Chapter 15

Review Techniques

Slide Set to accompany

Software Engineering: A Practitioner's Approach, 7/e

by Roger S. Pressman

Reviews

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

Jerry Weinberg

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

What Reviews Are Not

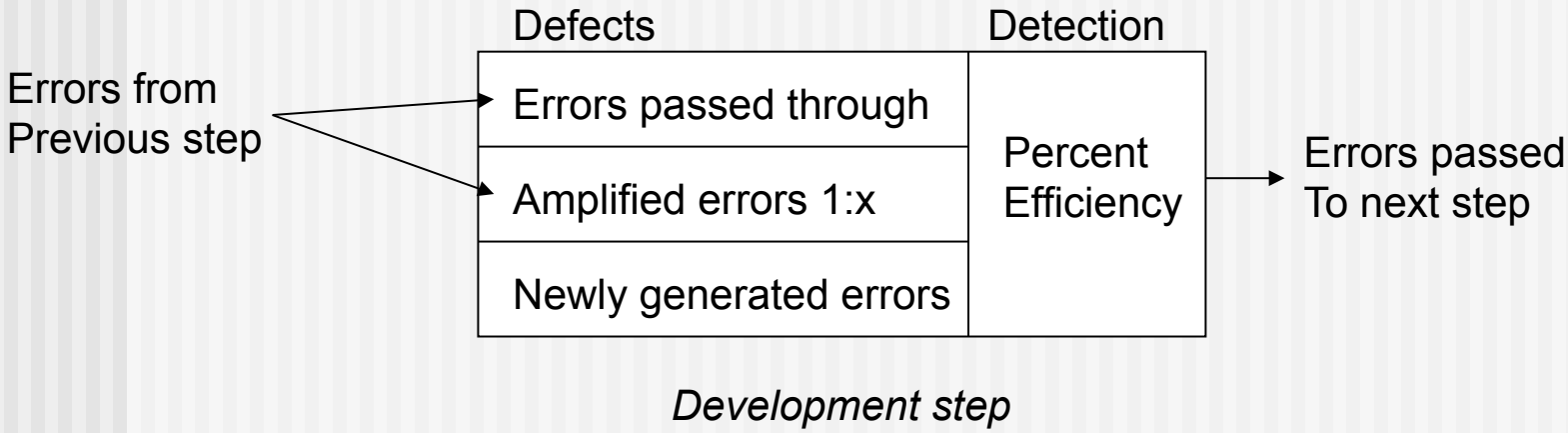
- A project summary or progress assessment
- A meeting intended solely to impart information
- A mechanism for political or personal reprisal!

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

Defect Amplification

- A *defect amplification model* [IBM81] can be used to illustrate the generation and detection of errors during the design and code generation actions of a software process.



Defect Amplification

- In the example provided in SEPA, Section 15.2,
  - a software process that does NOT include reviews,
    - yields 94 errors at the beginning of testing and
    - Releases 12 latent defects to the field
  - 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

- 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
- where …

Metrics

- *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)

An Example—I

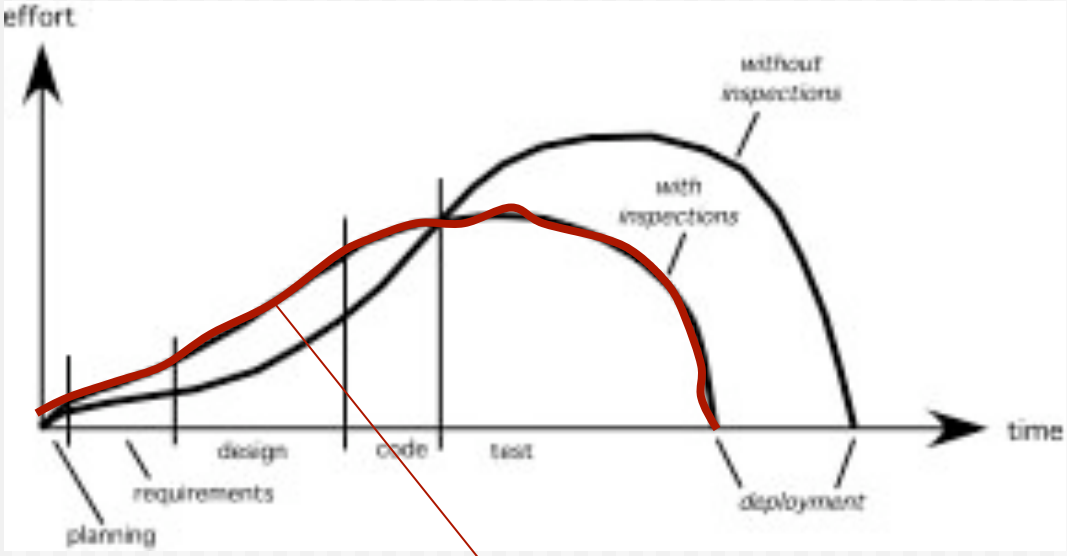
- If past history indicates that
  - the average defect density for a requirements model is 0.6 errors per page, and a new requirement model is 32 pages long,
  - a rough estimate suggests that your software team will find about 19 or 20 errors during the review of the document.
  - If you find only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

