

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN

BÀI GIẢNG
ĐẢM BẢO CHẤT LƯỢNG PHẦN MỀM
(Software Quality Assurance)

Giới thiệu

Trước những thách thức trong quá trình phát triển phần mềm, việc đảm bảo chất lượng phần mềm (Software Quality Assurance-SQA) là hết sức quan trọng, đòi hỏi phải nghiên cứu một cách nghiêm túc để thực thi hiệu quả. Tài liệu này cung cấp những kiến thức cơ bản về chất lượng phần mềm, đảm bảo chất lượng trong một dự án phát triển phần mềm. Qui trình xây dựng hệ thống đảm bảo chất lượng phần mềm cũng được trình bày trong nội dung bài giảng. Qua đó, sinh viên hiểu được cách thức xây dựng một hệ thống đảm bảo chất lượng phần mềm và vai trò của những thành viên trong hệ thống. Một số chuẩn đảm bảo chất lượng cũng được giới thiệu trong những chương cuối. Thông qua nội dung bài giảng sinh viên cũng sẽ nắm được kỹ năng rà soát và kiểm thử phần mềm.

Tài liệu được soạn phần lớn dựa trên cuốn sách *Software Quality Assurance From Theory to Implementation* của Daniel Galin và một số tài liệu về kỹ nghệ phần mềm, nhằm hỗ trợ cho sinh viên gặp khó khăn khi đọc các tài liệu nguyên gốc tiếng Anh.

Nội dung bài giảng được xây dựng gồm bảy chương:

Chương 1. Tổng quan về đảm bảo chất lượng phần mềm

Những khái niệm mở đầu của tài liệu được giới thiệu trong chương 1. Bắt đầu với khái niệm phần mềm, chất lượng phần mềm và đảm bảo chất lượng phần mềm, phần tiếp theo phân tích các yếu tố chất lượng phần mềm.

Chương 2. Các thành phần chất lượng phần mềm tiền dự án

Chương này trình bày những nội dung liên quan đến những thành phần đảm bảo chất lượng phần mềm tiền dự án bao gồm việc rà soát hợp đồng, kế hoạch phát triển dự án phần mềm và kế hoạch chất lượng phần mềm.

Chương 3. Các thành phần SQA trong vòng đời dự án

Chương 3 đề cập đến các thành phần đảm bảo chất lượng phần mềm trong vòng đời dự án phần mềm. Những nội dung được trình bày trong chương này bao gồm : phân tích một số mô hình phát triển phần mềm phổ biến, các phương pháp rà soát, bảo trì phần mềm và các công cụ CASE. Riêng kiểm thử phần mềm là bước quan trọng sẽ được trình bày riêng ở chương 4.

Chương 4. Kiểm thử phần mềm

Những nội dung được trình bày trong chương này bao gồm : khái niệm cơ bản, các mức kiểm thử, các kỹ thuật kiểm thử, và quá trình kiểm thử. Các loại thành phần được dùng trong kiểm thử phần mềm, các phần mềm phục vụ kiểm thử và thư viện JUnit được sử

dụng rộng rãi trong kiểm thử đơn vị cho ngôn ngữ lập trình Java.

Chương 5. Các thành phần cơ bản của chất lượng phần mềm

Các thành phần cơ bản của chất lượng phần mềm bao gồm các thủ tục (procedure), chỉ dẫn (instruction), khuôn mẫu (templates), checklists (danh mục kiểm tra). Đó chính là nội dung được trình bày trong phần đầu của chương 6. Phần tiếp theo sẽ trình bày các hoạt động đảm bảo chất lượng phần mềm khác như : đào tạo và cấp chứng chỉ, ngăn ngừa và sửa lỗi, quản lý cấu hình và kiểm soát tài liệu.

Chương 6. Các thành phần quản lý chất lượng phần mềm

Ngoài yếu tố kỹ thuật, trong các dự án phát triển phần mềm hiện đại, yếu tố quản lý đóng vai trò hết sức quan trọng. Chương 7 trình bày các vấn đề liên quan đến quản lý chất lượng phần mềm như : điều khiển tiến độ dự án, độ đo chất lượng phần mềm, chi phí chất lượng phần mềm.

Chương 7. Các chuẩn đảm bảo chất lượng

Chương này đề cập tới các chuẩn quản lý chất lượng như ISO 9001 và ISO 9000-3, CMM và CMMI và các chuẩn tiến trình dự án như IEEE/EIA Std 12207, IEEE Std 1012, IEEE Std 1028.

Chương 1. Tổng quan về đảm bảo chất lượng phần mềm

1.1 Giới thiệu về phần mềm

1.1.1 Phần mềm

Phần mềm bao gồm những thành phần sau đây:

- Chương trình máy tính
- Các thủ tục
- Tài liệu liên quan
- Dữ liệu cần thiết cho sự vận hành của hệ thống

Mỗi thành phần phần mềm đều có chức năng riêng và chất lượng của chúng đóng góp vào chất lượng chung của phần mềm và bảo trì phần mềm như sau:

1. Chương trình máy tính được cần thiết là hiển nhiên vì chúng giúp máy tính vận hành thực thi các yêu cầu ứng dụng.
2. Những thủ tục được yêu cầu để định nghĩa theo một thứ tự và lịch biểu của một chương trình khi thực thi, phương thức được triển khai và người chịu trách nhiệm cho thực thi các hoạt động cần thiết cho việc tác động vào phần mềm
3. Nhiều kiểu tài liệu là cần thiết cho người phát triển, người sử dụng và người có nhiệm vụ duy trì. Tài liệu phát triển (báo cáo yêu cầu, báo cáo thiết kế, mô tả chương trình, v.v) cho phép sự phối hợp và cộng tác hiệu quả giữa các thành viên trong đội ngũ phát triển và hiệu quả trong việc xem lại và rà soát cá sản phẩm lập trình và thiết kế. Tài liệu sử dụng (thường là hướng dẫn sử dụng) cung cấp một sự miêu tả cho ứng dụng sẵn sàng và những phương pháp thích hợp cho họ sử dụng. Tài liệu bảo trì (tài liệu cho người phát triển) cung cấp cho đội bảo trì tất cả những thông tin yêu cầu về mã nguồn và công việc và cấu trúc cho từng module. Thông tin này được sử dụng để tìm nguyên nhân lỗi (bugs) hoặc thay đổi hoặc bổ sung thêm vào phần mềm có sẵn.
4. Dữ liệu bao gồm các tham số đầu vào, mã nguồn và danh sách tên thích hợp với phần mềm để đặc tả những cái cần thiết cho người sử dụng thao tác với hệ thống. Một kiểu khác của dữ liệu cần thiết là chuẩn dữ liệu test, sử dụng để xác định rõ những thứ thay đổi không mong muốn trong mã nguồn hoặc dữ liệu phần mềm đã từng xảy ra và những loại sự cố phần mềm nào có thể được lường trước.

1.1.2 Các vấn đề hay gặp phải khi phát triển phần mềm

Có thể nói phần mềm là một sản phẩm đặc biệt, nó không giống như các sản phẩm công nghiệp khác nên người ta thường gọi là phát triển phần mềm. Để phân biệt sự khác nhau giữa sản phẩm phần mềm với các sản phẩm khác ta sẽ xem xét ba đặc điểm sau :

- (1) Độ phức tạp của sản phẩm : Độ phức tạp của sản phẩm có thể được đo bằng số lượng phương thức vận hành của sản phẩm. Một sản phẩm công nghiệp thậm chí là một máy tiên tiến cũng không cho phép nhiều hơn vài trăm phương thức vận hành. Trong khi đó, một gói phần mềm có thể có tới hàng triệu khả năng vận hành. Do đó, vấn đề đảm bảo vô số khả năng vận hành được xác định và phát triển đúng là một thách thức chính của công nghiệp phần mềm.
- (2) Tính trực quan của sản phẩm : Trong khi các sản phẩm công nghiệp có thể nhìn thấy được, thì các sản phẩm phần mềm đều vô hình. Hầu hết các nhược điểm của một sản phẩm công nghiệp đều có thể phát hiện trong tiến trình sản xuất. Hơn nữa, rất dễ dàng nhận thấy được sự khuyết thiếu một phần nào đó trong một sản phẩm công nghiệp (ví dụ : một cái ôtô không có cửa sổ). Trái lại, các nhược điểm trong các sản phẩm phần mềm (được lưu trữ trong các đĩa mềm hay CD) đều không nhìn thấy được, vì vậy, thực tế là các phần của một gói phần mềm có thể thiếu ngay từ đầu.
- (3) Tiến trình sản xuất và phát triển phần mềm : Các pha trong tiến trình sản xuất một sản phẩm
 - Phát triển sản phẩm : trong sản xuất công nghiệp, người thiết kế và các nhân viên đảm bảo chất lượng kiểm tra nguyên mẫu để phát hiện các khuyết điểm của chúng. Trong sản xuất phần mềm, các chuyên gia đảm bảo chất lượng và đội phát triển có xu hướng tìm ra các lỗi sản phẩm vốn có. Kết quả cuối cùng của pha này là một nguyên mẫu đã được phê chuẩn, sẵn sàng để sản xuất.
 - Lập kế hoạch sản xuất sản phẩm : tại pha này, trong các ngành công nghiệp, tiến trình sản xuất và các công cụ được thiết kế và chuẩn bị. Một số dòng sản phẩm đặc biệt cần phải được thiết kế và xây dựng. Do đó, pha này đã tạo thêm cơ hội xem xét sản phẩm, và có thể phát hiện ra các khuyết điểm đã bị người rà soát và kiểm thử bỏ qua trong pha phát triển. Ngược lại, đây là pha không yêu cầu trong tiến trình sản xuất phần mềm, bởi việc sản xuất các bản copy phần mềm và in các sách hướng dẫn phần mềm được thực hiện tự động. Điều này được áp dụng cho bất kỳ sản phẩm phần mềm nào, từ nhỏ tới lớn.

- Sản xuất : Trong pha này, các thủ tục đảm bảo chất lượng trong sản xuất công nghiệp được áp dụng để phát hiện lỗi sản xuất. Các khuyết điểm trong sản phẩm được phát hiện ra ở giai đoạn đầu tiên của quá trình sản xuất có thể được hiệu chỉnh bằng một thay đổi trong thiết kế sản phẩm hoặc nguyên liệu, hay trong các công cụ sản xuất...Nhờ đó có thể tránh được các khuyết điểm này trong các sản phẩm được sản xuất trong tương lai. Ngược lại, như đã nói ở phần trước, việc sản xuất phần mềm đơn giản chỉ là sao chép các sản phẩm và in các sách hướng dẫn, do đó việc phát hiện các khuyết điểm của sản phẩm rất khó khăn.

Kỹ nghệ phần mềm đã có những bước phát triển đáng kể và vượt qua nhiều giai đoạn khủng hoảng. Những kết quả nghiên cứu về kỹ nghệ phần mềm đã giúp các tổ chức phát triển phần mềm một cách chuyên nghiệp hơn. Môi trường phát triển phần mềm cũng mang những nét đặc trưng riêng. Với bảy đặc trưng sau ta có thể hiểu rõ hơn về môi trường phát triển cũng như môi trường bảo trì phần mềm chuyên nghiệp:

- (1) Các điều kiện hợp đồng : Là kết quả của các cam kết và điều kiện trong bản hợp đồng giữa nhà phát triển phần mềm và khách hàng, các hoạt động bảo trì và phát triển phần mềm cần đương đầu với các vấn đề :

- Một danh sách các yêu cầu chức năng được xác định mà phần mềm được phát triển và công việc bảo trì nó phải thực hiện.
- Ngân sách dự án.
- Thời gian biểu dự án.

Nhà quản lý việc phát triển phần mềm và bảo trì dự án cần nỗ lực lớn trong việc giám sát các hoạt động để đạt được các yêu cầu của hợp đồng.

- (2) Mối quan hệ khách hàng – nhà cung cấp : Trong suốt quá trình phát triển và bảo trì phần mềm, các hoạt động đều nằm dưới sự giám sát của khách hàng. Đội dự án phải hợp tác liên tục với khách hàng : để xem xét các yêu cầu thay đổi, để thảo luận những gì khách hàng không bằng lòng về các khía cạnh khác nhau của dự án, và để đạt được sự chấp thuận cho các thay đổi theo sáng kiến của đội phát triển.

- (3) Yêu cầu làm việc theo nhóm : 3 nhân tố thường thúc đẩy việc thành lập một đội dự án thay vì giao dự án cho một chuyên gia :

- Các yêu cầu về thời gian biểu. Nói cách khác, khối lượng công việc được thực hiện trong suốt thời kỳ dự án đòi hỏi sự tham gia của nhiều người nếu muốn dự án hoàn thành

đúng thời hạn.

- Để thực hiện được dự án cần có nhiều chuyên ngành khác nhau.
 - Sự rà soát lại và hỗ trợ lẫn nhau của các chuyên gia sẽ làm tăng chất lượng dự án.
- (4) Hợp tác và phối hợp với các đội phần mềm khác : Để thực hiện được các dự án, đặc biệt là các dự án có quy mô lớn, cần nhiều hơn một đội dự án. Đây là điều rất phổ biến trong công nghiệp phần mềm. Trong các trường hợp như thế, có thể đòi hỏi phải hợp tác với :
- Các đội phát triển phần mềm khác trong cùng một tổ chức.
 - Các đội phát triển phần cứng trong cùng một tổ chức.
 - Các đội phát triển phần cứng và phần mềm của các nhà cung cấp khác.
 - Các đội phát triển phần cứng và phần mềm của khách hàng – những người tham gia một phần vào sự phát triển dự án.
- (5) Các giao diện với các hệ thống phần mềm khác : Ngày nay, hầu hết hệ thống phần mềm đều có các giao diện với các gói phần mềm khác nhau. Các giao diện này cho phép các dữ liệu dưới dạng điện tử được “chạy” giữa các hệ thống phần mềm. Có thể định nghĩa các loại giao diện chính sau đây :
- Các giao diện đầu vào – nơi các hệ thống phần mềm khác truyền dữ liệu tới hệ thống phần mềm của bạn.
 - Các giao diện đầu ra – nơi hệ thống phần mềm của bạn truyền dữ liệu đã được xử lý tới các hệ thống phần mềm khác.
 - Các giao diện đầu vào và đầu ra tới các bảng điều khiển của máy, như trong các hệ thống kiểm soát thí nghiệm và các hệ thống y tế, thiết bị chế biến kim loại...
- (6) Sự cần thiết phải tiếp tục thực hiện một dự án mặc dù thành viên đội có sự thay đổi : Việc các thành viên trong đội rời khỏi đội trong thời gian phát triển dự án là khá phổ biến, do việc thăng chức với các công việc cấp cao hơn, chuyển sang một thành phố khác... Người lãnh đạo đội phải thay thế các thành viên trong đội bởi các nhân viên khác hoặc bởi một nhân viên mới được tuyển dụng. Không kể đến bao nhiêu nỗ lực cần đầu tư vào việc đào tạo một thành viên mới, việc thay đổi thành viên sẽ kéo theo thời gian thực hiện dự án sẽ thay đổi.
- (7) Sự cần thiết phải tiếp tục thực hiện việc bảo trì phần mềm trong một thời gian dài: Các khách hàng mua hoặc phát triển một hệ thống phần mềm mong đợi sẽ tiếp tục sử dụng

nó trong một thời gian dài, thường là từ 5-10 năm. Trong suốt thời kỳ dịch vụ, cuối cùng cũng cần tới sự bảo trì. Trong hầu hết trường hợp, dịch vụ bảo trì cần được cung cấp trực tiếp bởi nhà phát triển. Trong trường hợp các phần mềm được phát triển “trong nhà”, các khách hàng “nội bộ” sẽ cùng chia sẻ vấn đề bảo trì phần mềm trong suốt thời kỳ dịch vụ của hệ thống phần mềm.

1.2 Chất lượng phần mềm

Theo IEEE, chất lượng phần mềm được định nghĩa như sau :

Chất lượng phần mềm là:

Mức độ mà một hệ thống, thành phần hoặc một tiến trình đạt được yêu cầu đã đặc tả

Mức độ mà một hệ thống, thành phần hoặc một tiến trình đạt được những nhu cầu hay mong đợi của khách hàng hoặc người sử dụng.

Ban đầu đảm bảo chất lượng phần mềm có mục tiêu đạt được các yêu cầu đề ra, tuy nhiên thực tế phát triển phần mềm tồn tại rất nhiều ràng buộc đòi hỏi người phát triển cần tối ưu hóa công tác quản lý.

1.3 Đảm bảo chất lượng phần mềm

Theo Daniel Galin, khái niệm đảm bảo chất lượng phần mềm được xác định như sau :

Đảm bảo chất lượng phần mềm là một tập các hoạt động đã được lập kế hoạch và có hệ thống, cần thiết để cung cấp đầy đủ sự tin cậy vào quy trình phát triển phần mềm hay quy trình bảo trì phần mềm của sản phẩm hệ thống phần mềm phù hợp với các yêu cầu chức năng kỹ thuật cũng như với các yêu cầu quản lý mà giữ cho lịch biểu và hoạt động trong phạm vi ngân sách.

1.3.1 Những mục tiêu đảm bảo chất lượng phần mềm

Phát triển phần mềm luôn đi đôi với bảo trì, vì vậy các hoạt động bảo đảm chất lượng phần mềm đều có mối liên quan chặt chẽ đến bảo trì. Những mục tiêu đảm bảo chất lượng phần mềm tương ứng với giai đoạn phát triển và bảo trì được xác định cụ thể như sau :

- Phát triển phần mềm (hướng tiến trình)

1. Đảm bảo một mức độ chấp nhận được rằng phần mềm sẽ thực hiện được các yêu cầu chức năng.
2. Đảm bảo một mức độ cấp nhận được rằng phần mềm sẽ đáp ứng được các yêu cầu về lịch biểu và ngân sách

3. Thiết lập và quản lý các hoạt động để cải thiện và nâng cao hiệu quả của phát triển phần mềm và các hoạt động SQA.

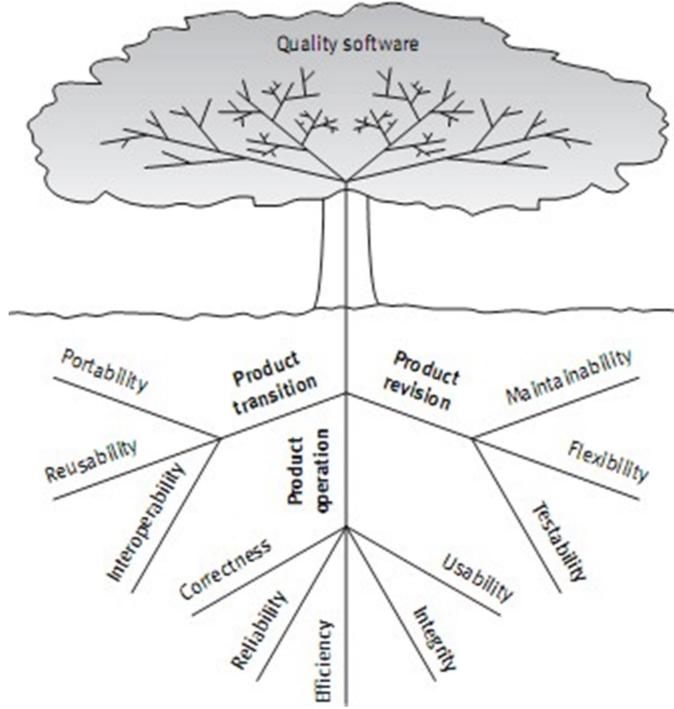
- Bảo trì phần mềm (hướng dẫn phẩm)

1. Đảm bảo một mức độ chấp nhận được rằng các hoạt động bảo trì phần mềm sẽ đáp ứng được các yêu cầu chức năng.
2. Đảm bảo một mức độ cấp nhận được rằng các hoạt động bảo trì phần mềm sẽ đáp ứng được các yêu cầu về lịch biểu và ngân sách
3. Thiết lập và quản lý các hoạt động để cải thiện và nâng cao hiệu quả của bảo trì phần mềm.

1.3.2 Phân loại yêu cầu phần mềm ứng với các yếu tố chất lượng phần mềm

Đã có nhiều tác giả nghiên cứu về các yếu tố chất lượng phần mềm từ các yêu cầu cả nó. Theo thời gian có thể quan niệm về việc đảm bảo chất lượng phần mềm có phần thay đổi, tuy nhiên mô hình các yếu tố đảm bảo chất lượng phần mềm của McCall ra đời vào những năm 70 của thế kỷ trước vẫn còn được nhiều người nhắc đến như là cơ sở tham chiếu các yêu cầu phần mềm. Sau McCall cũng có một số mô hình được quan tâm như mô hình do Evans, Marcinak do hay mô hình của Deutsch và Willis, tuy nhiên những mô hình này chỉ bổ sung hay sửa đổi một vài yếu tố chất lượng. Theo McCall, các yếu tố chất lượng phần mềm được chia làm ba loại :

- Các yếu tố hoạt động của sản phẩm bao gồm tính chính xác, tin cậy, hiệu quả, tính toàn vẹn, sử dụng được
- Các yếu tố rà soát bao gồm tính bảo trì, linh hoạt, có thể test được
- Các yếu tố chuyển giao bao gồm tính khả chuyên, có khả năng sử dụng lại, có khả năng giao tác.



Hình Cây mô hình yếu tố chất lượng phần mềm theo McCall

Chi tiết các thuộc tính được phân tích như sau :

(4) Các yếu tố vận hành sản phẩm : Sự chính xác, độ tin cậy, tính hiệu quả, tính toàn vẹn và khả năng sử dụng được :

- Sự chính xác : Các yêu cầu về độ chính xác được xác định trong một danh sách các đầu ra cần thiết của hệ thống phần mềm, như màn hình hiển thị truy vấn số dư của khách hàng trong một hệ thống thông tin kế toán bán hàng... Các đặc tả đầu ra thường là đa chiều, một số chiều thông dụng là :
 - Nhiệm vụ đầu ra (ví dụ : bản in hóa đơn bán hàng, hay đèn báo động đỏ khi nhiệt độ tăng lên trên 250 độ F)
 - Độ chính xác yêu cầu của các đầu ra này; chúng có thể bị ảnh hưởng bất lợi bởi các tính toán không chính xác hay các dữ liệu không chính xác.
 - Tính đầy đủ của thông tin đầu ra; chúng có thể bị ảnh hưởng bất lợi bởi dữ liệu không đầy đủ.
 - Up-to-dateness của thông tin (xác định bằng thời gian giữa sự kiện và việc xem xét hệ thống phần mềm).
 - Độ sẵn sàng của thông tin (thời gian đáp ứng : được định nghĩa là thời gian cần thiết

để có được các thông tin yêu cầu)

- Các chuẩn cho việc code và viết tài liệu cho hệ thống phần mềm.
- Độ tin cậy : Các yêu cầu về độ tin cậy giải quyết các lỗi để cung cấp dịch vụ. Chúng xác định tỷ lệ lỗi hệ thống phần mềm tối đa cho phép, các lỗi này có thể là lỗi toàn bộ hệ thống hoặc một hay nhiều chức năng riêng biệt của nó.
- Tính hiệu quả : Các yêu cầu về tính hiệu quả giải quyết vấn đề về các tài nguyên phần cứng cần thiết để thực hiện tất cả các chức năng của hệ thống phần mềm với sự phù hợp của tất cả các yêu cầu khác. Các tài nguyên phần cứng chính được xem xét ở đây là khả năng xử lý của máy tính (được đo bằng MIPS – triệu lệnh/giây; MHz – triệu chu kỳ/giây...); khả năng lưu trữ dữ liệu (dung lượng bộ nhớ, dung lượng đĩa – được đo bằng MBs, GBs, TBs...) và khả năng truyền dữ liệu (thường được đo bằng MBPS, GBPS). Các yêu cầu này có thể bao gồm cả các giá trị tối đa tài nguyên phần cứng được sử dụng trong hệ thống phần mềm. Một yêu cầu khác về tính hiệu quả đó là thời gian giữa các lần phải sạc điện đối với các hệ thống nằm trên các máy tính xách tay hay các thiết bị di động.
- Các yêu cầu về tính toàn vẹn giải quyết các vấn đề về bảo mật hệ thống phần mềm, các yêu cầu này để ngăn chặn sự truy cập trái phép, để phân biệt giữa phần lớn nhân viên chỉ được phép xem thông tin với một nhóm hạn chế những người được phép thêm và thay đổi dữ liệu...
- Các yêu cầu về khả năng sử dụng được sẽ đưa ra phạm vi của tài nguyên nhân lực cần thiết để đào tạo một nhân viên mới và để vận hành hệ thống phần mềm.

