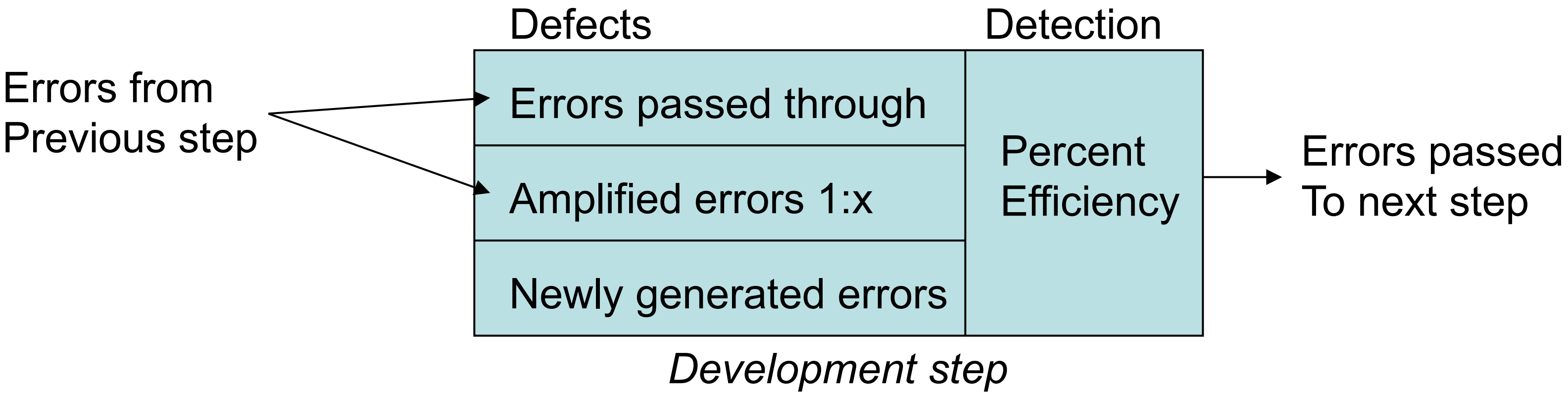
Ch.20 Review Techniques

20.1 Overview

- What Are 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
- Errors and defects
  - Error—a quality problem found before the software is released to end users
  - Defect—a quality problem found only after the software has been released to end-users
- However, the temporal distinction made between errors and defects in this book is not mainstream thinking

