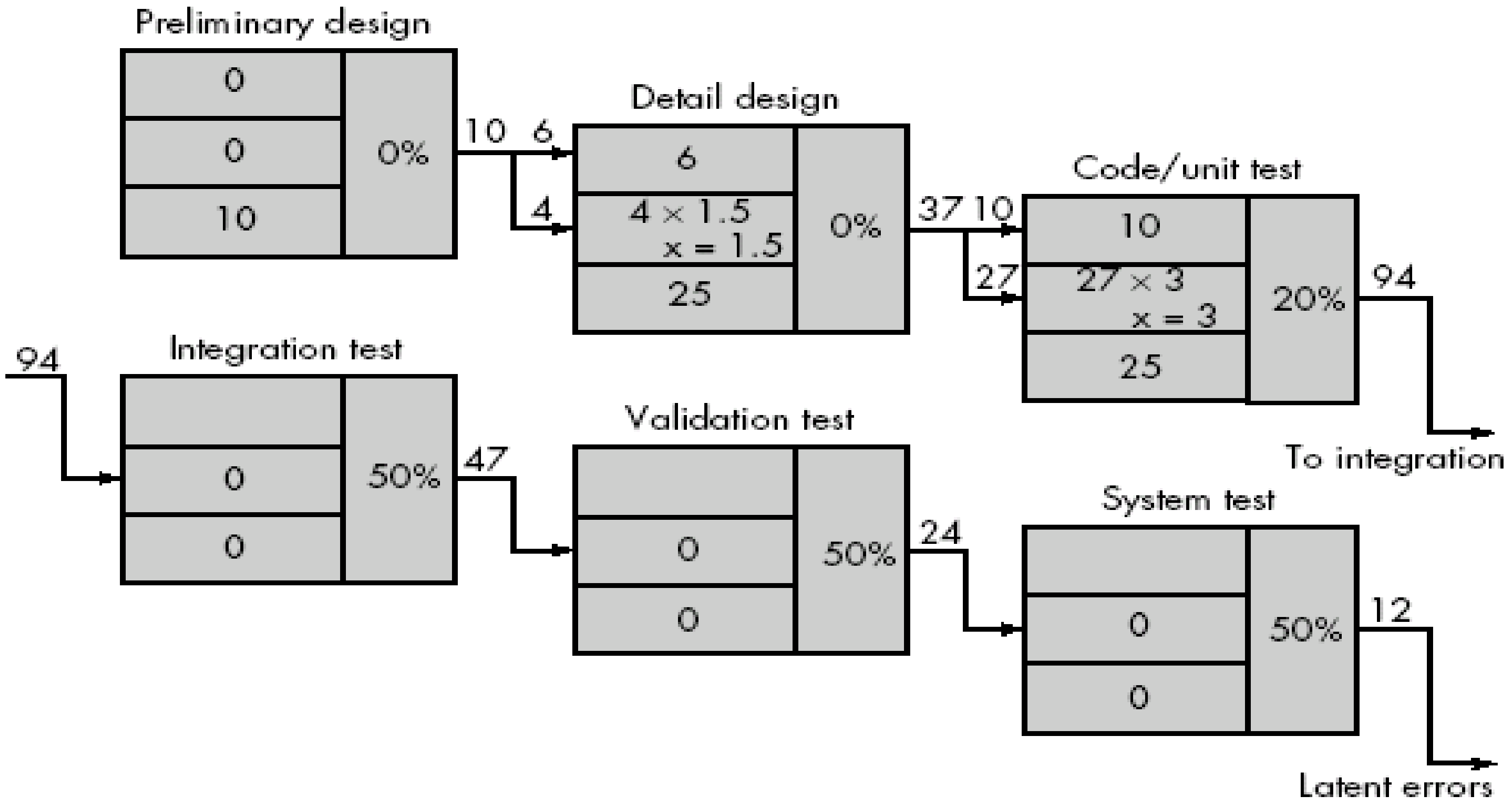
Defect Amplification and Removal

- A defect amplification model can be used to illustrate the generation and detection of errors during the preliminary design, detail design, and coding steps of the software engineering process.
- A box represents a software development step. During the step, errors may be by mistake generated.
- Review may fail to uncover newly generated errors and errors from previous steps, resulting in some number of errors that are passed through.
- 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.

