# **1. Khái niệm chất lượng mã nguồn**

Chất lượng mã xác định mã tốt (high quality) - và mã nguồn xấu (low quality).

Chất lượng, tốt, xấu - đều là chủ quan. Các nhóm khác nhau có thể sử dụng các định nghĩa khác nhau, dựa trên ngữ cảnh. Mã nguồn được coi là chất lượng cao có thể có ý nghĩa đối với một nhà phát triển phần mềm. Và nó có thể có nghĩa khác đối với một nhà phát triển ứng dụng web.

Vì lý do đó, chúng tôi giải thích chất lượng mã nguồn là gì, cách cải thiện chất lượng mã, các chỉ số chất lượng mã quan trọng là gì và các công cụ chất lượng mã có thể trợ giúp như thế nào.

**2. Tại sao chất lượng mã lại quan trọng**

Chất lượng mã nguồn rất quan trọng, vì nó ảnh hưởng đến chất lượng phần mềm tổng thể. Và chất lượng ảnh hưởng đến mức độ an toàn, bảo mật và đáng tin cậy của cơ sở mã của bạn.

Chất lượng cao là yếu tố quan trọng đối với nhiều nhóm phát triển ngày nay. Và nó đặc biệt quan trọng đối với những người đang phát triển các hệ thống quan trọng về bảo mật.

**3. Phân tích chất lượng mã: Mã tốt so với Mã xấu**

Mã nguồn tốt (chất lượng cao). Và đó là mã sạch. Nó đứng trước thử thách của thời gian.

Mã xấu ( chất lượng thấp). Nó sẽ không tồn tại lâu đâu.

Về cơ bản, mã được coi là tốt:

* Làm đúng công việc.
* Tuân theo một phong cách nhất quán.
* Dễ hiểu.
* Đã được ghi chép đầy đủ.
* Đã được kiểm tra.

**Kiểm tra là không đủ**

Lập trình viên không hoàn hảo. Kiểm tra và đánh giá mã thủ công sẽ không bao giờ tìm thấy mọi lỗi trong mã.

Một nghiên cứu về “ [Nguồn gốc và phương pháp loại bỏ lỗi phần mềm](https://www.ifpug.org/Documents/Jones-SoftwareDefectOriginsAndRemovalMethodsDraft5.pdf) ” cho thấy rằng các lập trình viên cá nhân có hiệu suất thấp hơn 50% trong việc tìm ra lỗi trong phần mềm của họ. Và hầu hết các hình thức kiểm tra chỉ đạt hiệu quả 35%. Điều này gây khó khăn cho việc xác định chất lượng.

**Lỗi mã hóa dẫn đến rủi ro**

Chất lượng của mã nguồn trong lập trình là quan trọng. Khi mã có chất lượng thấp, nó có thể gây ra [rủi ro về](https://www.perforce.com/blog/qac/3-best-practices-secure-software-development) an toàn hoặc bảo mật . Nếu phần mềm bị lỗi - do vi phạm bảo mật hoặc lỗ hổng an toàn - thì kết quả có thể rất thảm khốc hoặc gây tử vong.

**Chất lượng là trách nhiệm của mọi người**

Chất lượng là công việc của mỗi người. Người sản xuất. Người thử nghiệm. Người quản lý. Chất lượng cao phải là mục tiêu xuyên suốt quá trình phát triển.

**4. Làm thế nào để đo lường chất lượng mã nguồn**

Không có cách nào để đo lường chất lượng mã nguồn. Những gì đo lường có thể khác với những gì nhóm phát triển khác đo lường.

**Các khía cạnh chất lượng chính của mã nguồn để đo lường**

Dưới đây là năm đặc điểm chính để đo lường chất lượng cao hơn.

* *độ tin cậy*

Độ tin cậy đo xác suất hệ thống sẽ chạy mà không bị lỗi trong một khoảng thời gian hoạt động cụ thể. Nó liên quan đến số lượng lỗi và tính khả dụng của phần mềm.

Số lượng khuyết tật có thể được đo bằng cách chạy một [công cụ phân tích tĩnh](https://www.perforce.com/blog/qac/what-static-code-analysis) . Tính khả dụng của phần mềm có thể được đo bằng [thời gian trung bình giữa các lỗi (MTBF)](https://whatis.techtarget.com/definition/MTBF-mean-time-between-failures) . Số lượng lỗi thấp đặc biệt quan trọng để phát triển một cơ sở mã đáng tin cậy.

* *Khả năng bảo trì*

Khả năng bảo trì đo lường mức độ dễ dàng bảo trì của phần mềm. Nó liên quan đến kích thước, tính nhất quán, cấu trúc và độ phức tạp của codebase. Và việc đảm bảo mã nguồn có thể bảo trì dựa vào một số yếu tố, chẳng hạn như khả năng kiểm tra và khả năng hiểu.

