Chapter 20

Review Techniques

Slide Set to accompany

Software Engineering: A Practitioner's Approach

by Roger S. Pressman

Reviews

... there is no particular reason why your friend and colleague cannot also be your sternest critic.

Jerry Weinberg

Table of Contents

20.1 Cost Impact of Software Defects

- Software bugs, or errors cost the U.S. economy an estimated $59.5 billion (= 60조원) annually, which is about 0.6% of the gross domestic product (NIST, 2002).

- The study also found that, although all errors cannot be removed, more than a third of these costs, or an estimated $22.2 billion, could be eliminated by an improved testing infrastructure.

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 (SQA) mechanism
- A training ground
- What Reviews Are **NOT**
  - A project summary or progress assessment
  - A meeting intended solely to impart information

What Do We Look For?

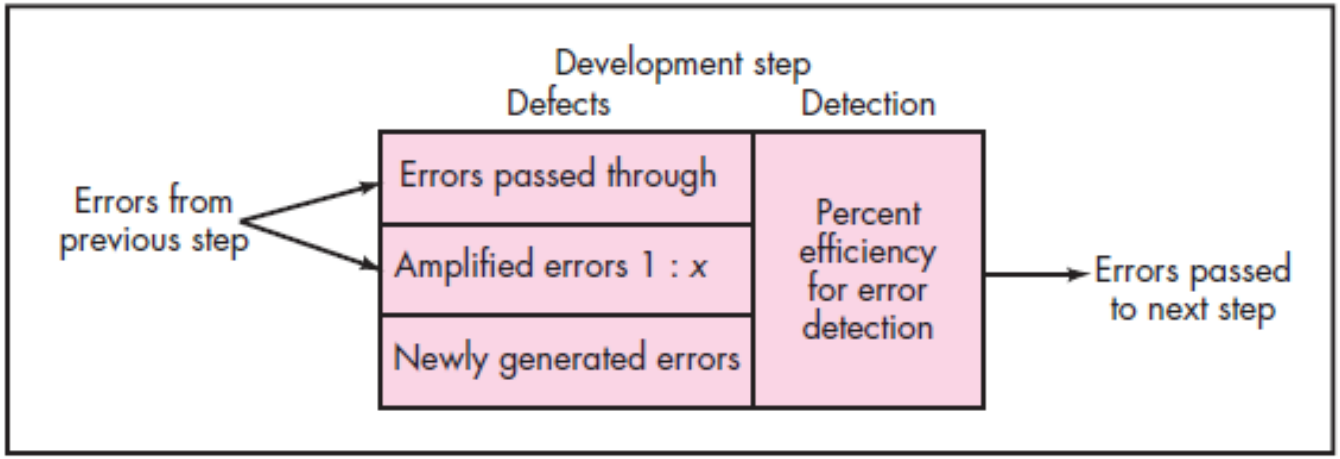
- Errors and defects
  - Error — a quality problem found before the software is released to end users
  - Defect — a quality problem found only after the software has been released to end-users
    - ☚ We make this distinction because errors and defects have very different economic, business, psychological, and human impact

- **More commonly,**

  Fault (= Defect) → Error → Failure (of Service)

