**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 này 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 triển khai 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 kiến thiết**. 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 kiến thiết** 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, việc **tái kiến thiết** 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 kiến thiết**.

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 cách duy nhất để có thể tái tạo mã nguồn là sử dụng trình phân dịch(Disasembler) để tạo mã máy và sau đó tạo một công cụ (có thể là trình biên dịch ngược) để cố gắng tái tạo mã nguồn của ngôn ngữ cấp cao ban đầu. Phương pháp này có một số vấn đề khó có thể tránh khỏi như:

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

• Cách tối ưu hóa mã của nhiều trình biên dịch gây khó khăn cho việc tái tạo mã nguồn.

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

Do đó, trên thực tế, sản phẩm đang tồn tại được coi như là một hộp đen và kỹ thuật đảo ngược được sử dụng để suy ra các thông số kỹ thuật từ cách thức hoạt động của sản phẩm hiện tại. **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.**

**16.9: Kiểm thử trong bảo trì hậu giao hàng**

Trong khi sản phẩm đang được phát triển, nhiều thành viên của nhóm phát triển có cái nhìn tổng quan về toàn bộ sản phẩm, nhưng do sự luân chuyển nhân sự chóng mặt trong ngành công nghiệp máy tính, chưa chắc rằng các thành viên của nhóm bảo trì sau giao hàng có tham gia vào việc phát triển ban đầu của dự án. Do đó, người bảo trì thường nhìn sản phẩm giống như một tập hợp các thành phần có liên quan lỏng lẻo và thường không nhận thức được rằng thay đổi đối với một mã tạo tác có thể ảnh hưởng nghiêm trọng đến một hoặc nhiều tạo tác khác nói riêng và cả sản phẩm nói chung. Ngay cả khi người bảo trì muốn hiểu mọi khía cạnh của sản phẩm thì áp lực phải sửa chữa hoặc mở rộng sản phẩm nói chung không cho phép người bảo trì có thời gian cần thiết để đạt được điều này. Hơn nữa, trong nhiều trường hợp, gần như không có chút tài liệu nào để hỗ trợ việc đạt được hiểu biết đó. Một cách để giảm thiểu khó khăn này là sử dụng kiểm tra hồi quy, nghĩa là kiểm tra sản phẩm đã thay đổi sử dụng các test case trước đó để đảm bảo rằng sản phẩm vẫn hoạt động chính xác.

Vì lý do này, việc lưu trữ tất cả các test case cùng với kết quả của chúng (ở dạng máy tính có thể đọc được) rất quan trọng. **Do những thay đổi được thực hiện đối với sản phẩm, một số test case nhất định được lưu trữ có thế sẽ phải được sửa đổi.**

- **Ví dụ**:

+ Nếu các phần trăm khấu trừ của tiền lương bị thay đổi do luật thuế, thì kết quả đầu ra chính xác từ bảng lương cho mỗi test case liên quan đến việc khấu trừ cũng sẽ thay đổi theo.

+ Tương tự , nếu các quan sát vệ tinh dẫn đến hiệu chỉnh về vĩ độ và kinh độ của một hòn đảo, thì kết quả đầu ra chính xác từ một sản phẩm phần mềm dùng để tính toán vị trí của một máy bay bằng cách sử dụng tọa độ của hòn đảo phải cũng sẽ phải thay đổi tương ứng. Dựa vào việc bảo trì được thực hiện, một số test case hợp lệ trở nên không hợp lệ. Nhưng việc tính toán để sửa các test case được lưu trữ về cơ bản giống như việc tạo mới dữ liệu kiểm thử để kiểm tra việc bảo trì đã được thực hiện chính xác hay chưa. Vì thế việc bảo trì file chứa các test case và kết quả đầu ra kỳ vọng của chúng là không cần thiết.

