

Chapter 20

■ Review Techniques

Slide Set to accompany

Software Engineering: A Practitioner's Approach, 8/e

by Roger S. Pressman

Slides copyright © 1996, 2001, 2005, 2009, 2015 by Roger S. Pressman

For non-profit educational use only

May be reproduced ONLY for student use at the university level when used in conjunction with *Software Engineering: A Practitioner's Approach, 7/e*. Any other reproduction or use is prohibited without the express written permission of the author.

All copyright information MUST appear if these slides are posted on a website for student use.

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

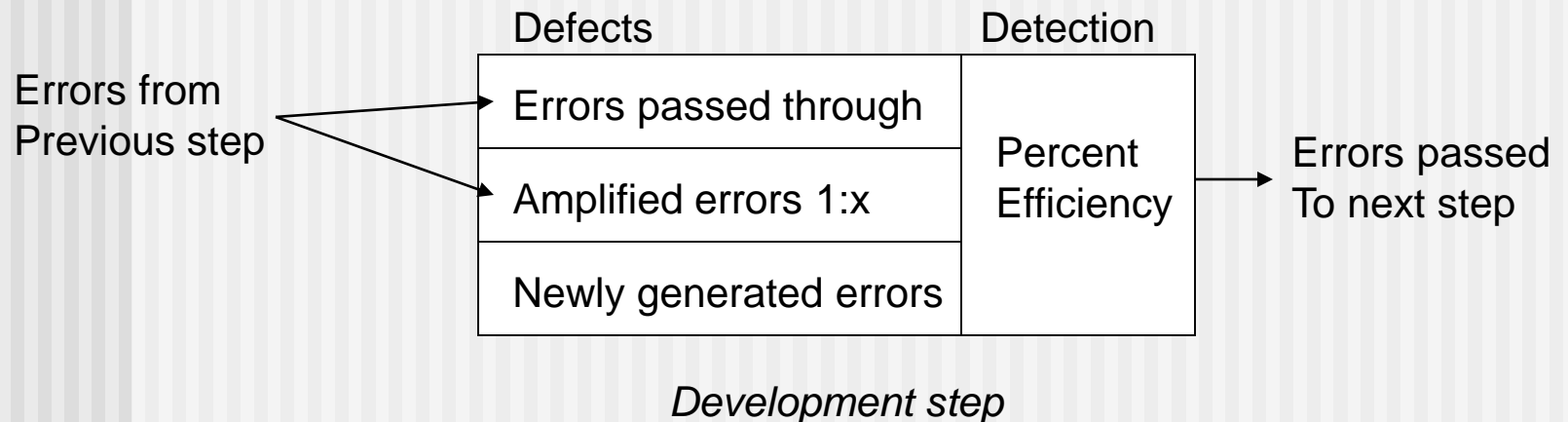
- A project summary or progress assessment
- A meeting intended solely to impart information
- A mechanism for political or personal reprisal!

What Do We Look For?

