TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN

THIẾT KẾ PHẦN MỀM

BÁO CÁO BÀI TẬP

Broken window Theory

Nhóm 18 – Mewing

Sinh viên:

22120046	**Nguyễn Ngọc Đăng**
22120181	**Nguyễn Duy Lâm**
22120183	**Nguyễn Đặng Minh Lân**

Tp. Hồ Chí Minh, ngày 28 tháng 03 năm 2025

MỤC LỤC

I. Giới thiệu

Lý thuyết cửa sổ vỡ, được giới thiệu bởi **James Q. Wilson và George L. Kelling** trong bài viết năm 1982 trên tạp chí The Atlantic Monthly **[1]**, là một khái niệm trong ngành tội phạm học, cho rằng các dấu hiệu nhỏ của rối loạn, như cửa sổ bị vỡ không được sửa, có thể tạo ra môi trường đô thị khuyến khích thêm tội phạm và rối loạn, bao gồm cả tội phạm nghiêm trọng. Thí nghiệm của Philip Zimbardo năm 1969, để hai xe ô tô không biển số ở khu vực Bronx (New York) và Palo Alto (California), minh chứng điều này. Xe ở Bronx bị phá hoại trong vòng 10 phút, trong khi xe ở Palo Alto chỉ bị phá sau khi Zimbardo tự phá hủy bằng búa tạ, cho thấy sự bỏ bê có thể dẫn đến hành vi phá hoại thêm. **[2]**

Lý thuyết này không chỉ áp dụng trong đô thị mà còn được mở rộng sang nhiều lĩnh vực, bao gồm lập trình phần mềm, nơi nó nhấn mạnh tầm quan trọng của việc duy trì chất lượng mã nguồn để tránh sự suy thoái dần dần.

II. Nguồn gốc và Bối cảnh

Lý thuyết cửa sổ vỡ xuất phát từ quan sát rằng cộng đồng có dấu hiệu rối loạn, như graffiti, hành vi say xỉn nơi công cộng, hoặc cửa sổ bị vỡ không được sửa, có thể dẫn đến cảm giác không ai quan tâm, từ đó khuyến khích hành vi phạm tội thêm. Nó được phổ biến vào những năm 1990 bởi William Bratton, ủy viên cảnh sát thành phố New York, và liên quan đến các chính sách như "stop-and-frisk," dù gây tranh cãi. Thí nghiệm của Zimbardo cho thấy xe ở Bronx bị cướp phá nhanh chóng, trong khi xe ở Palo Alto chỉ bị phá sau khi có hành động phá hoại ban đầu, minh chứng rằng môi trường có thể ảnh hưởng đến hành vi.

III. Ứng dụng trong Lập trình

Trong lập trình, lý thuyết cửa sổ vỡ được áp dụng để duy trì chất lượng mã nguồn, với ý tưởng rằng nếu bỏ qua các lỗi nhỏ hoặc mã nguồn kém, chúng có thể dẫn đến sự suy thoái dần dần, gọi là "nợ kỹ thuật" (technical debt). Sách "The Pragmatic Programmer" của Andrew Hunt và David Thomas, xuất bản năm 1999, là nguồn chính phổ biến khái niệm này trong cộng đồng lập trình **[3]**. Họ khuyên rằng không nên "sống chung với cửa sổ vỡ," nghĩa là sửa ngay các thiết kế xấu, quyết định sai lầm hoặc mã nguồn kém chất lượng. Nếu không có thời gian sửa hoàn toàn, có thể "che chắn" tạm thời, như bình luận mã lỗi, hiển thị thông báo "Chưa thực hiện," hoặc thay thế bằng dữ liệu giả, để ngăn chặn thiệt hại thêm **[4]**.