Không thể sử dụng một số liệu duy nhất để đảm bảo khả năng bảo trì. Một số số liệu có thể xem xét để cải thiện khả năng bảo trì là số lượng cảnh báo theo phong cách và các biện pháp độ phức tạp Halstead. Cả người đánh giá tự động và con người đều cần thiết để phát triển các cơ sở mã nguồn có thể bảo trì.

* *Khả năng kiểm tra*

Khả năng kiểm thử đo lường mức độ phần mềm hỗ trợ các nỗ lực kiểm thử. Nó phụ thuộc vào mức độ bạn có thể kiểm soát, quan sát, cô lập và tự động hóa thử nghiệm, trong số các yếu tố khác.

Khả năng kiểm tra có thể được đo lường dựa trên số lượng trường hợp kiểm thử cần để tìm ra các lỗi tiềm ẩn trong hệ thống. Kích thước và độ phức tạp của phần mềm có thể ảnh hưởng đến khả năng kiểm tra. Vì vậy, việc áp dụng các phương pháp ở cấp mã nguồn - chẳng hạn như [độ phức tạp](https://www.perforce.com/blog/qac/what-cyclomatic-complexity) theo chu kỳ - có thể giúp cải thiện khả năng kiểm tra của thành phần.

* *Tính di động*

Khả năng di động đo lường mức độ khả dụng của cùng một phần mềm trong các môi trường khác nhau. Nó liên quan đến tính độc lập của nền tảng.

Không có một thước đo cụ thể nào về tính di động. Nhưng có một số cách có thể đảm bảo mã di động. Điều quan trọng là phải thường xuyên kiểm tra mã trên các nền tảng khác nhau, thay vì đợi cho đến khi kết thúc quá trình phát triển. Cũng nên đặt mức cảnh báo trình biên dịch của mình càng cao càng tốt - và sử dụng ít nhất hai trình biên dịch. Việc thực thi một tiêu chuẩn mã hóa cũng giúp cải thiện tính di động.

* *Khả năng tái sử dụng*

Khả năng tái sử dụng đo lường liệu các tài sản hiện có, chẳng hạn như mã nguồn có thể được sử dụng lại hay không. Nội dung được tái sử dụng dễ dàng hơn nếu chúng có các đặc điểm như tính mô đun hoặc khớp nối lỏng lẻo.

Khả năng tái sử dụng có thể được đo lường bằng số lượng phụ thuộc lẫn nhau. Chạy một trình phân tích tĩnh có thể giúp xác định những sự phụ thuộc lẫn nhau này.

**5. Các chỉ số chất lượng mã nào sẽ sử dụng**

Có một số chỉ số có thể sử dụng để định lượng chất lượng mã nguồn

* *Các chỉ số khiếm khuyết*

Số lượng khiếm khuyết và mức độ nghiêm trọng của các vấn đề đó là những thước đo quan trọng của chất lượng tổng thể.

Điêu nay bao gôm:

* Xác định giai đoạn bắt nguồn của khiếm khuyết.
* Số lượng báo cáo lỗi mở.
* Thời gian để xác định và sửa chữa các vấn đề.
* Mật độ khiếm khuyết (ví dụ, số lượng khiếm khuyết trên mỗi dòng mã).
* *Chỉ số độ phức tạp*

Các chỉ số đo độ phức tạp có thể giúp ích trong việc đo lường chất lượng.

Độ phức tạp chu kỳ đo lường số lượng đường dẫn độc lập tuyến tính thông qua mã nguồn của chương trình.

Một cách khác để hiểu chất lượng là tính toán các thước đo độ phức tạp của Halstead. Các biện pháp này:

* Từ vựng chương trình
* Thời gian chạy của chương trình
* Độ dài chương trình được tính toán

**6. Cách cải thiện chất lượng mã nguồn**

Đo lường chất lượng giúp bạn hiểu được vị trí của mình. Sau khi đã đo lường, có thể thực hiện các bước để cải thiện chất lượng tổng thể.

Bốn cách bạn có thể cải thiện chất lượng mã:

1. Sử dụng một tiêu chuẩn mã hóa.

2. Phân tích mã nguồn - trước khi đánh giá mã nguồn.

3. Thực hiện theo các phương pháp hay nhất về đánh giá mã nguồn.

4. Refactor mã nguồn kế thừa (khi cần thiết)

**Cải thiện chất lượng mã nguồn:**

* **Sử dụng tiêu chuẩn mã hóa**

Sử dụng [tiêu chuẩn mã hóa](https://www.perforce.com/resources/qac/coding-standards) là một trong những cách tốt nhất để đảm bảo mã nguồn chất lượng cao.

Tiêu chuẩn mã hóa đảm bảo mọi người sử dụng đúng kiểu. Nó cải thiện tính nhất quán và khả năng đọc của cơ sở mã. Đây là chìa khóa để giảm độ phức tạp và chất lượng cao hơn.