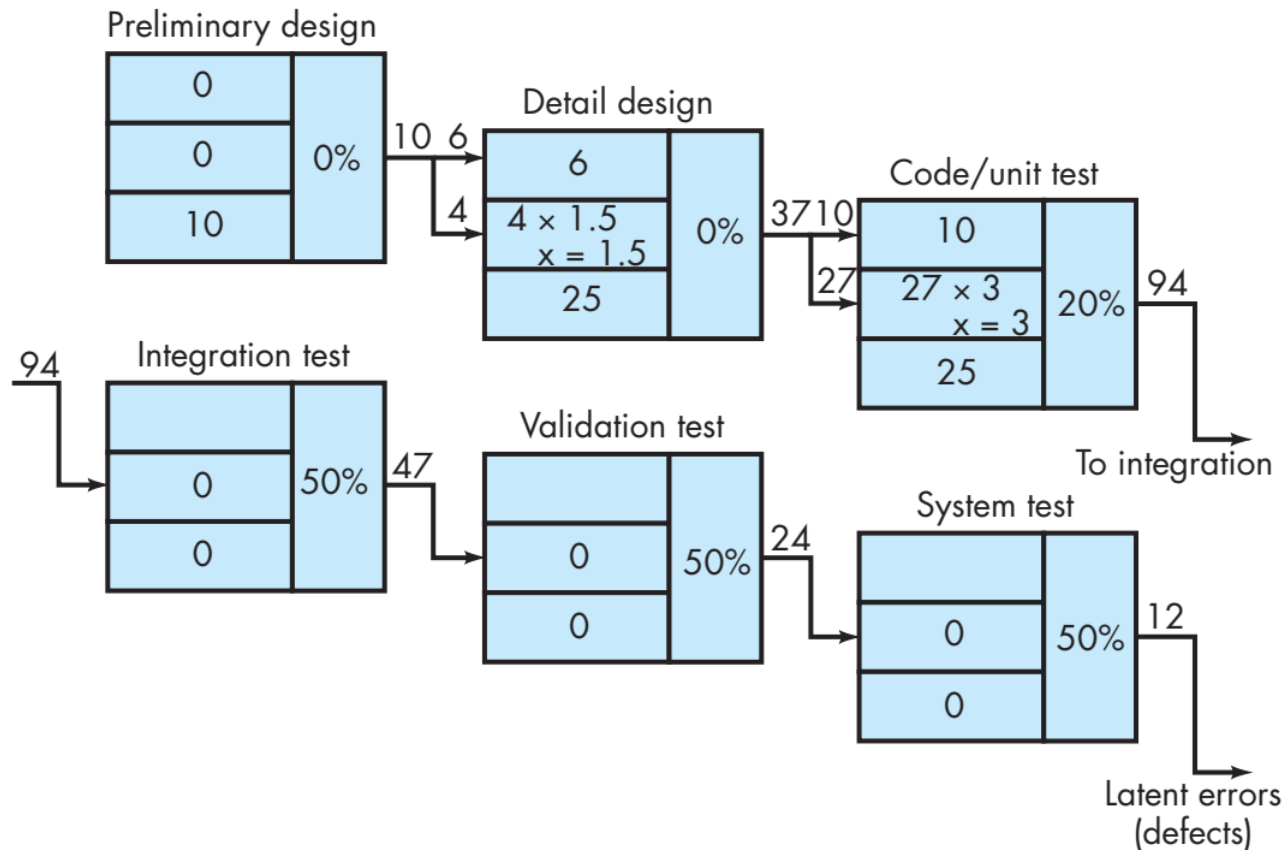
- Errors and defects
 - *Error*—a quality problem found *before* the software is released to end users
 - *Defect*—a quality problem found only *after* the software has been released to end-users
- We make this distinction because errors and defects have very different economic, business, psychological, and human impact
- However, the temporal distinction made between errors and defects in this book is *not* mainstream thinking