(5) Các yếu tố về rà soát sản phẩm : bảo trì được, linh động và kiểm tra được :

- Khả năng bảo trì được : Các yêu cầu về khả năng bảo trì được sẽ xác định người dùng và nhân viên bảo trì phải nỗ lực thế nào để xác định được nguyên nhân của các lỗi phần mềm, để sửa lỗi và để xác nhận việc sửa lỗi thành công. Các yêu cầu của yếu tố này nói tới cấu trúc modul của phần mềm, tài liệu chương trình nội bộ và hướng dẫn sử dụng của lập trình viên...
- Tính linh động : Các yêu cầu về tính linh động cũng bao gồm cả các khả năng và nỗ lực cần thiết để hỗ trợ các hoạt động bảo trì. Chúng gồm các nguồn lực (man- day) cần thiết để thích nghi với một gói phần mềm, với các khách hàng trong cùng nghề, với các mức độ hoạt động khác nhau, với các loại sản phẩm khác nhau...Các yêu cầu về yếu tố này cũng hỗ trợ các hoạt động bảo trì trở nên hoàn hảo, như thay đổi và bổ sung vào phần mềm để tăng dịch vụ của nó và để thích nghi với các thay đổi

trong môi trường thương mại và kỹ thuật của công ty.

- **Khả năng test** được : Các yêu cầu về khả năng kiểm tra được nói tới việc kiểm tra sự vận hành có tốt hay không của các hệ thống thông tin. Các yêu cầu về khả năng kiểm tra được liên quan tới các tính năng đặc biệt trong chương trình giúp người tester dễ dàng thực hiện công việc của mình hơn, ví dụ như đưa ra các kết quả trung gian. Các yêu cầu về khả năng kiểm tra được liên quan tới vận hành phần mềm bao gồm các chuẩn đoán tự động được thực hiện bởi hệ thống phần mềm trước khi bắt đầu hệ thống, để tìm hiểu xem có phải tất cả các thành phần của hệ thống phần mềm đều làm việc tốt hay không, và để có một bản báo cáo về các lỗi đã được phát hiện. Một loại khác của yêu cầu này là việc check các dự đoán tự động, được các kỹ thuật viên bảo trì sử dụng để phát hiện nguyên nhân gây lỗi phần mềm.

(6) Các yếu tố về chuyển giao sản phẩm : tính lưu động (khả năng thích nghi với môi trường), khả năng tái sử dụng và khả năng cộng tác được :

- **Tính lưu động** : các yêu cầu về tính lưu động nói tới khả năng thích nghi của hệ thống phần mềm với các môi trường khác, bao gồm phần cứng khác, các hệ điều hành khác... Các yêu cầu này đòi hỏi các phần mềm cơ bản có thể tiếp tục sử dụng độc lập hoặc đồng thời trong các trường hợp đa dạng.
- **Khả năng tái sử dụng** : Các yêu cầu về khả năng tái sử dụng nói tới việc sử dụng các modul phần mềm trong một dự án mới đang được phát triển mà các modul này ban đầu được thiết kế cho một dự án khác. Các yêu cầu này cũng cho phép các dự án tương lai có thể sử dụng một modul đã có hoặc một nhóm các modul hiện đang được phát triển. Tái sử dụng phần mềm sẽ tiết kiệm tài nguyên phát triển, rút ngắn thời gian phát triển và tạo ra các moduls chất lượng cao hơn. Chất lượng modul cao hơn là dựa trên giả định rằng hầu hết các lỗi phần mềm đều được phát hiện bởi các hoạt động đảm bảo chất lượng phần mềm thực hiện trên phần mềm ban đầu, bởi những người sử dụng phần mềm ban đầu và trong suốt những lần tái sử dụng trước của nó. Các vấn đề về tái sử dụng phần mềm đã trở thành một phần trong chuẩn công nghiệp phần mềm (IEEE, 1999).
- **Khả năng cộng tác** : Các yêu cầu về khả năng cộng tác tập trung vào việc tạo ra các giao diện với các hệ thống phần mềm khác. Các yêu cầu về khả năng cộng tác có thể xác định tên của phần mềm với giao diện bắt buộc. Chúng cũng có thể xác định cấu trúc đầu ra được chấp nhận như một tiêu chuẩn trong một ngành công nghiệp cụ thể hoặc một lĩnh vực ứng dụng.

1.4 Kế hoạch đảm bảo chất lượng

- Mô tả chất lượng mong muốn, thiết lập các tiêu chuẩn chất lượng và cách đánh giá (đo) các thuộc tính chất lượng.
- Định rõ qui trình đánh giá chất lượng.
- Định rõ các chuẩn mực về quản lí (dùng chuẩn có sẵn/thiết lập mới).

1.5 Đội ngũ đảm bảo chất lượng phần mềm

Bất cứ tổ chức phát triển phần mềm nào cũng có thể được tạo ra bởi 3 mức cấu trúc quản lý: người quản lý cấp cao (top management), người quản lý mức trung bình (middle management) và người quản lý dự án (project management).

Người quản lý cấp cao là những người quản lý chung trong tổ chức. Người quản lý mức trung bình có thể có nhiều loại, tùy thuộc vào quy mô của từng tổ chức, bao gồm một số trách nhiệm và một số mức như: các giám đốc bộ phận và các trưởng phòng... Người quản lý dự án cũng có nhiều loại, phụ thuộc vào dự án và quy mô của dự án, bao gồm các giám đốc dự án, trưởng nhóm...

Với mục đích thảo luận, chúng ta chỉ xem xét 3 mức cơ bản trong tổ chức phát triển phần mềm :

- **Người quản lý cấp cao** (top management): quản lý và thực thi chung, ví dụ như các CEO.
- **Giám đốc bộ phận** (department management): những người quản lý việc phát triển, bảo trì và kiểm thử phần mềm
- **Người quản lý dự án** và trưởng nhóm phát triển dự án và các dịch vụ bảo trì

Chương 2. Các thành phần chất lượng phần mềm tiền dự án

2.1. Rà soát hợp đồng

Một hợp đồng tồi chắc chắn là khó có thể chấp nhận được. Từ quan điểm của SQA, một hợp đồng tồi – thường mô tả các yêu cầu không chặt chẽ và đưa ra kế hoạch cũng như ngân sách phi thực tế - thì sẽ dẫn đến một phần mềm có chất lượng tồi. Vì thế, một chương trình SQA cần được thực hiện để đảm bảo chất lượng phần mềm bằng cách rà soát lại những đề xuất ban đầu và sau đó là bản dự thảo hợp đồng (hoạt động “rà soát hợp đồng” bao gồm cả 2 hoạt động trên). Cả hai hoạt động rà soát trên là nhằm mục đích cải thiện ngân sách và thời gian biểu, là những cơ sở cho những đề nghị và hợp đồng sau này, đồng thời có thể biết được những rủi ro tiềm năng sớm (trong mục tiêu ban đầu và trong bản dự thảo hợp đồng).

2.1.1. Tiến trình rà soát hợp đồng và các bước thực hiện

Có khá nhiều tình huống có thể giúp một công ty phần mềm (“nhà cung cấp”) ký hợp đồng với một khách hàng. Phổ biến nhất là:

- Tham gia trong một cuộc đấu thầu
- Đưa ra bản phác thảo dựa trên yêu cầu đề xuất (RFP-Request For Proposal) của khách hàng
- Nhận một đặt hàng từ một khách hàng của công ty
- Nhận một yêu cầu từ bên trong hoặc từ phòng ban khác trong một tổ chức

Rà soát hợp đồng là một thành phần của SQA được nghĩ ra để hướng dẫn xem xét lại những bản dự thảo của những tài liệu đề xuất và hợp đồng. Nếu có thể, rà soát lại hợp đồng còn cung cấp sự giám sát những hợp đồng được thực hiện với những đối tác dự án tiềm năng và các nhà thầu phụ. Tiến trình rà soát có thể được chia thành hai giai đoạn:

- Giai đoạn 1: rà soát lại bản dự thảo đề xuất trước khi giao cho khách hàng tiềm năng (“rà soát bản dự thảo đề xuất”). Giai đoạn này rà soát lại bản dự thảo cuối cùng và những cơ sở đề xuất: những tài liệu yêu cầu của khách hàng, chi tiết yêu cầu thêm của khách hàng và dự diễn giải các yêu cầu, các ước lượng chi phí và tài nguyên, những hợp đồng hiện tại hoặc là những bản dự thảo hợp đồng của nhà cung cấp với các đối

tác và nhà thầu phụ.

- Giai đoạn 2: rà soát lại bản dự thảo hợp đồng trước khi kí (“rà soát bản dự thảo hợp đồng”). Giai đoạn này rà soát lại bản dự thảo hợp đồng dựa trên đề xuất và sự hiểu biết (bao gồm cả những thay đổi) đã đạt được trong quá trình thương thảo hợp đồng.

Quá trình rà soát có thể bắt đầu khi những tài liệu dự thảo liên quan đã hoàn thành. Những cá nhân thực hiện rà soát phải xem xét kỹ lưỡng bản dự thảo trong khi đề cập đến một phạm vi toàn diện các đối tượng đang rà soát. Một danh sách kiểm tra là rất hữu ích cho việc đảm bảo xem xét hết các vấn đề liên quan.

Sau khi hoàn thành giai đoạn rà soát, việc cần thiết những sự thay đổi, cái thêm vào và sự hiệu chỉnh phải được thông báo bởi đội đề xuất (sau khi rà soát bản dự thảo đề xuất) và bởi ban phụ trách về luật pháp (sau khi rà soát lại bản dự thảo hợp đồng)

2.1.2. Các mục tiêu rà soát hợp đồng

- Những mục đích của công việc rà soát bản dự thảo đề xuất

Mục đích của việc rà soát bản dự thảo đề xuất là để đảm bảo rằng những hoạt động sau được thực hiện một cách thỏa đáng.

- 1) Những yêu cầu của khách hàng đã được giải thích chi tiết và có chú giải

Những tài liệu yêu cầu đề xuất (RFP) và những tài liệu công nghệ tương tự có thể quá chung chung và mơ hồ cho những mục tiêu của dự án. Kết quả là có nhiều chi tiết cần được thêm vào từ khách hàng. Việc giải thích chi tiết những yêu cầu mập mờ và những cập nhật của chúng nên được ghi lại trong một tài liệu riêng biệt đã được sự chấp nhận của cả khách hàng và công ty phần mềm.

- 2) Lựa chọn những phương pháp thực hiện dự án đã được kiểm tra.

Thông thường, những lựa chọn có triển vọng và phù hợp mà trên đó thể hiện một đề xuất thì đã được xem xét đầy đủ (nếu tất cả) bởi đội đề xuất. Điều kiện này đặc biệt muốn đề cập đến việc hoàn thành thay thế bao gồm tái sử dụng phần mềm, và những quan hệ đối tác hoặc là thầu lại với những công ty mà có hiểu biết chuyên môn hoặc nhân viên có chuyên môn có thể đảm bảo những điều khoản của đề xuất

- 3) Những khía cạnh hình thức của mối quan hệ giữa khách hàng và công ty

phần mềm phải được ghi rõ.

Đề xuất nên định nghĩa những thủ tục bao gồm:

- Sự giao tiếp với khách hàng và những kênh giao diện
- Việc chuyển giao dự án và tiêu chuẩn được chấp nhận
- Tiến trình phê chuẩn pha hình thức
- Phương thức tiếp theo khách hàng thiết kế và kiểm tra
- Thủ tục khách hàng thay đổi yêu cầu

4) Xác định những rủi ro khi phát triển

Những rủi ro khi phát triển, như không đủ kiến thức chuyên môn liên quan đến lĩnh vực nghiệp vụ của dự án hoặc cách sử dụng những công cụ phát triển yêu cầu, cần được xác định và giải quyết.

5) Ước lượng đầy đủ những tài nguyên và thời gian biểu của dự án

Việc ước lượng tài nguyên để cập đến đội ngũ nhân viên chuyên nghiệp và ngân sách của dự án, bao gồm cả chi phí cho các nhà thầu con. Việc ước lượng thời gian nên đưa vào những yêu cầu về thời gian của tất cả những bên tham gia vào dự án.

Lưu ý

Trong một vài tình huống, một nhà cung cấp có ý đề nghị cung cấp một chi phí thấp, xem xét các yếu tố như tiềm năng bán hàng. Trong trường hợp này, khi mà đề xuất dựa trên ước lượng thực tế của thời gian, ngân sách và khả năng chuyên môn, những tổn thất phát sinh được coi là một mắt mát có thể tính được, không phải là một hợp đồng thất bại

6) Kiểm tra năng lực của công ty đối với dự án.

Việc kiểm tra này nên xem xét đến năng lực chuyên môn cũng như là khả năng sẵn sàng của những thành viên trong đội được yêu cầu và những khả năng phát triển trong thời gian đã được lập lịch.

7) Kiểm tra năng lực của khách hàng để đáp ứng những yêu cầu của mình

Việc kiểm tra này đề cập đến khả năng tài chính và tổ chức của khách hàng,

như tuyển dụng và đào tạo nhân sự, cài đặt phần cứng yêu cầu và nâng cấp các thiết bị liên lạc.

8) Định nghĩa đối tác và nhà thầu phụ tham gia

Điều này bao gồm các vấn đề bảo đảm chất lượng, lịch trình thanh toán, phân phối thu nhập, lợi nhuận của dự án, và hợp tác giữa quản lý dự án và các đội.

9) Định nghĩa và bảo vệ quyền sở hữu.

Yếu tố này có tầm quan trọng trong trường hợp tái sử dụng phần mềm, khi việc có thêm một gói mới vào hoặc có tái sử dụng phần mềm hiện nay trong tương lai hay không cần phải được quyết định. Nó cũng đề cập đến việc sử dụng các file độc quyền của các dữ liệu quan trọng cho hoạt động của hệ thống và các biện pháp an ninh.

Những mục tiêu của rà soát lại bản dự thảo đề xuất được tổng kết trong bảng sau:

Những mục tiêu của việc rà soát bản dự thảo đề xuất
9 mục tiêu của việc rà soát bản dự thảo đề xuất đảm bảo rằng những hành động sau đây được thực hiện một cách thỏa đáng: <ol style="list-style-type: none">1. Những yêu cầu của khách hàng đã được giải thích chi tiết và có chú giải2. Lựa chọn những phương pháp thực hiện dự án đã được kiểm tra3. Những khía cạnh hình thức của mối quan hệ giữa khách hàng và công ty phần mềm phải được ghi rõ4. Xác định những rủi ro khi phát triển5. Ước lượng đầy đủ những tài nguyên và thời gian biểu của dự án6. Kiểm tra năng lực của công ty đối với dự án7. Kiểm tra năng lực của khách hàng để đáp ứng những yêu cầu của mình8. Định nghĩa đối tác và nhà thầu phụ tham gia9. Định nghĩa và bảo vệ quyền sở hữu

- Những mục tiêu của rà soát dự thảo hợp đồng.

Những mục tiêu của việc rà soát bản dự thảo hợp đồng để đảm bảo rằng những hoạt động sau đây được thực hiện một cách thỏa đáng:

- 1) Không có vấn đề chưa rõ ràng nào vẫn còn lại trong dự thảo hợp đồng
- 2) Tất cả những thỏa thuận đạt được giữa các khách hàng và công ty phải được giải thích đầy đủ và chính xác trong hợp đồng và phụ lục của nó. Những hiểu biết này được dùng để giải quyết tất cả các vấn đề chưa rõ ràng và khác biệt giữa khách hàng và công ty mà đã được đưa ra cho đến nay
- 3) Không có sự thay đổi, bổ sung, hoặc thiếu sót nào không được thảo luận và sự thoả thuận nên được đưa vào dự thảo hợp đồng. Việc thay đổi, dù có ý hay không, có thể dẫn đến sự bổ sung đáng kể và những nhiệm vụ bất ngờ trong một bộ phận của nhà cung cấp.

Những mục tiêu của việc rà soát lại dự thảo hợp đồng có thể được tổng kết trong bảng 5.2

Những mục tiêu của việc rà soát dự thảo hợp đồng

Ba mục tiêu của việc rà soát dự thảo hợp đồng nhằm đảm bảo những hoạt động sau đây được thực hiện một cách thỏa đáng:

- 1) Không có vấn đề chưa rõ ràng nào vẫn còn lại trong dự thảo hợp đồng
- 2) Mọi thỏa thuận đã đạt được sau khi xem xét những đề xuất phải được chú giải một cách chính xác
- 3) Không có sự thay đổi, bổ sung, hoặc thiếu sót đưa vào bản dự thảo hợp đồng

2.1.3. Thực thi rà soát hợp đồng

Duyệt hợp đồng khác nhau về độ lớn của chúng, tùy thuộc vào các đặc tính của dự án đề xuất. Phức tạp này có thể là kỹ thuật hoặc tổ chức. Theo đó, mức độ khác nhau của nguồn lực chuyên môn được điều chỉnh phù hợp cho những duyệt hợp đồng khác nhau. Những nguồn lực chuyên môn đặc biệt cần thiết cho những đề xuất chính.

- Những yếu tố ảnh hưởng tới phạm vi của một bản duyệt hợp đồng

Các yếu tố quan trọng nhất của dự án xác định mức độ của hợp đồng nỗ lực xem xét lại yêu cầu là:

- Độ lớn của dự án, thường được đo bằng các nguồn lực man-month.
- Kỹ thuật phức tạp của dự án.
- Trình độ và sự hiểu biết của nhân viên có kinh nghiệm trong lĩnh vực của dự án. Sự hiểu biết với các lĩnh vực dự án là thường xuyên được liên kết với khả năng tái sử dụng phần mềm; trong trường hợp có thể tái sử dụng phần mềm là cao, mức độ duyệt được giảm xuống
- Tổ chức dự án phức tạp. Càng với số lượng lớn của các tổ chức (nghĩa là, các đối tác, nhà thầu phụ, và khách hàng) tham gia các dự án, thì càng yêu cầu công sức rà soát hợp đồng lớn hơn.

Do đó chúng tôi có thể cho rằng "đơn giản" việc duyệt hợp đồng sẽ được thực hiện bởi một người xem, họ sẽ tập trung vào một vài chủ đề và đầu tư ít thời gian để xem lại. Tuy nhiên, hợp đồng quy mô lớn có thể yêu cầu sự tham gia của một đội để nghiên cứu một loạt các vấn đề, một quá trình đòi hỏi sự đầu tư của nhiều giờ làm việc.

- Tác nhân thực thi duyệt hợp đồng

Nhiệm vụ duyệt lại hợp đồng có thể được hoàn thành bởi các cá nhân khác nhau, được liệt kê ở đây theo thứ tự tăng dần thông qua sự phức tạp của dự án:

- Các nhà lãnh đạo hoặc thành viên khác của nhóm đề xuất.
- Các thành viên của đội đề xuất.
- Một đội ngũ nhân viên chuyên gia bên ngoài hoặc nhân viên của công ty những người không phải là một thành viên của nhóm đề xuất.
- Một nhóm các chuyên gia bên ngoài. Thông thường, một nhóm rà soát lại hợp đồng bao gồm các chuyên gia bên ngoài được mời đến để đưa ra các đề xuất chuyên môn đặc biệt. Các chuyên gia có thể được mời đến để rà soát lại hợp đồng trong các tổ chức phát triển phần mềm nhỏ do không thể tìm thấy đủ các nhân viên thích hợp trong đội ngũ nhân viên của họ.

- Thực hiện rà soát lại hợp đồng cho đề xuất chính

Đề xuất chính được đề nghị cho các dự án đặc trưng bởi ít nhất một số các yếu tố sau: quy mô dự án rất lớn, kỹ thuật rất cao và phức tạp, lĩnh vực chuyên môn mới, tổ chức phức tạp cao (nhận ra bởi một số lượng lớn các tổ chức, nghĩa là, đối tác, nhà thầu phụ, và khách hàng, tham gia một phần trong dự án này). Thực hiện quá trình rà soát lại hợp đồng cho một dự án lớn thường liên quan đến đáng kể đến khó khăn của một tổ chức. Một số con đường để vượt qua những khó khăn được đề nghị ở đây, sau sự xem lại của các yếu tố giới thiệu nhiều khó khăn cho một hoành thành mịn của nhiệm vụ.

2.1.4. Những khó khăn của thực hiện xem lại hợp đồng cho các đề xuất chính

Hầu hết mọi người đều đồng ý rằng xem xét lại hợp đồng là một thủ tục chính cho giảm nguy cơ thất bại của dự án lớn. Về căn bản, việc rà soát hợp đồng là khó khăn, đặc biệt là đối với những tình huống rà soát các yêu cầu đề xuất chính.

- Áp lực thời gian. Cả hai giai đoạn của việc xem xét hợp đồng, đề nghị xem xét lại dự thảo và xem xét dự thảo hợp đồng thường được thực hiện khi nhóm đề xuất là dưới áp lực đáng kể về thời gian. Kết quả là, mỗi giai đoạn của việc xem lại hợp đồng phải được hoàn tất trong vòng một vài ngày để giúp cho người tiếp theo điều chỉnh các văn bản được diễn ra.

- Quy tắc duyệt lại hợp đồng yêu cầu phải làm việc chuyên nghiệp. Chuyên nghiệp hiệu suất của mỗi giai đoạn của việc xem lại hợp đồng yêu cầu sự đầu tư chuyên nghiệp đáng kể của các chuyên gia (số lượng thời gian yêu cầu khác nhau, tất nhiên, tùy theo bản chất của dự án).
- Các thành viên tiềm năng trong nhóm rà soát Hợp đồng đều rất bận rộn. Những thành viên tiềm năng của đội duyệt hợp đồng thường là nhân viên cấp cao và các chuyên gia và họ thường cam kết thực hiện thường xuyên nhiệm vụ của họ tại tất cả thời gian được xem là cần thiết. Các chuyên gia nhàn rỗi có thể do đó là một vấn đề hậu cần quan trọng.

2.1.5. Khuyến cáo cho việc thực hiện duyệt lại những hợp đồng chính

Một kế hoạch xem lại hợp đồng một cách cẩn thận quyết định thành công của nhóm rà soát hợp đồng. Các bước sau đây được thực hiện để tạo điều kiện cho quy trình rà soát.

- Các hợp đồng nên được rà soát theo lịch trình. Xem xét lại các hoạt động rà soát hợp đồng nên được đưa vào lịch trình chuẩn bị đề xuất, để lại đầy đủ thời gian cho việc xem lại và chỉnh tiếp theo sẽ được thực hiện.
- Một nhóm thực hiện rà soát lại các hợp đồng. Làm cho nó có thể làm việc theo nhóm để phân phối các khối lượng công việc giữa các thành viên trong nhóm sao cho mỗi thành viên của đội xem lại hợp đồng có thể có đủ thời gian để làm (có thể bao gồm việc chuẩn bị một bản báo cáo bằng văn bản mà tóm tắt của mình những phát hiện và kiến nghị).
- Lãnh đạo của đội rà soát hợp đồng nên được bổ nhiệm. Điều quan trọng là trách nhiệm cho các tổ chức, quản lý và kiểm soát các hoạt động rà soát hợp đồng được xác định, thích hợp hơn bằng cách bổ nhiệm một lãnh đạo của đội. Cái hoạt động của các lãnh đạo đội bao gồm:
 - Tuyển dụng của các thành viên trong đội
 - Phân phối các nhiệm vụ rà soát giữa các thành viên của nhóm nghiên cứu
 - Phối hợp giữa các thành viên của đội duyệt
 - Phối hợp giữa các đội tuyển xem xét và đề xuất

- Theo dõi các hoạt động, đặc biệt là việc tuân thủ theo lịch biểu
- Tổng kết những phát hiện và phân phối chúng từ nhóm đề nghị

Lưu ý:

Theo xem lại hợp đồng có thể áp đặt một khối lượng công việc đáng kể và bối sung áp lực vào các nhóm đề xuất, nghĩ nên được trao cho khi nó có thể được thích hợp để tránh không tiến hành rà soát hợp đồng. Tình huống có thể xảy ra với các dự án quy mô nhỏ, hoặc dự án quy mô chi phí vừa và nhỏ. Các thủ tục rà soát hợp đồng nên được xác định thành những loại dự án mà rà soát lại hợp đồng là không bắt buộc. Đối với các loại khác được định nghĩa dự án "đơn giản", nó khuyến cáo rằng quyền được trao cho một người quản lý cao cấp để đưa ra quyết định để xem ở đâu để thực hiện xem lại.

2.1.6. Các đối tượng rà soát hợp đồng

Duyệt hợp đồng được thực hiện bởi nhiều đối tượng, dựa trên việc đánh giá mục tiêu của hợp đồng. Bảng kiểm mục (checklist) là công cụ hữu ích cho việc giúp đội xem xét để tổ chức công việc của mình và đạt được nhiều thông tin của những vấn đề liên quan. Rõ ràng là rất nhiều vấn đề trên các danh sách này là không thích hợp đối với bất kỳ dự án cụ thể. Tại cùng một thời điểm, ngay cả một danh sách kiểm tra toàn diện có thể loại trừ một số vấn đề quan trọng có liên quan đến một đề xuất dự án đã định. Đây là nhiệm vụ của đội rà soát hợp đồng, nhưng đặc biệt là của các nhà lãnh đạo, để xác định danh sách các vấn đề thích hợp cho các đề xuất dự án cụ thể.