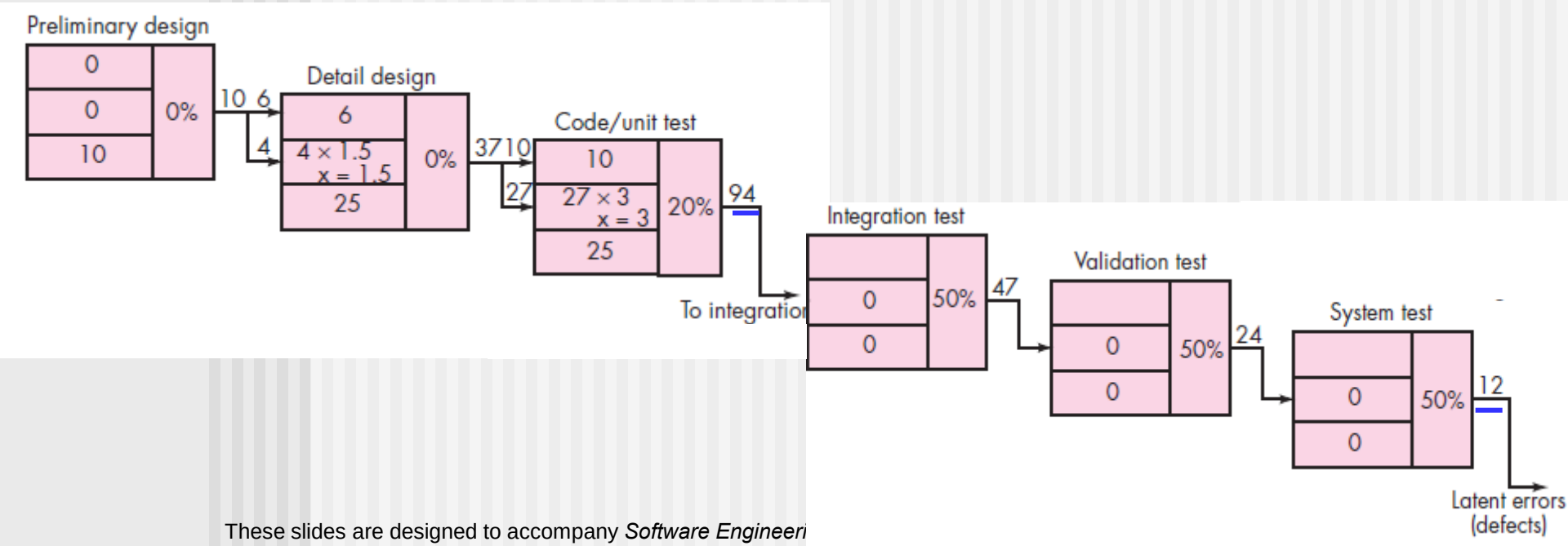
20.2 Defect Amplification and Removal

- A defect amplification model [IBM81]
  - Illustrates the generation and detection of errors during the design and code generation actions.



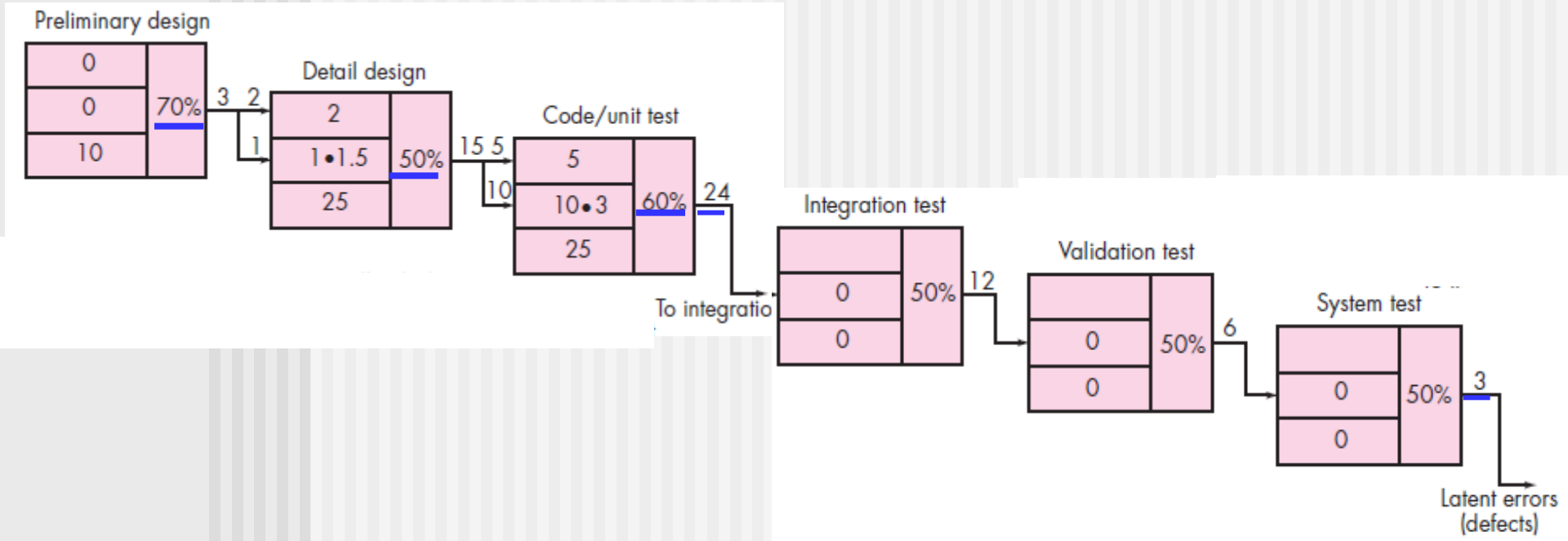
Example 1

- A software process that does **NOT** include reviews
  - yields 94 errors at the beginning of testing
  - releases 12 latent defects to the field



Example 2

- 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
  - 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

