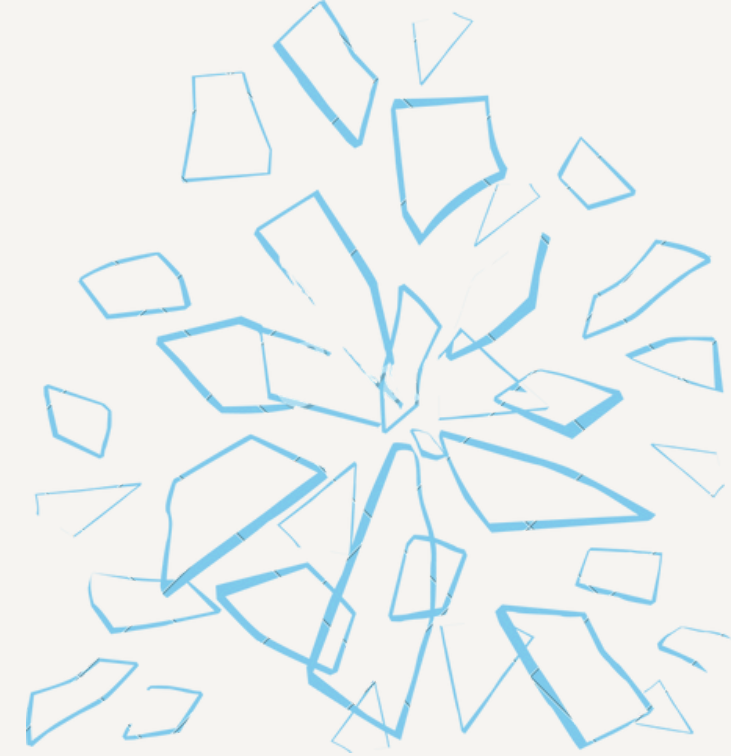




BROKEN WINDOW THEORY & BOY SCOUT RULE



Tại sao hai nguyên tắc này quan trọng trong phát triển phần mềm?



BROKEN WINDOW THEORY

"Consider a building with a few broken windows. If the windows are not repaired, the tendency is for vandals to break a few more windows."

— James Q. Wilson and George L. Kelling

Broken Window Theory



Nội dung: sự xuống cấp của một dự án phần mềm, sự rối loạn trong hệ thống sẽ ngày càng tăng nếu không được kiểm soát. Những lỗi nhỏ nếu không sửa sẽ tích tụ, làm giảm chất lượng sản phẩm và gây khó khăn trong bảo trì.

Các hành vi cần tránh dựa theo nguyên tắc:

- Mã nguồn lộn xộn, khó đọc, không tuân thủ nguyên tắc.
- Sử dụng giải pháp tạm bợ thay vì xử lý triệt để vấn đề.
- Thiếu comment, tài liệu hướng dẫn, gây khó hiểu cho đồng đội.
- Để lại mã rác, không dọn dẹp sau khi chỉnh sửa.
- Bỏ qua cảnh báo, lỗi khi biên dịch hoặc chạy ứng dụng.



Ví dụ

Hệ thống ban đầu tốt

Dự án bắt đầu với ý định tốt, có nguyên tắc rõ ràng.

Không xử lý lỗi

Nợ kỹ thuật bị bỏ qua hoặc trì hoãn sửa chữa.

Hệ thống sụp đổ

Code trở nên cứng nhắc, khó bảo trì, tốn kém để sửa chữa.

Xuất hiện lỗi nhỏ

Do áp lực hoặc thiếu kiến thức, lập trình viên mắc lỗi.

Lỗi lan rộng

Mọi người bắt chước sai lầm, lỗi ngày càng nhiều.



BOY SCOUT RULE

"Always leave the campground cleaner than you found it."

— Boy Scouts of America




Boy Scout Rule



Nội dung: đảm bảo rằng luôn để lại mã nguồn bạn đang làm việc tốt hơn khi bạn được nhận ban đầu.

Nguyên tắc:

- Khi sửa lỗi, tối ưu luôn phần mã xung quanh.
 - Khi thêm tính năng mới, hãy cải thiện cả mã cũ liên quan.
 - Khi review code, hãy đề xuất cách cải thiện thay vì chỉ chấp nhận.
 - Khi thay đổi mã nguồn, hãy cập nhật test case phù hợp.
 - Chủ động dọn dẹp hoặc nâng cấp mã cũ (legacy code).
- 

Tổng kết

Broken Window Theory & Boy Scout Rule là những nguyên tắc giúp duy trì chất lượng và trật tự trong một hệ thống, giảm thiểu sự xuống cấp của hệ thống và khuyến khích tư duy cải tiến liên tục.

Lời khuyên:

- Không bỏ qua những vấn đề nhỏ, khi phát hiện lỗi hoặc code xấu, hãy sửa ngay trước khi nó trở thành một vấn đề lớn.
- Luôn để lại mọi thứ tốt hơn, dù chỉ là một cải tiến nhỏ.
- Duy trì văn hóa chất lượng, cùng thực hiện hai nguyên tắc trên, sản phẩm sẽ bền vững hơn và dễ bảo trì hơn về lâu dài.

Tác hại khi vi phạm

Hậu quả:

- Codebase ngày càng xuống cấp, khó bảo trì.
- Gia tăng lỗi và sự cố phần mềm.
- Giảm hiệu suất làm việc của đội ngũ phát triển.
- Khó mở rộng và phát triển thêm tính năng mới.
- Ảnh hưởng đến danh tiếng và uy tín của công ty.




Ví dụ

Đặt vấn đề:

- Code không có test ban đầu vì áp lực thời gian để kịp deadline. Người sau không thấy test, nghĩ rằng không cần, tiếp tục không viết test.
 - Một dev viết code kém (sai cấu trúc, dùng abstraction không hợp lý). Người khác copy cách làm đó thay vì cải thiện, khiến code xấu lan rộng.
- => Vấn đề ngày càng lớn, chất lượng code giảm dần.

```
if (user.role == "admin" || user.role == "moderator") {  
    grantAccess();  
}
```



```
if (user.role == "admin" || user.role == "moderator" || user.role == "editor" ||  
    user.role == "contributor") {  
    grantAccess();  
} else if (user.role == "guest") {  
    restrictAccess();  
} else if (user.role == "banned") {  
    blockUser();  
}
```

Chất lượng code giảm do không cải thiện.