2.1.7. Rà soát hợp đồng cho các dự án nội bộ

Một số lượng đáng kể, nếu không phải là đa số, các dự án phần mềm là dự án các dự án nội bộ - dự án "trong nhà" - được thực hiện bởi một bộ phận của một tổ chức cho một bộ phận của tổ chức đó. Trong trường hợp như vậy, việc đơn vị phát triển phần mềm là nhà cung cấp, trong khi các đơn vị khác có thể được coi là khách hàng. Thường xuyên, dự án phát triển phần mềm nội bộ không dựa trên những gì sẽ được xem là một mối quan hệ đầy đủ khách hàng -nhà cung cấp . Trong nhiều trường hợp, các dự án này được dựa trên các thỏa thuận chung, với thiện chí đóng vai trò quan trọng trong mối quan hệ giữa hai đơn vị. Nó sau rằng các đơn vị đang phát triển sẽ thực hiện chỉ một đoạn ngắn và "nhẹ" duyệt hợp ,

hoặc không ai cả.

Bảng : Chuẩn dự án nội bộ và khách hàng trong cùng tổ chức

Kiểu của dự án	Khách hàng	Ví dụ
Quản trị hoặc các phần mềm thao tác dùng trong nội bộ	Quản trị hoặc thao tác	Hệ thống bán hàng và hàng tồn kho Hệ thống quản lý tài chính
Những gói phần mềm được dự định để bán theo kiểu “phần mềm đóng gói”	Thị trường phần mềm	Trò chơi máy tính Phần mềm giáo dục Xử lý ngôn ngữ
Những vi chương trình được nhúng trong sản phẩm của công ty	Bộ phận phát triển điện tử và cơ khí	Công cụ điện tử và các phần mềm điều khiển

Thật không may, mỗi quan hệ lỏng thường được đặc trưng bởi không đủ thí nghiệm các yêu cầu của dự án, lịch biểu, tài nguyên và phát triển các rủi ro. Kết quả là, những vấn đề sau đây có khả năng xảy ra:

- (1) Quá trình định nghĩa không tương xứng với yêu cầu dự án.
- (2) Nghèo nàn trong việc đánh giá yêu cầu.
- (3) Thời khóa biểu / lập lịch trình nghèo nàn.
- (4) Không tương xứng nhận thức về những nguy cơ về rủi ro.

Theo danh sách này cho thấy, chúng ta có thể dễ dàng kết luận rằng “in hourse”, thực hiện dự án cho các khách hàng nội bộ được nhiều dễ bị thất bại hơn là những hợp đồng dự án bên ngoài .

Có thể kết luận rằng mối quan hệ khách hàng-nhà cung cấp và việc xem lại hợp đồng đó được chứng minh là có hiệu quả cho các dự án bên ngoài nên được áp dụng cho các dự án nội bộ. Các cơ hội tránh néu trên những vấn đề tiềm năng có thể được cải thiện

đáng kể bằng cách thực hiện thủ tục đó sẽ xác định:

- Một đề xuất phù hợp cho dự án nội bộ
- Áp dụng một tiến trình xem lại hợp đồng đúng cách cho các dự án nội bộ
- Thỏa thuận thích đáng giữa các khách hàng nội bộ và các nội bộ nhà cung cấp.

- Nhược điểm của "các mối quan hệ lỏng" nội bộ dự án

Vấn đề	Nhược điểm của khách hàng nội bộ	Nhược điểm của nhà phát triển nội bộ
Quá trình định nghĩa không tương xứng với yêu cầu dự án	Cài đặt lệch hướng so với yêu cầu của chương trình Hài lòng thấp	Tốn nhân lực để đưa ra những thay đổi có thể tránh
Quá trình định nghĩa không tương xứng với yêu cầu dự án	Mong đợi phi hiện thực một hệ thống mềm dẻo	Độ lệch lớn từ ngân sách phát triển Xung đột được gây ra bởi các yêu cầu cho việc thêm ngân sách
Thời khóa biểu / lập lịch trình nghèo nàn.	Thiếu lịch trình ngày phân phối sản phẩm mới	Hoạt động phát triển dưới thời gian áp lực và khó chịu từ sản phẩm kém chất lượng
Không tương xứng nhận thức về những nguy cơ về rủi ro.	Chưa chuẩn bị cho những rủi ro của dự án và những hậu quả của chúng.	Sự khởi đầu chậm chạp của nhân lực để khắc phục khó khăn.

2.2. Các kế hoạch phát triển và kế hoạch chất lượng

Hãy tưởng tượng rằng bạn vừa được bổ nhiệm làm người đứng đầu một dự án khá lớn. Như là thường xảy ra trong ngành công nghiệp phần mềm, bạn chịu nhiều áp lực kinh khủng về thời gian từ ngày đầu tiên. Bởi vì bạn đã là thành viên của đội đề xuất và tham gia vào hầu hết các cuộc họp với đại diện của khách hàng, bạn có tự tin rằng bạn đã biết tất cả những gì là cần thiết cho công việc. Bạn định đề nghị sử dụng kế hoạch và tài liệu nội bộ mà đội đã chuẩn bị như bản kế hoạch về phát triển và chất lượng. Bạn đã chuẩn bị để dựa vào các tài liệu này bởi vì bạn biết rằng các đề nghị và ước lượng của nó, bao gồm cả thời gian biểu, yêu cầu nghiệp vụ, danh sách các tài liệu dự án, đánh giá thiết kế dự kiến, và danh sách các rủi ro khi phát triển thông qua xem xét kỹ lưỡng bởi nhóm xem xét lại hợp đồng.

Vì vậy bạn có một chút thất vọng rằng tại thời điểm rất quan trọng này của dự án, phòng quản lý phát triển yêu cầu bạn ngay lập tức chuẩn bị các kế hoạch phát triển dự án mới và riêng biệt ("Kế hoạch phát triển") và các kế hoạch chất lượng ("Kế hoạch chất lượng"). Khi bạn cho rằng đề nghị hoàn thành và phụ lục của nó có thể phục vụ như là các kế hoạch yêu cầu, người quản lý nhấn mạnh rằng họ có thể cập nhật, với các chủ đề mới và toàn diện hơn để đảm bảo đầy đủ nhất của kế hoạch. "Bằng cách này," người quản lý đề cập gần như là sang một bên", đừng quên rằng khoảng thời gian bảy tháng từ giữa các việc chuẩn bị đề nghị và ký kết hợp đồng. Đây như là một khoảng thời gian chết trong dự án của chúng ta "

Bạn nên mong muốn rằng người quản lý của bạn là đúng. Công sức và vốn đầu tư trong việc chuẩn bị các kế hoạch phát triển và chất lượng chắc chắn sẽ mang lại lợi ích. Bạn có thể nhận ra rằng một số thành viên trong đội sẽ không sẵn sàng tại các ngày theo thời gian biểu do chậm trễ trong việc hoàn thành công việc hiện tại của họ, hay rằng các công ty tư vấn đã đồng ý cung cấp hỗ trợ chuyên môn trong một lĩnh vực chuyên môn hoá cao và quan trọng đã bị thiệt hại nặng và bị phá sản trong thời gian ngắn. Đây chỉ là hai trong các loại của các vấn đề có thể nảy sinh.

Tổng kết lại, dự án cần các bản kế hoạch về phát triển và chất lượng như sau:

- Phải dựa trên các đề nghị sơ khai mà đã được kiểm tra lại một cách kỹ lượng và cập nhật liên tục.
- Phải toàn diện hơn so với đề nghị phê duyệt, đặc biệt là với quan điểm về thời gian biểu, đánh giá tài nguyên, đánh giá rủi ro và phát triển.

- Bao gồm bổ sung các đối tượng vắng mặt từ đề nghị phê duyệt.
- Chúng ta đã chuẩn bị ở giai đoạn đầu của dự án để có thể cảnh báo về khó khăn trong việc lập kế hoạch, tiềm năng thiếu nhân viên, các nhân tố phát triển, các vấn đề với các cuộc họp tại các mốc quan trọng trong hợp đồng, những rủi ro phát triển được sửa đổi,...

Kế hoạch phát triển và chất lượng là những yếu tố chính cần thiết cho việc tuân thủ các tiêu chuẩn dự án với 9000.3. Đây cũng là một yếu tố quan trọng trong Capability Maturity Model (CMM) để đánh giá sự trưởng thành của tổ chức phát triển phần mềm.

2.2.1. Những mục tiêu của kế hoạch phát triển và kế hoạch chất lượng

Kế hoạch, như là một quá trình, có nhiều mục tiêu, mỗi mục tiêu trong số đó có nghĩa là để chuẩn bị đầy đủ trên cơ sở sau đây:

- 1) Các hoạt động lập thời gian biểu phát triển sẽ dẫn đến thành công và kịp thời gian hoàn thành dự án, và ước lượng được nguồn nhân lực, yêu cầu và ngân sách.
- (2) Các thành viên đội tuyển dụng và việc phân bổ nguồn lực phát triển (theo lịch trình hoạt động và yêu cầu ước lượng nguồn nhân lực).
- (3) Giải quyết các rủi ro phát triển.
- (4) Cài đặt các hoạt động được yêu cầu của SQA.
- (5) Cung cấp việc quản lý với dữ liệu cần thiết để kiểm soát dự án.

2.2.2. Các thành phần của kế hoạch phát triển

Căn cứ vào tài liệu đề nghị, kế hoạch phát triển của dự án là chuẩn bị sẵn sàng để hoàn thành các mục tiêu trên. Các yếu tố sau đây áp dụng cho các thành phần của dự án khác nhau bao gồm kế hoạch phát triển dự án.

(1) Sản phẩm dự án:

Kế hoạch phát triển bao gồm các sản phẩm sau:

- Tài liệu thiết kế chi tiết xác định ngày hoàn thành, chỉ ra những gì được chuyển giao cho khách hàng ("phân phối").
- Sản phẩm phần mềm (xác định ngày hoàn thành và cách cài đặt).
- Đào tạo nghiệp vụ (chỉ rõ ngày tháng, người tham gia và các site).

(2) Giao diện dự án:

Dự án bao gồm các giao diện:

- Giao tiếp với các gói phần mềm hiện có (giao diện phần mềm).
- Giao tiếp với các phần mềm khác và / hoặc nhóm phát triển phần cứng đang làm việc trên cùng một hệ thống, dự án (tức là, hợp tác và phối hợp liên kết).
- Giao tiếp với phần cứng hiện tại (giao diện phần cứng).

(3) Phương pháp luận và công cụ phát triển dự án được áp dụng ở mỗi giai đoạn của dự án.

Chỉ dẫn thực hiện

Khi đánh giá sự phù hợp của dự án với các phương pháp và công cụ phát triển được đề xuất, chúng ta cũng nên đưa vào kinh nghiệm chuyên môn của đội ngũ nhân viên, bao gồm cả nhân viên của nhà thầu phụ, thậm chí nếu tạm thời.

(4) Các thủ tục và các chuẩn phát triển phần mềm:

Nên có một danh sách các chuẩn bà các thủ tục được áp dụng trong dự án.

(5) Sự ánh xạ các quá trình phát triển:

Việc ánh xạ các quá trình phát triển bao gồm việc cung cấp các định nghĩa chi tiết cho từng giai đoạn của dự án. Những mô tả bao gồm các định nghĩa các yếu tố đầu vào và đầu ra, và các hoạt động cụ thể được lên kế hoạch.

Hoạt động mô tả bao gồm:

- (a) Ước tính về thời gian của hoạt động. Những ước tính phụ thuộc nhiều vào kinh nghiệm thu được trong các dự án trước đó.
- (b) Chuỗi các hoạt động hợp lý, trong đó mỗi hoạt động được thực hiện, bao gồm mô tả sự phụ thuộc của hoạt động vào các hoạt động trước đó đã hoàn thành.
- (c) Các kiểu nguồn lực chuyên môn cần thiết và ước lượng cần bao nhiêu nguồn lực cần thiết cho từng hoạt động.

Chỉ dẫn thực hiện:

Các hoạt động SQA, như xem xét lại thiết kế và test phần mềm, nên được bao gồm trong các hoạt động dự án theo lịch trình. Việc này cũng được áp dụng cho các hoạt động thiết kế và chỉnh sửa mã. Không có kế hoạch các hoạt động này có thể

gây ra sự chậm trễ unanticipated trong việc khởi xướng các hoạt động tiếp theo.

Một số phương pháp có sẵn cho việc lập lịch biểu và trình bày bằng hình ảnh (đồ họa) quá trình phát triển. Một trong những phương pháp phổ biến nhất là sử dụng biểu đồ Gantt, hiển thị các hoạt động khác nhau bởi các thanh ngang có độ dài tỉ lệ thuận với thời gian của hoạt động. Các thanh đại diện cho các hoạt động của mình, và được đặt theo chiều dọc, theo kế hoạch và bắt đầu kết luận của chúng. Một vài công cụ trên máy tính có thể chuẩn bị các biểu đồ Gantt, thêm vào để cung cấp một danh sách các hoạt động bởi thời gian cần thiết để bắt đầu và kết thúc của chúng, v.v.

Thêm một phương pháp lập kế hoạch nâng cao, như CPM và Pert, cả hai đều thuộc loại phân tích con đường quan trọng, có trình tự phụ thuộc vào thời gian hoạt động thêm vào. Chúng cho phép tính toán thời gian sớm nhất và muộn nhất được chấp nhận cho mỗi hoạt động. Sự khác nhau giữa các thời gian bắt đầu phụ thuộc vào tính linh hoạt của các hoạt động được lập lịch. Đặc biệt chú ý đến các hoạt động thiếu thời gian biểu linh hoạt (được gọi là các hoạt động “critical path”), và hoàn thành vội vàng có thể gây chậm trễ trong việc ký kết của toàn bộ dự án.

Một số gói phần mềm, sử dụng kết hợp các phương pháp, hỗ trợ việc lập kế hoạch, báo cáo và theo dõi thời gian biểu của dự án. Một ví dụ về một gói phần mềm của loại hình này là dự Microsoft Project™. Để thảo luận chi tiết hơn về việc lập lịch, hãy tham khảo các sách viết về quản lý dự án.

(6) Các mốc dự án:

Đối với mỗi mốc quan trọng, thời gian hoàn thành và các sản phẩm của dự án (các tài liệu và mã) được xác định.

(7) Tổ chức nhân viên trong dự án:

Kế hoạch tổ chức bao gồm:

- Cơ cấu tổ chức: định nghĩa các đội dự án và nhiệm vụ của họ, bao gồm các đội được bao gồm cả các công nhân của nhà thầu phụ.
- Các yêu cầu chuyên môn: chứng chỉ chuyên môn, kinh nghiệm trong một ngôn ngữ lập trình cụ thể hoặc công cụ phát triển, kinh nghiệm với một sản phẩm phần mềm cụ thể và các loại,.. v.v.
- Số thành viên của nhóm cần thiết cho từng thời kỳ, theo các hoạt động đã được

lập lịch Dự kiến các đội sẽ bắt đầu hoạt động của họ vào các thời điểm khác nhau, và kích thước đội của họ có thể khác nhau tại mỗi khoảng thời gian, tùy thuộc vào các hoạt động đã được lập kế hoạch.

■ Tên của các nhà lãnh đạo đội và các thành viên nhóm. Những khó khăn được dự kiến sẽ phát sinh đối với việc giao lâu dài của nhân viên để các đội vì sự thay đổi trong tập unanticipated hiện tại của họ.

Vì vậy, tên của các nhân viên được yêu cầu giúp theo dõi sự tham gia của họ như thành viên của nhóm.

Chỉ dẫn thực hiện:

Tính sẵn có lâu dài của nhân viên dự án nên được kiểm tra cẩn thận. Chậm lại trong hoàn thành nhiệm vụ trước đây có thể dẫn đến sự chậm trễ trong việc gia nhập nhóm dự án, trong đó tăng rủi ro không họp lại được tại các mốc quan trọng của dự án. Ngoài ra, nhân viên bốc hơi "" gây ra bởi chức và / hoặc các chương trình khuyến mại, hiện tượng đó là đặc biệt thường xuyên trong ngành công nghiệp phần mềm, có thể gây ra tình trạng thiếu nhân viên. Vì vậy, ước lượng khả năng sẵn có của nhân viên nên được kiểm tra định kỳ để tránh những "ngạc nhiên". Cảnh báo sớm về tình trạng thiếu nhân viên bất khả kháng, làm cho nó dễ dàng để giải quyết vấn đề hơn.

(8) Các nhân tố phát triển:

Các nhân tố phát triển bắt buộc bao gồm phần cứng, phần mềm và công cụ phát triển phần cứng, không gian văn phòng, và các mặt hàng khác. Đối với từng nhân tố, khoảng thời gian cần thiết cho việc sử dụng nó nên được ghi trên thời gian biểu.

(9) Các rủi ro phát triển:

Rủi ro phát triển vốn có trong bất kỳ dự án. Để hiểu pervasiveness của chúng, và làm thế nào có thể kiểm soát được chúng, đầu tiên chúng ta phải xác định các khái niệm. Một rủi ro phát triển là "một tiểu bang hoặc tài sản của một công việc hoặc môi trường phát triển, trong đó, nếu bỏ qua, sẽ tăng khả năng thất bại của dự án" (Ropponen và Lyytinen, 2000). Các rủi ro phát triển tiêu biểu là:

■ Các khoảng trống công nghệ - Thiếu kiến thức chuyên môn phù hợp và đầy đủ và kinh nghiệm để thực hiện các nhu cầu của các hợp đồng phát triển.

- Thiếu nhân viên - hụt Unanticipated của đội ngũ nhân viên chuyên nghiệp.
- Sự phụ thuộc lẫn nhau của các yếu tố tổ chức - Các khả năng của các nhà cung cấp phần cứng hoặc phần mềm chuyên dụng của nhà thầu phụ, ví dụ, họ sẽ không thực hiện nghĩa vụ của họ như trên thời gian biểu.

Hệ thống hoạt động quản lý rủi ro nên được khởi xướng để đối phó với chúng. Quy trình quản lý rủi ro bao gồm các hoạt động sau đây: nhận biết rủi ro, đánh giá rủi ro, lập kế hoạch hành động quản lý rủi ro (RMAs), thực hiện RMAs, và giám sát các RMAs.

Phần mềm RMAs được đưa vào kế hoạch phát triển.

Tầm quan trọng ngày càng tăng của phần mềm quản lý rủi ro được thể hiện trong mô hình xoắn ốc của các vòng đời phần mềm. Để đối phó với các loại rủi ro, một giai đoạn đặc biệt dành riêng để đánh giá rủi ro phần mềm được chỉ định cho mỗi chu kỳ xoắn ốc (Boehm, 1988, 1998).

(10) Các phương pháp kiểm soát:

Để kiểm soát việc thực hiện dự án, quản lý dự án và quản lý phòng áp dụng một loạt các giám sát thực tiễn khi chuẩn bị báo cáo tiến độ và phối hợp với các cuộc họp.

(11) Ước lượng chi phí dự án:

Ước lượng chi phí dự án là dựa trên đề xuất dự toán chi phí, sau đó là xem xét kỹ lưỡng về sự liên quan tiếp tục của chúng dựa trên các số ước lượng tài nguyên cập nhật con người, thương lượng hợp đồng với nhà thầu phụ và nhà cung cấp, và ..v..v. Ví dụ, một phần của dự án, kế hoạch sẽ được thực hiện bởi một nhóm phát triển nội bộ, cần phải được thực hiện bởi một nhà thầu phụ, do độ chưa sẵn sàng của đội. Một sự thay đổi của bản chất này thường được tham gia vào một ngân sách bổ sung đáng kể.

2.2.3. Các thành phần của kế hoạch chất lượng

Tất cả hay một số mục sau đây, tùy thuộc vào các dự án, bao gồm các thành phần của một kế hoạch quản lý chất lượng dự án :

(1) Những mục tiêu của quản lý chất lượng

Thuật ngữ “quality goals” dùng để chỉ các yêu cầu chất lượng thực chất của việc phát triển hệ thống phần mềm. Đơn vị định lượng được ưa thích hơn so với

các đơn vị định tính khi lựa chọn mục tiêu chất lượng, bởi chúng cung cấp cho nhà phát triển các đánh giá khách quan hơn về hiệu suất phần mềm trong suốt quá trình phát triển tiến trình và kiểm thử hệ thống. Tuy nhiên, không chỉ có một loại mục tiêu thích ứng với tất cả. Các thẻ thay thế chất lượng với đơn vị định lượng được minh họa trong ví dụ sau đây

Ví dụ

Một hệ thống phần mềm phục vụ các công việc văn phòng của một nhà máy sản xuất điện được phát triển. Hệ thống trợ giúp công việc văn phòng (HDS – Help desk system) được dự định hoạt động 100 giờ mỗi tuần. Đội quản lý chất lượng phần mềm được yêu cầu chuẩn bị trước 1 danh sách các mục tiêu chất lượng định lượng phù hợp với những yêu cầu chất lượng nhất định, như thể hiện trong bảng sau

HDS qualitative requirements (yêu cầu chất lượng)	Các mục tiêu định lượng chất lượng liên quan
HDS: thân thiện với người sử dụng	Một thao tác mới trợ giúp công việc văn phòng có thể học chi tiết theo một khóa học kéo dài ít hơn 8 giờ, và làm chủ hoạt động trong ít hơn 5 ngày làm việc
HDS: rất tin cậy	HDS luôn sẵn sàng trên 99,5% (thời gian chết của HDS không được quá 30 phút mỗi tuần)
HDS: hoạt động liên tục	Thời gian phục hồi hệ thống không được quá 10 phút trong trường hợp HDS bị lỗi

HDS: có hiệu quả cao	Một thao tác HDS phải có khả năng xử lý ít nhất 100 yêu cầu của khách hàng trong 8 giờ thay đổi
HDS: cung cấp dịch vụ chất lượng cao cho khách hàng	Thời gian chờ đợi cho một thao tác phản hồi không vượt quá 30 giây trong 99% lời yêu cầu.Những thành công của mục tiêu này phụ thuộc vào sự kết hợp các tính năng của phần mềm và số lượng cài đặt và vận hành các trạm làm việc

Những mục tiêu chất lượng phải phản ánh những tiêu chí chính được chấp nhận và được chỉ ra trong tài liệu yêu cầu của khách hàng(ví dụ các tài liệu RFP). Như vậy, những mục tiêu chất lượng như là đơn vị các thành công của các yêu cầu chất lượng của khách hàng

(2) Kế hoạch đánh giá hoạt động

Kế hoạch quản lý chất lượng phải cung cấp một danh sách đầy đủ tất cả các kế hoạch đánh giá hoạt động: đánh giá thiết kế (Design Reviews –DRs), kiểm tra thiết kế, kiểm tra mã (code).., với những xác định sau cho từng hoạt động :

- Phạm vi đánh giá hoạt động
- Các loại hình đánh giá hoạt động
- Lập lịch đánh giá hoạt động (được định nghĩa bởi độ ưu tiên và các hoạt động thành công của tiến trình dự án)
- Các thủ tục cụ thể được áp dụng
- Ai chịu trách nhiệm thực hiện các hoạt động đánh giá lại ?

(3) Kế hoạch kiểm thử phần mềm

Kế hoạch quản lý chất lượng phải cung cấp một danh sách đầy đủ kế hoạch kiểm thử phần mềm, với những thiết kế sau cho mỗi lần kiểm tra:

- Đơn vị, hệ thống tích hợp hay hoàn chỉnh để kiểm tra
- Các loại hình của hoạt động kiểm thử sẽ được thực hiện, bao gồm các đặc điểm kỹ thuật của các lần kiểm thử phần mềm trên máy tính được áp dụng
- Lịch lịch cho kế hoạch kiểm thử (được định nghĩa bởi thứ tự ưu tiên của nó và những hoạt động thành công của tiến trình dự án)

- Các thủ tục cụ thể được áp dụng
- Ai chịu trách nhiệm thực hiện kiểm tra

(4) Kế hoạch kiểm thử sự chấp nhận cho phần mềm phát triển bên ngoài

Một danh sách đầy đủ của kế hoạch kiểm thử sự chấp nhận cho phần mềm phát triển bên ngoài phải được cung cấp trong kế hoạch quản lý chất lượng bao gồm các mục :

- (a) Phần mềm được mua
- (b) Phần mềm được phát triển bởi các nhà thầu phụ
- (c) Phần mềm được khách hàng cung cấp

Kiểm thử sự chấp nhận cho phần mềm phát triển bên ngoài nên song song với việc kiểm thử phần mềm phát triển bên trong.

(5) Quản lý cấu hình

Kế hoạch quản lý chất lượng phải xác định các công cụ và thủ tục quản lý cấu hình, bao gồm cả các thủ tục kiểm soát thay đổi và phải áp dụng trong suốt dự án.