20.2 Defect Amplification and Removal

- **Defect Amplification Model**



Defects

Detection

Errors from Previous step

Errors passed through

Amplified errors 1:x

Newly generated errors

Percent Efficiency

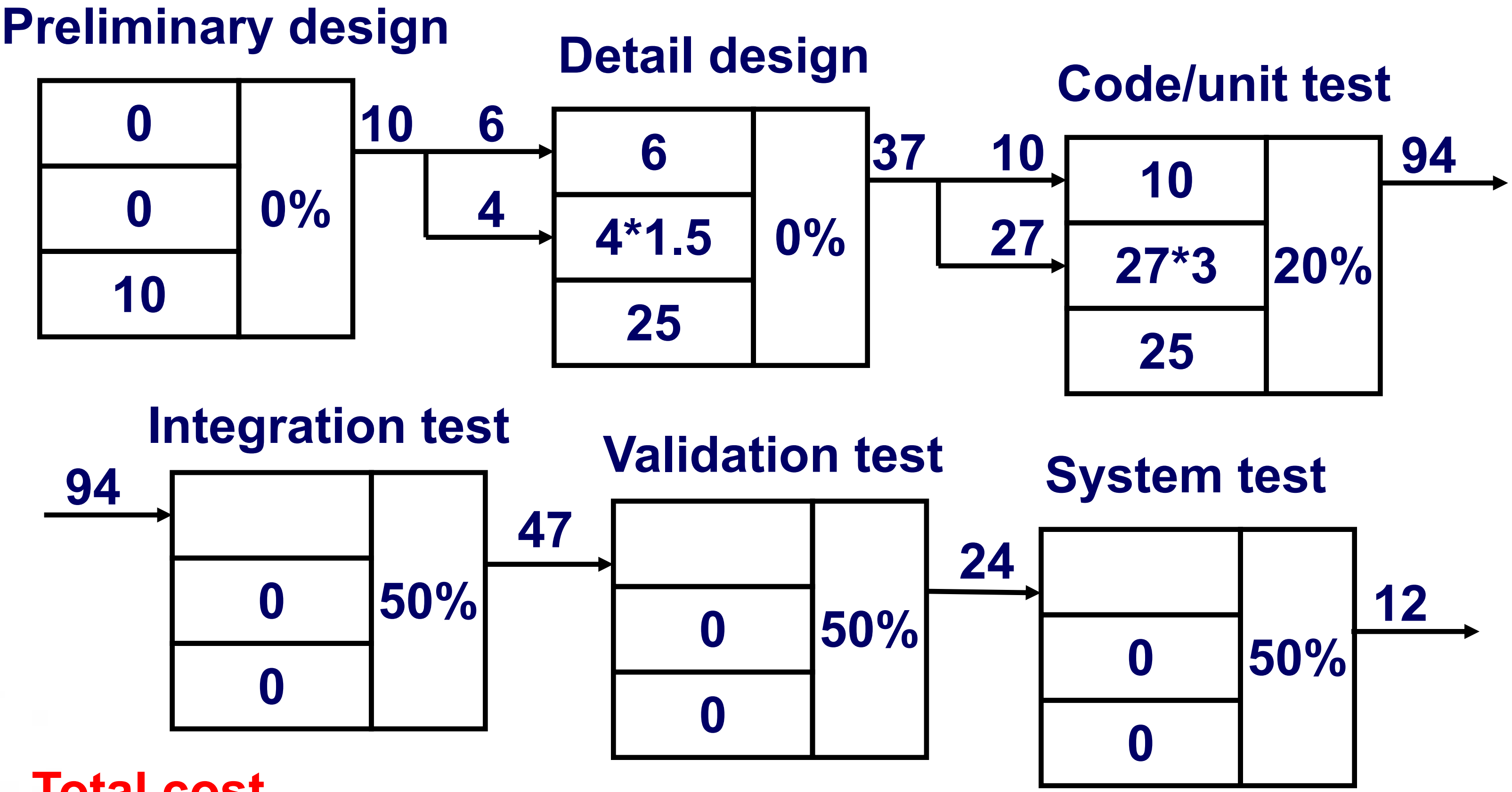
Errors passed To next step

Development step

- Assume that an error uncovered during **design** will cost **1.5** 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 **67** and **100** units.
- A number of studies indicate that **design activities** introduce between **50% - 65%** of all errors during the software process. However, **formal review technique** have been shown to be up to **75%** effective in uncovering design flaws.