An Example—II

- 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.

Overall

- Effort expended with and without reviews



Reference Model

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.

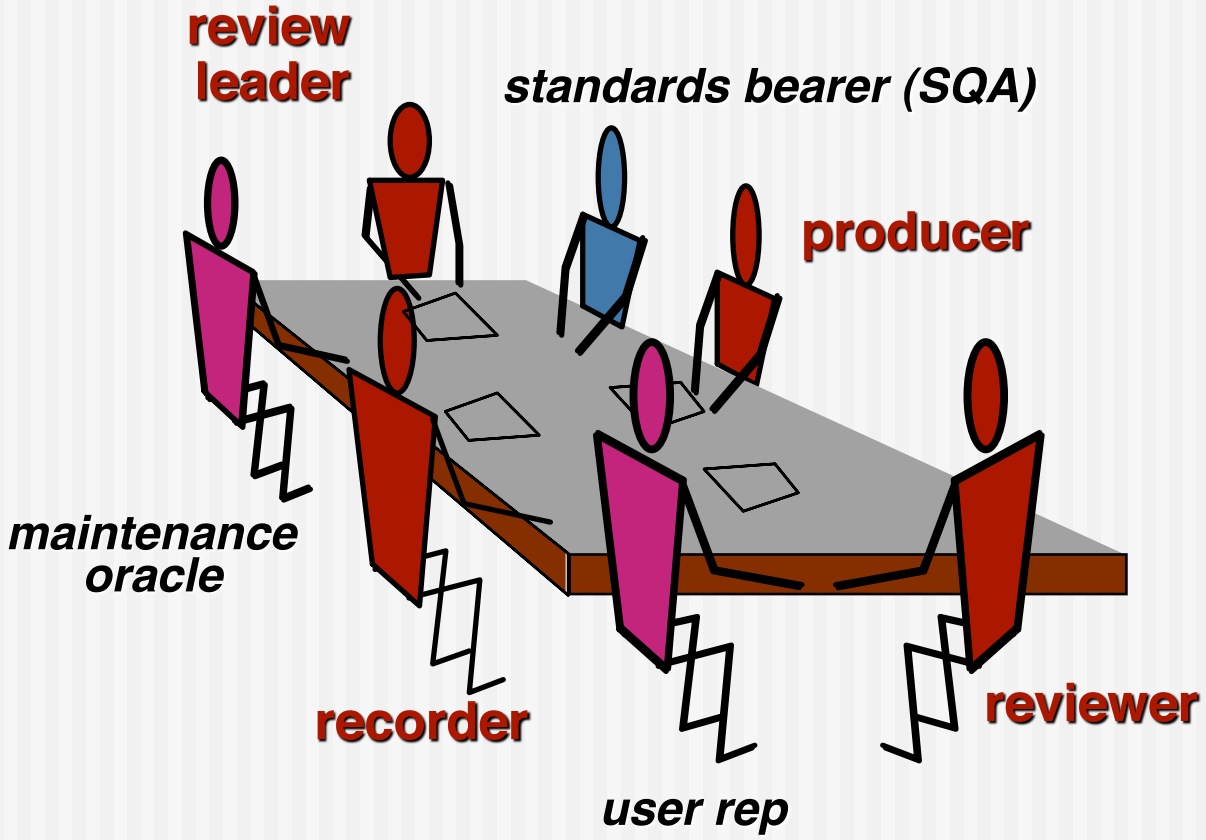
Formal Technical Reviews

- 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*.

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



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)*—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.

Conducting the Review

- *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.*

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 WT—Walkthrough
IN—Inspection RRR—round robin review

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.

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