20.3 Review Metrics and Their Use

- 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 (in person-hours) that is expending during the actual review
- Rework effort, E_r— the effort (in person-hours) 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, or the number of lines of code)
- 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)

Metrics Derived from Reviews

- inspection time per page of documentation
- inspection time per KLOC or FP
- inspection effort per KLOC or FP
- errors uncovered per reviewer hour
- errors uncovered per preparation hour
- errors uncovered per SE task (e.g., design)
- number of minor errors (e.g., typos)
- number of major errors (e.g., nonconformance to req.)
- number of errors found during preparation

20.3.1 Analyzing Metrics

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

E_p: Preparation effort
E_a: Assessment effort
E_r: Rework effort
Err_{minor}: Minor errors found
Err_{major}: Major errors found

- Defect density represents the errors found per unit of work product reviewed.
  - Defect density = Err_{tot} / WPS

Err_{tot}: Total review effort
WPS: Work Product Size

Example 1

- If past history indicates that
  - the average defect density for a requirements model is

    0.6 errors / page

  and a new requirement model is

    32 pages long,

  then 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.

Example 2

- Effort to correct a minor model error (immediately after the review)
  –> 4 person-hours.
- Effort for a major requirement error
  –> 18 person-hours.
- You find that minor errors occur about 6 times more frequently than major errors.
  - The average effort to find and correct a requirements error is about 6 person-hours.
    (4 x 6 + 18 x 1) /7 = 6
- Requirements related errors uncovered during testing require an average of 45 person-hours to find and correct. Using the averages noted, we get:

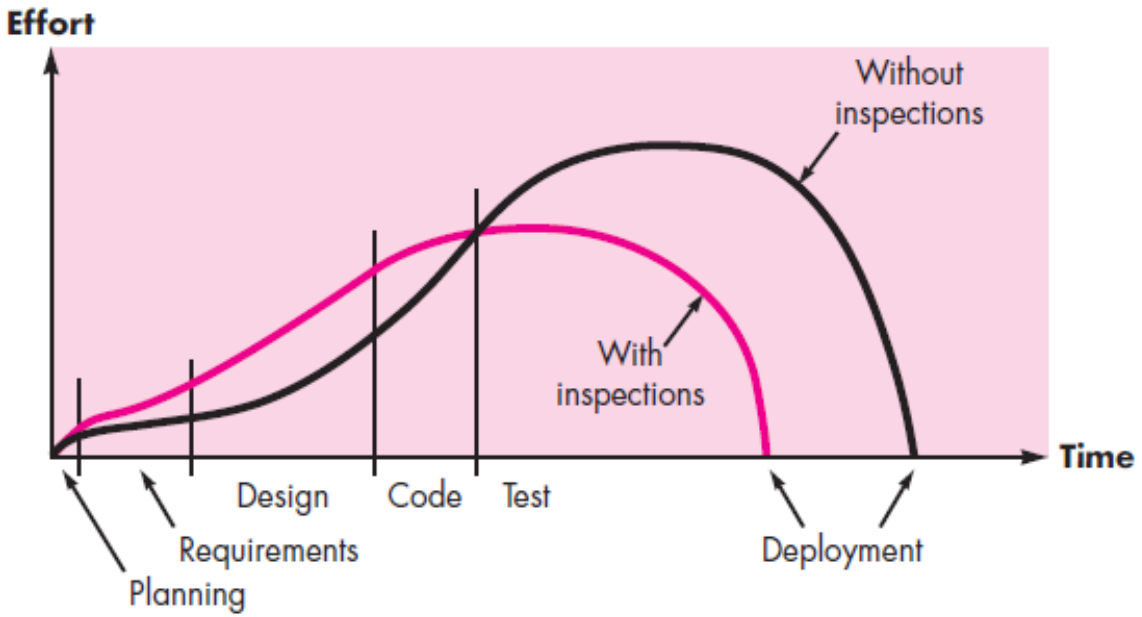
  Effort saved per error $= E_{testing} - E_{reviews}$

  45 – 6 = 39 person-hours/error
- Since 22 errors were found during the review of the requirements model, a saving of about 858 person-hours of testing effort would be achieved.
  And that's just for requirements-related errors.