Defect Amplification model

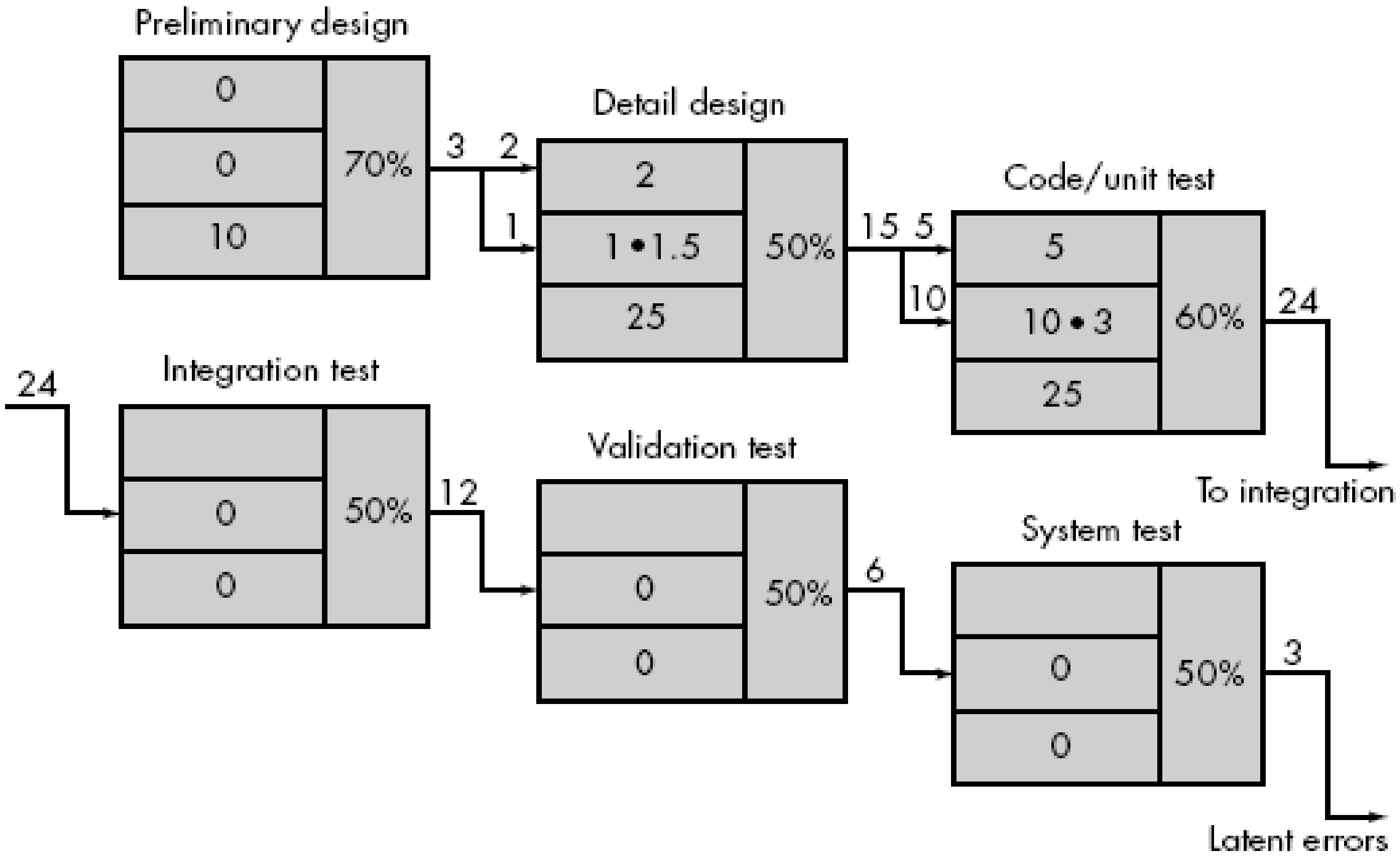


Defect Amplification – No Review



- Because of no review conducted, Ten preliminary design defects are amplified to 94 errors before testing commences.
- Each test step is assumed to correct 50 percent of all incoming errors without introducing any new errors.
- Twelve latent errors are released to the field or to the customers.

Defect Amplification –Review conducted



- Considers the same conditions except that design and code reviews are conducted as part of each development step.
- In this case, ten initial preliminary design errors are amplified to 24 errors before testing commences.
- Only three latent errors exist.
- Now we can estimate overall cost (with and without review for our hypothetical example)
- Using these data, the total cost for development and maintenance when reviews are conducted.
- To conduct reviews, a software engineer must expend time and effort and the development organization must spend money.