Ví dụ, nếu một đoạn mã có lỗi cú pháp nhỏ không được sửa, lập trình viên khác có thể cảm thấy không ai quan tâm đến chất lượng, dẫn đến việc thêm mã kém chất lượng hơn, cuối cùng làm cho codebase trở nên khó bảo trì. **[5]**

IV. Tầm quan trọng của Chất lượng Mã

Chất lượng mã là yếu tố quan trọng đối với thành công dự án phần mềm. Nghiên cứu cho thấy mã chất lượng cao dẫn đến ít lỗi, dễ bảo trì, hiệu suất tốt hơn và bảo mật

cao hơn. Ngược lại, bỏ qua chất lượng mã có thể dẫn đến thời gian phát triển kéo dài, chi phí bảo trì cao, độ tin cậy thấp và lỗ hổng bảo mật. Chất lượng mã ảnh hưởng đến sự hài lòng của lập trình viên, tỷ lệ nghỉ việc, và thành công thương mại, với mã tốt liên quan đến ít lỗi và ít lỗ hổng bảo mật **[6]**. Một ví dụ cụ thể là một dự án sử dụng framework lỗi thời, dẫn đến hiệu suất chậm, được cải thiện sau khi chuyển sang HTML, minh chứng cho tác động của chất lượng mã **[7]**.

V. Thực tiễn Áp dụng

Để áp dụng lý thuyết cửa sổ vỡ trong lập trình, các đội phát triển có thể:

1. **Xem xét mã định kỳ:** Thực hiện xem xét mã (code review) thường xuyên để phát hiện và sửa lỗi nhỏ trước khi chúng trở thành vấn đề lớn.
2. **Kiểm thử tự động:** Sử dụng công cụ kiểm thử tự động để đảm bảo mã hoạt động đúng và phát hiện lỗi hồi quy sớm.
3. **Công cụ phân tích mã:** Sử dụng các công cụ như SonarQube, CodeScene, ESLint, và Semgrep để phân tích mã và phát hiện vấn đề tiềm ẩn **[8]**. Ví dụ, SonarQube giúp phát hiện lỗ hổng bảo mật và đảm bảo tiêu chuẩn chất lượng mã.
4. **Mã sạch và dễ đọc:** Viết mã dễ hiểu, tuân thủ tiêu chuẩn mã hóa và thực hành tốt nhất, như sử dụng tài liệu rõ ràng và tránh lặp lại mã.
5. **Tích hợp liên tục và triển khai liên tục (CI/CD):** Sử dụng CI/CD để tự động hóa quá trình xây dựng, kiểm thử và triển khai, đảm bảo thay đổi mã được kiểm tra kỹ trước khi tích hợp.

VI. Kết luận

Lý thuyết cửa sổ vỡ, khi áp dụng trong lập trình, nhấn mạnh tầm quan trọng của việc duy trì chất lượng mã bằng cách xử lý lỗi nhỏ ngay lập tức. Bằng cách thực hiện xem xét mã định kỳ, kiểm thử tự động, và sử dụng công cụ phân tích, các đội phát triển có thể ngăn chặn nợ kỹ thuật và đảm bảo phần mềm đáng tin cậy, hiệu quả và dễ bảo trì. Dù có tranh cãi, bằng chứng cho thấy duy trì chất lượng mã từ đầu là chìa khóa cho thành công lâu dài.

TÀI LIỆU THAM KHẢO

[1] "The Atlantic," [Online]. Available: https://www.theatlantic.com/magazine/archive/1982/03/broken-windows/304465/.

[2] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Broken_windows_theory.

[3] "Artima," [Online]. Available: https://www.artima.com/articles/dont-live-with-broken-windows.

[4] "Coding Horror," [Online]. Available: https://blog.codinghorror.com/the-broken-window-theory/.

[5] "Medium," [Online]. Available: https://medium.com/@matryer/broken-windows-theory-why-code-quality-and-simplistic-design-are-non-negotiable-e37f8ce23dab.

[6] "StackOverflow," [Online]. Available: https://stackoverflow.blog/2021/10/18/code-quality-a-concern-for-businesses-bottom-lines-and-empathetic-programmers/.

[7] "Ekreative," [Online]. Available: https://www.ekreative.com/blog/how-code-quality-impacts-business-success/.

[8] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis.

[9] "Tech debt policy," [Online]. Available: https://techdebtpolicy.com/broken-windows-theory/.

[10] "Sonar," [Online]. Available: https://www.sonarsource.com/products/sonarqube/.