Các thành phần quản lý chất lượng phần mềm được yêu cầu được liệt kê trong bảng sau:

Các thành phần của quản lý chất lượng phần mềm
<ol style="list-style-type: none"> 1. Danh sách các mục tiêu chất lượng 2. Đánh giá các hoạt động 3. Kiểm thử phần mềm 4. Kiểm thử sự chấp nhận phần mềm phát triển bên ngoài 5. Các công cụ và thủ tục quản lý cấu hình

2. Đánh giá các hoạt động
3. Kiểm thử phần mềm

4. Kiểm thử sự chấp nhận phần mềm phát triển bên ngoài

5. Các công cụ và thủ tục quản lý cấu hình

*** Tài liệu quản lý chất lượng, sự phê duyệt và định dạng của nó**

Quản lý chất lượng có thể được chuẩn bị như một phần của kế hoạch phát triển hay như một tài liệu độc lập. Trong một số trường hợp, kế hoạch được chia thành một số tài liệu bởi hạng mục , như kế hoạch DR, kế hoạch kiểm thử, và kế hoạch kiểm tra sự chấp nhận phần mềm phát triển bên ngoài . Đánh giá và chấp thuận của kế hoạch quản lý chất lượng nên được tiến hành theo thủ tục chuẩn của tổ chức cho các kế hoạch như vậy.

2.2.4. Các kế hoạch phát triển và kế hoạch chất lượng cho các dự án nhỏ và các dự án nội bộ

Việc các nhà lãnh đạo dự án cố gắng tránh khỏi những “rắc rối” trong quá trình chuẩn bị kế hoạch phát triển và quản lý chất lượng là điều tự nhiên. Điều này phản ánh xu hướng tránh việc “làm việc quan liêu” và kiểm soát chung chung mà khách hàng có thể dự tính thực hiện. Xu hướng này đặc biệt thường thấy trong 2 trường hợp : các dự án nhỏ và các dự án nội bộ. Sự chuẩn bị kế hoạch cho các dự án như vậy sẽ được thảo luận trong 2 phần sau.

- Kế hoạch phát triển và quản lý chất lượng cho các dự án nhỏ

❖ một dự án chỉ có thời hạn 40 ngày, có thể được thực hiện bởi một chuyên gia và hoàn thành trong 12 tuần, với 1 man-days có phải chuẩn bị lập kế hoạch quản lý chất lượng và phát triển toàn thể ?

❖ một dự án được thực hiện bởi 3 chuyên gia với tổng vốn là 30 man-days và hoàn thành trong 5 tuần, đòi hỏi phải lập kế hoạch toàn thể?

Cần làm rõ rằng các thủ tục lập kế hoạch quản lý chất lượng và phát triển áp dụng cho các dự án lớn có thể sẽ không áp dụng được cho các dự án nhỏ. Các thủ tục đặc biệt là cần thiết. Các thủ tục này xác định cách giải quyết các dự án như trong các câu hỏi trên với việc chú trọng tới các kế hoạch :

(1) trường hợp không cần lập kế hoạch quản lý chất lượng và phát triển. Ví dụ các dự án đòi hỏi khoảng 15 man-days

(2) trường hợp việc chuẩn bị các kế hoạch phụ thuộc vào quyết định của lãnh đạo dự án. Ví dụ : một dự án đòi hỏi dưới 50 man-days mà không có rủi ro quan trọng đã được xác định. Trong trường hợp này có thể quyết định rằng việc lập kế hoạch là không cần thiết. Một ví dụ khác :

một dự án nhỏ nhưng phức tạp cần được hoàn thành trong 30 ngày, trong đó sẽ có một hình phạt nghiêm trọng khi không hoàn thành đúng thời gian. Trong trường hợp này, lập kế hoạch là cần thiết để đối phó với những khó khăn của dự án.

(3) Trường hợp việc lập kế hoạch phát triển và quản lý chất lượng là bắt buộc.

- *Kế hoạch phát triển :*

➢ Các sản phẩm dự án, chỉ ra “sự phân phôi”

➢ Các điểm chuẩn của dự án

➢ Các rủi ro trong việc phát triển

➢ Ước lượng giá thành dự án

- *Kế hoạch quản lý chất lượng :*

➢ Mục tiêu chất lượng

Một số lợi ích của các dự án nhỏ được lập kế hoạch so với các dự án không được lập kế hoạch có thể được xác định, thậm chí cho các kế hoạch “suy giảm”:

- (1) Sự hiểu biết đầy đủ và triệt để hơn các nhiệm vụ để có thể hoàn thành chúng
- (2) Trách nhiệm lớn hơn có thể được phân công
- (3) Dễ dàng hơn cho người quản lý và khách hàng trong việc chia sẻ quyền kiểm soát dự án và sớm phát hiện ra các sự chậm trễ không mong muốn
- (4) Hiểu biết tốt hơn về các yêu cầu và thời gian biểu được đặt ra giữa người phát triển và khách hàng

- Kế hoạch phát triển và quản lý chất lượng đối với các dự án nội bộ

Các dự án nội bộ là các dự án dự định dành cho các bộ phận khác trong tổ chức hoặc cho toàn bộ tổ chức, hoặc dùng các dự án này trong việc phát triển các gói phần mềm cho thị trường phần mềm. Nhìn chung, tất cả các dự án loại này sẽ không có sự tham gia của các khách hàng bên ngoài trong việc phát triển dự án. Các dự án nội bộ có quy mô vừa hoặc lớn. Tuy nhiên, ngay cả trong trường hợp này cũng có xu hướng tránh việc chuẩn bị các kế hoạch phát triển và quản lý chất lượng một cách đầy đủ. Ví dụ sau minh họa hậu quả tiêu cực của một dự án nội bộ không được lập kế hoạch :

Bộ phận tiếp thị của công ty Toyware, một công ty sản xuất trò chơi vi tính mới, đã có kế hoạch tung ra thị trường “Super – Monster 2000” – trò chơi vi tính mới của công ty – trong mùa Giáng sinh sắp tới. Bộ phận phát triển phần mềm cho rằng việc xây dựng trò chơi nên được bắt đầu ngay lập tức để hoàn thành dự án đúng thời gian. Do đó, việc chuẩn bị một buổi thảo luận giữa bộ phận tiếp thị và bộ phận phát triển; và các kế hoạch phát triển và quản lý chất lượng không được xem là cần thiết. Bộ phận phát triển ước tính ngân sách khoảng 240 000\$, và đã được chuyển giao. Theo lịch tiếp thị, kiểm thử hệ thống phải được hoàn thành trước 1 tháng 10 để bộ phận tiếp thị thực hiện các yêu cầu xúc tiến và quảng cáo chiến dịch trong mùa bán hàng của dịp Giáng sinh. Khi dự án tiến triển, nó có thể có một sự chậm trễ. Vào cuối tháng 6, người ta nhận ra 3 tháng chậm trễ là không thể tránh khỏi. Các hoạt động quảng cáo diễn ra trước 30.6 trở thành vô nghĩa. Cuối cùng, dự án hoãn thành vào cuối tháng 2. Giá thành của dự án vượt đáng kể - thực tế chi phí vượt quá 385 000\$ - nhưng thiệt hại lớn nhất là công ty mất cơ hội khai thác thị trường mùa Giáng sinh. Tuần trước, quản lý công ty quyết định sẽ tránh tất cả các dự án nội bộ phát triển trò chơi vi tính trong tương lai.

Ví dụ này cho thấy rõ ràng là việc chuẩn bị các kế hoạch phát triển và quản lý chất lượng toàn bộ cho các dự án nội bộ - kể cả việc giám sát thường xuyên – có thể rất có lợi cho việc thực hiện các dự án nội bộ.

Bộ phận phát triển phần mềm có thể đạt được những thuận lợi sau đây với việc chuẩn bị kế hoạch :

- (1) Tránh vượt quá ngân sách. Điều này có tầm quan trọng đặc biệt trong việc đảm bảo lợi nhuận.
- (2) Tránh thiệt hại cho các dự án khác do các chuyên gia tham gia vào dự án nội bộ bị chậm trễ nên chưa thể tham gia vào dự án khác.
- (3) Tránh trường hợp mất thị trường, đặc biệt liên quan đến danh tiếng của công ty, do các dự án nội bộ chậm trễ kéo theo các dự án bên ngoài phụ thuộc vào nó cũng bị trễ.

Khách hàng nội bộ có thể có những thuận lợi sau :

- (1) Ít trễ hơn và ngân sách ít bị vượt quá hơn.
- (2) Kiểm soát tốt hơn quá trình phát triển, trong đó có việc xác định sự chậm trễ sớm hơn, từ đó có thể tìm cách giải quyết nguyên nhân của

chúng.

(3) Ít thiệt hại chậm trễ nội bộ
hơn Tổ chức có thể có những lợi
ích sau :

- (1) Giảm rủi ro mất thị trường do sản phẩm bị trễ
- (2) Giảm rủi ro bị kiện do hoàn thành các sản phẩm bị muộn, vì thế, giảm
thiệt hại vì vi phạm hợp đồng
- (3) Giảm nguy cơ danh tiếng của công ty bị ảnh
hưởng Giảm rủi ro phải bổ sung ngân sách

Chương 3. Các thành phần SQA trong vòng đời dự án

3.1. Tích hợp các hoạt động chất lượng trong vòng đời dự án

Các mô hình phát triển phần mềm cung cấp một tập các khái niệm và các phương pháp luận cần thiết để xây dựng phần mềm.Thêm vào đó, mô hình phát triển bao gồm các định nghĩa của các hoạt động chính cần cho sự phát triển của dự án, cho các điểm mốc của của dự án. Người quản trị dự án sẽ xác định cách dự án thực hiện bằng việc xác định mô hình phát triển và áp dụng vào trong dự án. Hầu hết các hoạt động đảm bảo chất lượng được đặt trong cấu hình với các điểm mốc (milestones), trong đó yêu cầu xem xét lại các hoạt động phát triển sản phẩm đã được hoàn thiện trước đó. Do đó, những người đảm bảo chất lượng phần mềm nên hiểu biết về các mô hình kỹ nghệ phần mềm khác nhau để có thể chuẩn bị một kế hoạch chất lượng mà sẽ được tích hợp thành kế hoạch dự án.

3.1.1. Phương pháp phát triển phần mềm truyền thống và các phương pháp khác

Bốn mô hình của quy trình phát triển được bàn luận trong phần này là:

- Mô hình vòng đời phát triển phần mềm (SDLC).
- Mô hình bản mẫu nhanh.
- Mô hình xoắn ốc.
- Mô hình hướng đối tượng.

Mô hình vòng đời phát triển phần mềm (SDLC) là một mô hình truyền thống (hiện nay vẫn có thể áp dụng được); nó cung cấp sự mô tả toàn diện nhất về quy trình phát triển phần mềm. Mô hình chỉ ra cần xây dựng những khối chính cho toàn bộ quá trình phát triển, được mô tả như là một chuỗi tuyến tính. Trong phần đầu của quy trình phát triển phần mềm, các tài liệu thiết kế sản phẩm được chuẩn bị, việc đánh giá phiên bản đầu tiên của chương trình máy tính chỉ diễn ra ở giai đoạn cuối của quy trình. Mô hình SDLC có thể đóng vai trò như một khung (framework) để biểu diễn các mô hình khác.

Mô hình bản mẫu dựa trên sự thay thế của một hoặc nhiều pha trong mô hình SDLC bằng một quy trình đánh giá, các bản mẫu phần mềm được sử dụng cho quá trình giao tiếp giữa người phát triển, người sử dụng cuối và khách hàng. Các bản mẫu được đưa ra để người sử dụng đánh giá. Sau đó, người phát triển tiếp tục phát

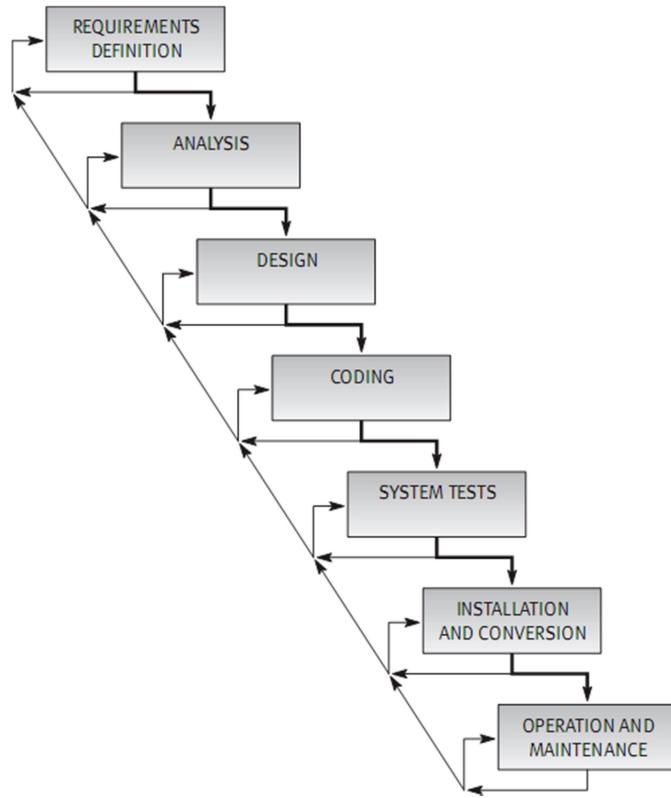
triển một bản mẫu nâng cao, bản mẫu này cũng phải được đưa ra để đánh giá. Quá trình đánh giá này tiếp tục cho đến khi dự án phần mềm được hoàn thiện hoặc bản mẫu phần mềm đã đạt được những gì mà pha đó yêu cầu. Trong trường hợp này, phần còn lại của quy trình phát triển có thể được thực thi theo một phương pháp luận khác, chẳng hạn như theo mô hình SDLC truyền thống.

Mô hình xoắn ốc cung cấp một phương pháp luận nhằm đảm bảo hiệu quả về mặt hiệu năng của mỗi pha trong mô hình SDLC truyền thống. Mô hình này liên quan đến một quy trình lặp, tích hợp các yêu cầu thay đổi của khách hàng, phân tích và giải quyết các rủi ro, các hoạt động về mặt kỹ nghệ và kế hoạch cho phát triển phần mềm. Một trong những tương tác của mô hình xoắn ốc có thể là yêu cầu hoàn thiện ở mỗi pha của dự án trong mô hình SDLC.

Mô hình hướng đối tượng kết hợp chặt chẽ việc sử dụng lại các phần mềm cũ lớn bằng cách kết hợp các mô đun có thể sử dụng lại được thành các hệ thống phần mềm mới. Trong trường hợp không có mô đun phần mềm nào có thể sử dụng được (theo thuật ngữ đối tượng hoặc thành phần) sẵn có, người phát triển có thể thực hiện một quy trình bản mẫu hoặc quy trình SDLC để hoàn thành việc phát triển một hệ thống mới.

- **Mô hình vòng đời phát triển phần mềm (SDLC)**

SDLC là mô hình tuyến tính bắt đầu bằng việc định nghĩa các yêu cầu và kết thúc bằng sự vận hành của hệ thống và quá trình bảo trì. Thể hiện phổ biến nhất của SDLC là mô hình waterfall (thác nước). Mô hình này được minh họa trong hình dưới đây:



Hình Mô hình thác nước

Mô hình gồm có 7 pha: xác định yêu cầu, phân tích, thiết kế, coding, kiểm thử hệ thống, cài đặt và chuyển giao, vận hàng và bảo trì.

Xác định yêu cầu: Mục đích của pha xác định yêu cầu là xác định các chức năng của hệ thống cần xây dựng, khách hàng phải đưa ra và xác định các quyết định của họ. Trong nhiều trường hợp hệ thống phần mềm là một phần của hệ thống lớn hơn. Thông tin về các phần khác của hệ thống được mở rộng sẽ giúp thiết lập sự cộng tác giữa đội dự án và các giao diện thành phần phát triển.

Phân tích: Nỗ lực chính ở đây là phân tích sự ẩn ý của những yêu cầu thành mô hình khởi tạo của hệ thống phần mềm.

Thiết kế: Giai đoạn này bao gồm việc định nghĩa chi tiết đầu vào, đầu ra và các thủ tục xử lý, bao gồm cấu trúc dữ liệu, cơ sở dữ liệu, cấu trúc phần mềm, ...

Coding: Trong pha này, toàn bộ thiết kế được chuyển thành code. Việc coding bao gồm những hoạt động đảm bảo chất lượng phần mềm như inspection, unit test, và test tích hợp.

Test hệ thống (System test): Test hệ thống được thực hiện sau khi pha coding hoàn thiện. Mục đích chính của việc kiểm thử là càng tìm ra nhiều lỗi phần mềm càng tốt để đạt được mức độ chấp nhận của chất lượng phần mềm. Kiểm thử hệ thống được thực hiện bởi những người phát triển trước khi bàn giao cho khách hàng. Trong nhiều trường hợp, khách hàng thực hiện kiểm thử phần mềm một cách độc lập (còn gọi là test chấp nhận sản phẩm) để đảm bảo rằng những người phát triển đã thỏa mãn tất cả những hứa hẹn và những phản ứng phần mềm bất ngờ hoặc lỗi có thể được thấy trước. Đây là điều khá phổ biến, khách hàng yêu cầu người phát triển để họ được tham gia vào trong quá trình kiểm thử hệ thống, một thủ tục đã tiết kiệm được về mặt thời gian và tài nguyên cho việc phân tách riêng biệt giữa test hệ thống và test chấp nhận.

Cài đặt và chuyển giao: Sau khi hệ thống phần mềm được phê chuẩn, hệ thống được cài đặt để phục vụ như là một phần sụn, có nghĩa là, nó như là một phần của hệ thống thông tin, biểu diễn một thành phần cơ bản của hệ thống được mở rộng. Nếu hệ thống thông tin mới được xây dựng để thay thế hệ thống đang tồn tại, một quy trình chuyển giao phải được khởi tạo để đảm bảo rằng các hoạt động của tổ chức vẫn được tiếp tục mà không bị gián đoạn trong suốt pha chuyển giao.

Vận hành và bảo trì: Vận hành phần mềm bắt đầu sau khi pha cài đặt và chuyển giao đã được xong. Xuyên suốt thời kì vận hành, thường ít nhất là một vài năm hoặc cho đến khi xuất hiện phần mềm mới, việc bảo trì là cần thiết. Việc bảo trì kết hợp ba kiểu dịch vụ: thích ứng – sử dụng các đặc trưng của phần mềm đang tồn tại để thực hiện các yêu cầu mới; hoàn thiện – thêm một số đặc trưng để cải thiện hiệu năng của phần mềm; sửa lỗi – sửa các lỗi phần mềm được tìm thấy bởi người sử dụng trong quá trình vận hành.

Số lượng các pha có thể thay đổi tùy theo đặc trưng của từng dự án. Trong các dự án phức tạp, các mô hình phần mềm cỡ lớn, một số pha có thể bị chia ra, do đó, số lượng các pha có thể lên đến tám, chín hoặc thậm chí là nhiều pha hơn nữa. Trong các dự án nhỏ, một số pha lại có thể bị gộp lại, giảm số lượng pha của SDLC xuống còn sáu, năm, thậm chí là chỉ còn bốn pha.

Ở cuối mỗi pha, các đầu ra được xem xét và đánh giá bởi người phát triển, và trong một số trường hợp cũng có thể là khách hàng tham gia. Các kết quả có thể của quá trình xem xét lại bao gồm:

- Sự phê chuẩn của các đầu ra ở pha đó và quy trình của pha tiếp theo
- Các yêu cầu chỉnh sửa, làm lại hoặc thay đổi các phần của pha cuối cùng; trong trường hợp đảm bảo tính chắc chắn, có thể trả về các pha gần nhất theo yêu cầu.

Độ rộng của đường kết nối giữa các hình hộp chữ nhật trong hình minh họa phản ánh khả năng có những kết quả khác nhau. Do đó, hầu hết các quy trình được thực hiện như là một dòng tuyếng tinh. Tuy nhiên, cần chú ý rằng, mô hình nhấn mạnh các hoạt động phát triển trực tiếp và không chỉ ra sự tham gia của khách hàng trong quy trình phát triển.

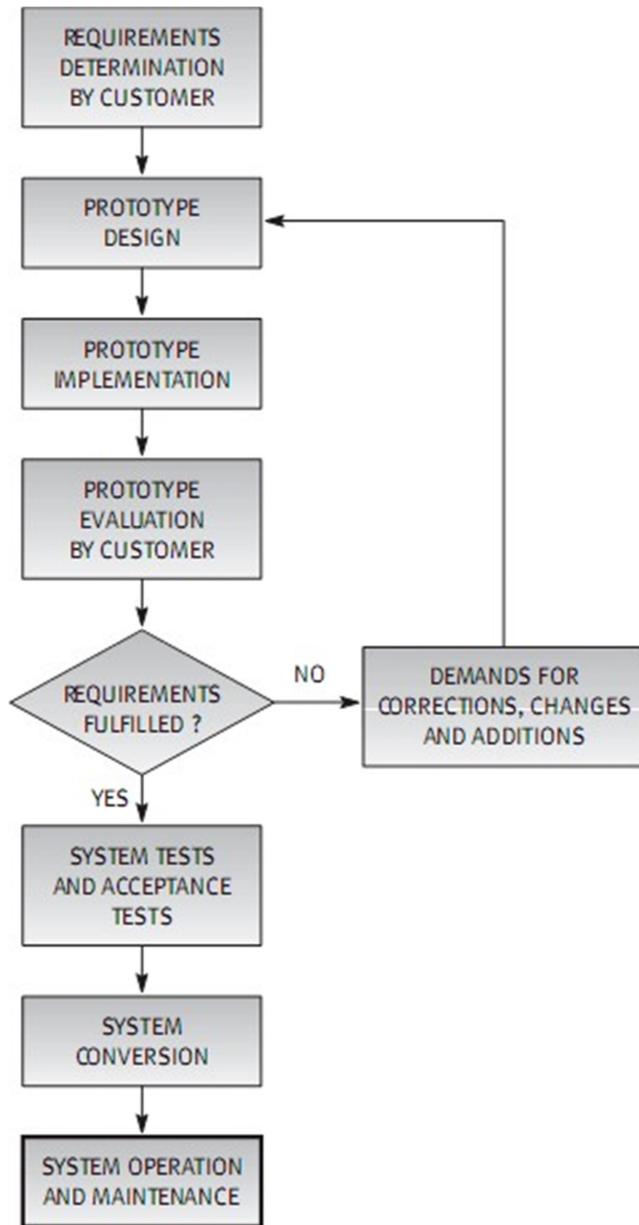
Mô hình Waterfall đã được đề xuất bởi Royce (1970) và sau đó đưa ra trong mô hình chung của nó được biết đến như là Boehm (1981). Mô hình này cung cấp các nền tảng cho phần lớn các chuẩn đảm bảo chất lượng phần mềm đã được triển khai, chẳng hạn như chuẩn IEEE 1012, IEEE 12207, ...

- Mô hình bản mẫu

Phương pháp bản mẫu:

- Cho phép phát triển nhanh và dễ dàng các bản mẫu phần mềm.
- Kết hợp với sự tham gia của khách hàng và người sử dụng trong quy trình phát triển để kiểm tra và đánh giá bản mẫu.

Khi áp dụng phương pháp bản mẫu, những người sử dụng tương lai của hệ thống được yêu cầu bình luận các phiên bản của bản mẫu phần mềm khác nhau đã được chuẩn bị bởi người phát triển. Từ sự phản hồi của khách hàng và người sử dụng, người phát triển sẽ sửa bản mẫu cho phù hợp, cho đúng và thêm các phần vào hệ thống theo cách biểu diễn như là một phiên bản phần mềm tiếp theo để người sử dụng đánh giá. Quy trình này được lặp lại cho đến khi đạt được mục đích của bản mẫu hoặc toàn bộ hệ thống phần mềm được hoàn thiện. Ứng dụng điển hình của phương pháp luận bản mẫu được minh họa trong hình dưới đây:



Mô hình bản mẫu

Phương pháp bản mẫu có thể được áp dụng trong sự kết hợp với các phương pháp luận khác hoặc có thể xem như là một phương pháp luận độc lập. Nói cách khác, sự mở rộng phương pháp bản mẫu có thể thay đổi, từ thay thế một pha SDLC (hoặc một pha trong phương pháp luận khác) đến bản mẫu hoàn thiện của toàn bộ hệ thống phần mềm.

Phương pháp bản mẫu như là một phương pháp luận phát triển phần mềm đã được áp dụng có hiệu quả cho các loại dự án có kích thước từ nhỏ đến trung bình. Ưu điểm

cũng như thiếu sót chính của phương pháp bản mẫu so với SDLC được tổng kết lại trong bảng dưới đây:

Ưu điểm:

- Quy trình phát triển ngắn.
- Tiết kiệm tài nguyên (man-days).
- Phù hợp với các yêu cầu của khách hàng và giảm thiểu được rủi ro của dự án.
- Sự tiếp nhận (lĩnh hội) của người dùng về hệ thống mới nhanh hơn và dễ dàng hơn.