20.2 Defect Amplification and Removal

- **Example: Defect Amplification No Reviews**



Preliminary design

0

0

10

0%

10

6

4

Detail design

6

4*1.5

25

0%

37

10

27

Code/unit test

10

27*3

25

20%

94

Integration test

94

0

0

50%

47

Validation test

0

0

50%

24

System test

0

0

50%

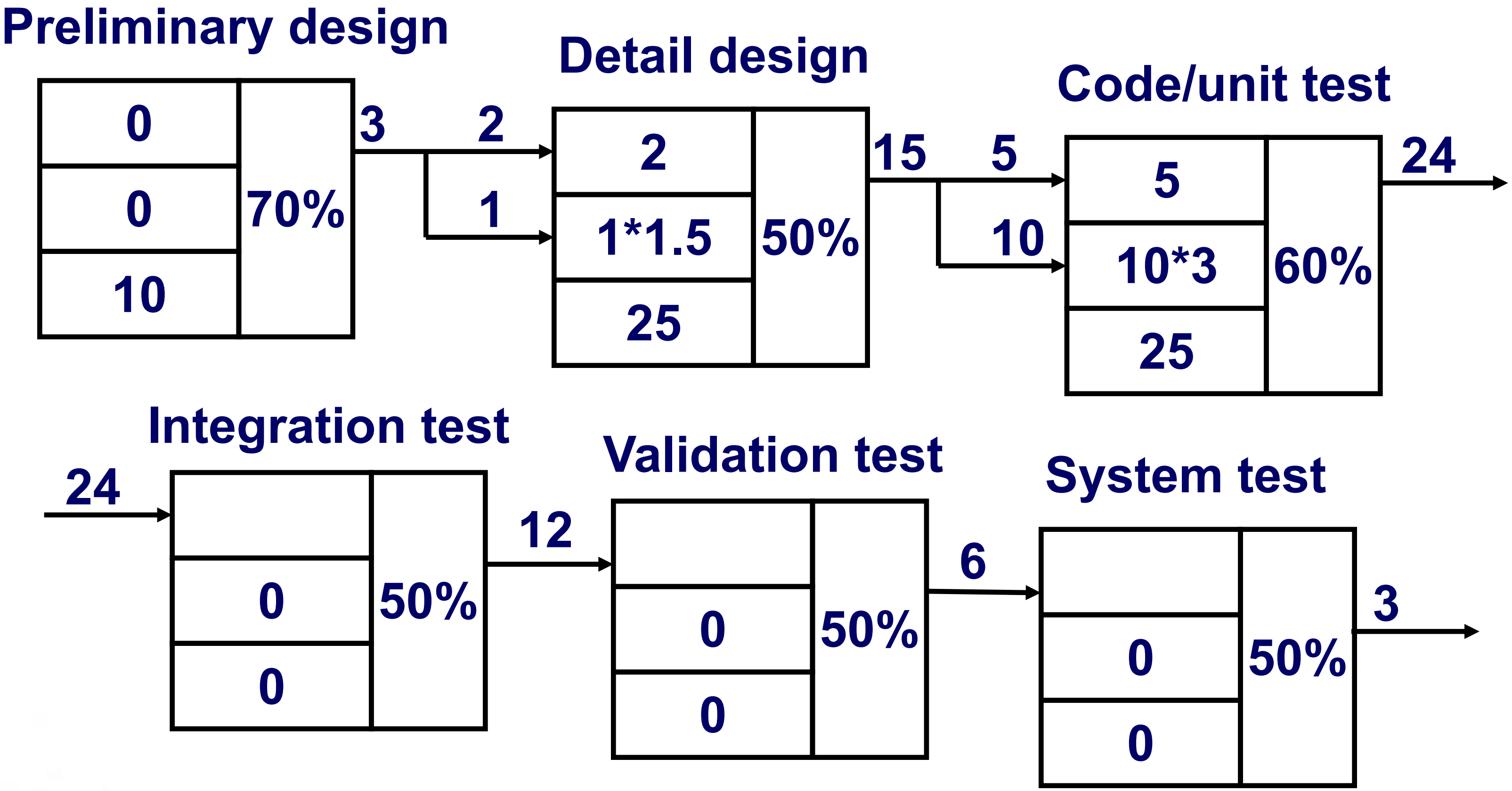
12

Total cost

= (10+27*3+25)*20%*6.5 + (94+47+24)*50%*15 + 12*67 = 2177

20.2 Defect Amplification and Removal

- **Example: Defect Amplification With Reviews**



Preliminary design

0

0

10

70%

3

2

1

Detail design

2

1*1.5

25

50%

15

5

10

Code/unit test

5

10*3

25

60%

24

Integration test

24

0

0

50%

12

Validation test

0

0

50%

6

System test

0

0

50%

3

Total cost = (10*70%+28.5*50%)*1.0 + (5+10*3+25)*60%*6.5 + (24+12+6)*50%*15 + 3*67 = 771

20.3 Review Metrics and Their Use

- The total review effort and the total number of errors discovered are defined as:
  - $E_{review} = E_p + E_a + E_r$
  - $Err_{tot} = Err_{minor} + Err_{major}$
- *Defect density* represents the errors found per unit of work product reviewed.
  - Defect density = Err_{tot} / WPS