FORMAL TECHNICAL REVIEWS

- It is a software quality assurance activity performed by software engineers

Objectives of the FTR are

- To uncover errors in function, logic, or implementation for any representation of the software;
- To verify that the software under review meets its requirements;
- To ensure that the software has been represented according to predefined standards;
- To achieve software that is developed in a uniform manner;
- To make projects more manageable.

- The FTR is actually a class of reviews that includes walkthroughs, inspections, round-robin reviews and other small group technical assessments of software.

FTR- Review meeting

- Every review meeting should abide by the following constraints:
- Between three and five people (typically) should be involved in the review.
- Advance preparation should occur but should require no more than two hours of work for each person.
- The duration of the review meeting should be less than two hours.
- FTR focuses on a specific (and small) part of the overall software.
- e.g. rather than attempting to review an entire design, are conducted for each component or small group of components.

- The focus of the FTR is on a work product. (e.g. requirement, anlaysis,design, coding)
- *The producer*—informs the project leader that the work product is complete and that a review is required.
- The project leader contacts a *review leader,* who evaluates the product for readiness, generates copies of product materials, and distributes them to two or three reviewers for advance preparation.
- Each reviewer is expected to spend between one and two hours reviewing the product & making notes.
- Concurrently, the review leader also reviews the product and establishes an agenda for the review meeting, which is typically scheduled for the next day.
- The FTR begins with an introduction of the agenda and a brief introduction by the producer.
- The producer then proceeds to "walk through" the work product, explaining the material, while reviewers raise issues based on their advance preparation.

Contd.

- When valid problems or errors are discovered, the recorder notes each.
- At the end of the review, all attendees of the FTR must decide whether to
  - Accept the product without further modification,
  - Reject the product due to severe errors
  - Accept the product provisionally
- The decision made, all FTR attendees complete a sign-off, indicating their participation in the review and their concurrence with the review team's findings.

Review Reporting and Record Keeping

- During the FTR, a reviewer (the recorder) actively records all issues that have been raised.
- These are summarized at the end of the review meeting and a review issues list is produced.
- A *review summary report* answers three questions:
  1. What was reviewed?
  2. Who reviewed it?
  3. What were the findings and conclusions?
- *Review summary report* becomes part of the project historical record and may be distributed to the project leader and other interested parties.

- The *review issues list* serves two purposes:
  - To identify problem areas within the product
  - To serve as an action item checklist that guides the producer as corrections are made.
- An issues list is normally attached to the summary report.
- It is important to establish a follow-up procedure to ensure that items on the issues list have been properly corrected. Unless this is done, it is possible that issues raised can “fall between the cracks.”
- One approach is to assign the responsibility for follow-up to the review leader.

Review Guidelines

- *Review the product, not the producer.*
  - *Don't point out errors harshly. One way to be gentle is to ask a question that enables the producer to discover his or her own error.*
- *Set an agenda and maintain it.*
  - An FTR must be kept on track and on schedule.
- *Limit debate and rebuttal:*
  - Rather than spending time debating the question, the issue should be recorded for further discussion off-line
- *Enunciate problem areas, but don't attempt to solve every problem noted.*
  - Review only some small part of component.
- *Take written notes.*
  - make notes on a wall board, so that wording and priorities can be assessed by other reviewers

- *Limit the number of participants and insist upon advance preparation.*
  - Keep the number of people involved to the necessary minimum. However, all review steam members must prepare in advance.
- *Develop a checklist for each product that is likely to be reviewed.*
  - helps the review leader to structure the FTR meeting and helps each reviewer to focus on important issues.
- *Allocate resources and schedule time for FTRs*
- *Conduct meaningful training for all reviewers.*
  - To be effective all review participants should receive some formal training
- *Review your early reviews.*

SOFTWARE RELIABILITY

- *Software reliability* is defined in statistical terms as "the probability of failure-free operation of a computer program in a specified environment for a specified time“.
- What is meant by the term *failure?*
  - In the context of any discussion of software quality and reliability, failure is nonconformance to software requirements.
  - Correction of one failure may in fact result in the introduction of other errors that ultimately result in other failures.
- Software reliability can be measured directed and estimated using historical and developmental data.

Measures of Reliability and Availability

- 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.

- MTBF is a far more useful measure than defects/KLOC or defects/FP.
- Stated simply, an end-user is concerned with failures, not with the total error count. Because each error contained within a program does not have the same failure rate, the total error count provides little indication of the reliability of a system.
- In addition to a reliability measure, we must develop a measure of availability.

- *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)] 100%
- The MTBF reliability measure is equally sensitive to MTTF and MTTR.
- The availability measure is somewhat more sensitive to MTTR, an indirect measure of the maintainability of software.