Lỗi (bug) là gì

Bởi:

Khoa CNTT ĐHSP KT Hưng Yên

Bạn vừa được tìm hiểu về một số vấn đề có thể xảy ra khi phần mềm bị lỗi. Nó có thể dẫn đến những phiền phức, giống như khi một máy chơi *game* không thể làm việc một cách hợp lý, hoặc nó có thể dẫn đến một thảm họa khủng khiếp nào đó. Số tiền để giải quyết vấn đề và sửa lỗi có thể lên tới hàng triệu *dollar*. Trong các ví dụ ở trên, rõ ràng phần mềm không hoạt động như dự tính ban đầu. Nếu là một tester, bạn sẽ phải tìm thấy hầu hết những lỗi của phần mềm. Hầu hết là những lỗi đơn giản và tinh vi, có khi quá nhỏ đến nỗi không thể phân biệt được cái nào là lỗi và cái nào không phải là lỗi.

NHỮNG THUẬT NGỮ VỀ CÁC LỖI PHẦN MỀM:

Phụ thuộc vào nơi mà bạn được làm việc (như một tester), bạn sẽ sử dụng những thuật ngữ khác nhau để mô tả: điều gì sẽ xảy đến khi phần mềm bị lỗi. Dưới đây là một số thuật ngữ:

Defect nhược điểm

Fault khuyết điểm

Failure sự thất bại

Anomaly sự dị thường

Variance biến dị

Incident việc rắc rối

Problem vấn đề

Error lỗi

Bug lỗi

Feature đặc trưng

Inconsistency sự mâu thuẫn

(Chúng ta có một danh sách các thuật ngữ không nên nhắc đến, nhưng hầu hết chúng được sử dụng riêng biệt, độc lập giữa các lập trình viên)

Bạn có thể thấy ngạc nhiên rằng có quá nhiều từ để mô tả một lỗi phần mềm. Tại sao lại như vậy? Có phải tất cả chúng đều thật sự dựa trên văn hóa của công ty và quá trình mà công ty sử dụng để phát triển phần mềm của họ. Nếu bạn tra những

từ này trong từ điển, bạn sẽ thấy rằng tất cả chúng đều có ý nghĩa khác nhau không đáng kể. Chúng cũng có ý nghĩa được suy ra từ cách mà chúng được sử dụng trong các cuộc đàm thoại hàng ngày.

Ví dụ, *fault, failure* và *defect* có xu hướng ám chỉ một vấn đề thật sự quan trọng, thậm chí là nguy hiểm. Dường như là không đúng khi gọi một biểu tượng (*icon*) không được tô đúng màu là 1 lỗi (*fault*). Những từ này cũng thường ám chỉ một lời khiển trách: chính là do nó (*fault*) mà phát sinh lỗi phần mềm (*software failure*) ("it's his fault that the software failure.")

Anomaly, incident, variance thì không có vẻ là quá tiêu cực và thường được sử dụng để đề cập tới sự vận hành không được dự tính trước thay vì hoàn toàn là lỗi (*all-out failure*). "Tổng thống đã tuyên bố rằng nó là một sự dị thường phần mềm đã làm cho tên lửa đi sai lịch trình của nó" ("The president stated that it was a software anomaly that caused the missile to go off course.")

Có lẽ, *Problem, error* và *bug* là những thuật ngữ chung nhất thường được sử dụng.

JUST CALL IT WHAT IT IS AND GET ON WITH IT (Hãy chỉ gọi nó là cái mà nó là và tiếp tục với nó)

Một điều thú vị khi một số công ty và các đội sản xuất đã tiêu tốn khá nhiều thời gian quý báu của quá trình phát triển phần mềm vào việc thảo luận và tranh cãi về những thuật ngữ được sử dụng. Một công ty máy tính nổi tiếng đã mất hàng tuần để thảo luận với những kỹ sư của họ trước khi quyết định đổi tên *Product Anomaly Report (PARs)* thành *Product Incident Report(PIRs).* Một số tiền lớn đã được sử dụng cho việc quyết định thuật ngữ nào là tốt hơn. Một khi quyết định đã được đưa ra (Once the decision was made), tất cả các công việc liên quan đến giấy tờ, phần mềm, định dạng… phải được cập nhật để phản ảnh thuật ngữ mới. Nó sẽ không được biết tới nếu nó gây ra bất kỳ sự khác biệt nào đối với hiệu quả làm việc của lập trình viên và tester.