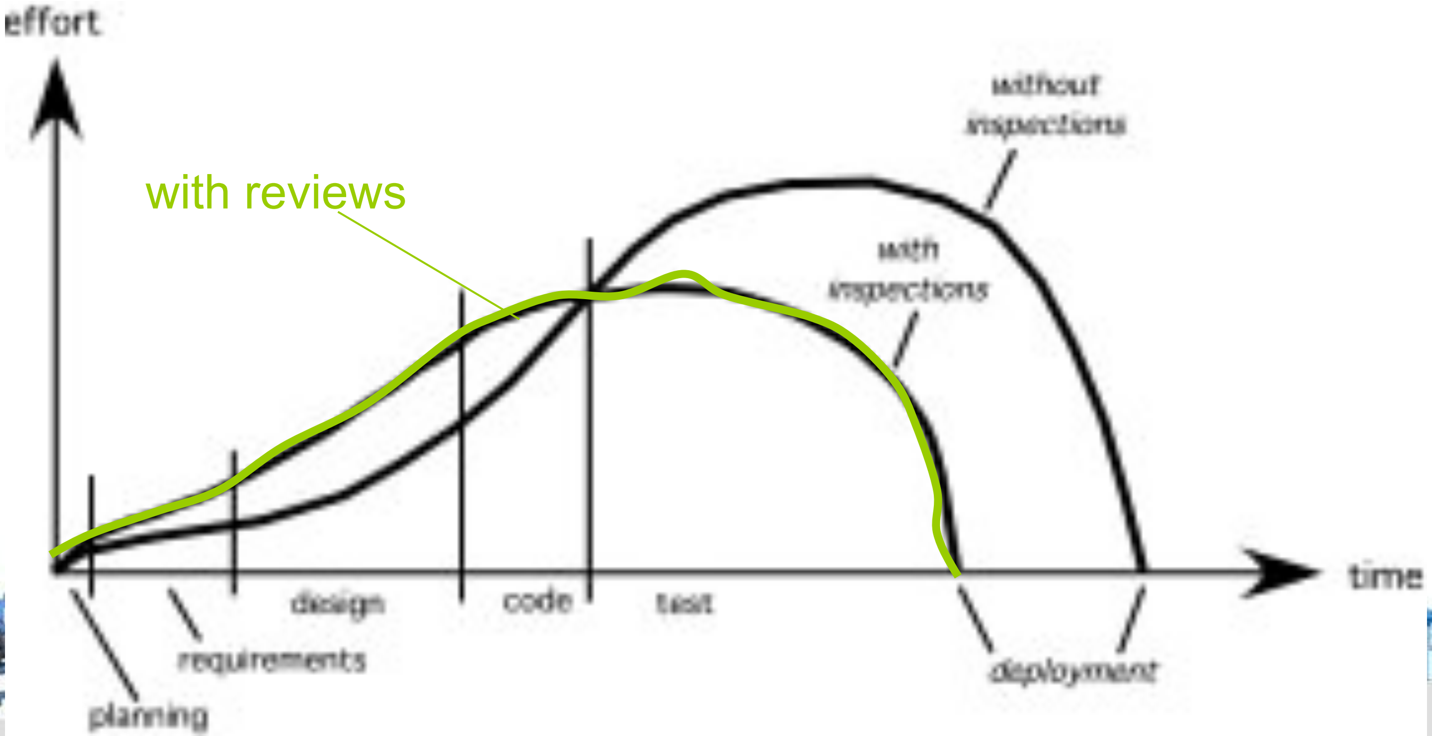
- *Preparation effort, E_p* — the effort (in person-hours) required to review a work product prior to the actual review meeting
- *Assessment effort, E_a* — the effort that is expending during the actual review
- *Rework effort, E_r* — the effort that is dedicated to the correction of those errors uncovered during the review
- *Work product size, WPS* — a measure of the size of the work product that has been reviewed (e.g., the number of UML models, or the number of document pages)
- *Minor errors found, Err_{minor}* — the number of errors found that can be categorized as minor (requiring less than some pre-specified effort to correct)
- *Major errors found, Err_{major}* — the number of errors found that can be categorized as major (requiring more than some pre-specified effort to correct)

Evaluate Saving: An Example—I

- The effort required to correct a minor model error (immediately after the review) was found to require 4 person-hours.
- The effort required for a major requirement error was found to be 18 person-hours.
- Examining the review data collected, you find that minor errors occur about 6 times more frequently than major errors. Therefore, you can estimate that the average effort to find and correct a requirements error during review is about 6 person-hours.
- Requirements related errors uncovered during testing require an average of 45 person-hours to find and correct. Using the averages noted, we get:
- Effort saved per error = $E_{testing} – E_{reviews}$
- $45 – 6 = 30$ person-hours/error
- Since 22 errors were found during the review of the requirements model, a saving of about 660 person-hours of testing effort would be achieved. And that' s just for requirements-related errors.

