**Lý thuyết Cửa sổ vỡ**

**trong Phát triển phần mềm**

*(The broken window theory in Software Development)*

**Giới thiệu**

Hồi còn nhỏ, trước khi có chiếc máy tính đầu tiên, tôi luôn rong ruổi trên chiếc xe đạp của mình. Đối diện nhà tôi có một khu rừng nhỏ với vài con đường mòn hẹp, nơi tôi thường xuyên đạp xe qua.

Tôi thường tưởng tượng rằng chiếc xe BMX của mình là một chiếc xe đua công thức 1, và tôi đang thi đấu với Nigel Mansell, Alain Prost hoặc Ayrton Senna. Rồi, ngay trước vạch đích, tôi thực hiện những pha vượt mặt ngoạn mục và giành chiến thắng sít sao.

Ở cuối khu rừng, có một nhà kho cũ. Trên cao, gần mái nhà, có một số ô cửa sổ nhỏ để ánh sáng lọt vào, nhưng chúng ở đủ cao để không ai có thể nhìn trộm bên trong.

Tôi vẫn nhớ lần đầu tiên dừng lại để nghỉ ngơi và quan sát tòa nhà đó. Tôi nghĩ: "Lạ thật! Sao nó không nằm gần con đường nào?... Ai đó đã ném đá làm vỡ một số cửa sổ… Tại sao chúng không được sửa? Nó trông bí ẩn nhưng ảm đạm. Tốt nhất là nên tránh xa, có gì đó không ổn." Rồi tôi lại đạp xe đi và chẳng mấy khi quay lại khu vực đó nữa.

Nhiều năm trôi qua, khu rừng bị cắt ngang bởi một con đường, và dần dần, một số tòa nhà mới mọc lên hai bên. Nhưng trong một thời gian khá dài, nhà kho cũ vẫn tồn tại. Tuy nhiên, mỗi năm trôi qua, ngày càng nhiều cửa sổ bị vỡ bởi những viên đá ném vào.

Thực lòng mà nói, những kẻ phá hoại những ô cửa sổ đó có lẽ chẳng hề bận tâm. Vì một số ô cửa sổ đã vỡ và không ai sửa, thì thêm vài cái nữa cũng chẳng sao. Dù sao thì nếu sửa, cũng phải thay toàn bộ.

Vì thế, những ô cửa sổ của nhà kho cũ trở thành một mục tiêu lý tưởng để luyện kỹ năng ném, và đôi khi tôi thấy vài thanh niên lớn hơn ăn mừng khi đóng góp vào việc "phá dỡ sớm" tòa nhà bằng cách làm vỡ thêm một ô cửa sổ khác.

Đối với tôi và một số bạn học, con đường mới giúp rút ngắn quãng đường đến trường. Nhưng chúng tôi vẫn thích đi đường cũ hơn dù nó dài hơn một chút, vì đó là con đường quen thuộc và giúp tránh xa tòa nhà xuống cấp ấy, nơi khiến chúng tôi cảm thấy bất an. Ít nhất, nó giúp chúng tôi bớt lo lắng.

Tôi chia sẻ câu chuyện này từ một thời chưa có điện thoại di động để đưa bạn quay lại tuổi thơ của chính mình, để gợi lên những cảm giác tương tự mà bạn có thể đã trải qua khi nhìn thấy một tòa nhà đổ nát hoặc một khu vực hoang tàn. Đồng thời, đây cũng là cách để giới thiệu về ***Lý thuyết Cửa sổ Vỡ* *(The Broken Windows Theory)****.*

Nói ngắn gọn, lý thuyết cửa sổ vỡ cho rằng môi trường suy thoái khuyến khích hành vi phạm tội, chống đối xã hội hoặc vô tổ chức. Nếu không có cửa sổ nào bị vỡ, khả năng xảy ra tình trạng hỗn loạn sẽ giảm đi, tạo điều kiện cho sự yên bình và tuân thủ luật pháp. Nhưng một khi sự suy thoái đã xuất hiện, nó sẽ dễ dàng lan rộng hơn.

Nguyên tắc này cũng có thể áp dụng vào quá trình phát triển phần mềm, nơi mà một "broken window" có thể được xem là một thiết kế kém, một lựa chọn mã nguồn tệ có thể nói là code bị “smell” hoặc một quyết định sai lầm gây ra cảm giác thờ ơ, chán nản cho các thành viên trong đội ngũ phát triển. Theo thời gian, sự thờ ơ này sẽ dẫn đến sự xuống cấp về chất lượng và khả năng bảo trì của phần mềm.

**Một ví dụ về “broken window” trong phát triển phần mềm**

Một ví dụ điển hình về "broken window" trong lĩnh vực phần mềm là lỗi chính tả trong HTTP Referer header, vốn là một sai sót từ "referrer". Lỗi này đã tồn tại trong nhiều năm và gây không ít khó khăn cho các lập trình viên trên toàn thế giới.

