Chương 15

- **Kĩ thuật đánh giá**

Slide đi cùng với sách

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

by Roger S. Pressman

Đánh giá

... không có lí do cụ thể vì sao mà bạn bè và đồng nghiệp của bạn không thể trở thành nhà phê bình nghiêm khắc nhất của bạn

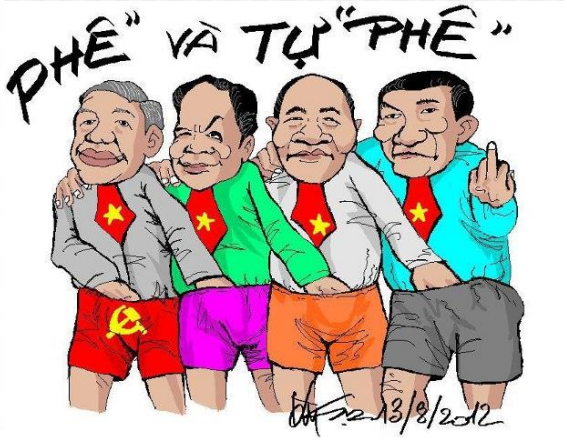
Jerry Weinberg

Đánh giá là gì?

- Một cuộc họp thực hiện bởi dân kĩ thuật dành cho dân kĩ thuật
- Một đánh giá kĩ thuật về sản phẩm được tạo ra trong quá trình sản xuất phần mềm
- Một cơ chế đảm bảo chất lượng phần mềm
- Một mặt bằng đào tạo

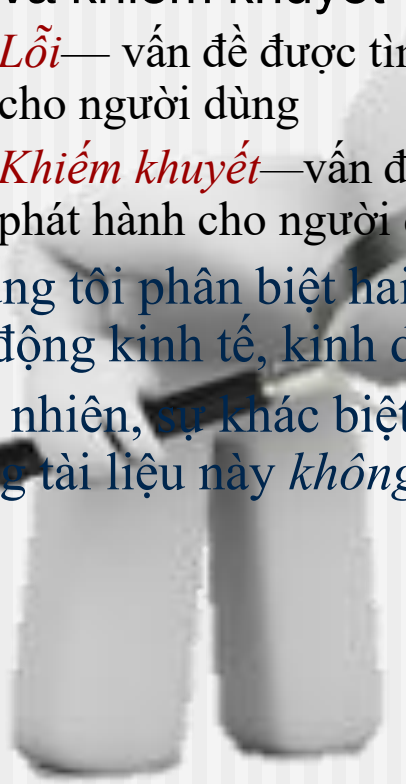
Đánh giá không phải là

- Một bản tóm tắt dự án hay đánh giá quá trình
- Một cuộc họp chỉ để truyền đạt thông tin
- Một cơ chế vì mục đích chính trị hay trả thù cá nhân



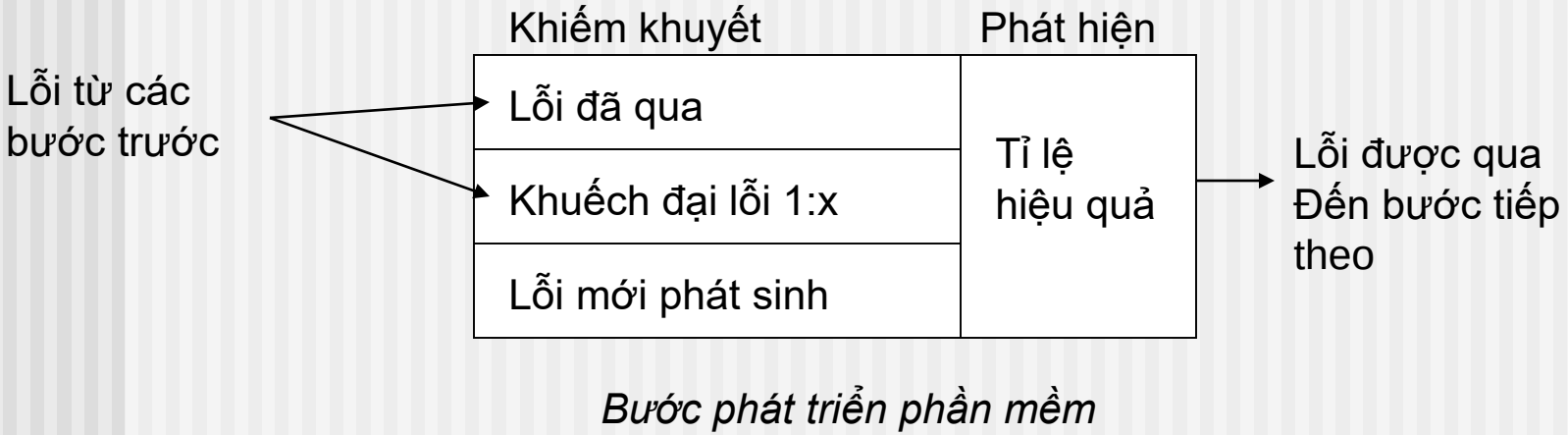
Chúng ta tìm kiếm gì?