20.3 Review Metrics

- Effort expended with and without reviews
  - The effort expended when reviews are used does increase earl, but this early investment for reviews pays dividends because testing and corrective effort is reduced.
  - The development date with reviews is sooner than the development date without reviews. Reviews don’t take time, they save it.

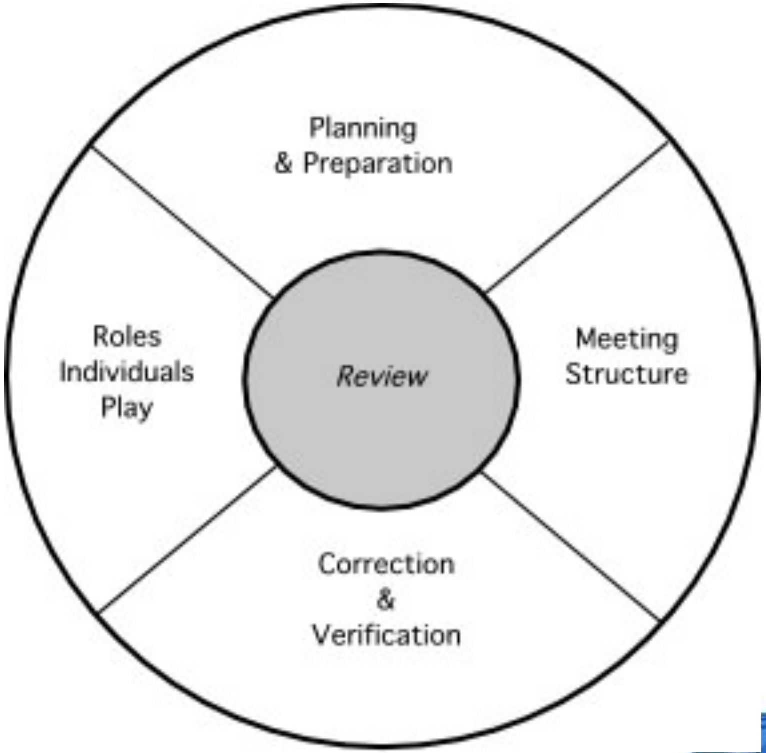


Prediction Work Performance: An Example—II

- If past history indicates that
  - the average defect density for a requirements model is 0.6 errors per page, and a new requirement model is 32 pages long,
  - a rough estimate suggests that your software team will find about 19 or 20 errors during the review of the document.
  - If you find only 6 errors, you've done an extremely good job in developing the requirements model *or* your review approach was not thorough enough.

20.4 Reference Model

- The formality of a review increases when:
  - Distinct roles are explicitly defined for the reviewers.
  - There is a sufficient amount of planning and preparation for the review.
  - A distinct structure for the review is defined.
  - Follow-up by the reviewers occurs for any corrections that are made.



20.5 Informal Reviews

- Informal reviews include:
  - a simple desk check of a software engineering work product with a colleague
  - a casual meeting (involving more than 2 people) for the purpose of reviewing a work product, or
  - the review-oriented aspects of pair programming
- *pair programming* encourages continuous review as a work product (design or code) is created.
  - The benefit is immediate discovery of errors and better work product quality as a consequence.