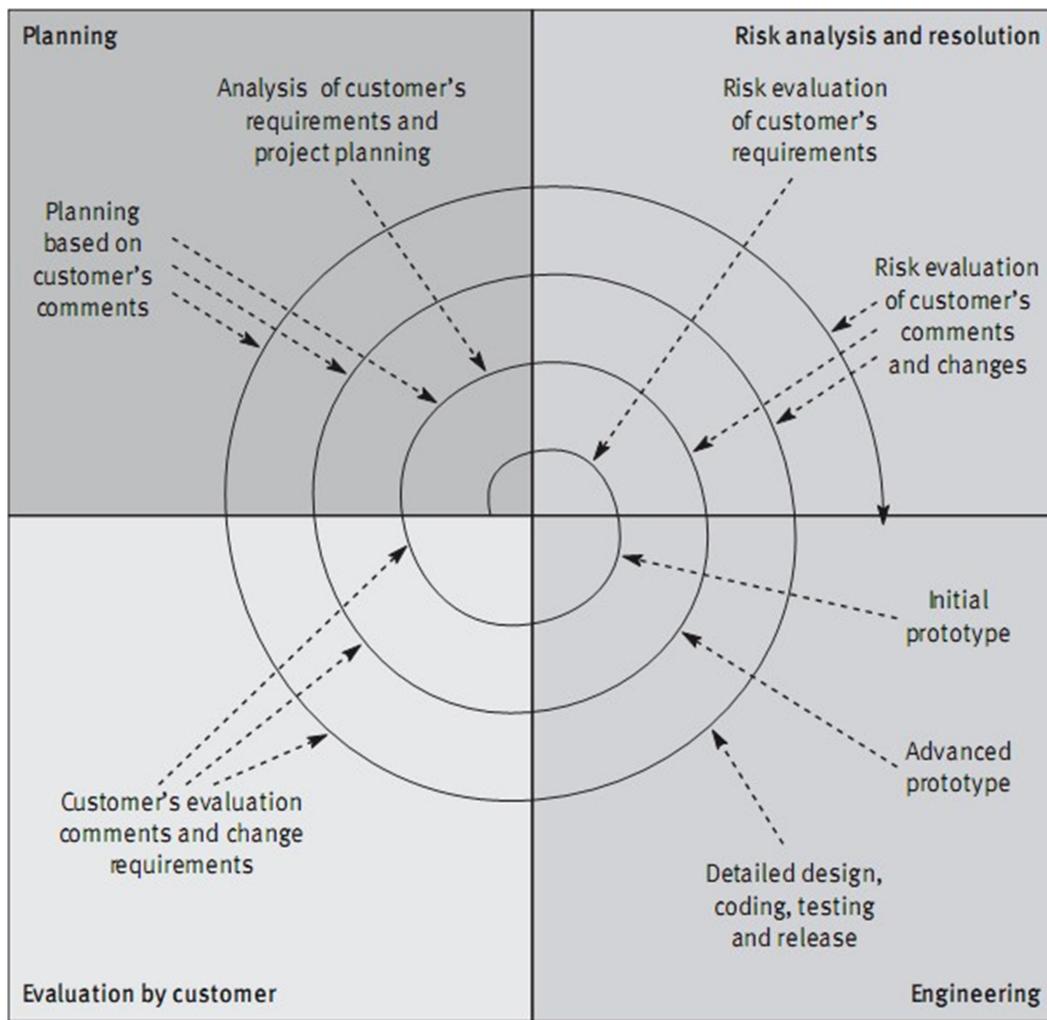
Nhược điểm:

- Giảm bớt khả năng mềm dẻo và thích nghi đối với những thay đổi và những bổ sung.
- Giảm sự chuẩn bị cho các trường hợp lỗi không mong đợi.

- Mô hình xoắn ốc

Mô hình xoắn ốc, được xem xét bởi Boehm (1988, 1998), đưa ra một phương pháp luận cải thiện nhằm phục vụ cho các dự án cỡ lớn và phức tạp.

Mô hình này được minh họa như hình dưới đây:

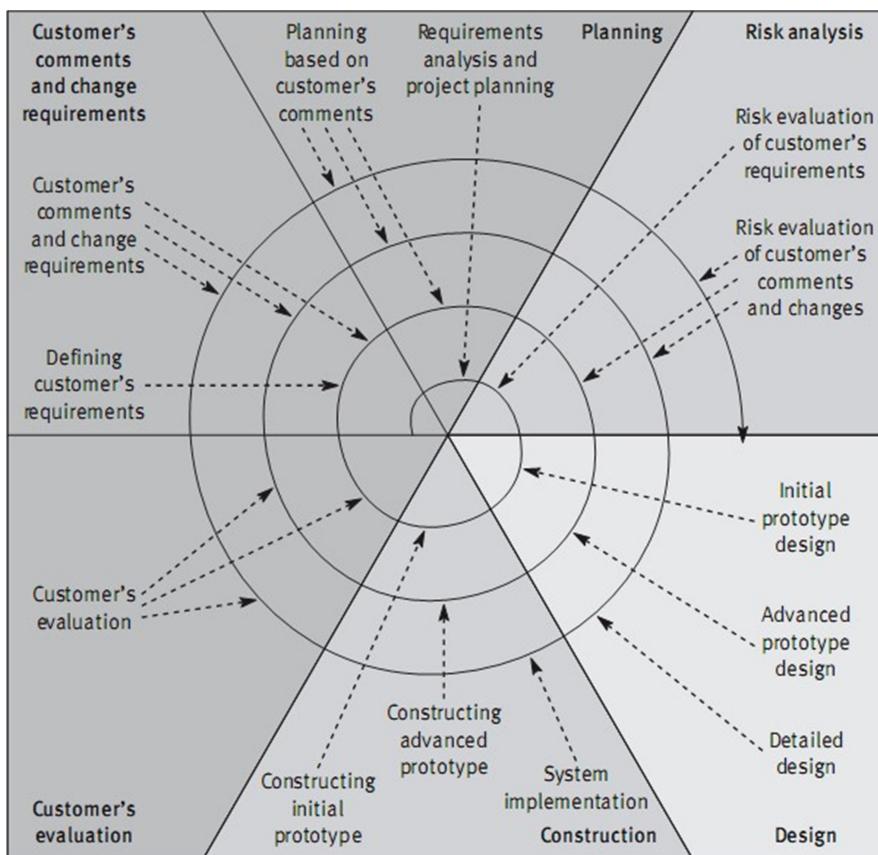


Mô hình xoắn ốc

Theo mô hình xoắn ốc, sự phát triển phần mềm được hiểu như làm một quy trình lặp; tại mỗi lần lặp, các hoạt động sau được thực hiện:

- Lập kế hoạch.
- Phân tích và giải quyết rủi ro.
- Các hoạt động kỹ nghệ theo giai đoạn của dự án: thiết kế, coding, test, cài đặt và chuyển giao.
- Sự đánh giá của khách hàng bao gồm các bình luận, các thay đổi và các yêu cầu bổ sung, ...

Một mô hình xoắn ốc tiên tiến, là mô hình xoắn ốc win - win. Mô hình xoắn ốc tiên tiến nhấn mạnh vào sự giao tiếp và thương lượng giữa khách hàng và người phát triển. Tên của mô hình ám chỉ thực tế rằng, bằng việc sử dụng mô hình này, khách hàng có thể nhận được hệ thống thỏa mãn nhất cái họ cần, và nhà phát triển có khả năng nhận được nhiều tiền và hoàn thành dự án đúng ngày. Điều này đạt được bằng việc nhán mạnh vào sự tham gia của khách hàng và những hoạt động kĩ nghệ. Việc xem xét lại trong quá trình phát triển được diễn tả bằng hai đoạn đồ thị của xoắn ốc, với sự tham gia của khách hàng: đầu tiên là sự đánh giá của khách hàng, thứ hai là sự phản hồi và những yêu cầu thay đổi của khách hàng. Tương tự như vậy, những hoạt động kĩ nghệ được miêu tả bằng hai đoạn trong hình xoắn ốc: đầu tiên là thiết kế, và thứ hai là xây dựng. Bằng sự đánh giá tiến trình dự án ở cuối mỗi đoạn, nhà phát triển có thể điều chỉnh tốt hơn toàn bộ quá trình phát triển.



Mô hình xoắn ốc cải tiến

Vì vậy, trong mô hình xoắn ốc cải tiến, sáu hoạt động sau phải thực hiện trong mỗi lần lặp:

- Những đặc tả yêu cầu, những phản hồi và những đòi hỏi thay đổi của

khách hàng.

- Những hoạt động lập kế hoạch của nhà phát triển.
- Phân tích và giải quyết những rủi ro của nhà phát triển.
- Những hoạt động thiết kế của nhà phát triển.
- Những hoạt động xây dựng của nhà phát triển, liên quan tới việc coding, test, cài đặt và release.
- Sự đánh giá của khách hàng.

- Mô hình hướng đối tượng

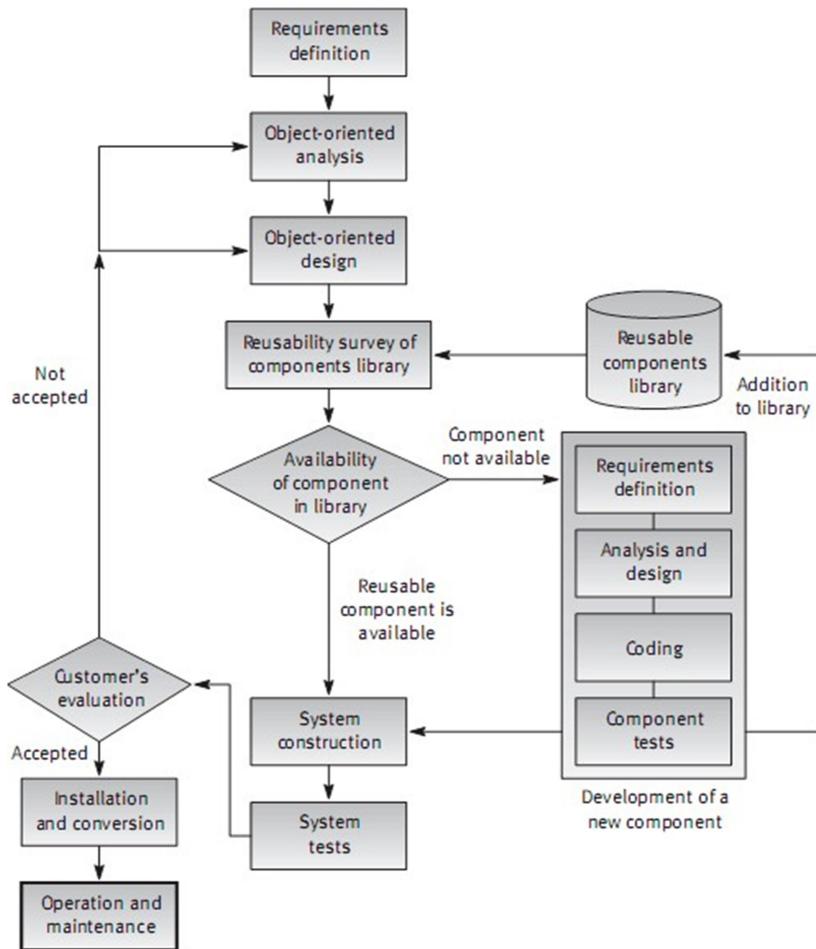
Mô hình hướng đối tượng khác với các mô hình khác bởi nó hướng tới việc sử dụng lại các thành phần phần mềm. Đặc trưng của phương pháp luận này là nó dễ dàng tích hợp các module phần mềm đang tồn tại (được gọi là các đối tượng hoặc các thành phần) vào trong những hệ thống mới. Một thư viện thành phần phần mềm phục vụ cho mục đích này bằng việc cung cấp những thành phần phần mềm cho việc sử dụng lại.

Vì thế, theo mô hình hướng đối tượng được chỉ ra trong hình 7.5, quy trình phát triển bắt đầu với chuỗi phân tích và thiết kế hướng đối tượng. Pha thiết kế được thực hiện bằng sự thu nhận các thành phần phù hợp từ thư viện phần mềm có thể sử dụng lại, khi các thành phần đó là ở trạng thái sẵn sàng trong thư viện. Sự phát triển “thông thường - regular” được thực hiện theo cách khác. Sao chép những thành phần phần mềm mới và sau đó đưa nó vào trong thư viện phần mềm cho mục đích sử dụng lại trong tương lai. Phương pháp này mong đợi sự lớn lên của kho thư viện thành phần phần mềm sẽ cho phép tăng việc sử dụng và chắc chắn sử dụng lại của phần mềm, một hướng sẽ cho phép mang lại ưu điểm lớn về mặt tài nguyên như sau:

- Kinh tế: Chi phí tích hợp thành phần phần mềm sử dụng lại ít hơn so với giá phát triển một phần mềm mới.
- Cải thiện chất lượng: Các thành phần phần mềm được sử dụng ít lỗi hơn là thành phần phần mềm mới xây dựng.
- Thời gian phát triển ngắn hơn: Việc tích hợp các thành phần phần mềm sử dụng lại làm giảm áp lực về mặt lịch biểu.

Như vậy, ưu điểm của phương pháp luận hướng đối tượng so với phương pháp luận khác là sẽ làm tăng sự phát triển của kho lưu trữ phần mềm có thể sử dụng

lại được.



Mô hình hướng đối tượng

3.1.2. Các yếu tố ảnh hưởng hoạt động đảm bảo chất lượng phần mềm

Những hoạt động đảm bảo chất lượng là hướng quy trình, nói cách khác, có liên kết tới sự hoàn thành của một pha dự án, sự hoàn tất một mốc dự án, và nhiều hơn nữa. Những hoạt động đảm bảo chất lượng được tích hợp vào trong kế hoạch phát triển.

Những người lên kế hoạch hoặc đảm bảo chất lượng cho một dự án cần xác định:

- Danh sách những hoạt động đảm bảo chất lượng cần thiết cho dự án.
- Với mỗi hoạt động đảm bảo chất lượng:
 - Thời gian.
 - Loại hoạt động đảm bảo chất lượng áp dụng.

➢ Thời gian.

➢ Loại hoạt động đảm bảo chất lượng áp dụng.

❖ Người thực hiện hoạt động và tài nguyên yêu cầu.

❖ Những tài nguyên yêu cầu cho việc khắc phục những nhược sai sót và những thay đổi.

Cường độ của những hoạt động đảm bảo chất lượng đã được lên kế hoạch được chỉ ra bởi số những hoạt động yêu cầu. Những yếu tố dự án và nhóm ảnh hưởng tới cường độ như sau:

Yếu tố dự án:

- Độ lớn của dự án.
- Sự phức tạp và khó của kỹ thuật.
- Phạm vi của những thành phần sử dụng lại.
- Hậu quả nếu dự án bị lỗi.

Yếu tố nhóm:

- Trình độ chuyên môn của những thành viên trong nhóm.
- Sự quen thuộc của nhóm với dự án và Kinh nghiệm của nhóm trong lĩnh vực.
- Tính sẵn sàng của những thành viên có thể trợ giúp nhóm.
- Sự hiểu biết giữa những thành viên trong nhóm, hay nói cách khác là số thành viên mới trong nhóm.

3.1.3. Xác minh, thẩm định và đánh giá chất lượng

Ba khía cạnh đảm bảo chất lượng của sản phẩm phần mềm được sát hạch dưới những dạng Verification, Validation, và Qualification.

- **Verification:** quá trình đánh giá một hệ thống hay một thành phần để xác định xem những sản phẩm của một pha phát triển xác định có thỏa mãn những điều kiện được đặt gia khi bắt đầu pha đó hay không.
- **Validation:** quá trình đánh giá một hệ thống hay một thành phần trong suốt hoặc khi kết thúc quá trình phát triển để xác định xem nó có thỏa mãn những yêu cầu đã đặc tả hay không.
- **Qualification:** quá trình xác định một hệ thống hoặc một thành phần có phù hợp với việc sử dụng hay không.

Theo những định nghĩa của IEEE, verification kiểm tra tính nhất quán của sản phẩm đang phát triển với những sản phẩm đã được phát triển ở pha trước. Khi thực hiện, người thẩm tra đi sau quy trình phát triển và giả sử rằng tất cả những pha phát triển đãng trước đã được hoàn thành một cách chính xác hoặc là như kế hoặc gốc hoặc là sau khi đã sửa chữa những sai sót được phát hiện.

Validation miêu tả sự quan tâm của khách hàng, bằng cách thẩm tra những yêu cầu gốc của họ.

Qualification tập trung vào những khía cạnh hoạt động, ở đó việc bảo trì là vấn đề chính. Một thành phần phần mềm đã được phát triển và tài liệu hóa theo những chuẩn, kiểu, và cấu trúc chuyên nghiệp sẽ dễ dàng để bảo trì hơn những thành phần có code lạ.

Người lên kế hoạch cần phải xác định xem những khía cạnh nào nên được sát hạnh trong mỗi hoạt động đảm bảo chất lượng phần mềm.

3.2. Rà soát

Định nghĩa của IEEE (1990), quá trình rà soát là:

“Một quá trình hoặc một cuộc họp mà trong đó một sản phẩm công việc hoặc một tập các sản phẩm công việc được đưa ra tới toàn thể cá nhân tham gia vào dự án, các giám đốc, người dùng, khách hàng và các bên quan tâm đến dự án nhằm lấy ý kiến phê bình và phê chuẩn”

Một số phương pháp dùng để xem xét lại tài liệu:

- Xem xét lại thiết kế hình thức (Formal design reviews)
- Xem xét lại ngang hàng (Peer reviews)
- Ý kiến chuyên gia

3.2.1. Mục tiêu rà soát

Mục đích được chia làm 2 loại: mục đích trực tiếp và gián tiếp. Mục đích trực tiếp:

- Phát hiện lỗi phân tích và thiết kế.
- Xác định các rủi ro mới.
- Xác định sự sai lệch so với mẫu, các kiểu thủ tục và qui ước.

- Để phê chuẩn sản phẩm của phân tích hoặc thiết kế.

Mục đích gián tiếp:

- Nơi họp mặt không chính thức để trao đổi về những kiến thức chuyên môn.
- Ghi lại những lỗi phân tích và thiết kế sẽ hỗ trợ một cơ sở cho những hoạt động sửa chữa lỗi trong tương lai.

3.2.2. Những rà soát thiết kế hình thức

Rà soát thiết kế hình thức(DRs-formal Design Reviews) là rà soát duy nhất cần thiết cho việc phê duyệt sản phẩm thiết kế

Rà soát thiết kế hình thức có thể được thực hiện tại bất cứ mốc phát triển nào yêu cầu sự hoàn thiện của tài liệu phân tích hay thiết kế.

Một danh sách các review thiết kế chính thức :

- DPR – Development Plan Review : Review kế hoạch phát triển
- SRSR – Software Requirement Specification Review : Review đặc tả yêu cầu phần mềm
- PDR – Preliminary Design Review : Review thiết kế sơ bộ
- DBDR- Detailed Design Review : Review thiết kế chi tiết
- TPR – Test Plan Review : Review kế hoạch kiểm thử
- STPR – Software Test Procedure Review : Review thủ tục kiểm thử phần mềm
- VDR- Version Description Review : Review mô tả phiên bản
- OMR- Operator Manual Review : Review vận hành thủ công
- SMR- Support Manual Review :Review trợ giúp thủ công
- TRR- Test Readiness Review : Review sự sẵn sàng kiểm thử
- PRR- Product Release Review : Review bản phát hành sản phẩm
- IPR-Installation Plan Review : Review kế hoạch cài đặt

Các nhân tố ảnh hưởng tới DRs

- Những người tham gia
- Sự chuẩn bị trước

- Phiên DR
- Các hoạt động sau DR được đề xuất

- Những người tham gia rà soát thiết kế

Gồm có Review leader và review team.

Review Leader :

- Có kiến thức và kinh nghiệm trong việc phát triển kiểu dự án được review
- Có thâm niên ở mức độ bằng với hoặc cao hơn của project leader
- Có mối quan hệ tốt với project leader và đội dự án
- Có vị trí bên ngoài đội dự án

Review team

- Phần lớn không thuộc đội dự án
- Kích thước từ 3-5 người
- Đa dạng về kinh nghiệm và phương pháp

- Sự chuẩn bị cho một phiên bản DR

Được hoàn thành bởi 3 thành viên: review leader, review team và development team..

Chuẩn bị của Review leader

- Bổ nhiệm các thành viên nhóm
- Lập lịch các phiên review
- Phân chia tài liệu thiết kế cho các thành viên của nhóm

Chuẩn bị của Review team

- Xem lại tài liệu thiết kế
- Danh sách bình luận

Chuẩn bị của Development team

- Trình diễn ngắn tài liệu thiết kế
- Checklist các công việc review

- Phiên DR

Một cuộc họp phiên DR thông thường gồm có :

1. Trình diễn ngắn gọn về tài liệu thiết kế
2. Các bình luận của các thành viên review team
3. Kiểm tra và xác nhận thảo luận mỗi bình luận
4. Các quyết định về tài liệu thiết kế để xác định tiến trình dự án. Các quyết định có thể có 3 loại :
 - Phê duyệt đầy đủ
 - Phê duyệt từng phần
 - Từ chối phê duyệt

- Các hoạt động hậu review

- Báo cáo Review
 - Review leader thực hiện sau phiên review
 - Bao gồm :
 - Tổng kết các thảo luận review
 - Quyết định về sự tiếp tục của dự án
 - Danh sách các hoạt động cần thiết phải làm
 - Tên thành viên chịu trách nhiệm theo sát việc hiệu chỉnh
- Tiến trình theo dõi
 - Review leader thực hiện
 - Chắc rằng việc hiệu chỉnh được thực hiện đúng đắn
 - Việc theo dõi cần được ghi lại

3.2.3. Các rà soát ngang hàng (peer review)

Mục đích chính của peer review là xác định lỗi và độ lệch dựa vào các chuẩn. Có hai phương pháp peer reviews:

- xét duyệt (inspection)
- kiểm tra từng bước (walkthrough).

Walkthrough phát hiện sai sót và ghi chú lên tài liệu.

Inspection phát hiện sai sót và kết hợp với nỗ lực để cải tiến.

- **Những người tham gia vào peer reviews**

Một đội peer review tối ưu 3-5 người tham gia.

Tất cả những người tham gia nên là những người cùng địa vị của nhà thiết kế hệ thống phần mềm.

Một đội peer review đề cử bao gồm:

- Một leader review.
- Một người thực thi (author).
- Các chuyên gia đặc biệt (specialized professionals). **Phân công trách nhiệm trong đội (team assignments)** Hai trong số các thành viên sẽ là:
 - một người dẫn chương trình
 - một người viết tài liệu trong cuộc thảo luận.

- **Chuẩn bị cho phiên peer review**

- **Leader:**
 - Xác định những đoạn trong tài liệu thiết kế sẽ được review
 - Lựa chọn thành viên nhóm
 - Lập lịch cho những phiên review
 - Đưa tài liệu cho các thành viên trong đội trước phiên review
- **Đội peer review:** yêu cầu của inspection khá tinh vi, còn walkthrough chỉ yêu cầu đơn giản.
 - Inspection: Đọc & liệt kê chú thích của họ
 - Walkthrough : Đọc các đoạn sẽ được review

- **Phiên peer review**

- Inspection

- Presenter đọc một đoạn tài liệu và thêm vào nếu cần thiết.
- Những người liên quan hoặc đưa ra chủ thích, hoặc phản ứng với những lời chủ thích trong tài liệu.
- Walkthrough
- Bắt đầu bằng sự trình bày ngắn của author (thường ko phải là presenter) hoặc tổng quan về dự án và những đoạn thiết kế sẽ được review.

Scribe ghi lại vị trí, mô tả, kiểu, đặc điểm (sai sót, những phần thiếu, những phần thêm vào) của mỗi lỗi được chấp nhận.

Quy tắc thời gian: phiên không nên vượt quá 2 giờ, hoặc lập lịch không nhiều hơn 2 ngày.

Tài liệu sau mỗi phiên review:

- Báo cáo những phát hiện trong phiên inspection
- Báo cáo tóm tắt của phiên inspection

- Các hoạt động sau peer review (post-peer review activities) Inspection:

- Nhắc nhở, sửa chữa hiệu quả, làm lại tất cả các lỗi
- Chuyển giao các bản báo cáo inspection tới CAB để phân tích.

- Hiệu quả của peer review (the efficiency of peer reviews)

Một vài độ đo phổ biến ước lượng hiệu suất của peer review:

- Số giờ trung bình trên một lỗi.
- Mật độ phát hiện thiếu sót (Số thiếu sót trung bình trên một trang tài liệu thiết kế).
- Hiệu năng peer review bên trong.