- Lỗi và khiếm khuyết
  - *Lỗi*— vấn đề được tìm thấy *trước* khi phần mềm được phát hành cho người dùng
  - *Khiếm khuyết*—vấn đề chỉ được tìm thấy *sau* khi phần mềm được phát hành cho người dùng
- Chúng tôi phân biệt hai loại này vì lỗi và khiếm khuyết có các tác động kinh tế, kinh doanh, tâm lí, con người rất khác nhau
- Tuy nhiên, sự khác biệt thời gian tạo ra lỗi hay khiếm khuyết trong tài liệu này *không phải* tư tưởng chính



Khuếch đại khiếm khuyết

- *Mô hình khuếch đại khiếm khuyết* [IBM81] có thể sử dụng để biểu diễn sự phát sinh và phát hiện lỗi trong hoạt động thiết kế và viết code của một quy trình phần mềm.



Khuếch đại khiếm khuyết

- Trong ví dụ được nêu ở SEPA, Mục 15.2,
  - Một quy trình phần mềm KHÔNG có đánh giá,
    - mắc 94 lỗi tại lúc bắt đầu kiểm tra và
    - phát hành 12 khiếm khuyết tiềm tàng
  - Một quy trình phần mềm có đánh giá,
    - mắc 24 lỗi khi bắt đầu kiểm tra và
    - phát hành 3 khiếm khuyết tiềm tàng
  - Một phân tích chi phí chỉ ra rằng quy trình KHÔNG có đánh giá tốn hơn khoảng 3 lần quy trình có đánh giá, nếu xét tới chi phí sửa chữa các khiếm khuyết tiềm tàng

Độ đo

- Tổng các nỗ lực đánh giá và tổng số lỗi phát hiện được định nghĩa:
  - $E_{review} = E_p + E_a + E_r$
  - $Err_{tot} = Err_{minor} + Err_{major}$
- *Mật độ khiếm khuyết* biểu diễn số lỗi phát hiện trên một đơn vị sản phẩm được đánh giá
  - Mật độ khiếm khuyết = Err_{tot} / WPS
- Trong đó …

Độ đo

- *Nỗ lực chuẩn bị, E_p*—sự nỗ lực (theo giờ làm việc) cần thiết để đánh giá chất lượng sản phẩm trước khi đánh giá thực tế
- *Nỗ lực đánh giá, E_a*— sự nỗ lực (theo giờ làm việc) trong lúc đánh giá thực tế
- *Nỗ lực sửa chữa, E_r*— sự nỗ lực (theo giờ làm việc) dành cho việc sửa chữa các lỗi được phát hiện trong quá trình đánh giá
- *Kích thước sản phẩm, WPS*—một đơn vị đo lường kích thước của sản phẩm được đánh giá (số mô hình UML, số trang tài liệu, số dòng code…)
- *Các lỗi nhỏ được phát hiện, Err_{minor}*—số các lỗi được phát hiện và phân loại là nhỏ (cần ít thời gian sửa chữa hơn một mốc định trước)
- *Các lỗi lớn được phát hiện, Err_{major}*— số các lỗi được phát hiện và phân loại là lớn (cần nhiều thời gian sửa chữa hơn một mốc định trước)