Defect Amplification

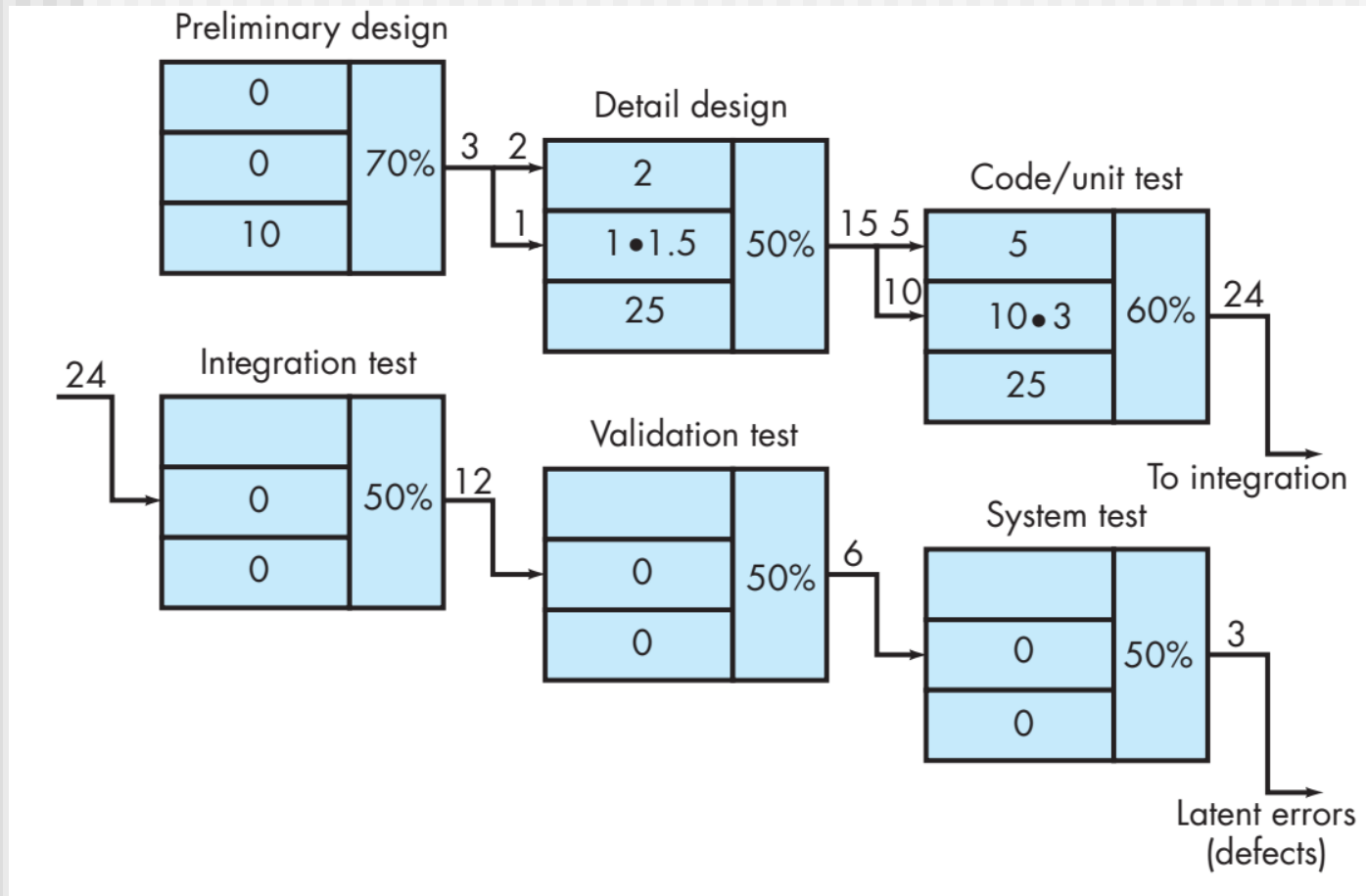
- A *defect amplification model* [IBM81] can be used to illustrate the generation and detection of errors during the design and code generation actions of a software process.



Defect Amplification (no reviews)



Defect Amplification (reviews conducted)



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

- *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

- 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 ...

An Example—I

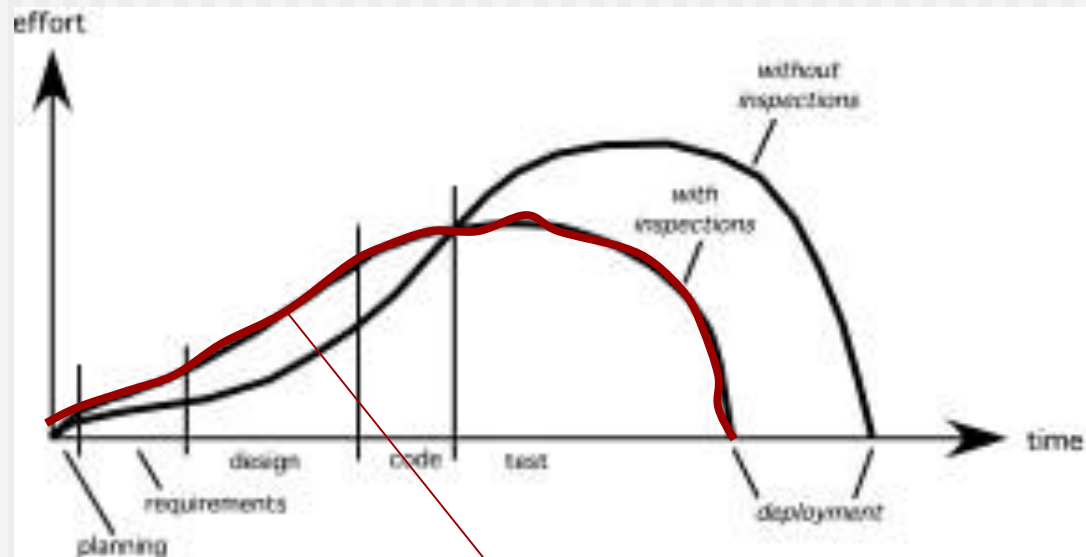
- If past history indicates that
 - the **average defect density** for a requirements model is 0.6 errors per page, and a new requirement model is 32 pages long,
 - a rough estimate suggests that your software team will find about 19 or 20 errors during the review of the document.
 - If you find only 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 = $E_{\text{testing}} - E_{\text{reviews}}$
- $45 - 6 = 39$ 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