- Peer review coverage

Là tỉ lệ nhỏ của tài liệu và toàn bộ code đã từng trải qua peer review

3.2.4. Các ý kiến của chuyên gia

Ý kiến của chuyên ra rất hữu ích trong những trường hợp sau:

- Thiếu sự hiểu biết đầy đủ về lĩnh vực nào đó.
- Tạm thời thiếu những người chuyên nghiệp để tham gia vào đội xem xét lại

- Các thành viên chuyên nghiệp cao cấp trong tổ chức không thống nhất được với nhau.
- Trong các tổ chức nhỏ số lượng ứng viên phù hợp cho đội xem xét lại là không đủ.

Để rõ hơn những đặc điểm của các phương pháp rà soát có thể tham khảo bảng so sánh giữa ba phương pháp rà soát sau đây:

Thuộc tính	Formal design reviews	Inspections	Walkthroughs
Mục đích h trực tiếp	<ul style="list-style-type: none"> - Phát hiện lỗi - Xác định rủi ro mới - Phê chuẩn tài liệu thiết kế 	<ul style="list-style-type: none"> - Phát hiện lỗi - Xác định sự sai lệch so với tiêu chuẩn 	Phát hiện lỗi
Mục đích h gián tiếp	Trao đổi kiến thức	<ul style="list-style-type: none"> - Trao đổi kiến thức - Hỗ trợ hoạt động sửa chữa 	Trao đổi kiến thức
Lãnh đạo việc xem xét lại	Người đứng đầu kỹ nghệ phần mềm hoặc là thành viên cao cấp	Người điều hành chỉ đạo (ngang hàng)	Người tổ chức
Người tham gia	Các thành viên cấp cao và đại diện khách hàng	Những người ngang hàng	Những người ngang hàng
Sự tham gia của trưởng dự án	Có	Có	Có, thông thường là khi bắt đầu xem xét lại

Thành viên chuyên gia trong đội		<ul style="list-style-type: none"> - Người thiết kế - Lập trình viên - Người kiểm thử 	<ul style="list-style-type: none"> - Người giám sát tiêu chuẩn - Chuyên gia bảo trì - Đại diện người dùng
------------------------------------	--	------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------

Hợp đồng quan	Không	Có	Có
Sự chuẩn bị của những người tham gia	Có – chuẩn bị kỹ lưỡng	Có – chuẩn bị kỹ lưỡng	Có – chuẩn bị tóm lược
Phiên xem xét lại	Có	Có	Có
Bám sát việc sửa chữa	Có	Có	Có
Cơ sở hạ tầng			
Đào tạo chính thức cho người tham gia	Không	Có	Không
Sử dụng checklist	Không	Có	Không
Lỗi liên quan để thu thập dữ liệu	Không yêu cầu hình thức	Yêu cầu hình thức	Không yêu cầu hình thức
Tài liệu hóa việc xem xét lại	Báo cáo xem xét lại thiết kế hình thức	<ul style="list-style-type: none"> - Báo cáo đánh giá phiên thanh tra - Báo cáo tổng kết phiên thanh tra 	Báo cáo đánh giá phiên walkthrough

3.3. Đảm bảo chất lượng của các thành phần bảo trì phần mềm

3.3.1. Giới thiệu

Giai đoạn chính của vòng đời phần mềm là giai đoạn hoạt động, thường kéo dài từ 5 đến 10 năm, mặc dù cũng có những trường hợp hoạt động kéo dài đến 15 năm và thậm chí là hơn thế nữa cũng không phải là hiếm. Vậy điều gì đã khiến cho gói phần mềm có thể đạt được “tuổi thọ cao” với nhóm người dùng mà đã hài lòng nó, trong khi gói phần mềm khác cũng phục vụ đối tượng đó lại “bị hủy sớm”? Yếu tố chính chịu trách nhiệm cho dịch vụ dài hay thành công là bảo trì chất lượng. Việc bảo trì phần mềm quan trọng như thế nào có thể được giả định thông qua việc chú ý đến chủ đề được đưa ra trong chuẩn ISO 9000 – 3 (xem ISO(1977), phiên bản 4.19 và ISO/IEC (2001), phiên bản 7.5), IEEE (1998) và Oskarsson và Glass (1996).

Dưới đây là 3 thành phần của dịch vụ bảo trì và nó được xem như là bản chất của sự thành công:

- Bảo trì sửa lỗi: những dịch vụ hỗ trợ người dùng và sửa lỗi phần mềm.
- Bảo trì thích ứng: làm cho các gói phần mềm thích ứng với những yêu cầu mới của khách hàng và điều kiện môi trường thay đổi.
- Bảo trì cải thiện chất lượng: kết hợp (1) bảo trì hoàn thiện những chức năng mới được thêm vào phần mềm cũng như nâng cao hiệu suất, cùng với (2) bảo trì phòng ngừa – những hoạt động cải thiện độ tin cậy và cơ sở hạ tầng hệ thống cho dễ dàng làm cho việc bảo trì trong tương lai hiệu quả hơn.

Những dịch vụ hỗ trợ người dùng (“những trung tâm hỗ trợ người dùng”) trong bảo trì sửa lỗi có thể cần phải rõ ràng (clarification). Những dịch vụ hỗ trợ giải quyết tất cả những khó khăn xuất hiện khi sử dụng hệ thống phần mềm của người dùng; những dịch vụ sửa lỗi phần mềm thường được tích hợp trong dịch vụ này. Những khó khăn của người dùng có thể được gây ra bởi nguyên nhân sau:

- Lỗi code (thường được dùng với thuật ngữ “thất bại phần mềm”-software failure”).
- Lỗi viết tài liệu thủ công, giúp những màn ảnh hoặc các hình thức tài liệu hướng dẫn được chuẩn bị sẵn sàng cho người dùng. Trong trường hợp này, dịch vụ hỗ trợ có thể cung cấp cho người dùng với những chỉ dẫn đúng (mặc dù không có sự chính xác trong chính tài liệu phần mềm được thực

hiện).

- Không đầy đủ, mơ hồ hoặc tài liệu không chính xác.
- Thiếu kiến thức về hệ thống phần mềm của người dùng hoặc thất bại của họ khi sử dụng tài liệu được cung cấp. Trong những trường hợp này không có xét đến thất bại hệ thống phần mềm.

Ba nguyên nhân đầu tiên ở trên được xem như là những thất bại của hệ thống phần mềm. Thêm vào đó, sự tích hợp của những dịch vụ hỗ trợ người dùng và những dịch vụ hiệu chỉnh phần mềm nói chung là đã được hoàn thành trong sự kết hợp gần gũi, với nhiều thông tin chia sẻ. Những thành phần khác của dịch vụ bảo trì – cải thiện chất lượng và bảo trì thích ứng – khuynh hướng không được bắt đầu bởi những dịch vụ hỗ trợ người dùng. Trong hầu hết các trường hợp, những công việc cải thiện chức năng và khả năng thích ứng sẽ trình bày những đặc tính của một dự án nhỏ hoặc lớn, điều đó phụ thuộc vào những mong muốn của khách hàng. Trong trường hợp này, những công việc trên có thể được thực thi như là một phần của quá trình phát triển phần mềm. Khi xem xét ở trên, việc bao gồm những dịch vụ hỗ trợ người dùng trong những hoạt động bảo trì sửa đổi là hợp lý.

Nói chung, chúng ta có thể nói rằng việc bảo trì sửa lỗi bảo đảm những người dùng hiện thời có thể thao tác hệ thống như đã được vạch rõ, bảo trì thích ứng có thể mở rộng cho những đối tượng người dùng, và bảo trì cải thiện chất lượng đưa ra giai đoạn dịch vụ của gói.

Như đã đề cập trong mục trước, việc kết hợp 3 thành phần bảo trì phần mềm chiếm hơn 60% tổng số tài nguyên thiết kế và lập trình dành cho hệ thống phần mềm trong suốt vòng đời của nó (Pressman, 2000). Những ước lượng khác chia sẻ những tài nguyên bảo trì kéo dài từ trên 50% (Lientz và Swanson, 1980) đến 65 – 75 % tổng số những tài nguyên phát triển dự án.

Mục tiêu hoạt động QA bảo trì phần mềm:

- (1) Đảm bảo, với mức độ tin cậy được chấp nhận, rằng những hoạt động bảo trì phù hợp với những yêu cầu kỹ thuật chức năng.
- (2) Đảm bảo, với mức độ tin cậy được chấp nhận, rằng những hoạt động bảo trì phù hợp với những yêu cầu quản lý lập lịch và ngân sách.
- (3) Những hoạt động khởi đầu và quản lý nhằm cải thiện và tăng hiệu quả cho bảo trì phần mềm và những hoạt động SQA. Điều này liên quan đến việc cải thiện cái nhìn toàn cảnh để đạt được những yêu cầu về chức năng và quản lý trong khi giá thành giảm.

3.3.2. Cơ sở cho chất lượng bảo trì cao

Nhiều người cho rằng chất lượng của gói phần mềm được bảo trì có lẽ là cơ sở quan trọng nhất nằm dưới chất lượng của những dịch vụ bảo trì. Nhưng cũng có những nhà phê bình khác cho rằng đó là chính sách bảo trì. Dưới đây là những thảo luận về chủ đề này.

3.3.2.1. Cơ sở 1: Chất lượng gói phần mềm

Chất lượng của gói phần mềm được bảo trì rõ ràng bắt nguồn từ sự thành thạo và những nỗ lực của nhóm phát triển cũng như những hoạt động SQA được thực hiện xuyên suốt quá trình phát triển. Nếu chất lượng của gói phần mềm là nghèo nàn thì cũng dẫn đến việc bảo trì nghèo nàn hoặc không có hiệu quả. Khi mà hiểu sâu sắc cơ sở này, chúng ta sẽ lựa chọn việc nhấn mạnh 7 trong 11 nhân tố đảm bảo chất lượng ban đầu mà có tác động trực tiếp trong bảo trì phần mềm. Đặc biệt là, chúng ta sẽ thảo luận về 2 trong 5 nhân tố thao tác sản phẩm, 3 nhân tố xem xét lại sản phẩm và 2 trong 3 nhân tố chuyển giao sản phẩm.

Hai nhân tố thao tác sản phẩm:

- **Sự chính xác** (Correctness) – bao gồm:
 - Sự chính xác của đầu ra: Việc hoàn thành những đầu ra được chỉ rõ (nói cách khác là không có đầu ra mà đã được chỉ ra trước đó bị thiếu), sự đúng đắn của những đầu ra

(những đầu ra của hệ thống được xử lý một cách chính xác), đầu ra hợp thời (thông tin được xử lý luôn được cập nhật như đã chỉ rõ) và đầu ra có tính sẵn sàng (Những lần tương tác không vượt những giá trị cực đại xác định, đặc biệt là những ứng dụng trực tuyến và thời gian thực).

- **Sự chính xác** của tài liệu hướng dẫn: Chất lượng của tài liệu hướng dẫn: tính đầy đủ, sự đúng đắn, kiểu và cấu trúc tài liệu. Những hình thức tài liệu bao gồm bản sao trên giấy và những file máy tính – được in ấn thông thường cũng như những file “trợ giúp” điện tử - trong khi phạm vi của nó chưa đựng những tài liệu cài đặt, tài liệu hướng dẫn người dùng và tài liệu của người lập trình viên.
- **Viết mã** đúng quy cách: phù hợp với những hướng dẫn mã, đặc biệt là những hạn chế và sự phức tạp trong mã biến đổi cũng như xác định kiểu viết mã chuẩn.
- **Độ tin cậy.** Tần số của những thất bại của hệ thống cũng như những lần phục hồi.

Ba nhân tố xem xét lại sản phẩm:

- **Sự bảo trì:** Những yêu cầu này được thực hiện đầu tiên và tốt nhất bằng cách làm theo cấu trúc phần mềm và những yêu cầu kiểu cũng như là theo cài đặt những yêu cầu trong tài liệu của lập trình viên.
- **Tính linh động:** Đạt được thông qua việc thiết kế, lập kế hoạch thích hợp và những đặc tính cung cấp không gian ứng dụng lớn hơn nhu cầu của người dùng hiện tại.
- **Có thể test được:** Có thể test bao gồm tính sẵn sàng của những chuẩn đoán hệ thống được đưa ra bởi người dùng cũng như những chuẩn đoán thất bại được áp dụng bởi trung tâm hỗ trợ hoặc những nhân viên bảo trì tại vị trí người dùng.

Cuối cùng là 2 nhân tố chuyển phát sản phẩm:

- (1) **Tính khả chuyển:** Các ứng dụng tiềm tàng của phần mềm trong môi trường phần cứng và hệ điều hành khác nhau, nó gồm những hoạt động mà cho phép thực hiện những ứng dụng đó.
- (2) **Thao tác giữa các phần:** Khả năng của các gói giao tiếp với các gói hoặc thiết bị tính toán khác. Bằng việc cung cấp khả năng đáp ứng những chuẩn về giao diện và áp dụng những giao diện đó một cách phù hợp kết hợp với những hướng dẫn của thiết bị sản xuất và phần mềm, thao tác giữa các phần có thể đạt được ở mức cao.

3.3.2.2. Cơ sở 2: Chính sách bảo trì

Những thành phần của chính sách bảo trì chính ảnh hưởng đến sự thành công của việc bảo trì phần mềm là sự phát triển các phiên bản và thay đổi chính sách để được áp dụng trong suốt vòng đời phần mềm.

Chính sách phát triển phần mềm

Chính sách này có quan hệ mật thiết với câu hỏi có bao nhiêu phiên bản phần mềm được vận hành cùng một lúc. Rõ ràng rằng, đây không phải là vấn đề dịch vụ của một tổ chức về phần mềm được đặt hàng mà vấn đề chính ở đây là số lượng các phiên bản hay chính là giá thành (COSTs) của những gói phần mềm được lên kế hoạch để phục vụ cho nhiều khách hàng khác nhau. Chính sách phát triển phiên bản sau này có thể được thực hiện dưới dạng “tuần tự” hoặc “cây”. Khi áp dụng chính sách tuần tự, chỉ một phiên bản được tạo ra sẵn cho toàn bộ khách hàng. Phiên bản này gồm số lượng lớn những ứng dụng mà biểu lộ sự dư thừa cao, một thuộc tính mà cho phép phần mềm phục vụ tất cả những mong muốn của khách hàng. Phần mềm phải được xem lại một cách định kỳ nhưng một khi một phiên bản mới được hoàn thành, nó sẽ thay thế phiên bản đang được sử dụng hiện thời bởi toàn bộ đối tượng người dùng.

Khi áp dụng chính sách phiên bản “cây”, nhóm bảo trì phần mềm hỗ trợ những nỗ lực marketing (quảng cáo/tiếp thị) bằng việc phát triển một phiên bản chuyên dụng, nhằm tới những nhóm khách hàng hay khách hàng chính một khi nó được yêu cầu. Một phiên bản mới được bắt đầu bằng việc thêm những ứng dụng đặc biệt hoặc bỏ qua những ứng dụng, phụ thuộc vào những gì liên quan đến nhu cầu của khách hàng. Những phiên bản này thay đổi theo sự phức tạp và mức của

ứng dụng – những ứng dụng hướng công nghiệp được hướng tới, vv ... Nếu chính sách này được chấp nhận, gói phần mềm có thể tiến triển thành một gói đa phiên bản sau vài năm của dịch vụ, có nghĩa rằng giống như một cái cây, với vài nhánh chính và nhiều nhánh phụ, từng phần nhánh đại diện cho một phiên bản đã được duyệt lại một cách chuyên dụng. Trái với phiên bản phần mềm dạng tuần tự, việc bảo trì và quản lý của phiên bản phần mềm dạng cây phức tạp và tốn thời gian hơn nhiều. Xem xét sự thiếu sót này, những tổ chức phát triển phần mềm có gắng áp dụng chính sách phiên bản dạng cây một cách giới hạn, chỉ cho phép một số lượng nhỏ những phiên bản phần mềm được phát triển.

Chính sách thay đổi

Chính sách thay đổi đề cập đến phương thức để kiểm tra mỗi yêu cầu thay đổi và tiêu chuẩn sử dụng cho những phương pháp đó. Một chính sách là rõ ràng nếu nó được thực thi bởi CCB(The Change Control Board _ Ban điều khiển thay đổi) hoặc những người được ủy quyền để phê duyệt những thay đổi đó. Chính sách cân bằng đòi hỏi có một cuộc kiểm tra tỉ mỉ những yêu cầu thay đổi , điều này là rất phù hợp vì như vậy nó cho phép nhân viên tập trung vào những thay đổi quan trọng và có ích nhờ thế chúng ta mới có thể thực thi công việc trong thời gian hợp lý và theo những chuẩn chất lượng mong muốn.

3.3.3. Các thành phần chất lượng phần mềm tiền bảo trì

Giống như những thành phần SQA tiền dự án, những hoạt động SQA trước bảo trì cần được hoàn thành để khởi tạo các dịch vụ bảo trì cần thiết cũng rất quan trọng. Nó bao gồm thứ tự các bước sau:

- Xem xét lại hợp đồng bảo trì
- Xây dựng kế hoạch bảo trì.

3.3.3.1. Xem xét lại hợp đồng bảo trì

Khi xem xét hợp đồng bảo trì, quan điểm mở rộng nên được khai quát. Đặc biệt, những quyết định được yêu cầu về các loại hình dịch vụ cần được ký kết trong hợp đồng. Những quyết định này phụ thuộc vào các loại hình dịch vụ khách hàng: dành cho khách hàng mà có gói custom-made được phát triển; dành cho khách hàng mà mua gói phần mềm COST và những khách hàng bên trong. Vì vậy, trước khi bắt đầu cung cấp những dịch vụ bảo trì phần mềm tới từng nhóm khách hàng trên thì hợp đồng bảo trì tương ứng phải được hoàn thành để nó đánh giá

tổng số những trách nhiệm bảo trì theo những điều kiện liên quan.

Giả định sự thực thi

Những dịch vụ bảo trì tới khách hàng bên trong thường không có hợp đồng. Trong một số trường hợp cụ thể, một vài dịch vụ được cung cấp trong quá trình thực hiện mà không có sự xác định rõ ràng cho những dịch vụ đó. Trong những trường hợp như vậy thì sự không hài lòng dễ xảy ra trong cả 2 phía: những khách hàng bên trong cảm thấy rằng họ mong muốn được hỏi

ý kiến một cách có thiện ý thay vì nhận những dịch vụ một cách đều đặn mà họ không mong chờ, bên cạnh đó thì nhóm phát triển lại luôn yêu cầu thực thi việc bảo trì như là một việc bắt buộc một khi họ đã làm việc trong một dự án khác.

Để tránh tình trạng căng thẳng, một “hợp đồng dịch vụ bên trong” nên được viết. Trong tài liệu này, những dịch vụ này được cung cấp bởi nhóm bảo trì bên trong tới những khách hàng bên trong được xác định một cách rõ ràng. Với việc bỏ qua hầu hết những hiểu sai liên quan đến những dịch vụ ảo, hợp đồng này có thể đáp ứng một cách cơ bản những bảo trì thỏa đáng tới những khách hàng bên trong.

Những hoạt động xem xét lại hợp đồng bảo trì bao gồm việc nhìn lại những bản nháp đề xuất cũng như là nhìn lại những bản hợp đồng nháp. Thực tế, việc nhìn lại mục tiêu và sự thực thi bản hợp đồng bảo trì là đi theo từng dòng trong việc xem xét lại bản hợp đồng tiền dự án.

Dưới đây là những mục tiêu chính trong việc xem xét lại hợp đồng bảo trì phần mềm:

- **Sự rõ ràng trong yêu cầu của khách hàng**

Những vấn đề sau đây đặc biệt được quan tâm:

- Loại dịch vụ bảo trì sửa đổi được yêu cầu: Danh sách những dịch vụ từ xa và những dịch vụ tại chỗ được cung cấp, giờ phục vụ, thời gian phản hồi,....
- Số lượng người sử dụng và kiểu ứng dụng được dùng.
- Những người dùng địa phương, ở khoảng cách xa (hoặc qua đại dương) và các kiểu dịch vụ được cài đặt tại mỗi nơi đó.
- Cung cấp bảo trì cải thiện chức năng, khả năng thích ứng và thủ tục cho những yêu cầu của dịch vụ cũng như đề xuất và phê duyệt việc

thực thi cho những dịch vụ này.

Xem xét lại những phương pháp tiếp cận khác về các điều khoản trong văn bản bảo trì

Xem xét những suy xét cụ thể dưới đây để đưa ra lựa chọn phù hợp:

- Những hợp đồng con cho những site (địa điểm) hoặc kiểu dịch vụ
- Việc thực thi một vài dịch vụ thông qua khách hàng cùng với sự hỗ trợ từ nhóm bảo trì của nhà cung cấp.

Nhìn lại sự ước lượng về tài nguyên bảo trì được yêu cầu

Đầu tiên, những ước lượng nên được kiểm tra lại trên cơ sở của những dịch vụ được yêu cầu, sắp xếp theo đề xuất của nhóm. Sau đó, dựa vào năng lực của công ty để xem có đáp ứng được những khía cạnh chuyên môn cũng như là tính sẵn sàng của nhóm bảo trì đã phân tích hay không.

Xem xét lại những dịch vụ bảo trì được cung cấp bởi những hợp đồng con và/hoặc khách hàng.

Việc nhìn lại này đề cập tới việc định danh lại những dịch vụ được cung cấp bởi mỗi người tham gia, số tiền trả cho những hợp đồng con, đảm bảo chất lượng và những thủ tục tiếp theo được đáp ứng.

Xem xét lại những ước lượng về giá thành bảo trì

Những ước lượng này nên được nhìn lại dựa trên cơ sở của những tài nguyên được yêu cầu.

3.3.3.2. Lập kế hoạch bảo trì

Lập kế hoạch bảo trì phải được chuẩn bị cho tất cả các khách hàng trong và ngoài. Những kế hoạch này nên cung cấp framework trong việc tổ chức những điều khoản bảo trì.

Lập kế hoạch đó bao gồm những bước sau:

Danh sách những dịch vụ bảo trì được ký kết

- Khách hàng bên trong và ngoài, số lượng người sử dụng, sự định vị vị trí cho mỗi site khách hàng.

- Đặc tính của những dịch vụ bảo trì sửa đổi (xa hoặc tại chỗ).
- Trách nhiệm của việc bảo trì cải thiện chức năng và khả năng thích ứng phục vụ điều khoản của mỗi khách hàng.

Mô tả tổ chức của nhóm bảo trì

Kế hoạch tổ chức nhóm bảo trì tập trung vào những yêu cầu về nhân sự. Điều đó có thể được xem xét cẩn thận theo những tiêu chuẩn sau:

- Số lượng thành viên nhóm được yêu cầu.
- Chất lượng thành viên nhóm được yêu cầu theo công việc bảo trì, bao gồm những người quen với gói phần mềm cần được bảo trì.
- Cấu trúc tổ chức của những nhóm bảo trì, bao gồm tên của những người lãnh đạo nhóm.
- Định nghĩa các công việc (trách nhiệm của khách hàng, kiểu ứng dụng,...) cho mỗi nhóm.
- Đào tạo khi cần thiết

Danh sách điều kiện thuận lợi cho bảo trì

Những điều kiện thuận lợi cho bảo trì – cơ sở hạ tầng mà có thể cung cấp dịch vụ bao gồm:

- Trung tâm hỗ trợ bảo trì với thiết bị phần cứng và phần mềm đã được cài đặt để cung cấp những dịch vụ hỗ trợ cho người dùng và sửa đổi phần mềm.
- Tài liệu chính thức chứa tập các tài liệu đã hoàn thành:
 - (1) Tài liệu phần mềm, bao gồm tài liệu phát triển.
 - (2) Những hợp đồng dịch vụ
 - (3) Cấu hình phần mềm cho mỗi khách hàng và phiên bản của những gói phần mềm được cài đặt tại mỗi địa điểm và được cung cấp bởi nhà quản lý cấu hình.
 - (4) Những thông tin bảo trì lịch sử được ghi lại cho mỗi người dùng và khách hàng.

Danh sách những rủi ro dịch vụ bảo trì được xác định

Rủi ro dịch vụ bảo trì liên quan đến hoàn cảnh mà thất bại xảy ra để cung cấp bảo trì tương ứng được dự đoán trước. Những rủi ro này bao gồm:

- Thiếu nhân viên trong suốt những dịch vụ bảo trì của tổ chức, trong trung tâm hỗ trợ bảo trì cụ thể hoặc trong những ứng dụng cụ thể.
- Sự hiểu biết và trình độ chuyên môn không tươngứng với từng phần của các gói phần mềm liên quan để thực thi những dịch vụ hỗ trợ người dùng và/hoặc những công việc bảo trì sửa đổi.
- Những thành viên nhóm không đủ khả năng để thực hiện những công việc cải thiện chức năng cũng như khả năng thích ứng.