Ví dụ —I

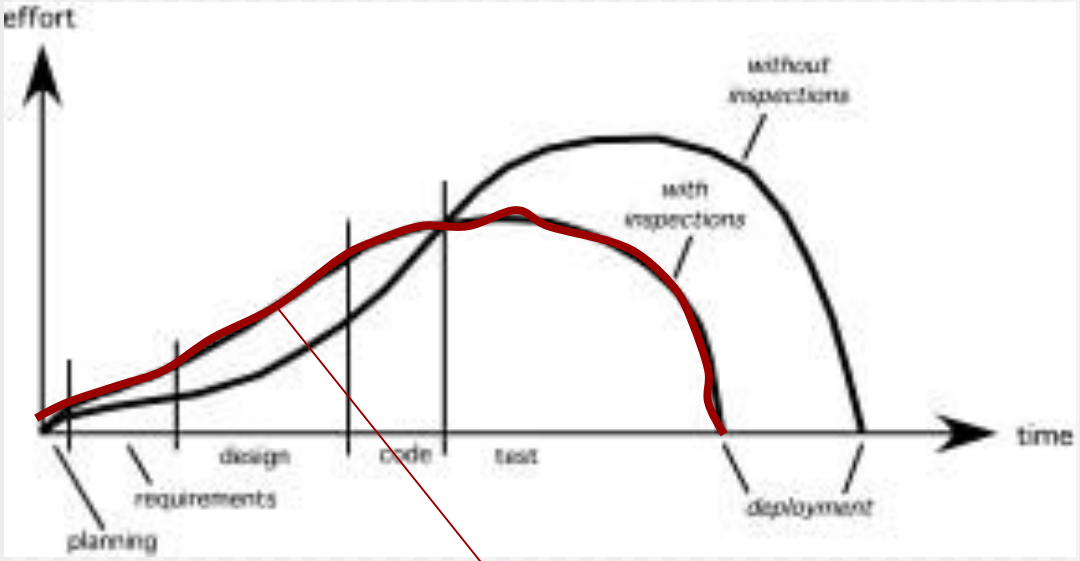
- Nếu quá khứ chỉ ra rằng
  - mật độ khiếm khuyết trung bình với một mẫu yêu cầu là 0.6 lỗi mỗi trang, và mẫu yêu cầu này có 32 trang
  - Một ước tính thô đoán rằng nhóm phần mềm của bạn sẽ tìm thấy khoảng 19 hoặc 20 lỗi trong quá trình đánh giá tài liệu đó
  - Nếu bạn chỉ tìm thấy 6 lỗi, bạn đã làm cực tốt trong quá trình phát triển mẫu yêu cầu hoặc quá trình đánh giá của bạn không đủ kĩ lưỡng

Ví dụ—II

- Nỗ lực cần thiết để sửa một lỗi nhỏ (ngay sau khi đánh giá) là 4 giờ làm việc
- Nỗ lực cần thiết để sửa chữa một lỗi lớn là 18 giờ làm việc
- Ví dụ sau khi đánh giá, bạn thấy số lỗi nhỏ nhiều gấp 6 lần số lỗi lớn. Do đó, bạn có thể ước lượng nỗ lực trung bình để sửa chữa một lỗi là khoảng 6 giờ làm việc.
- Các lỗi được phát hiện trong quá trình kiểm tra đòi hỏi 45 giờ làm việc để phát hiện và sửa chữa. Sử dụng kí hiệu trung bình, ta có
- Nỗ lực tiết kiệm được trên một lỗi = $E_{testing} – E_{reviews}$
- $45 – 6 = 30$ giờ làm việc/lỗi
- Nếu có 22 lỗi được phát hiện trong khi đánh giá mẫu yêu cầu, một khoảng thời gian 660 giờ làm việc có thể được tiết kiệm trong quá trình kiểm thử. Và nó chỉ là các lỗi liên quan đến yêu cầu.

Tổng thể

- Nỗ lực cần thiết khi có và không có đánh giá



with reviews

Mô hình tham chiếu

Đánh giá không chính thức

- Đánh giá không chính thức bao gồm:
  - một kiểm tra tại chỗ một sản phẩm phần mềm với một đồng nghiệp
  - một cuộc họp ngẫu nhiên (liên quan đến hơn 2 người) với mục đích đánh giá một sản phẩm, hoặc
  - các đánh giá có định hướng của cặp lập trình
- *Cặp lập trình* khuyến khích đánh giá liên tục trong khi sản phẩm được tạo ra.
  - Lợi ích là ngay lập tức phát hiện lỗi và kết quả là chất lượng sản phẩm tốt hơn

