**16.7: Kỹ năng bảo trì sau giao hàng so với kỹ năng phát triển phần mềm.**

Những kỹ năng cần thiết cho việc bảo trì đã đc nhắc đến ở các phần trước:

* Đối với việc bảo trì sửa lỗi, kỹ năng xác định nguyên nhân trong 1 sản phẩm lớn được coi là cần thiết. Tuy nhiên kỹ năng không chỉ được sử dụng trong duy nhất công việc bảo trì mà còn được dùng xuyên suốt quá trình tích hợp và kiểm thử sản phẩm.
* Một kỹ năng tối quan trọng nữa là khả năng làm việc hiệu quả khi không có một lượng tài liệu đầu đủ. Tại vì tài liệu khó có thể được hoàn thiện trong khi sản phẩm vẫn còn đang trong quá trình tích hợp và kiểm thử.
* Kỹ năng liên quan đến hoạt động phân tích, thiết kế, cài đặt và kiểm thử cũng khá quan trọng trong việc bảo trì thích ứng và hoàn thiện. Đây là những hoạt động được tiến hành trong suốt quá trình phát triển và mỗi hoạt động để được thực hiện chuẩn chỉ lại đòi hỏi những kỹ năng chuyên biệt.

Nói cách khác, những kỹ năng cần có để bảo trì sản phẩm sau ki giao hàng đều là những kỹ năng cần thiết trong các mảng khác của quá trình phát triển sản phẩm. **Điểm mấu chốt đó là 1 lập trình viên bảo trì không chỉ đơn thuần là có kỹ năng trong nhiều lĩnh vực mà còn phải có trình độ cao trong tất cả các lĩnh vực đó**. Mặc dù một nhà phát triển phần mềm có thể chuyên về 1 lĩnh vực trong ngành như là thiết kế, kiểm thử nhưng người bảo trì phần mềm bắt buộc phải là 1 chuyên gia trong ngành sản xuất phần mềm. Sau cùng ngoài những thứ cần thiết ở trên thì bảo trì với phát triển phần mềm là như nhau.

**16.8: Kỹ thuật đảo ngược.**

Như đã được chỉ ra, đôi khi tài liệu duy nhất dành cho bảo trì chỉ là **source code** (thực tế chyên này xảy ra khá thường xuyên khi bảo trì các hệ thống được kế thừa, đó là những phần mềm được phát triển hơn 15-20 năm trước). Trong trường hợp đó, bảo trì code trở nên cực kỳ khó nhằn. Để giải quyết vấn đề này thì chỉ có cách xem source code và cố tái tạo các bản thiết kế hay thậm chí là cả những thông só kỹ thuật. **Quá trình này được gọi là kỹ thuật đảo ngược**.

Hỗ trợ cho quá trình này chúng ta có **CASE tools** (Hệ thống các công cụ được sử dụng để thiết kế và phát triển các phần mềm với sự trợ giúp của máy tính). Một trong số đó là Pretty Printer, phần mềm giúp trình bày source code một cách rõ ràng, trực quan. Có những công cụ giúp chúng ta tạo lưu đồ trực tiếp từ Source Code như **UML diagram** hay **flow-chart.** Những công cụ hổ trợ việc trực quan hóa này sẽ giúp ta tái tạo lại thiết kế của phần mềm.

Khi nhóm bảo trì đã xây dựng lại thiết kế, có hai khả năng xảy ra. Một là cố gắng tạo lại các thông số kỹ thuật cụ thể, chỉnh sửa các thông số kỹ thuật nhằm rút ra những thay đổi cần thiết và hoàn thiện sản phẩm như bình thường. **(Trong ngữ cảnh của kỹ thuật đảo ngược, quy trình phát triển thông thường được thực hiện từ phân tích cho đến thiết kế và rồi thực thi được gọi là kỹ thuật chuyển tiếp. Quá trình đảo ngược xong tiếp đó đến kỹ thuật chuyển tiếp được gọi là tái cấu trúc.)** Trong thực tế, việc xây dựng lại các thông số kỹ thuật cụ thể là một công việc cực kỳ khó. Thông thường, thiết kế được tái tạo được chỉnh sửa và **thiết kế sau chỉnh sửa đó được** **triển khai theo kỹ thuật chuyển tiếp**.

Một hoạt động liên quan thường được thực hiện trong quá trình bảo trì đó là việc tái cấu trúc. Kỹ thuật đảo ngược đưa sản phẩm từ mức trừu tượng thấp hơn lên mức trừu tượng cao hơn, ví dụ**, từ mức source code lên mức thiết kế**. Kỹ thuật chuyển tiếp đưa sản phẩm từ cấp cao hơn xuống mức độ trừu tượng đến mức thấp hơn. Tuy nhiên, việc tái cấu trúc diễn ra ở cùng một cấp độ.

Đó là quá trình cải tiến sản phẩm mà không làm thay đổi chức năng của nó. **Pretty Printing** là một hình thức tái cấu trúc và việc chuyển đổi mã từ hình thức không có cấu trúc sang hình thức có cấu trúc cũng vậy. Nói chung, tái cấu trúc được thực hiện để giúp mã nguồn (thiết kế hoặc cơ sở dữ liệu) **dễ bảo trì hơn**. Khi một quy trình nhanh (Phần 2.9.5) được sử dụng, việc sửa đổi thiết kế được hay còn được gọi là thay đổi cấu trúc mà ko làm thay đổi hành vi bên ngoài của hệ thống (Refactoring) là một ví dụ khác của việc tái cấu trúc.

Tình huống xấu hơn xảy ra nếu mã nguồn bị mất và phiên bản chạy được của sản phẩm là tất cả những gì có sẵn. Nhìn thoáng, có vẻ như đây là cách duy nhất có thể để tạo lại mã nguồn là sử dụng trình phân dịch(Disasembler) để tạo mã máy và sau đó tạo một công cụ

(có thể được gọi là trình biên dịch ngược) để cố gắng tái tạo mã của ngôn ngữ cấp cao ban đầu.Phương pháp này có một số vấn đề khó có thể tránh được như:

• Tên của các biến sẽ bị mất là hệ quả của việc biên dịch ngược.

• Nhiều trình biên dịch tối ưu hóa mã mà bằng 1 cách nào đó gây khó khăn cho việc cố gắng tái tạo mã nguồn.

• Một cấu trúc chẳng hạn như một vòng lặp trong trình hợp dịch hoàn toàn có thể giống với một số cấu trúc có trong mã nguồn.

Do đó, trên thực tế, sản phẩm đang tồn tại được coi như một hộp đen và kỹ thuật đảo ngược được sử dụng để suy ra các thông số kỹ thuật từ hoạt động của sản phẩm hiện tại. **Các các thông số kỹ thuật có được từ việc tái cấu trúc được sửa đổi theo yêu cầu và một phiên bản mới của sản phẩm được triển khai theo kỹ thuật chuyển tiếp từ các thông số kỹ thuật đó.**