with 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.

Informal Reviews

- One way to improve the efficacy of a desk check review is to develop a set of simple review checklists for each major work product produced by the software team.
 - Is the layout designed using standard conventions? Left to right? Top to bottom?
 - Does the presentation need to be scrolled?
 - Are color and placement, typeface, and size used effectively?
 - Are all navigation options or functions represented at the same level of abstraction?
 - Are all navigation choices clearly labeled?

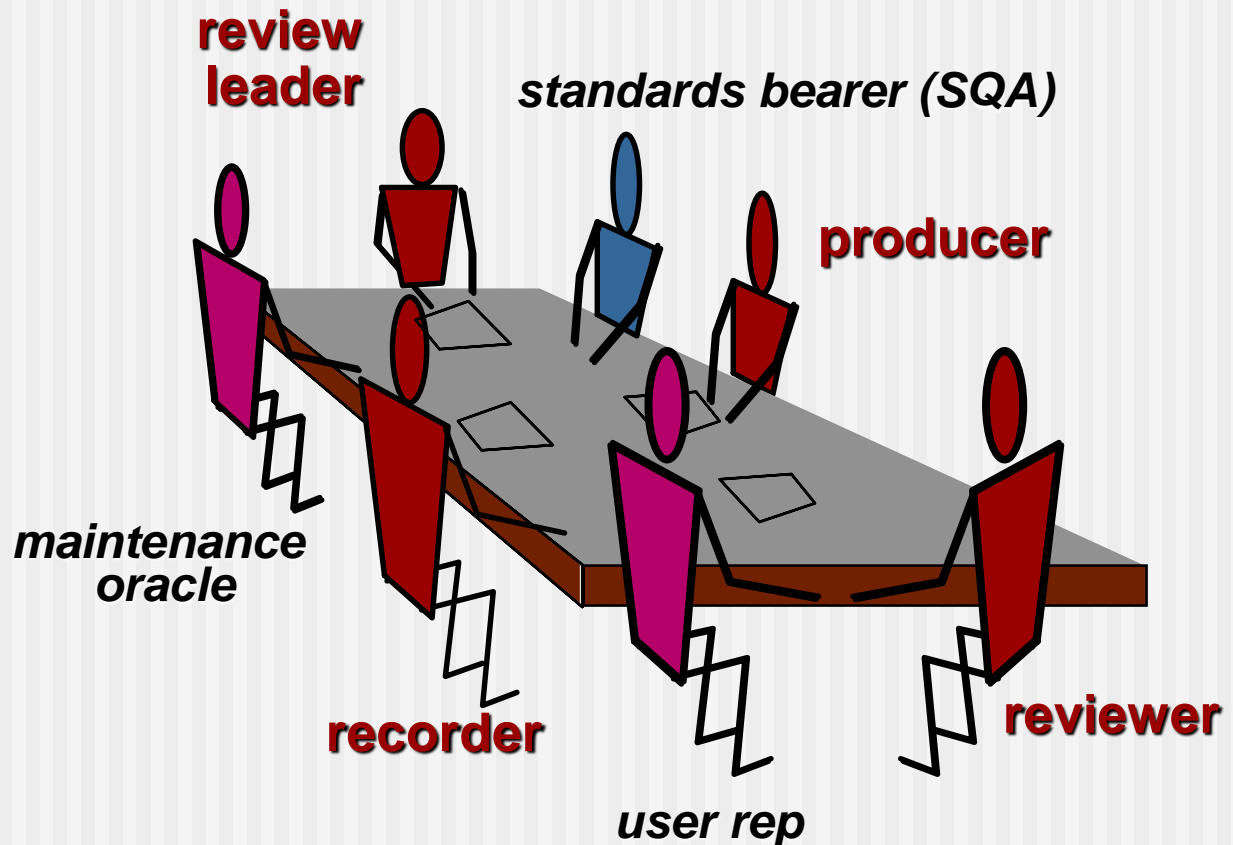
Formal Technical Reviews

- 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.*

Inspection

- از مهم‌ترین تکنیک‌های صحت‌سنجی و اعتبارسنجی است که هدف اصلی آن تشخیص خطاها و کاستی‌ها پیش از آغاز تست نرم‌افزار است؛ از این رو به شدت در افزایش کیفیت کلی نرم‌افزار نقش دارد. بازرسی یک فرآیند با قاعده و دارای ساختار است که در ۵ گام زیر انجام می‌پذیرد:
- مرور کلی (Overview): هدف از آن آشنایی اعضای تیم با محصول مورد بررسی است تا زمان مورد نیاز برای آماده‌سازی تا حدی کاهش یابد.
- آماده‌سازی (Preparation): در آن، اعضای تیم به صورت انفرادی محصول را بررسی و مشکلات را لیست می‌کند و سپس این لیست را به سرپرست بازرسی تحویل می‌دهند. وی بر اساس لیست دریافتی از تمام اعضا، دسته‌بندی و تحلیل آن‌ها، برای برگزاری جلسه بازرسی و مدت آن تصمیم‌گیری می‌نماید.

Inspection