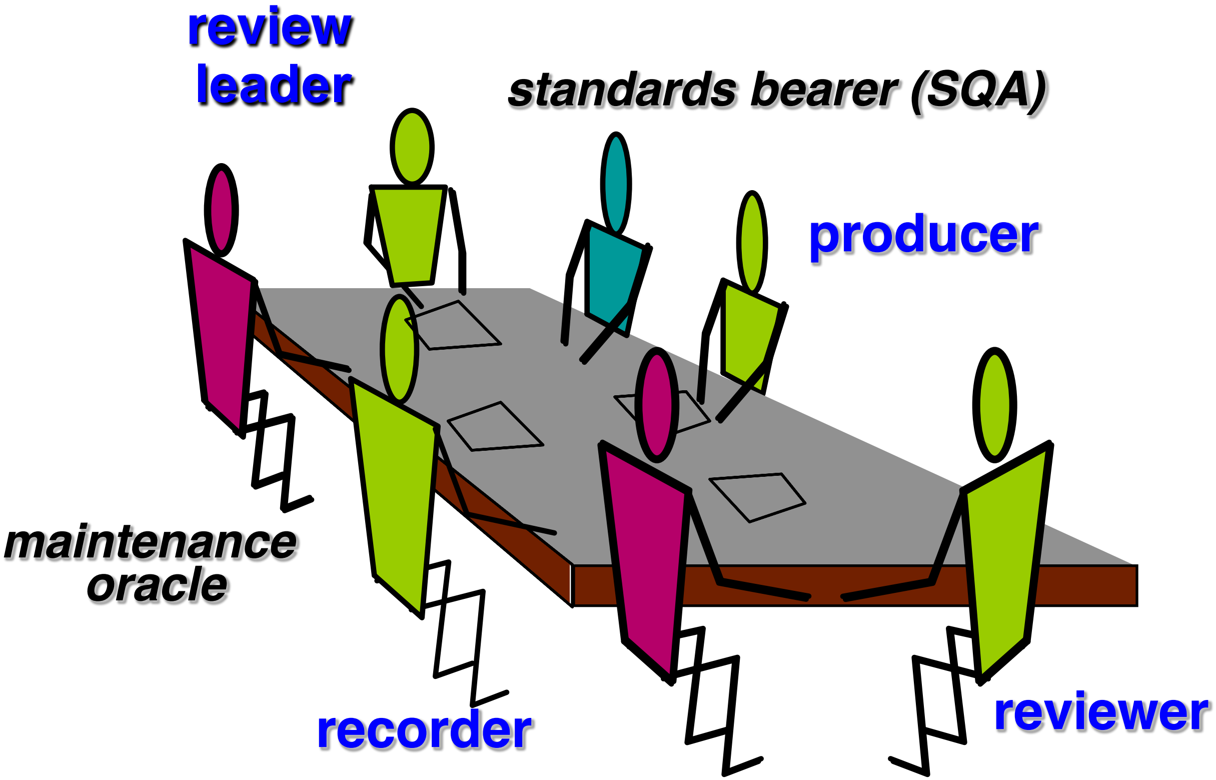
20.6 Formal Technical Reviews

- The objectives of an 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* and *inspections*.

20.6.1 The Review Meeting

- 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.
- Focus is on a work product (e.g., a portion of a requirements model, a detailed component design, source code for a component)

The Players


review leader
standards bearer (SQA)
producer
maintenance oracle
recorder
reviewer

Process

- ***Role of players***
  - *Producer*—the individual who has developed the work product
    - informs the project leader that the work product is complete and that a review is required
  - *Review leader*—evaluates the product for readiness, generates copies of product materials, and distributes them to two or three *reviewers* for advance preparation.
  - *Reviewer(s)*—expected to spend between one and two hours reviewing the product, making notes, and otherwise becoming familiar with the work.
  - *Recorder*—reviewer who records (in writing) all important issues raised during the review.
- ***Process***
  - *Preparation phase: producer->review leader->reviewers->problem list*
  - *Perform phase: producer introduction -> reviewers raise issue->recorder*
  - *Track phase: conclusion、SQA Report；*

20.6.3 Review Guidelines

- Review the product, not the producer.
- Set an agenda and maintain it.
- Limit debate and rebuttal.
- Enunciate problem areas, but don't attempt to solve every problem noted.
- Take written notes.
- Limit the number of participants and insist upon advance preparation.
- Develop a checklist for each product that is likely to be reviewed.
- Allocate resources and schedule time for FTRs.
- Conduct meaningful training for all reviewers.
- Review your early reviews.

20.6.4 Sample-Driven Reviews (SDRs)

- SDRs attempt to quantify those work products that are primary targets for full FTRs.

To accomplish this …

- Inspect a fraction a_i of each software work product, *i.* Record the number of faults, f_i found within a_i.
- Develop a gross estimate of the number of faults within work product *i* by multiplying f_i by $1/a_i$.
- Sort the work products in descending order according to the gross estimate of the number of faults in each.
- Focus available review resources on those work products that have the highest estimated number of faults.