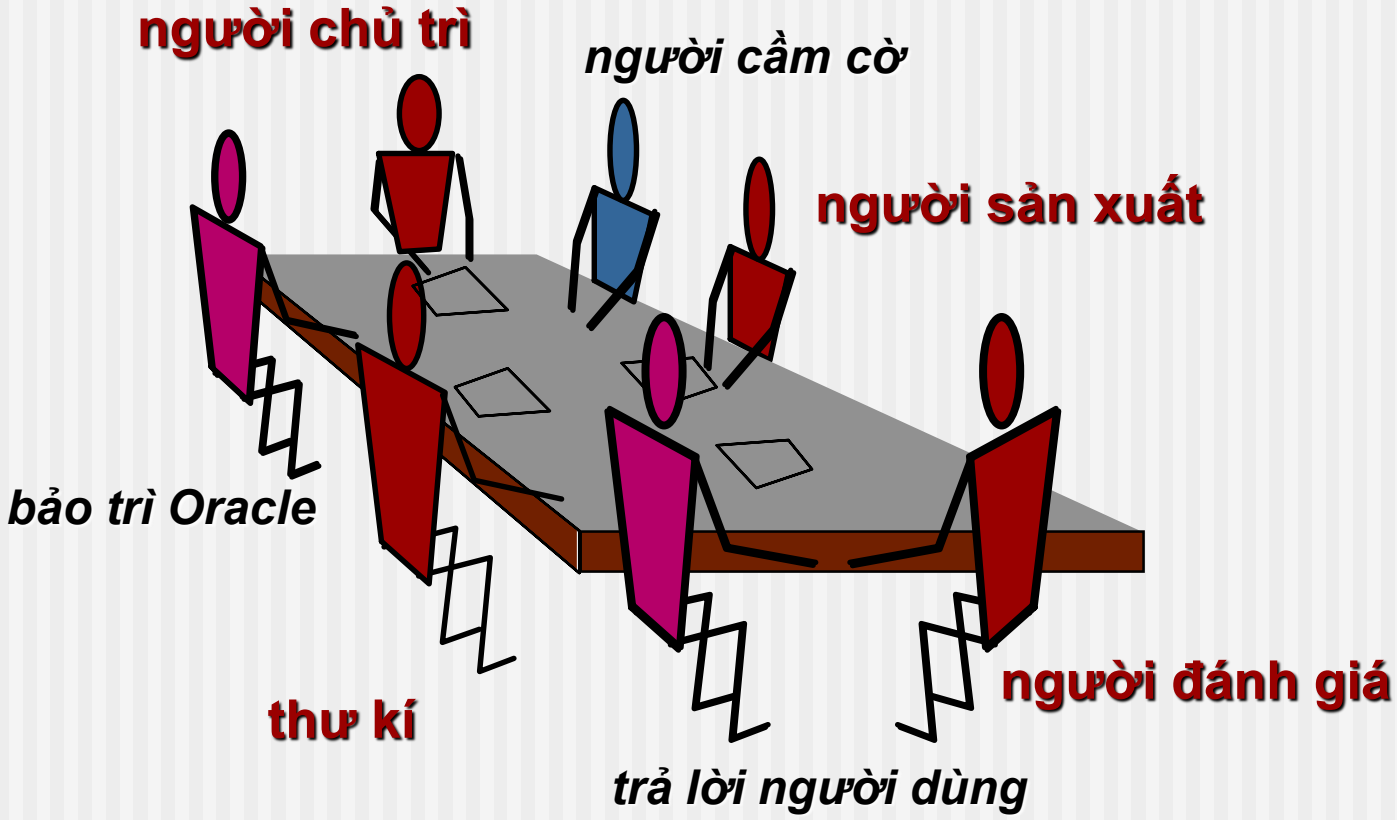
Đánh giá kĩ thuật chính thức (FTR)

- Các mục tiêu của FTR:
  - Để phát hiện ra lỗi trong chức năng, logic hoặc hiện thực bất kì mô tả nào của phần mềm
  - Để xác minh rằng phần mềm đang đánh giá đáp ứng đúng yêu cầu
  - Để đảm bảo rằng phần mềm đã được mô tả theo đúng chuẩn được đĩnh nghĩa trước
  - Để phần mềm được phát triển theo một cách thống nhất
  - Để cho dự án dễ quản lí hơn
- FTR là một lớp đánh giá, bao gồm *thực hiện các bước* và *kiểm tra*.