Nhiều người đã từng đặt câu hỏi về nguyên nhân của sai sót này và ai là người chịu trách nhiệm. Tuy nhiên, qua thời gian, các lý do phổ biến như "quá bận" hoặc "sẽ sửa sau" thường được đưa ra, phản ánh một thực tế phổ biến trong ngành phần mềm: nhiều lỗi nhỏ nhưng quan trọng vẫn chưa được khắc phục do không được ưu tiên.

Việc một lỗi chính tả đơn giản kéo dài trong nhiều năm đặt ra câu hỏi về trách nhiệm của cộng đồng phát triển phần mềm. Thay vì sửa lỗi, có những quan điểm cho rằng chỉ cần hướng dẫn người dùng chấp nhận và làm theo cách sai đó. Một cách tiếp cận mỉa mai hơn là thay đổi từ điển để biến lỗi này thành tiêu chuẩn, từ đó nó sẽ không còn được xem là một sai sót nữa.

Điều này cho thấy tầm quan trọng của việc khắc phục các vấn đề kỹ thuật, dù nhỏ, nhằm tránh tích lũy các lỗi có thể gây ảnh hưởng tiêu cực đến hệ thống trong tương lai.

**“Broken window” kéo dài trong các tổ chức**

Từ thực tế, có thể nhận thấy rằng trong các tổ chức lớn với đội ngũ quản lý có nhiều kinh nghiệm và phần mềm đã tồn tại lâu năm, các vấn đề "broken windows" có xu hướng kéo dài hoặc thậm chí không bao giờ được khắc phục.

Việc phát hành phần mềm sớm và thường xuyên thường đi kèm với sự suy giảm về chất lượng, khi nhiều tính năng chưa hoàn thiện mà đã được đưa ra thị trường. Điều này không phản ánh tính linh hoạt (Agile) trong phát triển phần mềm mà chỉ thể hiện sự thiếu cẩn trọng trong quy trình phát triển.

Khi các “broken window” không được xử lý kịp thời, chúng sẽ tích lũy và trở thành nợ kỹ thuật (Technical debt). Nợ kỹ thuật không chỉ bao gồm các lỗi nhỏ mà còn là những quyết định tạm thời, giải pháp không tối ưu hoặc thiếu tài liệu chi tiết. Theo thời gian sẽ gây tăng chi phí, lỗ hổng hệ thống và khó khăn khi mở rộng.

Dù có nhiều lý do có thể được đưa ra để biện minh cho thực trạng này, nhưng hầu hết các phần mềm không thể đứng vững khi có sự xuất hiện của cuộc tấn công mạng. Các công ty khởi nghiệp thường là mục tiêu dễ bị tấn công bởi tội phạm mạng do hệ thống của họ còn nhiều lỗ hổng bảo mật.

**Nguyên nhân “broken window” không được sửa chữa**

Ở các tổ chức đã trưởng thành, một trong những nguyên nhân chính khiến các "broken windows" không được khắc phục là do:

* Tỷ suất lợi nhuận thấp
* Sự ưu tiên cho các nhiệm vụ quan trọng hơn.

Bên cạnh đó, các tổ chức này còn phải tuân thủ nhiều quy định và chính sách, khiến họ không thể dễ dàng thực hiện những thay đổi có thể gây rủi ro lớn cho hệ thống và xã hội.

Chính vì vậy, các tập đoàn lớn thường không trực tiếp tham gia vào các lĩnh vực mới hoặc thị trường chưa ổn định ngay từ đầu. Thay vào đó, họ để các công ty khởi nghiệp (startup) tiên phong thử nghiệm những ý tưởng mới, phát triển công nghệ và mở rộng thị trường trước. Sau một thời gian, khi thị trường đã ổn định hơn và các quy định pháp lý cũng rõ ràng hơn, các tập đoàn lớn sẽ xem xét mua lại những startup thành công để tận dụng công nghệ, khách hàng và lợi thế mà startup đó đã xây dựng được.

**Tâm lý “chấp nhận lỗi”**

Xã hội thường đề cao những cá nhân và tổ chức thành công, đồng thời ít chấp nhận những thất bại. Điều này dẫn đến tâm lý phòng thủ khi mắc lỗi, thể hiện qua những lý do phổ biến như:

* "Đó không thực sự là lỗi."
* "Không ai sử dụng theo cách đó."
* "Nó vẫn đang hoạt động như mong đợi."

Khi các lỗi tồn tại trong thời gian dài mà không được sửa chữa, chúng thường được gọi một cách hài hước là "tính năng" thay vì "lỗi". Điều này phản ánh thực tế rằng nhiều lập trình viên, khi không thể giải quyết triệt để một vấn đề, sẽ cố gắng chấp nhận nó để giảm bớt áp lực tâm lý thay vì tiếp tục lo lắng về nó.

Việc nhận diện và xử lý các "broken windows" một cách có hệ thống là điều cần thiết để duy trì chất lượng phần mềm và giảm thiểu các rủi ro tiềm ẩn trong dài hạn.