Danh sách những thủ tục và điều khiển quản lý phần mềm được yêu cầu

Hầu hết những thủ tục được yêu cầu để cập đến những tiến trình được thực thi bởi những nhóm bảo trì sửa đổi và trung tâm hỗ trợ người dùng. Những thủ tục này giải quyết:

- Vận hành những ứng dụng của khách hàng.
- Xử lý những bản ghi lỗi của phần mềm
- Ghi chép định kỳ và liên tục những dịch vụ hỗ trợ người dùng
- Ghi chép định kỳ và liên tục những dịch vụ bảo trì sửa đổi.
- Đào tạo và cấp giấy chứng thực cho những thành viên bảo trì.

Ngân sách bảo trì phần mềm

Những ước lượng được sử dụng trong ngân sách bảo trì sửa đổi dựa trên kế hoạch tổ chức nhân sự, trình độ chuyên môn được yêu cầu và sự đầu tư vốn cần để tạo ra những điều kiện, những nhu cầu của nhóm và tác vụ khác. Họ phải được chuẩn bị mỗi khi nhân sự, trình độ chuyên môn và các thủ tục được xác định. Những ước lượng cho việc bảo trì cải thiện chất lượng và khả năng thích ứng được chuẩn bị theo sự mong đợi, và theo sự thay đổi cũng như là cải thiện khi chúng được thực hiện.

3.3.4. Các công cụ đảm bảo chất lượng bảo trì phần mềm

Sự đa dạng của các công cụ đảm bảo chất lượng phần mềm được sử dụng trong các chu kỳ thực hiện của vòng đời phần mềm. Bản chất riêng biệt của mỗi thành phần bảo trì phần mềm: bảo trì sửa đổi, bảo trì thích ứng và bảo trì cải thiện chức năng, yêu cầu các tập công cụ SQA khác nhau cần phải sử dụng cho mỗi loại. Hơn nữa, chu kỳ thực hiện của phần mềm điển hình thì mở rộng việc sử dụng

các công cụ SQA cơ sở và công cụ giám sát việc quản lý.

Ý tưởng về việc đưa SQA vào pha bảo trì được Perry đưa ra (1995). Trong 1 nghiên cứu ông thực hiện vào tháng 11 năm 1994, các bên tham gia được thông báo rằng: dựa trên kinh nghiệm của họ thì 31% kế hoạch bảo trì của họ đã được thực hiện để đảm bảo chất lượng.

Phần tiếp theo sẽ nói về các nội dung sau:

- 1) Công cụ SQA cho bảo trì sửa lỗi
- 2) Công cụ SQA cho bảo trì cải thiện chức năng
- 3) Công cụ SQA cơ sở cho bảo trì phần mềm
- 4) Công cụ SQA cho giám sát quản lý bảo trì phần mềm.

3.3.4.1. Công cụ SQA cho bảo trì sửa lỗi

Các hoạt động bảo trì sửa lỗi đưa ra đầu tiên: (a) các dịch vụ hỗ trợ người sử dụng và (b) sửa chữa phần mềm (sửa lỗi). Các dịch vụ hỗ trợ người sử dụng giải quyết các trường hợp lỗi về mã phần mềm và lỗi tài liệu, tài liệu không hoàn thành hoặc mập mờ. Chúng có thể chứa những câu lệnh mà người sử dụng không đủ kiến thức về phần mềm hoặc thất bại trong việc sử dụng tài liệu có sẵn. Các dịch vụ sửa chữa phần mềm – gỡ lỗi và sửa tài liệu được sử dụng trong trường hợp lỗi phần mềm (failure), và được cung cấp trong suốt chu kỳ khởi tạo của việc thực hiện (mặc dù sự nỗ lực này được đầu tư vào việc testing) và tiếp tục được yêu cầu mặc dù không thường xuyên lắm. Dù hai kiểu dịch vụ khác nhau nhưng tập các công cụ đảm bảo chất lượng đều tập trung vào cùng một mục đích là đảm bảo chất lượng dịch vụ. Trong nhiều trường hợp cùng một đội có thể thực hiện cả hai kiểu bảo trì sửa lỗi.

Thêm vào các công cụ SQA giám sát việc quản lý và cơ sở hạ tầng (được thảo luận ở phần sau của chương) thì hầu hết các công việc sửa lỗi yêu cầu sử dụng các công cụ SQA cho vòng đời nhỏ, mainly mini-testing (test những phần nhỏ quan trọng). Thủ tục mini-testing được yêu cầu cho việc xử lý các tác vụ về lỗi đường dẫn (repair patch) (small - scale), nó được đặc trưng bởi số lượng nhỏ dòng lệnh thay đổi cùng nhau để hoàn thành việc sửa lỗi nhanh chóng. Những vấn đề liên quan đến sửa chữa trì hoãn giống như một abridged-mini-form của testing thường được sử dụng. Tuy nhiên, sử dụng

những công cụ mini testing vẫn cần được thực hiện để tránh những trường hợp không thể test được.

Để đảm bảo chất lượng của mini testing, những chỉ dẫn sau nên được giữ vững:

- Testing được thực hiện bởi những tester chất lượng, không phải bởi người lập trình thực hiện sửa chữa
- Tài liệu thủ tục test (dài nhất là 2-3 trang) nên được chuẩn bị. Tài liệu bao gồm một bản mô tả về những ảnh hưởng biết trước được của việc sửa chữa, phạm vi sửa lỗi và một danh sách các test case được thực hiện. Tài liệu thủ tục trước test, tương tự như tài liệu thủ tục testing cũng nên được chuẩn bị để thực hiện test việc sửa lỗi được phát hiện trong test trước.
- Một bản ghi test đầy đủ viết về các lỗi phát hiện được dưới dạng tài liệu trong mỗi giai đoạn test và re-test nên được hoàn thành.
- Nhóm test chính xem lại tài liệu test về phạm vi sửa lỗi, tính đúng đắn của các test case và các kết quả test.
- Việc sửa chữa được cho là “simple và trivial”, đặc biệt đối với việc thực hiện tại site của khách hàng, mini-testing có thể được tránh đi.

Các dịch vụ bảo trì hợp đồng con, đặc biệt là các dịch vụ hỗ trợ người sử dụng đã trở nên khá phổ biến bát cứ khi nào có quá nhiều vấn đề hoặc không kinh tế cho nhà thầu bảo trì cung cấp các dịch vụ đó một cách trực tiếp. Công cụ chính để đảm bảo chất lượng của các dịch vụ bảo trì của nhà thầu phụ và mở đường cho các quan hệ suôn sẻ là hợp đồng contractor-subcontractor. Công cụ SQA được tích hợp vào hợp đồng nhằm mục đích:

- (i) Các thủ tục xử lý phân loại cụ thể các cuộc gọi bảo trì.
- (ii) Tài liệu đầy đủ về các thủ tục dịch vụ.
- (iii) Có sẵn các bản ghi được viết tài liệu kiểm thử chuyên nghiệp của thành viên đội bảo trì của nhà thầu phụ, cho nhà thầu xem xét lại.
- (iv) Cấp giấy phép cho nhà thầu thực hiện xem xét lại theo chu kỳ các dịch vụ bảo trì cũng như sự thỏa mãn của khách hàng.
- (v) Các điều kiện liên quan đến chất lượng thì yêu cầu chặt chẽ về hình phạt và sự kết thúc của ràng buộc hợp đồng con trong trường hợp

cực đoan.

Một khi bảo trì sẵn sàng thực hiện, nhà thầu nên quản lý đều đặn việc xem lại dịch vụ bảo trì và sự thỏa mãn của khách hàng.

3.3.4.2. Các công cụ SQA cho bảo trì cải thiện chức năng

Do sự giống nhau của việc bảo trì cải thiện chức năng và dự án phát triển phần mềm, các công cụ vòng đời dự án được áp dụng cho bảo trì cải thiện chức năng. Những công cụ này cũng được thực thi cho bảo trì thích ứng phạm vi rộng (large scale adaptive maintenance).

Các công cụ SQA mở rộng được thực thi cho bảo trì cải thiện chức năng là các công cụ điều khiển quản lý và cơ sở hạ tầng, được thảo luận thêm ở phần sau.

3.3.4.3. Các thành phần cơ sở hạ tầng SQA cho bảo trì phần mềm

Các công cụ cơ sở đảm bảo chất lượng phần mềm à các thành phần quan trọng của bảo trì phần mềm. Sự cân đối của mảng các công cụ SQA cơ sở là bản chất và được thực thi trong suốt vòng đời của hệ thống phần mềm. Hơn nữa, sự giống nhau của các tiến trình cải thiện chức năng phần mềm và phát triển phần mềm cho phép các tiến trình dùng chung các công cụ SQA cơ sở và chỉ có thay đổi nhỏ. Các công cụ cơ sở chuyên dụng được yêu cầu cho các hoạt động bảo trì sửa lỗi do các đặc trưng đặc biệt của các hoạt động này. Các hoạt động bảo trì thích ứng được phục vụ bởi các công cụ SQA cơ sở, theo các đặc trưng của nó. Các công cụ được áp dụng thường xuyên nhất là các công cụ SQA cải thiện chức năng, được theo bởi các công cụ SQA bảo trì sửa lỗi.

Thực tế, sự đóng góp của các công cụ SQA cơ sở cho việc bảo trì không bắt đầu tại thời điểm ban đầu của các tiến trình bảo trì. Rõ ràng là việc áp dụng thích hợp các công cụ SQA cơ sở bởi đội phát triển phần mềm đóng góp căn bản vào hiệu quả và hiệu năng các hoạt động của đội bảo trì. Nói cách khác, những công cụ này góp phần vào việc bảo trì đảm bảo chất lượng theo 2 cách: đâu tiên, bằng cách hỗ trợ đội phát triển phần mềm khi tạo ra phần mềm chất lượng cao, và thứ hai, bằng cách hỗ trợ đội bảo trì đáp ứng việc bảo trì của sản phẩm phần mềm giống nhau.

Các công cụ SQA cơ sở chuyên dụng được yêu cầu cho các tiến trình bảo trì phần mềm, đặc biệt là bảo trì sửa lỗi, thể hiện các đặc trưng đặc biệt. Ở đây chúng ta tập trung vào các công cụ SQA cơ sở chuyên dụng của các lớp sau:

- Các thủ tục bảo trì và các chỉ dẫn làm việc
- Hỗ trợ các thiết bị chất lượng
- Đào tạo và cấp chứng chỉ cho đội bảo trì
- Các hoạt động ngăn ngừa và sửa lỗi
- Quản lý cấu hình
- Viết tài liệu và điều khiển bản ghi chất lượng

Các thủ tục bảo trì và chỉ dẫn làm việc

Hầu hết các thủ tục bảo trì và các chỉ dẫn làm việc chuyên dụng được áp dụng cho các hoạt động bảo trì sửa lỗi và hỗ trợ người sử dụng, ví dụ:

- Xử lý từ xa các yêu cầu cho dịch vụ trong trường hợp lỗi phần mềm
- Xử lý tại chỗ các yêu cầu của khách hàng cho dịch vụ trong trường hợp phần mềm lỗi.
- Dịch vụ hỗ trợ người sử dụng
- Điều khiển đảm bảo chất lượng cho các hoạt động sửa lỗi phần mềm và hộ trợ người dùng
- Điều tra sự thỏa mãn khách hàng.
- Cấp chứng chỉ cho các thành viên đội bảo trì sửa lỗi và hỗ trợ người sử dụng.

Các thiết bị hỗ trợ chất lượng

Bảo trì được mong đợi để phát triển các thiết bị chuyên dụng hỗ trợ các hoạt động sửa lỗi phần mềm và hỗ trợ người dùng: templates(khuôn mẫu), checklists (danh sách kiểm tra) và những cái tương tự thế. Các thiết bị có thể bao gồm:

- Danh sách kiểm tra các phần có thể gây ra lỗi – được áp dụng bởi nhà chuyên môn về bảo trì
- Các khuôn mẫu báo cáo lỗi phần mềm được giải quyết như thế nào, bao gồm sự phát hiện của các tiến trình sửa lỗi
- Danh sách kiểm tra cho việc chuẩn bị tài liệu thủ tục testing nhỏ.

Đào tạo và cấp chứng chỉ cho đội bảo trì

Đào tạo đội bảo trì giải quyết các công việc cải thiện chức năng không khác mấy so với việc đào tạo đội phát triển phần mềm khác. Tuy nhiên, đào tạo đặc biệt và cấp chứng chỉ mang tính cốt yếu cho đội bảo trì sửa lỗi.

Việc đào tạo các chuyên gia bảo trì sửa lỗi được thúc đẩy bởi yêu cầu hỗ trợ các dịch vụ cụ thể trong hợp đồng bảo trì. Do vậy, kế hoạch đào tạo nên cung cấp các giải pháp cho các yêu cầu nhân sự trong suốt các chu kỳ trọng tải cao nhất và các yêu cầu của tổ chức để thay thế nhân sự bị sa thải. Trong nhiều trường hợp, việc đào tạo các nhân viên bảo trì dự trữ là không đủ, phải đào tạo thêm hệ thống riêng biệt. Nói cách khác, chương trình đào tạo nghiêm ngặt được yêu cầu để cho phép tổ chức xác định phạm vi với mức độ ràng buộc của các dịch vụ riêng biệt cho các chu kỳ trọng tải cao và trong trường hợp thay đổi nhân viên bảo trì hoặc bất kỳ lý do nào khác.

Yêu cầu cấp chứng chỉ cho các nhân viên sửa lỗi phần mềm và hỗ trợ người dùng là gốc rễ trong các đặc trưng của các dịch vụ này. Sự quan tâm đặc biệt nên dành cho việc cấp chứng chỉ cho các nhân viên sửa lỗi phần mềm, người mà thường xuyên thực hiện công việc của họ dưới áp lực về thời gian lớn, làm việc 1 mình, và trong nhiều trường hợp làm việc tại site của khách hàng, nơi mà sự hỗ trợ chuyên gia từ team leader hoặc những người khác bị giới hạn.

Các hoạt động ngăn ngừa và sửa lỗi

Pha hoạt động của vòng đời phần mềm tạo ra thông tin có giá trị cao: các bản ghi lỗi phần mềm, sửa chữa các lỗi đó cũng như bản ghi các yêu cầu hỗ trợ khách hàng có thể dẫn tới các hoạt động ngăn ngừa và sửa lỗi do đó góp phần cải thiện hệ thống phần mềm đã có và phần mềm mới.

Để các tiến trình hoạt động hiệu quả, cần có các tiến trình thích hợp cho việc biểu diễn thông tin thu thập được, xem lại và phân tích những phát hiện, và đưa ra những gợi ý cho việc cải thiện các tiến trình bảo trì và phát triển liên quan. Những hoạt động SQA này được chỉ dẫn và điều khiển bởi ủy ban nội bộ - CAB (Corrective Action Board), được sáng lập trong tổ chức phát triển phần mềm chính.

Những vấn đề điển hình ban xúc tiến cần xem lại gồm:

Những thay đổi về nội dung và tính thường xuyên của khách hàng yêu cầu về các dịch vụ hỗ trợ người dùng.

Tăng thời gian trung bình được đầu tư để tuân theo yêu cầu hỗ trợ của khách hàng.

Tăng thời gian trung bình được đầu tư để sửa chữa các lỗi phần mềm của khách hàng.

Tăng phần trăm các lỗi hiệu chỉnh phần mềm.

Quản lý cấu hình

Nhóm bảo trì đưa ra hầu hết tập các phụ thuộc vào quản lý cấu hình. Sự phụ thuộc này là kết quả của các mối quan hệ dựa trên kinh nghiệm với các gói phần mềm dịch vụ qua nhiều năm, trong suốt các phiên bản mới được thêm vào, phiên bản cũ bị thay thế và nhiều sự cài đặt mới và các thay đổi phần mềm được thực hiện.

Hai ứng dụng chung dựa vào quản lý cấu hình là: (1) sửa lỗi và (2) sự thay thế nhóm (group replacement) của các phiên bản phần mềm đang được sử dụng bằng các phiên bản mới, được khởi tạo bởi tổ chức bảo trì.

1. Sửa lỗi (Failure repair): Trong vấn đề về sửa lỗi phần mềm, việc hỗ trợ cập nhật và độ tin cậy là cần thiết theo dạng của:

- Thông tin quan tâm tới phiên bản hệ thống phần mềm được cài đặt tại site của khách hàng.

- Bản copy code hiện thời và tài liệu của nó.

Sự đóng góp vào chất lượng phần mềm được thể hiện là lỗi ít hơn trong các ~~lỗi~~ kinh nghiệm hiệu chỉnh lỗi và giảm bớt được tài nguyên đầu tư vào sửa lỗi.

2. Group replacement (Thay thế nhóm): Thuật ngữ group trong SQA để cập đến tất cả khách hàng có cùng phiên bản phần mềm được cài đặt tại site của họ. Do vậy, group replacement chỉ rõ rằng tất cả khách hàng sử dụng phiên bản sẽ được nhận phiên bản mới được phát triển hoặc cập nhật tại cùng 1 thời gian. Quản lý cấu hình hỗ trợ cho vấn đề thay thế nhóm dựa trên thông tin về các thành viên của nhóm khách hàng

- Đưa ra quyết định về tính thích hợp của việc thực hiện thay thế nhóm dựa trên quy mô của việc thay thế và kiểu hợp đồng đã ký với khách hàng.

- Lên kế hoạch thay thế nhóm, phân phối tài nguyên và xác định thời gian.

Sự đóng góp vào chất lượng phần mềm thể hiện ở việc thay thế phiên bản phần mềm hiện thời bởi những phiên bản đã được cải thiện mà giảm lỗi phần mềm, yêu cầu ít sự hỗ trợ. Cải thiện chất lượng cũng góp phần bảo trì phần mềm hiệu quả, yêu cầu ít tài nguyên cho bảo trì sửa lỗi.

Tài liệu bảo trì và bản ghi chất lượng

Những yêu cầu cụ thể cho tài liệu và bản ghi chất lượng có quan hệ chặt chẽ nhất với các hoạt động sửa lỗi phần mềm và hỗ trợ người sử dụng. Tài liệu và bản ghi chất lượng được chuẩn bị để:

- Cung cấp dữ liệu cần thiết cho các hoạt động ngăn ngừa và sửa lỗi.
- Hỗ trợ việc xử lý các yêu cầu hỗ trợ người dùng và các báo cáo lỗi khách hàng trong tương lai.
- Cung cấp bằng chứng để đáp ứng những yêu sách từ phía khách hàng trong tương lai.

Những yêu cầu tài liệu được liệt kê trong các thủ tục bảo trì khác nhau nên đáp ứng với tất cả những yêu cầu tài liệu trên

3.3.4.4. Những công cụ SQA giám sát quản lý cho bảo trì phần mềm

Trong khi các công cụ SQA giám sát quản lý cụ thể được yêu cầu cho các hoạt động bảo trì sửa lỗi, thì sự giống nhau của các tiến trình phần mềm mô tả việc cải thiện chức năng và bảo trì thích ứng cũng như việc phát triển phần mềm cho phép những tiến trình này được sử dụng cùng một công cụ quản lý. Đặc biệt, các thành phần SQA quản lý có ý nghĩa giúp cải thiện việc điều khiển bảo trì bằng cách tạo ra những báo động sớm tín hiệu làm giảm chất lượng của các dịch vụ và tăng tỷ số thất bại dịch vụ.

Phần còn lại của mục này sẽ đưa ra những vấn đề điều khiển quản lý chất lượng, những phần chính này đạt tới dịch vụ sửa lỗi phần mềm và hỗ trợ người dùng ở trên:

Điều khiển hiệu năng cho các dịch vụ bảo trì sửa lỗi.

Đo chất lượng cho bảo trì sửa lỗi.

Chi phí chất lượng bảo trì phần mềm.

Điều khiển hiệu năng cho các dịch vụ bảo trì sửa lỗi

Điều khiển hiệu năng quản lý cho các dịch vụ bảo trì sửa lỗi khác với khi áp dụng các dịch vụ sửa lỗi phần mềm và các dịch vụ hỗ trợ người dùng. Các công cụ điều khiển quản lý mang lại, bên cạnh thông tin hiệu năng theo chu kỳ, còn là *những báo động sự chú ý quản lý*, như sau:

- **Sửa lỗi phần mềm**

- Tăng việc sử dụng tài nguyên
- Giảm tỉ lệ sửa chữa lỗi từ xa so với sửa chữa on-site của khách hàng.
- Tăng tỷ lệ sửa chữa on-site ở những vùng cục bộ ở xa so với các dịch vụ hải ngoại
- Tăng phần trăm thất bại gấp phải khi sửa chữa yêu cầu lập lịch
- Tăng tỉ lệ sửa lỗi, và liệt kê các trường hợp “model” cụ thể của các ~~thuống~~ lỗi cực đoan.
- Giảm bớt sự thỏa mãn khách hàng dựa trên các điều tra sự thỏa mãn ~~ủa~~ khách hàng.

- **Hỗ trợ người sử dụng**

- Tăng tỉ lệ các yêu cầu dịch vụ cho các hệ thống phần mềm cụ thể, cho các kiểu dịch vụ,...
- Tăng tài nguyên được sử dụng cho các dịch vụ hỗ trợ người dùng
- Tăng tỉ lệ thất bại của việc cung cấp các dịch vụ được yêu cầu
- Tăng tỉ lệ lỗi, và trường hợp cụ thể của các thất bại đáng chú ý
- Thông tin thỏa mãn khách hàng dựa trên các nghiên cứu sự thỏa mãn của khách hàng

Những điều khiển sửa chữa lỗi quản lý này (những cái được mong đợi để đưa ra những cảnh báo) được thực hiện qua báo cáo theo chu kỳ, qua buổi họp nhân viên đều đặn, qua việc xem xét các trung tâm hỗ trợ bảo trì cung cấp dịch vụ, qua phân tích báo cáo xử lý độ đo bảo trì phần mềm và giá thành chất lượng bảo trì. Thông tin được tích lũy hỗ trợ các quyết định quản lý quan tâm tới kế hoạch và việc thực hiện bảo trì sửa lỗi.

Đo chất lượng bảo trì phần mềm

Đo chất lượng bảo trì phần mềm được sử dụng chủ yếu để nhận biết khuynh hướng trong hiệu quả bảo trì, hiệu năng và sự thỏa mãn của khách hàng. Đơn vị đảm bảo chất lượng phần mềm thường thực hiện các tiến trình đo chất lượng. Thay đổi xu hướng tích cực hay tiêu cực, cung cấp cơ sở định lượng cho các quyết định quản lý, quan tâm tới:

❖ Ước lượng các yêu cầu tài nguyên khi lên kế hoạch bảo trì sửa chữa cho chu kỳ tiếp theo

❖ So sánh các phương thức thực hiện

❖ Khởi tạo các hoạt động ngăn ngừa và sửa lỗi

❖ Ước lượng các yêu cầu tài nguyên như một cơ sở cho các đề xuất các dịch vụ bảo trì mới hoặc đã được điều chỉnh.

Chi phí chất lượng phần mềm

Như trong những phần trước, ở đây chúng ta chỉ quan tâm tới vấn đề bảo trì sửa lỗi. Giá thành chất lượng của bảo trì sửa lỗi được phân loại thành 6 lớp. Sau đây là định nghĩa cho mỗi lớp và ví dụ:

- Chi phí của việc ngăn ngừa: Chi phí ngăn ngừa lỗi, ví dụ như chi phí của việc xây dựng và đào tạo đội bảo trì, chi phí của các hoạt động ngăn ngừa và sửa lỗi.
- Chi phí của việc đánh giá chất lượng: chi phí của việc phát hiện lỗi, ví dụ như chi phí của việc xem xét lại các dịch vụ bảo trì được thực hiện bởi nhóm SQA, đội bên trong và các điều tra sự thỏa mãn của khách hàng.
- Chi phí của việc chuẩn bị và giám sát quản lý: Chi phí của các hoạt động quản lý được thực hiện để ngăn ngừa lỗi, ví dụ như chi phí của việc chuẩn bị kế hoạch bảo trì, việc tuyển đội bảo trì và bám sát hiệu năng bảo trì (follow-up of maintenance performance)
- Chi phí lỗi bên trong: Chi phí của lỗi phần mềm được đề xướng bởi đội bảo trì (ưu tiên cho các yêu sách nhận được từ phía khách hàng)
- Chi phí lỗi bên ngoài: Chi phí của các lỗi phần mềm được đề xướng bởi các yêu sách của khách hàng.
- Chi phí của lỗi quản lý: Chi phí lỗi phần mềm gây ra bởi các hoạt động quản lý hoặc tương tác, ví dụ như chi phí của các thiệt hại từ việc thiếu nhân viên bảo trì hoặc bảo trì không chính xác.