  22 x 39 = 858

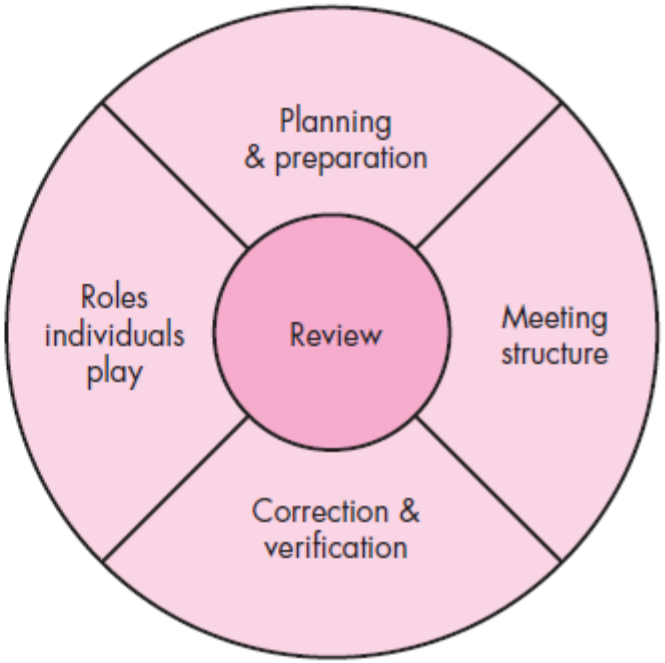
20.3.2 Cost Effectiveness of Reviews

Effort expended with and without reviews
Source: Adapted from [Fag86].



20.4 Reviews: A Formality Spectrum

Reference model for technical reviews



- Self Review: Review performed by an individual
- Peer Review: Review performed by peers
- Walkthrough
  - Review that is performed by the predetermined procedure by the reviewers who have predetermined roles

  Example Fagan Inspection

  [Fagan 76] Fagan, M.E., "Design and code inspections to reduce errors in program development", IBM Systems Journal, 15(3), 182-211, 1976.

  - Team: Producer, Review Leader, Reviewers, Reader, Recorder
  - Steps: 1) Overview – Determine the target and the participants
    2) Preparation – Training, Assignment of roles
    3) Inspection – Detect errors
    4) Rework & Follow-up – check fixes, check side-effects
  - High error detection efficiency, Significant cost

Inspection and Walkthrough [Fagan 76]

Table 4. Inspection and walk-through processes and objectives

Inspection		*Walk-through*	
Process Operations	*Objectives*	*Process Operations*	*Objectives*
1. Overview	Education (Group)	–	–
2. Preparation	Education (Individual)	1. Preparation	Education (Individual)
3. Inspection	Find errors! (Group)	2. Walk-through	Education (Group) Discuss design alternatives Find errors
4. Rework	Fix problems	–	
5. Follow-up	Ensure all fixes correctly installed	–	

Note the separation of objectives in the inspection process.

Review Options Matrix

	IPR*	WT	IN	RRR
trained leader	no	**yes**	**yes**	**yes**
agenda established	maybe	**yes**	**yes**	**yes**
reviewers prepare in advance	maybe	**yes**	**yes**	**yes**
producer presents product	maybe	**yes**	no	no
“reader” presents product	no	no	**yes**	no
recorder takes notes	maybe	**yes**	**yes**	**yes**
checklists used to find errors	no	no	**yes**	no
errors categorized as found	no	no	**yes**	no
issues list created	no	**yes**	**yes**	**yes**
team must sign-off on result	no	**yes**	**yes**	maybe

IPR*: Informal peer review
IN: Inspection
WT : Walkthrough
RRR: Round robin review

RRR: Each reviewer gets to present their comments on the product. The reviewers present one comment at a time. The person to the left of the leader starts, the reviewer to their left goes next, and so on, around the table.

20.5 Informal Reviews

- 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
- Pair programming encourages continuous review as a work product (design or code) is created.

  - Benefits
    - immediate discovery of errors
    - better work product quality.

20.6 Formal Technical Reviews (FTR)

20.6.1 The Review Meeting
20.6.2 Review Reporting and Record Keeping
20.6.3 Review Guidelines
20.6.4 Sample-Driven Reviews

Formal Technical Reviews (FTR)