Có ý kiến cho rằng việc kiểm thử hồi quy là việc làm phí thời gian vì kiểm thử hồi quy đòi hỏi sản phẩm hoàn thiện phải được kiểm thử lại với một nhóm test case, thứ mà rõ ràng chả có liên quan gì đến các tạo tác mã bị sửa đổi trong quá trình bảo trì sản phẩm. Nhưng mối nguy từ các tác dụng phụ vô tình tạo ra trong việc bảo trì là quá lớn vì thế kiểm thử hồi quy là một khía cạnh quan trọng của việc bảo trì trong tất cả các trường hợp

**16.10: Các công cụ hỗ trợ cho việc bảo trì hậu giao hàng.**

Lập trình viên bảo trì khó có thể tự mình theo dõi thủ công tất cả các số của các phiên bản và đánh số mới mỗi lần có mã tạo tác được cập nhật.

Trừ khi hệ điều hành có sẵn trình quản lý phiên bản thì máy nào cũng cần có một công cụ quản lý phiên bản như:

* sccs (source code control system – hệ thống kiểm soát source code)
* rcs (revision control system – hệ thống kiểm soát phiên bản)

Ngoài ra các công cụ kiểm soát cấu hình cũng rất cần thiết như:

* CVS (concurrent versions system – hệ thống các phiên bản đồng thời) (Mã nguồn mở)
* Subversion (Mã nguồn mở)
* CCC
* IBM Rational ClearCase

Kể cả tổ chức phần mềm không có nhu cầu mua một công cụ kiểm soát cấu hình hoàn chỉnh thì ít nhất một công cụ xây dựng phải được dùng kèm với công cụ kiểm soát phiên bản. Một danh mục khác của các công cụ hỗ trợ tương đối quan trọng với công việc bảo trì đó là 1 công cụ theo dõi quản lý lỗi, nó sẽ giũ một bản ghi gồm các lỗi chưa được sửa.

Việc theo dõi lỗi là một khía cạnh quan trọng trong việc bảo trì hậu giao hàng. Việc định trước trạng thái của các lỗi đã được ghi lại là rất quan trọng. IBM Rational ClearQuest là công cụ theo dõi lỗi thương mại của tập đoàn IBM, trong khi đó Bugzilla là công cụ theo dõi lỗi mã nguồn mở. Những công cụ này được dùng để ghi lại tính nghiêm trọng cua một lỗi và trạng thái của nó (đã được sửa hay chưa). Ngoài ra, một số công cụ theo dõi lõi còn có thể dẫn một báo cáo lỗi đến một công cụ quản lý cấu hình để từ đó khi có một phiên bản mới được dựng, lập trình viên bảo trì có thể chọn các bản vá báo cáo lỗi cụ thể để cho vào bản dựng.

Việc bảo trì hậu giao hàng rất khó và vất vả. Thứ tối thiếu mà quản lý có thể làm là cung cấp cho đội bảo trì các công cụ cần thiết để làm việc hiệu quả.

**16.11: Các số liệu cho bảo việc bảo trì hậu giao hàng.**

Các hoạt động chính của công việc bảo trì hậu giao hàng đó là phân tích, thiết kế, triển khai, kiểm thử và viết tài liệu. Từ đó, các số liệu dùng để đo các hoạt động trên được áp dụng như nhau với việc bảo trì. Ví dụ:

* Các số liệu phức tạp ở mục 15.3.2 có liên quan đến công việc bảo trì hậu giao hàng, trong dó một mã tạo tác với độ phức tạp cao là một ứng cử viên sáng giá cho việc gây ra lỗi hồi quy. Việc chỉnh sửa mã tạo tác như vậy cần phải hết sức cẩn thận.

Ngoài ra, các số liệu cụ thể cho công việc bảo trì hậu giao hàng còn bao gồm cả các giải pháp liên quan đến báo cáo lỗi phần mềm như là tổng số lượng lỗi được báo cáo và phân loại của các lỗi đó theo mức độ nghiêm trọng và loại lỗi. Thêm vào đó, thông tin về tình trạng của các bảo cáo lỗi cũng cần thiết. Ví dụ: Việc có 13 lỗi nghiêm trọng được báo cáo nhưng đã được sửa vào năm 2006 tương đối là khác so với việc chỉ có 2 lỗi nghiêm trọng được báo cáo những chưa cái nào được sửa.

**16.13: Các thách thức với công việc bảo trì hậu giao hàng.**

Việc bảo trì thường khó hơn việc phát triển thế nhưng các lập trình viên bảo trì thường bị các nhà phát triển coi thường và thường xuyên nhận lương thấp hơn các nhà phát triển.