Vậy, tại sao phải đưa ra chủ đề này? Thực sự là quan trọng khi một tester hiểu khả năng cá nhân đằng sau nhóm phát triển phần mềm mà bạn đang làm việc cùng. Cách thức họ đề cập tới các vấn đề về phần mềm của họ là dấu hiệu thông

thường về cách họ tiếp cận quá trình phát triển toàn bộ của họ. Họ có đề phòng, cẩn thận, thẳng thắn hay chỉ đơn giản là blunt?

Mặc dù tổ chức của bạn có thể chọn một cái tên khác, nhưng trong cuốn sách này tất các các vấn đề về phần mềm sẽ được gọi là các *bug*. Không thành vấn đề nếu lỗi là lớn, nhỏ, trong dự định, hay ngoài dự định, hoặc cảm giác của ai đó sẽ bị tổn thương bởi họ tạo ra chúng. Không có lý do gì để mổ xẻ các từ. *A bug's a bug's a bug.*

ĐỊNH NGHĨA VỀ LỖI PHẦN MỀM:

Các vấn đề về *software problems bugs* nghe có vẻ đơn giản, nhưng chưa hẳn đã giải quyết được nó. Bây giờ, từ *problem* cần được định nghĩa. Để tránh việc định nghĩa vòng quanh (*circular definitions*), điều quan trọng là phải mô tả được lỗi là gì?

Đầu tiên, bạn cần một thuật ngữ trợ giúp (*supporting term*): đặc tả phần mềm (*product specification*). Đặc tả phần mềm có thể gọi một cách đơn giản là *spec* hoặc *product spec,* là luận cứ của các đội phát triển phần mềm. Nó định nghĩa sản phẩm mà họ tạo ra, chi tiết là gì, hành động như thế nào, sẽ làm gì, và sẽ không làm gì? Luận cứ này có thể vạch ra phạm vi về hình thức từ một dạng hiểu biết về ngôn từ đơn giản, một email, hoặc 1 chữ viết nguệch ngoạc trên tờ giấy ăn, tới một tài liệu thành văn được hình thức hóa, chi tiết hơn. Trong bài 2, “Quy trình phát triển phần mềm”, bạn sẽ học về đặc tả phần mềm và quy trình phát triển, nhưng không phải là bây giờ, định nghĩa này là đầy đủ.

Một lỗi phần mềm xuất hiện khi 1 hoặc nhiều hơn trong 5 quy tắc dưới đây là đúng:

1. Phần mềm không thực hiện một số thứ giống như mô tả trong bản đặc tả phần mềm

2. Phần mềm thực hiện một số việc mà bản đặc tả yêu cầu nó không được thực hiện

3. Phần mềm thực hiện một số chức năng mà bản đặc tả không đề cập tới

4. Phần mềm không thực hiện một số việc mà bản đặc tả không đề cập tới, nhưng là những việc nên làm

5. Trong con mắt của người kiểm thử, phần mềm là khó hiểu, khó sử dụng, chậm đối với người sử dụng

Để tìm hiểu kỹ hơn về mỗi quy tắc, hãy cố gắng xem ví dụ dưới đây để áp dụng chúng cho phần mềm *calculator* trong máy tính.

Có lẽ, đặc tả của 1 *calculator* đã nói rõ rằng: nó sẽ thực thi phép cộng, phép trừ, phép nhân, phép chia đúng. Nếu bạn là một tester, nhận việc kiểm thử phần mềm *Calculator*, nhấn phím “+” và không có chuyện gì xảy ra, đó là một lỗi theo quy tắc 1. Nếu bạn nhận được câu trả lời sai, cũng có nghĩa rằng đó là một lỗi, bởi vì theo **quy tắc 1**.

Bản đặc tả phần mềm yêu cầu rằng, *calculator* sẽ không bao giờ bị đột ngột ngưng hoạt động, bị khóa lại hoặc bị đóng băng. Nếu bạn đập thình thịch lên các phím và nhận được thông điệp từ *calculator* là "*not responding"* (dừng quá trình hồi đáp với dữ liệu vào), đây là một lỗi theo **quy tắc 2**.