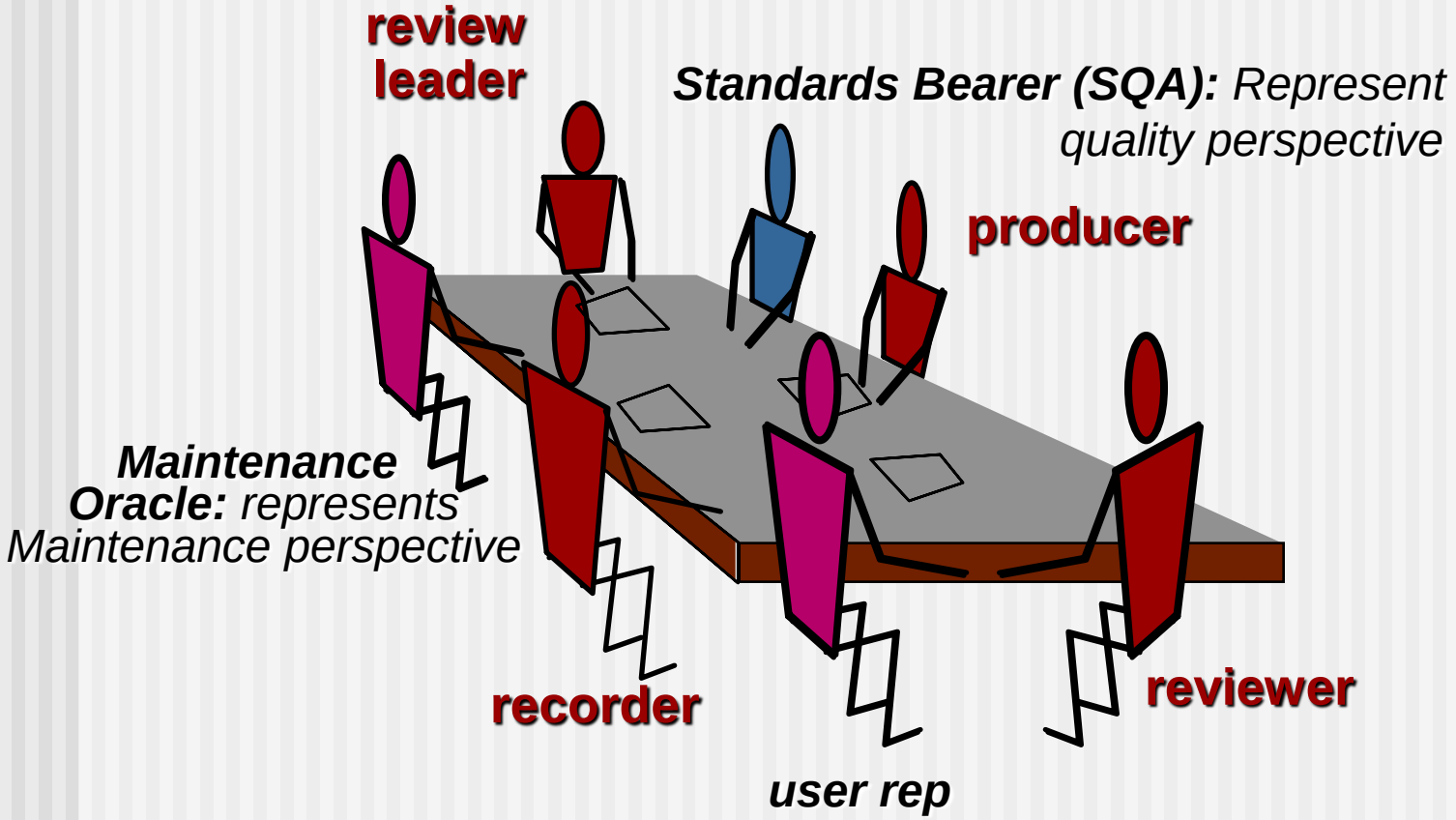
- Objectives:
  - to uncover errors in function, logic, or implementation for any representation of the software
  - to verify that the software under review meets its requirements
  - to ensure that the software has been represented according to predefined standards
  - to achieve software that is developed in a uniform manner
  - to make projects more manageable
- The FTR includes
  - Walkthroughs: more informal, overall
  - Inspections: more formal, focused

20.6.1 The Review Meeting

- 3 ~ 5 people (typically) should be involved
- Advance preparation
  - should require no more than two hours of work for each person.
- The duration of the review meeting
  - should be < 2 hours.
- Focus is on a work product

  Examples
  - a portion of a requirements model
  - a detailed component design
  - source code for a component

The Players



The 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) - review the product, make notes, and otherwise become familiar with the work.
- Recorder - reviewer who records (in writing) all important issues raised during the review.

20.6.2 Review Reporting and Record Keeping

- A review summary report answers the following:
  1. What was reviewed?
  2. Who reviewed it?
  3. What were the findings and conclusions?

20.6.3 Review Guidelines

1. Review the product, not the producer.
2. Set an agenda and maintain it.
3. Limit debate and rebuttal.
4. Identify problem areas, but don't attempt to solve every problem noted.
5. Take written notes.
6. Limit the number of participants and insist upon advance preparation.
7. Develop a checklist for each product that is likely to be reviewed.
8. Allocate resources and schedule time for FTRs.
9. Conduct meaningful training for all reviewers.
10. 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 …

Sample

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