**Một vài điều rút ra từ lí thuyết cửa sổ vỡ**

**Don’t leave windows broken, don’t leave software issues unaddressed** *(Đừng để cửa sổ vỡ, đừng bỏ mặc lỗi phần mềm)*

Bài học quan trọng từ lý thuyết cửa sổ vỡ trong phát triển phần mềm rất đơn giản: đừng để những ô cửa sổ vỡ mà không sửa chữa. Bất cứ khi nào bạn gặp một vấn đề dù là thiết kế kém, quyết định sai lầm, hay mã nguồn chất lượng thấp hãy giải quyết nó ngay lập tức.

Nếu bị giới hạn về thời gian và không thể sửa chữa triệt để những vấn đề đó, hãy thực hiện các biện pháp để ngăn chặn thiệt hại tránh làm nó lan rộng hơn; giống như đóng ván che cửa sổ bị vỡ lại. Điều này có thể hiểu trong quá trình phát triển phần mềm là tạm thời vô hiệu hóa đoạn mã đó, hoặc hiển thị lỗi khi nó được sử dụng. Điều quan trọng là phải thể hiện rằng bạn đang chú ý, quan tâm đến vấn đề đó và cam kết duy trì chất lượng phần mềm. Dù cho vấn đề đó không do bạn gây ra, nhưng nếu trong sản phẩm mà bạn được tham gia vào, bạn cũng cần có trách nhiệm đối với nó.

**Maintain a high standard** *(Duy trì tiêu chuẩn cao)*

Sự thờ ơ, thiếu trách nhiệm là yếu tố lớn nhất thúc đẩy quá trình suy thoái phần mềm làm mất đi những đặc điểm mà một phần mềm tốt nên có như tính bảo trì, tính mở rộng, ... Những phần mềm sạch sẽ, hoạt động tốt có thể xuống cấp rất nhanh khi các "broken window" bắt đầu xuất hiện ngày càng nhiều. Dù có nhiều yếu tố khác góp phần vào sự suy thoái của phần mềm, nhưng sự thờ ơ, thiếu trách nhiệm của các lập trình viên vẫn là nguyên nhân hàng đầu đẩy nhanh sự suy giảm chất lượng của các dự án phần mềm.

Các nhà phát triển nên hướng đến một tiêu chuẩn cao. Trong những dự án có mã nguồn được thiết kế tốt, sạch sẽ và tinh gọn, đội ngũ phát triển sẽ có xu hướng cẩn trọng hơn và cố gắng duy trì chất lượng hiện có. Tuy nhiên, nếu bạn làm việc trong một dự án có nhiều "broken window", bạn sẽ rất dễ rơi vào tư duy: *"Đa phần mã nguồn trong hệ thống đều tệ, nhưng để nhất quán thì tốt hơn hết là mình cứ làm theo như vậy đi."* Để ngăn chặn tư duy này, hãy ưu tiên sửa chữa các vấn đề mà hệ thống hiện có và duy trì tiêu chuẩn chất lượng cao trong mã nguồn nhất có thể. Một bước khởi đầu tốt là hãy tự mình đánh giá lại mã nguồn mỗi khi thực hiện merge, miễn là bạn thực sự trung thực với những gì mình đang xem xét.

Giải quyết sự xuống cấp của phần mềm là một thách thức, đặc biệt đối với sản phẩm phần mềm của công ty outsource. Mặc dù khách hàng thường mong đợi chất lượng của các ứng dụng này ngang bằng với sản phẩm do các công ty product phát triển, nhưng chi phí để duy trì chất lượng cao có thể rất lớn. Một sản phẩm phần mềm của công ty product có thể phân bổ chi phí cho hàng trăm hoặc hàng nghìn khách hàng, trong khi chi phí của một ứng dụng thuộc công ty outsource chỉ do một khách hàng gánh chịu. Vì lý do này, việc đặt ra kỳ vọng về chất lượng hợp lý ngay từ đầu trong bất kỳ mối quan hệ hợp tác nào giữa khách hàng và đối tác là vô cùng quan trọng.

**Fixing windows: Small choices and actions that keep entropy at bay** *(Sửa chữa những cửa sổ: Những lựa chọn và hành động nhỏ giúp ngăn chặn sự hỗn loạn)*

Sự suy thoái phần mềm (*software entropy*) là một thách thức thực sự trong phát triển phần mềm, và “the broken window” mang đến những bài học giá trị để kiểm soát nó. Bằng cách nhanh chóng sửa chữa các "broken windows" và duy trì tiêu chuẩn chất lượng mã nguồn cao, các dự án phần mềm có thể chống lại xu hướng tự nhiên của sự rối loạn và suy giảm.

**Kết luận**

Cuối cùng, những hành động và lựa chọn tuy nhỏ sẽ tạo ra sự khác biệt lớn đối với sự ổn định và thành công lâu dài của một dự án phần mềm. Vì vậy, đừng để sự suy thoái chiến thắng, hãy ưu tiên duy trì một mã nguồn sạch và kiểm tra nó một cách thường xuyên.