Mục đích là bạn nhận được phần mềm calculator để kiểm thử và thấy rằng bên cạnh các phép cộng, trừ, nhân, chia; nó còn thực hiện các phép căn bậc 2. Điều này chưa từng được nêu trong bản đặc tả. Một lập trình viên có nhiều tham vọng vừa thêm nó vào bởi vì anh ta cảm thấy nó sẽ là một đặc tính tuyệt vời (*great feature*). Đây không phải là một một *feature,* nó thật sự là một *bug* theo **quy tắc 3**. Phần mềm đang thực hiện một số công việc mà bản đặc tả phần mềm không hề đề cập tới. Mặc dù sự điều khiển không định hướng này có thể là tốt, nhưng nó sẽ yêu cầu thêm những lỗ lực lập trình và kiểm thử (vì có thể sẽ xuất hiện thêm nhiều lỗi). Lúc này chi phí cho việc sản xuất phần mềm cũng lớn hơn, điều này làm giảm hiệu quả kinh tế của quá trình sản xuất phần mềm.

Đọc **quy tắc thứ 4** có thể thấy hơi lạ với sự phủ định kép, nhưng mục đích của nó là tìm thấy những đặc điểm bị lãng quên, không được nhắc tới trong bản đặc tả. Bạn bắt đầu kiểm thử phần mềm *Calculator* và khám phá ra rằng, khi Pin yếu, bạn không nhận được những câu trả lời đúng cho quá trình tính toán của bạn nữa. Chưa ai từng xem xét xem *calculator* phản ứng lại như thế nào trong chế độ này. Một giả định không tốt được tạo ra rằng: pin luôn được nạp đầy. Bạn mong muốn rằng công việc sẽ được duy trì cho đến khi pin hoàn toàn hết, hoặc ít nhất bằng cách

nào đó báo cho bạn biết Pin đã yếu. Những phép tính đúng đã không xảy ra khi pin yếu, và nó cũng không chỉ rõ điều gì sẽ xảy đến. Quy tắc số 4 tạo nên lỗi này.

Quy tắc số 5 mang tính chất tổng hợp (*catch-all*). *Tester* là người đầu tiên thực sự sử dụng phần mềm như một khách hàng lần đầu sử dụng phần mềm. Nếu bạn phát hiện một vài điều mà bạn thấy không đúng vì bất cứ lý do gì, thì nó là một lỗi. Trong trường hợp của *calculator*, có lẽ bạn đã tìm thấy những nút có kích thước quá nhỏ. Hoặc có thể sự sắp xếp của nút "=" đã làm cho nó khó sử dụng. Hoặc sự bố trí màu sắc làm cho nó rất khó nhìn... Tất cả những điều này đều là lỗi (bug) theo quy tắc 5.

Chú ý: Mọi người sử dụng một phần của phần mềm sẽ có những mong đợi khác nhau và những ý kiến phần mềm nên hoạt động như thế nào. Sẽ không thể viết được phần mềm mà mọi người sử dụng đều thấy nó hoàn hảo. *Tester* sẽ luôn phải giữ những ý nghĩ này trong suy nghĩ của họ khi họ áp dụng quy tắc 5 để kiểm thử. Xét một cách thấu đáo, hãy sử dụng khả năng đánh giá tốt nhất của bạn, và **quan trọng nhất là phải hợp lý**. Ý kiến của bạn có giá trị, nhưng, bạn sẽ được tìm hiểu trong các phần sau, không phải tất cả các lỗi bạn tìm được có thể hoặc sẽ được sửa (*fix*).

Để có một số ví dụ đơn giản và điển hình, bạn hãy nghĩ xem các quy tắc được áp dụng như thế nào với phần mềm mà bạn sử dụng hàng ngày. Đâu là điều bạn mong đợi, đâu

là điều không mong đợi? Bạn thấy điều gì đã được chỉ rõ và điều gì bị bỏ quên? Và điều mà bạn hoàn toàn không thích về phần mềm này?

Định nghĩa trên về lỗi của một phần mềm đã bao quát những vấn đề cơ bản, nhưng nếu sử dụng tất cả 5 quy tắc trên sẽ giúp bạn định nghĩa được các loại vấn đề khác nhau trong phần mềm mà bạn đang kiểm thử.