- بازرسی: هدف در این جلسه ارزیابی یا بحث در مورد راه حل نیست، بلکه با استفاده از یک تکنیک مشخص (مثلاً چک لیست) مشکلات محصول بدون توجه به روش برطرف سازی آنها شناسایی و ثبت می گردند. این جلسه معمولاً بیشتر از دو ساعت طول نمی کشد و توسط سرپرست اداره می شود.
- دوباره کاری (rework): در این گام، تولیدکنندگان محصول بر اساس مشکلات شناخته شده در گام های قبل، به تصحیح این موارد می پردازند و اقدامات انجام شده برای این منظور را مستند می نمایند.
- پی گیری (follow up): در این گام، سرپرست بازرسی از تصحیح و برطرف سازی مشکلات شناسایی شده اطمینان می یابد. علاوه بر این ممکن است، سرپرست بر اساس نتایج حاصل از بازرسی چک لیست ها و آمارهای بروز خطا و نقص را به روزرسانی کند.

Inspection

		Planning	Overview	Preparation	Inspection Meeting	Rework	Follow-up
1976	Fagan's Inspection						
1985	Active Design Review						
1989	Two-Person Inspection						
1990	N-Fold Inspection						
1993	Phased Inspections						
1993	Inspection without meeting				Collection		
1993	Gilb Inspection				Process Brainstorming		

walkthrough

- یک روش مرور است که به منظور شناسایی ناهنجاری‌ها، بهبود محصول نرم‌افزاری، در نظر گرفتن گزینه‌های جایگزین برای پیاده‌سازی، مطابقت با استانداردها و مشخصات و ارزیابی قابلیت استفاده‌ی محصول مورد استفاده قرار می‌گیرد.
- علاوه بر این، هدف ثانویه‌ی آن آموزش شرکت‌کنندگان در رابطه با محصول مورد بررسی و تبادل تکنیک می‌باشد.
- این تکنیک در شکل معمول و سنتی بر روی طراحی و کد انجام می‌پذیرد که در آن پس از انتخاب ورودی‌های تست، شرکت‌کنندگان طراحی و کد را به صورت خط به خط بر مبنای ورودی‌های انتخاب شده مورد مرور و بررسی قرار می‌دهند.