Cách tốt nhất để sử dụng tiêu chuẩn mã hóa là:

* Đào tạo các nhà phát triển
* Giúp họ tuân thủ nó

Có thể thực hiện việc này bằng cách sử dụng [trình phân tích mã nguồn tĩnh.](https://www.perforce.com/blog/qac/how-enforce-coding-standards-automated-static-analysis)

* **Phân tích- Trước khi đánh giá mã nguồn**

Chất lượng nên được ưu tiên ngay từ khi bắt đầu phát triển. Không phải lúc nào thời gian cũng xa xỉ khi quá trình phát triển tiến triển. Đó là lý do tại sao điều quan trọng là phải phân tích mã *trước khi* bắt đầu xem xét mã. Và tốt nhất nên phân tích mã nguồn ngay sau khi nó được viết.

Trong [DevOps](https://www.perforce.com/solutions/devops) , phân tích mã diễn ra trong giai đoạn tạo. Máy phân tích tĩnh có thể được chạy qua mã ngay sau khi nó được viết. Điều này tạo ra một vòng phản hồi tự động, vì vậy các nhà phát triển có thể cải thiện mã trước khi nó chuyển sang giai đoạn xem xét mã.

Xét cho cùng, bạn phát hiện ra lỗi càng sớm thì việc giải quyết chúng càng nhanh, dễ dàng và rẻ hơn.

Cách tốt nhất để [cải thiện chất lượng là phân tích mã](https://www.perforce.com/webinars/how-improve-quality-static-code-analysis) tự động. Bằng cách chạy trình phân tích tĩnh trên mã sớm và thường xuyên, đảm bảo mã đến giai đoạn xem xét mã có chất lượng cao nhất có thể. Ngoài ra, có thể sử dụng máy phân tích tĩnh (chẳng hạn như [Helix QAC](https://www.perforce.com/products/helix-qac) và [Klocwork](https://www.perforce.com/products/klocwork) ) để theo dõi các chỉ số chất lượng chính.

* **Thực hiện theo các phương pháp hay nhất về đánh giá mã**

Việc xem xét mã thủ công vẫn rất quan trọng để xác minh mục đích của mã. Khi việc đánh giá mã được thực hiện tốt, chúng sẽ cải thiện chất lượng phần mềm tổng thể.

Cách tốt nhất để thực hiện [đánh giá mã là làm theo các phương pháp hay nhất](https://www.perforce.com/blog/qac/9-best-practices-code-reviews)  và tận dụng các công cụ tự động.

**4. Refactor Mã kế thừa (Khi cần thiết)**

Một cách để cải thiện chất lượng của cơ sở mã nguồn hiện có là thông qua cấu trúc lại. Cấu trúc lại mã kế thừa có thể giúp bạn làm sạch cơ sở mã của mình và giảm độ phức tạp của nó.

*Làm thế nào để làm nó*

Cách tốt nhất để cải thiện cơ sở mã nguồn kế thừa là làm điều đó dần dần. Dưới đây là [tám mẹo để cải thiện mã kế thừa](https://www.perforce.com/blog/qac/8-tips-working-legacy-code) (mà không ảnh hưởng đến phần mềm của bạn).

**7. phân tích chất lượng mã nguồn**

Việc phân tích và đo lường chất lượng có thể rất phức tạp, vì chất lượng có thể mang tính chủ quan. Có thể sử dụng một số số liệu để đánh giá khách quan mã, bao gồm cả độ phức tạp theo chu kỳ. Và có một số cách có thể giảm độ phức tạp và cải thiện chất lượng.

Mã hóa chất lượng có thể mất nhiều thời gian và nỗ lực hơn trong lần đầu tiên vượt qua. Nhưng bằng cách sớm giới thiệu chất lượng, bạn sẽ giảm chi phí bảo trì và sửa lỗi về lâu dài. Và bạn sẽ giảm nợ kỹ thuật của mình.

**Chọn đúng công cụ đánh giá chất lượng mã nguồn**

Sử dụng các [công cụ đánh giá chất lượng mã](https://www.perforce.com/solutions/static-analysis) nguồn phù hợp , bao gồm cả bộ phân tích tĩnh, là chìa khóa.

Các bộ phân tích tĩnh - chẳng hạn như [Helix QAC](https://www.perforce.com/products/helix-qac) và [Klocwork](https://www.perforce.com/products/klocwork) - giúp dễ dàng đảm bảo rằng mã của bạn có chất lượng cao. Bạn sẽ cải thiện chất lượng bằng cách:

* Áp dụng các tiêu chuẩn mã hóa.
* Phân tích mã tự động.
* Tuân theo các phương pháp hay nhất về mã hóa.
* Cấu trúc lại mã kế thừa.

Tài liệu tham khảo: https://www.perforce.com/blog/sca/what-code-quality-overview-how-improve-code-quality