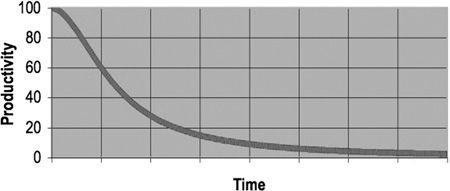
# Hiểu rõ hơn về clean code và phương pháp ứng dụng hiệu quả vào các dự án phần mềm.

Clean code hiểu đơn giản là code phải rõ ràng, dễ phát triển, bảo trì và mở rộng tính năng ứng dụng trong tương lai giúp giảm chi phí bảo trì và mở rộng dự án. Nhưng đó là nói đến đích đến, còn thực tế nó là cái gì và áp dụng vào dự án phần mềm như thế nào? Thì vẫn là câu hỏi với nhiều dự án thực tế ngày nay.



*Figure 1. Productivity vs. Time development*

Các chuyên gia phát triển phần mềm nổi tiếng trên thế giới cũng có những phát biểu khác nhau về việc này:

* *I like my code to be elegant and efficient. The logic should be straightforward to make it hard for bugs to hide, the dependencies minimal to ease maintenance, error handling complete according to an articulated strategy, and performance close to optimal so as not to tempt people to make the code messy with unprincipled optimizations. Clean code does one thing well.* - **Bjarne Stroustrup**
* *Clean code is simple and direct. Clean code reads like well-written prose. Clean code never obscures the designer’s intent but rather is full of crisp abstractions and straightforward lines of control.* - **Grady Booch**
* *Clean code can be read and enhanced by a developer other than its original author. It has unit and acceptance tests. It has meaningful names. It provides one way rather than many ways for doing one thing. It has minimal dependencies, which are explicitly defined, and provides a clear and minimal API. Code should be literate since depending on the language, not all necessary information can be expressed clearly in code alone.* - **Dave Thomas**
* *I could list all of the qualities that I notice in clean code, but there is one overarching quality that leads to all of them. Clean code always looks like it was written by someone who cares. There is nothing obvious that you can do to make it better. All of those things were thought about by the code’s author, and if you try to imagine improvements, you’re led back to where you are, sitting in appreciation of the code someone left for you—code left by someone who cares deeply about the craft. -* **Michael Feathers**
* *In recent years I begin, and nearly end, with Beck’s rules of simple code. In priority order, simple code: Runs all the tests; Contains no duplication; Expresses all the design ideas that are in the system; Minimizes the number of entities such as classes, methods, functions, and the like. -* **Ron Jeffries**

Còn nhìn từ khía cạnh chất lượng để giúp giảm chi phí phát triển, bảo trì và mở rộng trong tương lai chúng ta có thể sử dụng cách tiếp cận Total Productive Maintenance (TPM) từ Nhật có nền tảng là tập hợp các nguyên tắc 5S (Seiri, Seiton, Seiso, Seiketsu, Shitsuke) và thể hiện cuối cùng của chất lượng ứng dụng chính là source code. Như vậy chúng ta có thể ứng dụng chúng vào việc phát triển phần mềm như những quy tắc sau:

Seiri

Biết được

mọi thứ ở đâu bằng

cách sử dụng các cách tiếp cận như đặt tên phù hợp là rất quan

trọng.

Seiton

Tìm kiếm dễ

dàng đoạn code mình muốn tìm (nếu không hãy refactor lại chúng).

Seiso

Hãy giữ cho

code được sạch sẽ với việc xóa bỏ

code thừa và các comments

không cần thiết.

Seiketsu

Phong cách

code nhất quán và tập hợp các kinh nghiệm của nhóm.

Shitsuke

Kỷ luật phải

tuân theo quy định của nhóm.

Thể hiện cụ thể trong quá trình kiểm soát chất lượng source code là những quy tắc như:

* **Naming.** Tên ở khắp mọi nơi trong phần mềm như biến, hàm, đối số, lớp, tệp và gói, thư mục. Đặt tên phải cho bạn biết tại sao nó tồn tại, nó làm gì và sử dụng nó như thế nào và có thể phát âm và tìm kiếm. Đây là một kỹ thuật có mức độ trừu tượng cao vừa phải có kinh nghiệm lâu năm vừa phải hiểu nghiệp vụ hệ thống.
* **Quy tắc về các Function, Error Handling**. Các methods nên dài khoảng 20 đến 30 dòng và chỉ phụ trách một nhiệm vụ duy nhất, không vượt quá 3 đối số, có biến trả về để có thể testable hay việc xử lý lỗi trong method nên được throw ra (thay vì trả ra mã lỗi của method) và có một lớp riêng phụ trách về việc xử lý này và không return null (xử lý lỗi là quan trọng, nhưng nếu nó che khuất logic là nó sai). Ngoài ra việc đặt tên cũng phải tuân theo quy tắc Naming ở trên.
* **Quy tắc về Comments và Format**. Không cần thiết các comment thừa nếu tên đã đủ ý nghĩa hay việc comment file và sự thay đổi khi đã có các ứng dụng Source Control kiểm soát, tập trung vào các comment về mặt nghiệp vụ để làm sáng tỏ đoạn code. Và cũng cần phải chú ý đến định dạng code để làm sao cho việc đọc code trên một màn hình một cách dễ dàng cũng như việc xóa code thừa không dùng đến.
* **Quy tắc về việc sử dụng thư viện ngoài (Third-Party Libraries)**. Các thư viện phụ ngoài khi sử dụng cần phải wrapper (boundaries) lại những hàm mình cần dùng để sau này có thể nâng cấp thư viện hoặc việc thay thế thư viện không gây ảnh hưởng đến nghiệp vụ hệ thống.
* **Các quy tắc về TDD và SOLID**. Các quy tắc này tập trung cho việc thực hiện code design (viết sơ lược code như dạng detail design) và sử dụng kỹ thuật Test First. SOLID giúp chúng ta chia tách code thành những nhiệm vụ đơn và tăng cường sự kế thừa code cũng như giảm sự phụ thuộc lẫn nhau giữa các Layer code, đây là một bộ quy tắc phục vụ cho việc thiết kế hướng đối tượng. Còn TDD là tư duy phát triển ứng dụng đảm bảo chất lượng và có thể sử dụng được Automation test sau này…

Ngoài ra còn các quy tắc về Class, System, Concurrency…

# Ứng dụng vào dự án và những cải tiến

Với yêu cầu cao từ dự án cần chất lượng code tốt để giúp việc phát triển, bảo trì và mở rộng sau này dễ dàng hơn, chúng tôi đã đào tạo và áp dụng các kỹ thuật clean code và nguyên tắc SOLID cùng với một số tip nâng cao performance của hệ thống. Sau một thời gian đào tạo và áp dụng thì có những chuyển biến tích cực, code đã đơn giản, có trật tự và dẽ đọc hơn, dễ bảo trì và phát triển tính năng mới hơn, khớp nối giữa các components ít bị phụ thuộc vào nhau hơn dễ dàng hơn cho việc thay thế.

Nhưng cùng với những tiến bộ đó chúng tôi đã gặp phải những thách thức mới như một số code chất lượng kém vẫn bị lặp lại, việc đặt tên method và class, namespace vẫn chưa có tiến bộ nhiều, các methods viết ra với khả năng viết unit test cho nó thấp

và việc sử dụng 5 nguyên tắc SOLID vẫn còn yếu. Sau một thời gian phân tích chúng tôi đã đưa ra một số giải pháp như áp dụng quy trình kiểm soát chất lượng code cũng như tăng cường đào tạo nâng cao các kỹ thuật có mức độ trừu tượng cao cần phải có thời gian trải nghiệm:

* Áp dụng checklist cho các lỗi lặp lại nhiều lần
* Sử dụng Peer Review (sử dụng những người có kinh nghiệm để review code cho những bạn ít kinh nghiệm hơn)
* Đào tạo về Unit Test và Test-Driven Development
* Đào tạo về kỹ thuật phân tích yêu cầu theo góc nhìn của Developer và đào tạo kỹ năng nâng cao về Code Design như quy tắc Naming và SOLID.

Sau một thời gian áp dụng quy trình những lỗi cơ bản về chất lượng code gần như biến mất, việc đặt tên cũng tốt hơn và đặc biệt các code business đều có khả năng viết Unit Test. Các quy tắc SOLID như việc chia tách nghiệp vụ, sử dụng các bổ từ truy cập và phát triển các lớp trừu tượng đã chuẩn hơn đặc biệt là việc sử dụng interface để chia tách các layer code và các khớp nối tạo ra sự lỏng lẻo trong code đã được áp dụng tốt hơn.

Nhờ vào sự tiến bộ này chúng tôi đã rỡ bỏ một số quy trình như checklist cho những bạn gần như không còn tái phạm các lỗi cơ bản và Peer Review cho những bạn yếu hay mắc phải các lỗi về Naming và SOLID, TDD. Và cũng từ đây chúng tôi được khách hàng tin tưởng giao cho những công việc ở phân khúc cao hơn như phân tích nghiệp vụ và thiết kế cho các chức năng mới của hệ thống cùng với việc refactoring lại một số phần của hệ thống để đảm bảo việc phát triển thêm tính năng mới dễ dàng và dễ sửa đổi/fix bug đối với các tính năng đang có. Tốc độ fix bug và sửa đổi tính năng cũ đã rút ngắn lại còn ½ effort, quá trình phát triển thêm tính năng mới đã nhanh hơn và chỉ tốn 75% effort so với effort trước đây chúng tôi phải bỏ ra khi phát triển tính năng mới trên source code cũ.

Tại NTQ chúng tôi đã nhân rộng việc đào tạo/huấn luyện và triển khai áp dụng clean code vào hầu hết các dự án và đã đem lại giá trị cao về chất lượng source code khi deliver cho khách hàng, được khách hàng đánh giá cao về lợi ích mà nó mang lại cho việc phát triển hiện tại và bảo trì, mở rộng dự án sau này.