walkthrough

- این تکنیک نسبت به بازرسی دارای قاعده‌مندی کمتری است و برخلاف آن بر کنترل تیم فنی، توسط اعضای داخلی آن تأکید دارد و در آن معمولاً تولیدکننده‌ی محصول کاری، جلسه را هدایت می‌کند.
- بنابراین ممکن است تفاسیر تولیدکننده سهواً موجب پنهان ماندن برخی خطاها شود؛ اگرچه به علت آشنایی کامل وی با محصول مورد بررسی، امکان مرور محتوای بیشتری در جلسه‌ی walkthrough وجود دارد. در برخی از منابع همین عامل، موجب کارایی walkthrough شناخته شده است.
- بسیاری از مواردی که در بازرسی الزامی هستند (مثلاً استفاده از تکنیک‌های مرور، آماده‌سازی و یا پی‌گیری) در Walkthrough انجام نمی‌شود یا با شدت کمتری صورت می‌پذیرند.

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**

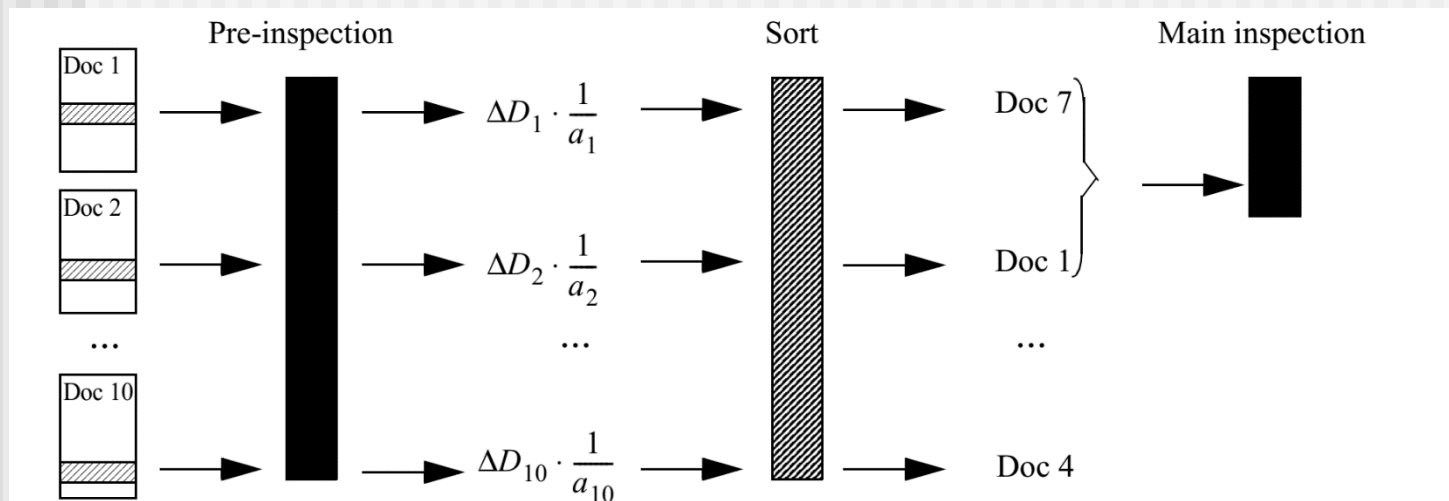
Sample-Driven Reviews (SDRs)

- 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.

Sample-Driven Reviews (SDRs)



Sample-Driven Reviews (SDRs)

■ Example:

assume that five design documents are to be inspected. The project manager selects five documents of equal importance and decides that 20% of the documents are to be inspected. Since the documents are about 10 pages, he/she selects two of the pages that he/she thinks is appropriate as a sample of the documents. Prior to the pre-inspection, the project manager has decided that only severe faults should be considered when sorting the documents.

The five documents contain 5, 5, 10, 15 and 20 severe faults. (Of course, this is not known beforehand.) In the 20% pre-inspection 0, 1, 2, 1 and 3 severe faults are found. The project manager decides that the focus in the main inspection will be put on the documents where 3 and 2 faults were found.

Sample-Driven Reviews (SDRs)

■ Example

approximate number of faults that have been exposed to an inspection

$$(0.20*(5+5+10+15+20)+0.80*(10+20)) = 35$$

$$35 / (5+5+10+15+20) = 35/55 = 64\% \text{ error}$$

$$(2+2+2+10+10)/50 = 26/50 = 52\% \text{ effort}$$

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