Buổi đánh giá

- Khoảng 3 đến 5 người có nhiệm vụ đánh giá sản phẩm
- Các chuẩn bị nâng cao nên được thực hiện nhưng yêu cầu không quá 2 giờ cho mỗi người
- Thời gian buổi đánh giá nên dưới 2 giờ
- Tập trung vào sản phẩm (ví dụ: một phần của mẫu yêu cầu, chi tiết thiết kế một bộ phận, mã nguồn của một thành phần…)

Người tham gia



Người tham gia

- *Người sản xuất*—các cá nhân phát triển sản phẩm
  - Thông báo cho lãnh đạo dự án rằng công việc đã hoàn thành và cần thiết phải có đánh giá
- *Người chủ trì*—chuẩn bị sẵn sàng sản phẩm, tạo các bản sao tài nguyên sản phẩm và phân phát chúng cho 2 hoặc 3 người đánh giá để chuẩn bị trước
- *Người đánh giá*—được kì vọng trong 1 đến 2 giờ có thể đánh giá sản phẩm, tạo ghi chú hoặc nắm rõ được công việc.
- *Thư kí*—là người ghi chép tất cả các vấn đề quan trọng trong buổi đánh giá.

Tiến hành đánh giá

- *Đánh giá sản phẩm, không phải đánh giá người sản xuất.*
- *Đặt ra một trình tự và duy trì nó*
- *Giới hạn sự tranh luận và bác bỏ.*
- *Phát biểu phạm vi vấn đề nhưng không cố gắng giải quyết từng vấn đề được nêu*
- *Tạo các ghi chú*
- *Giới hạn số người tham gia và nhấn mạnh sự chuẩn bị trước*
- *Tạo một danh mục có thể được xem xét với mỗi sản phẩm*
- *Phân bổ nguồn lực và thời gian cho các FTR*
- *Tiến hành huấn luyện cho tất cả người đánh giá*
- *Đánh giá các đánh giá trước đó*

Ma trận đánh giá chọn lựa

	IPR *	WT	IN	RRR
Người chủ trì	không	có	có	có
Lên chương trình	có thể	có	có	có
Người đánh giá chuẩn bị trước	có thể	có	có	có
NSX trình bày sản phẩm	có thể	có	không	không
"người nghe" trình bày sản phẩm	không	không	có	không
thư kí ghi chép	Có thể	có	có	có
Danh mục dùng để tìm lỗi	không	không	có	không
Phân loại lỗi khi tìm thấy	không	không	có	không
Danh sách các vấn đề	không	có	có	có
Nhóm phải kí vào kết quả	không	có	có	có thể

IPR—đánh giá không chính thức WT—thực hiện các bước
IN—kiểm tra RRR—kiểm tra theo vòng

Tiến trình đánh giá đơn giản (SDRs

- SDRs nỗ lực định lượng các sản phẩm là các mục tiêu chính của FTR đầy đủ.

Để thực hiện việc này...

- Kiểm tra một phần nhỏ a_i của mỗi sản phẩm phần mềm i, ghi lại số lượng lỗi f_i tìm thấy trong a_i.
- Thực hiện ước tính tổng số lỗi trong sản phẩm i bằng cách nhân f_i với $1/a_i$.
- Sắp xếp các sản phẩm theo thứ tự ước tính tổng số lỗi giảm dần.
- Tập trung có xem xét các sản phẩm có ước tính số lỗi cao nhất.

Số liệu cần có từ đánh giá

- Thời gian kiểm tra mỗi trang tài liệu
- Thời gian kiểm tra mỗi dòng code (LOC) hoặc mỗi h
- Nỗ lực kiểm tra mỗi LOC và FP
- Số lỗi phát hiện được mỗi giờ đánh giá
- Số lỗi phát hiện được mỗi giờ chuẩn bị
- Số lỗi phát hiện được trên mỗi phần việc (thiết kế…)
- Tổng số lỗi nhỏ (ví dụ: lỗi đánh máy…)
- Tổng số lỗi lớn (ví dụ: không đúng với yêu cầu…)

- Tổng số lỗi trong quá trình chuẩn bị