Sau khi xem lại các lớp định giá thành chất lượng phần mềm, được định nghĩa theo các lớp cũng như mô hình mở rộng, sẽ được thảo luận kỹ hơn ở chương

Giá thành lỗi bên trong của các hoạt động bảo trì sửa lỗi phần mềm

Để định nghĩa giá thành lỗi bên trong, có hai chu kỳ bảo trì phải được quan tâm riêng biệt. Đó là: (a) chu kỳ đảm bảo (warranty) (thường từ 3 đến 12 tháng sau khi cài đặt phần mềm) và (b) chu kỳ các dịch vụ bảo trì được ký hợp đồng, bắt đầu ở thời điểm kết thúc chu kỳ đảm bảo. Vấn đề ở đây yêu cầu một quyết định trong trường hợp nào nên quan tâm đến lỗi bên ngoài, chỉ sau khi đưa ra quyết định này thì việc định giá chất lượng mới được xác định và ước lượng. Những định nghĩa được đề xuất về chi phí lỗi bên ngoài và các tham số của chúng cho các dịch vụ sửa lỗi phần mềm và hỗ trợ người dùng dưới đây:

- **Cho sửa lỗi phần mềm:**

- Tất cả chi phí của sửa lỗi phần mềm được đưa ra bởi người sử dụng trong suốt chu kỳ đảm bảo là chi phí chất lượng bên ngoài bởi vì chúng đề cập tới kết quả trực tiếp từ lỗi phát triển phần mềm, hơn nữa người phát triển có trách nhiệm cho việc sửa lỗi của họ trong suốt chu kỳ này.
- Sửa lỗi phần mềm được thực hiện trong suốt chu kỳ bảo trì được ký kết, nó được xem như là một phần chính của dịch vụ, khi trách nhiệm của người phát triển đối với việc sửa lỗi bị giới hạn ở chu kỳ đảm bảo. Chi phí của các dịch vụ này được xem là chi phí các dịch vụ chính thức chứ không phải là chi phí chất lượng..
- Trong suốt chu kỳ bảo trì kí kết, sau những nỗ lực của việc sửa lỗi ban đầu thì chỉ có chi phí cho việc sửa lỗi lại (re-correction) mới được xem là chi phí của lỗi bên ngoài.

- **Cho các dịch vụ hỗ trợ người dùng:**

- Trong suốt chu kỳ đảm bảo, các dịch vụ hỗ trợ người dùng được xem như là một phần của nỗ lực làm theo chỉ dẫn (instruction effort), và do vậy không nên quan tâm tới chi phí lỗi bên ngoài.
- Suốt chu kỳ bảo trì được kí kết, tất cả các kiểu dịch vụ hỗ trợ người dùng, dù là giải quyết một lỗi phần mềm được xác định hay bàn (consultation) về các tùy chọn ứng dụng, là tất cả các phần của dịch

vụ chính thức, và chi phí của chúng không quan tâm tới chi phí lỗi bên ngoài.

- Trong cả hai chu kỳ bảo trì, một lỗi bên ngoài được định nghĩa như một trường hợp nơi sự bàn bạc thứ hai (second consultation) được yêu cầu sau khi sự bàn bạc ban đầu chứng minh là chính xác. Chi phí cung cấp cho sự bàn bạc thứ hai và kỹ hơn cho cùng một trường hợp được xem như là chi phí lỗi bên ngoài.

Như trường hợp tổng quát, thông tin chi phí bảo trì chất lượng, cùng với thông tin giám sát quản lý khác được mong đợi nhằm giúp đỡ cho việc quản lý đưa ra quyết định, được quan tâm như sau:

- Chỉ dẫn cho việc đầu tư trong việc cải thiện các dịch vụ bảo trì bằng cách đưa ra điểm yếu của chi phí chất lượng cực cao và điểm mạnh của chi phí chất lượng cực thấp.
- Phát triển phiên bản được cải tiến của phần mềm (trong trường hợp phần mềm được làm cho khách hàng) hoặc việc thay thế gói phần mềm đã được mua.

3.4. Các CASE tool và ảnh hưởng của nó lên chất lượng phần mềm

3.4.1. Khái niệm CASE tool

Định nghĩa: Công cụ CASE là những công cụ máy tính để phát triển phần mềm mà chúng hỗ trợ người phát triển thực hiện một hoặc nhiều pha trong vòng đời của phần mềm và/hoặc hỗ trợ việc bảo trì phần mềm.

Theo định nghĩa này thì những bộ biên dịch, hệ thống gỡ lỗi tương tác, hệ thống quản lý cấu hình và hệ thống kiểm thử tự động cũng được coi là công cụ CASE. Nói cách khác, những công cụ máy tính hỗ trợ việc phát triển phần mềm đã có từ lâu (ví dụ như những chương trình gỡ lỗi tương tác, bộ biên dịch hay hệ thống kiểm soát tiến trình dự án) có thể được xem là công cụ CASE cổ điển (*classic CASE tool*), trong khi đó những công cụ mới hỗ trợ người phát triển thực thi thành công các pha phát triển của dự án được xem là những công cụ CASE thực (*real CASE tool*). Khi nhắc đến những công cụ CASE thực ta cần phải phân biệt giữa công cụ CASE bên trên (*upper CASE tool*) hỗ trợ pha phân tích và thiết kế với công cụ CASE bên dưới (*lower CASE tool*) hỗ trợ cho pha *coding* (trong đó “bên trên” và “bên dưới” liên quan đến vị trí các pha trong mô hình thác nước), và công

cụ CASE tích hợp hỗ trợ pha phân tích, thiết kế và coding.

Thành phần chính của công cụ CASE thực là một kho chứa (*repository*) chứa tất cả các thông tin liên quan đến dự án. Thông tin dự án được tích luỹ trong kho chứa và được cập nhật mỗi khi có sự thay đổi trong suốt quá trình phát triển phần mềm và trong cả giai đoạn bảo trì. Kho chứa của pha phát triển phía trước sẽ là cơ sở cho pha tiếp theo. Thông tin phát triển tích luỹ được chứa trong kho chứa cung cấp sự hỗ trợ cho giai đoạn bảo trì trong đó những công việc bảo trì sửa đổi, đáp ứng hay cải thiện chức năng được thực hiện. Hệ thống máy tính quản lý kho chứa đảm bảo thông tin dự án được duy trì và phù hợp với phương pháp luận phát triển mềm cũng như là đảm bảo các quy chuẩn dựa theo phong cách (*style*), các thủ tục xây dựng và chỉ dẫn công việc. Điều đó làm các công cụ CASE có khả năng đưa ra tài liệu dự án đầy đủ và cập nhật bất cứ lúc nào. Một vài công cụ CASE bên dưới và CASE tích hợp có thể tự động sinh ra mã chương trình dựa hoàn toàn trên thông tin thiết kế được lưu trong kho chứa. Những công cụ kỹ nghệ ngược (Reverse engineering tool) cũng được xem là các công cụ CASE thực. Dựa trên mã hệ thống, các công cụ này được áp dụng chính cho việc khôi phục và tái tạo tài liệu thiết kế cho hệ thống phần mềm đang được sử dụng (phần mềm “kế thừa”). Nói cách khác, những công cụ CASE kỹ nghệ ngược này hoạt động ngược với công cụ CASE thông thường: thay vì tạo ra mã hệ thống dựa trên thông tin thiết kế, chúng tạo ra kho chứa đầy đủ, cập nhật và tài liệu thiết kế dựa trên mã hệ thống.

Sự hỗ trợ của các công cụ CASE cho người phát triển hệ thống được thể hiện trong bảng sau:

Kiểu công cụ CASE	Hỗ trợ
Sửa đổi và biểu đồ	Sửa đổi văn bản và biểu đồ, tạo biểu đồ thiết kế dựa trên các bản ghi trong kho chứa
Truy vấn kho chứa	Hiển thị các phần của văn bản thiết kế, biểu đồ... theo dõi yêu cầu

Viết tài liệu tự động	Tự động sinh ra các tài liệu được yêu cầu theo các bản ghi cập nhật trong kho chứa
Hỗ trợ thiết kế	Việc sửa đổi thiết kế được ghi lại bởi người phân tích hệ thống và quản lý từ điển dữ liệu
Sửa đổi mã	Biên dịch, thông dịch và gỡ lỗi tương tác mã cho ngôn ngữ lập trình cụ thể hoặc công cụ phát triển
Sinh mã	Chuyển từ các bản ghi thiết kế sang mẫu hoặc ứng dụng phần mềm thích hợp với ngôn ngữ lập trình (hoặc công cụ phát triển)
Quản lý cấu hình	Quản lý phiên bản của tài liệu thiết kế và mã phần mềm, kiểm soát sự thay đổi trong thiết kế và mã phần mềm.
Kiểm thử phần mềm	Kiểm thử tự động, kiểm thử tải và quản lý kiểm thử và sửa bản ghi
Kỹ nghệ ngược	Xây dựng kho chứa phần mềm và tài liệu thiết kế, dựa trên mã của hệ thống phần mềm thừa kế. Khi kho chứa của phần mềm đã có, nó có thể được cập nhật và sử dụng để tự động tạo ra phiên bản mới của hệ thống
Quản lý dự án và chỉ số phần mềm	Hỗ trợ việc quản lý dự án bằng cách theo

	dõi lịch và tính toán năng suất, chỉ số lỗi
--	---------------------------------------------

3.4.2. Đóng góp của CASE tool cho chất lượng sản phẩm phần mềm

Lợi ích của các công cụ CASE đối với chất lượng sản phẩm phần mềm đó là làm giảm một số lượng lớn các lỗi trong các pha phát triển phần mềm. Để đánh giá được lợi ích này, chúng ta cùng xem xét việc cải tiến chất lượng mà các công cụ CASE thực hiện được trong 9 nguyên nhân gây ra lỗi (được liệt kê trong phần 2.3). Đánh giá của chúng ta sẽ bao gồm cả công cụ CASE cổ điển và thực.

Bảng sau sẽ liệt kê những đóng góp của công cụ CASE tới việc cải thiện chất lượng phần mềm:

Nguyên nhân gây ra lỗi	Công cụ CASE cổ điển	Công cụ CASE thực
1. Lỗi trong xác định yêu cầu		Hầu như không đóng góp Việc kiểm tra tính cố định của yêu cầu hay sự chính xác bằng máy tính gần như hiếm xảy ra.
2. Việc thất bại trong giao tiếp giữa khách hàng và người phát triển		Hầu như không đóng góp Trong hầu hết các trường hợp, việc xác định sự thất bại trong giao tiếp bằng máy tính là không thể. Những thất bại này chỉ có thể được xác định và ngăn chặn khi có thay đổi hoặc khi những thông tin khác được tìm thấy trên môi

		thuần với những thông tin đã có.
3. Sự chênh lệch có chủ ý trong yêu cầu phần mềm		<p>Đóng góp lớn</p> <p>Dựa trên những thông tin đã có, những chênh lệch từ yêu cầu được xác định như những mẫu thuần và được liệt vào lỗi. Những chênh lệch này có thể được xác định bằng các công cụ theo dõi yêu cầu đã có và công cụ truy vấn (<i>cross-referenced query tools</i>).</p>
4. Những lỗi trong thiết		Đóng góp lớn

kế theo logic		<ul style="list-style-type: none"> Việc thiết kế lại (<i>re-engineering</i>) cho phép sản sinh tự động bản thiết kế cho những hệ thống kế thừa và những bản ghi của chúng. Việc sử dụng nơi lưu trữ nhằm xác định những thiếu sót trong thiết kế, những thay đổi và những mâu thuẫn mới có với những bản ghi đã có (bản ghi đã được lưu trữ từ trước).
5. Lỗi trong coding	<p>Đóng góp rất lớn</p> <p>Áp dụng những trình biên dịch (<i>compiler, interpreters</i>) và những trình gỡ lỗi tương tác (<i>debugger</i>) vào.</p>	<p>Đóng góp rất lớn</p> <p>Áp dụng các công cụ <i>CASE</i> bên dưới vào nhằm tạo ra một cách tự động những đoạn code tương thích hoàn toàn với thiết kế có sẵn. Thêm vào đó, vì code được tự động nên không có lỗi nào xảy ra.</p>
6. Không làm đúng với những chỉ dẫn về code và viết tài liệu	<p>Đóng góp có giới hạn</p> <p>Sử dụng những trình biên tập văn bản và kiểm định <i>code</i> để hỗ trợ chuẩn hóa cấu trúc, phong cách của văn bản, <i>code</i> và làm việc xác định những sai lệch</p>	<p>Đóng góp rất lớn</p> <p>Áp dụng những công cụ <i>CASE</i> bên dưới vào nhằm tạo ra những đoạn <i>code</i> đảm bảo làm đúng với những chỉ dẫn về <i>code</i> và viết tài liệu.</p>

	<p>không đúng dễ dàng hơn.</p>	
7. Những thiếu sót xảy ra trong quá trình test	<p>Đóng góp lớn Những công cụ kiểm tra tự</p>	<p>Đóng góp lớn Áp dụng những công cụ</p>

	động thực hiện hồi quy và kiểm tra việc nạp một cách tự động. Việc quản lý test và sửa lỗi bằng máy tính làm giảm lỗi.	CASE bên dưới, đặc biệt là những công cụ CASE được tích hợp vào để ngăn chặn lỗi trong <i>coding</i> và giảm lỗi thiết kế. Ứng dụng của những công cụ lưu trữ tới việc sửa lỗi và thay đổi trong suốt quá trình phát triển nhằm ngăn ngừa hầu hết các lỗi của phần mềm.
8. Lỗi thủ tục	Đóng góp lớn Việc điều khiển các phiên bản, xem xét lại và cài đặt phần mềm bằng các công cụ quản lý quản lý cấu hình.	Đóng góp có giới hạn Sử dụng những bản tài liệu đầy đủ và luôn cập nhật làm việc ngăn ngừa các lỗi bảo trì gây ra bởi những tài liệu không đầy đủ hoặc thiếu chính xác, đặc biệt nếu thiết kế đã được sửa lại nhiều lần.
9. Lỗi tài liệu	Đóng góp có giới hạn Áp dụng duy nhất trình biên tập văn bản vào.	Đóng góp lớn Sử dụng nơi lưu trữ một cách tự động nhằm sinh ra những bản tài liệu đầy đủ và luôn cập nhật trước mỗi lần sửa lỗi và thay đổi.

3.4.3. Đóng góp của CASE tool cho chất lượng bảo trì phần mềm

Các công cụ CASE (đặc biệt là công cụ CASE thực) có mặt trong nhiều loại của chất lượng bảo trì phần mềm theo nhiều cách khác nhau.

Bảo trì sửa chữa (Corrective maintenance):

- Tài liệu của phần mềm được đã được cập nhật và CASE được đưa ra đầy đủ sẽ giúp tìm ra nguyên nhân gây lỗi (*failure*) của phần mềm một cách dễ dàng và chính xác hơn.
- Các câu truy vấn *cross-referenced* cho phép xác định trước kết quả của kế hoạch sửa chữa đang đề ra một cách tốt hơn.
- Sửa chữa bằng các công cụ CASE tích hợp hay bên dưới hỗ trợ coding tự động mà sẽ không có lỗi (*error*) lập trình nào cũng như tài liệu tự động của việc sửa chữa.

Bảo trì thích nghi (*Adaptive maintenance*):

- Tài liệu phần mềm đầy đủ và được cập nhật bởi các công cụ CASE cho phép xem xét kỹ lưỡng khả năng thích nghi của gói phần mềm đối với ứng dụng mới, người dùng mới. **Bảo trì cải thiện chức năng (*Functional improvement maintenance*):**
- Việc sử dụng kho chúa cho phép những người thiết kế có thể đảm bảo tính nhất quán của các ứng dụng mới, các cải tiến mới với các hệ thống phần mềm vốn có.
- Các câu truy vấn kho chúa *cross-referenced* cho phép lên kế hoạch cho việc thay đổi, thêm chức năng một cách dễ dàng hơn
- Các thay đổi và việc thêm các chức năng thực hiện bằng các công cụ CASE tích hợp hay bên dưới cho phép *coding* tự động mà không có bất cứ lỗi (*error*) *coding* nào cũng như tài liệu tự động của các thay đổi và sửa chữa.

3.4.4. Đóng góp của CASE tool cho quản lý dự án

- (1) Phương pháp sử dụng CASE nâng cao mang lại tính kinh tế cao hơn phương pháp thông thường.
- (2) Chất lượng của việc quản lý cả hai dự án là giống nhau với việc ước lượng lịch biểu và tài nguyên dưới mức yêu cầu.

Thông thường, việc áp dụng các công cụ CASE được mong đợi sẽ làm giảm giá thành và thời gian phát triển của dự án (“*shorter time to market*”). Tuy nhiên, vai trò của các công cụ CASE đối với các khía cạnh chất lượng của quản lý dự án (gồm: điều khiển chi phí và thời gian) là trọng điểm sự quan tâm của chúng ta. Hiện nay, đã có trường hợp sử dụng công cụ CASE hiện đại làm giảm độ lệch giữa

kinh phí thực thi và lịch biểu theo kế hoạch, đặc biệt bởi vì chúng ngăn ngừa lượng lỗi (*error*) và cho phép sửa lỗi nhanh, dễ dàng hơn khi có yêu cầu. Để việc quản lý dự án được cải thiện tốt hơn, công cụ điều khiển dự án (ở đây được xem là loại của các công cụ CASE cổ điển) và các phương pháp luận ước lượng thời gian, kinh phí được cải thiện phải được phát triển.

3.5. Đảm bảo chất lượng phần mềm của các yếu tố bên ngoài cùng tham gia

3.5.1. Những thành phần bên ngoài đóng góp vào dự án phần mềm

Những người tham gia một dự án phát triển phần mềm – tổ chức quan tâm đến hệ thống phần mềm(khách hàng) và tổ chức cam kết tiến hành phát triển (nhà thầu) – ngày nay thường không chỉ là những người tham dự trong dự án. Những người tham gia bên ngoài có liên quan đến dự án phát triển phần mềm đóng góp cho dự án nhưng không phải là nhà thầu, mà cũng không phải là những thành viên của nhà thầu. Sự đóng góp của họ cho dự án được cấu trúc thông qua những thỏa thuận với nhà thầu (nhà thầu phụ hay những người cung cấp của COTS software) hoặc thông qua những điều khoản của hợp đồng dự án, các phần của dự án sẽ được thực thi bởi chính khách hàng của họ. Dự án lớn hơn và phức tạp hơn, có ý nghĩa hơn có thể đúng mà những người tham gia bên ngoài được yêu cầu, phần lớn công việc được chuyển giao hoặc chia ra. Mục đích để chuyển hướng những người tham gia bên ngoài dựa vào một số nhân tố, xác định khoảng cách từ kinh tế đến kỹ thuật và đến những cá nhân liên quan (to personnel – related interests), và mang lại một sự gia tăng mối quan tâm trong sự phân phối của công việc liên quan trong việc hoàn thành những dự án phức tạp .

Những người tham gia bên ngoài có thể được phân chia thành 3 nhóm chính:

- Subcontractors(những nhà thầu phụ, hiện nay được gọi là những tổ chức “outsourcing”) họ cam kết thực hiện các thành phần của 1 dự án, lớn hay nhỏ, tùy theo từng trường hợp. Những nhà thầu phụ thường đưa ra hợp đồng ở mức tối thiểu một trong những lợi ích: khả năng huy động nhân viên, ý kiến về mặt chuyên môn đặc biệt hoặc giá thấp.
- Những nhà cung cấp COTS software và những module phần mềm sử dụng lại. Lợi ích của sự tích hợp các thành phần đó là rất rõ ràng, sự xác định khoảng cách từ kế hoạch làm việc và

giảm bớt giá thành đến chất lượng. Một sự mong đợi mà sự tích hợp của những thành phần sẵn sàng để dùng sẽ được lưu trữ trong những mã nguồn phát triển, một kế hoạch làm việc ngắn hơn và phần mềm chất lượng cao hơn. Phần mềm chất lượng cao hơn thì được chờ đợi nhưng những thành phần đã được kiểm tra và được sửa chữa bởi những người phát triển, tốt như được sửa chữa theo những thiếu sót được xác định bởi khách hàng xem trước. Các tính chất của COTS software và các vấn đề chất lượng liên quan đến sử dụng chúng đã được nhận định bởi Basili và Boehm (2001).

- Khách hàng, bản thân họ như là người tham gia thực hiện dự án. Điều này khá chung để mỗi khách hàng thực hiện các phần của dự án: để áp dụng những chuyên môn đặc biệt của những khách hàng, đáp ứng cho giao dịch hoặc những yêu cầu bảo mật khác, giữ lại những nhân viên phát triển nội bộ đang sử dụng, ngăn ngừa những vấn đề bảo mật trong tương lai.v.v. Trường hợp này có những hạn chế trong những điều khoản của quan hệ khách hàng-người cung cấp cần thiết để thực hiện thành công một dự án. Vì thế, chắc chắn trường hợp này đã trở thành 1 thành phần chuẩn của nhiều dự án phát triển phần mềm và những quan hệ bằng hợp đồng.

3.5.2. Rủi ro và lợi ích của giới thiệu người tham dự ngoài.

Rủi ro chủ yếu tới chất lượng dự án liên quan với người giới thiệu tham gia từ bên ngoài trong cơ cấu của dự án là như sau:

- Sự trì hoãn hoàn thành của dự án.

Trong đó trường hợp người tham gia ở bên ngoài cung cấp chậm những thành phần cho hệ thống phần mềm, dự án nói chung sẽ bị hoãn lại. Sự chậm trễ này chủ yếu là do nhà thầu phụ và khách hàng gây ra chứ không phải do các nhà cung cấp phần mềm có sẵn (COTS).

Trong nhiều trường hợp, trách nhiệm kiểm soát sự phát triển phần mềm của nhà thầu phụ và khách hàng không cao, do đó gây ra tình trạng chậm chạp, trì hoãn và

không có thời gian cho sự thay đổi cũng như thời gian cần thiết để tổ chức lại gây ảnh hưởng tiêu cực với dự án.

- Các phần dự án chất lượng thấp được cung cấp bởi các thành viên bên ngoài

Các vấn đề về chất lượng có thể phân loại như

(a) sai sót: cao hơn so với số lượng sai sót mong đợi, thường cao hơn nhiều so với mong đợi

(b) Coding và tài liệu không chuẩn : sự vi phạm của phong cách và cấu trúc trong việc xây dựng và các thủ tục(theo giả thuyết như đã ấn định trong bất cứ hợp đồng nào). Phần mềm chất lượng thấp và không chuẩn sẽ gây ra khó khăn trong giai đoạn kiểm thử và sau đó trong giai đoạn bảo trì. Việc yêu cầu thêm thời gian để kiểm tra và chỉnh sửa chất lượng phần mềm chất lượng thấp có thể gây ra sự chậm trễ trong dự án đặc biệt trong trường hợp khi các thành viên bên ngoài hoàn thành nhiệm vụ của họ đúng thời hạn.

- Khó khăn về bảo trì trong tương lai.

Thực tế một số tổ chức tham gia việc phát triển nhưng chỉ một trong số họ, nhà thầu, là người trực tiếp gây nên 2 khó khăn trong việc bảo trì:

a,Nhà thầu có thể đối mặt với việc các coding và tài liệu không hoàn thành và/hoặc không đúng tiêu chuẩn từ các thành viên bên ngoài, gây ra sự kém chất lượng trong dịch vụ bảo trì của nhóm bảo trì và nhà thầu sẽ tốn chi phí cao hơn.

b. Các dịch vụ bảo trì được cung cấp bởi nhiều hơn một tổ chức, có thể nhà thầu phụ, nhà cung cấp phần mềm có sẵn COTS và các bộ phận phát triển của khách hàng.Khi mỗi phần này bị hạn chế khả năng đáp ứng, khách hàng buộc phải tìm kiếm người chịu trách nhiệm cho các lỗi cụ thể của phần mềm mỗi khi các lỗi này được phát hiện.

- Mất sự kiểm soát các bộ phận của dự án

Dù có ý hay không có ý,sự kiểm soát việc phát triển phần mềm của thành viên bên ngoài có thể tạo ra một bức tranh lạc quan không thực tế về tình trạng của dự án. Sự trao đổi với nhóm các thành viên bên ngoài có thể làm gián đoạn tới một vài tuần,gây cản trở việc đánh giá tiến độ của dự án..Kết quả là,cảnh báo về khó khăn trong phát triển, thiếu đội ngũ nhân viên và nhiều vấn đề khác đến muộn với các nhà thầu.

Nhận thức được trước các khó khăn này, nhà thầu phải xem xét kết hợp lợi ích và rủi ro được đưa ra bởi các thành viên bên ngoài trong một dự án.

3.5.3. Những mục tiêu đảm bảo chất lượng về sự đóng góp người tham gia bên ngoài

Những mục tiêu nào thu được bằng việc áp dụng công cụ SQA được cung cấp bởi các thành viên bên ngoài? Những mục tiêu dưới đây có thể thu được trực tiếp từ việc liệt kê các rủi ro đã đề cập ở trên:

- Để tránh trì hoãn hoàn thành nhiệm vụ và để đảm bảo cảnh báo sớm để tính trước sự trì hoãn.
- Để đảm bảo mức độ chất lượng có thể chấp nhận được của bộ phận triển khai và đón nhận cảnh báo sớm của phạm vi chất lượng yêu cầu.
- Để đảm bảo đủ tài liệu phục vụ cho nhóm bảo trì.
- Để đảm bảo liên tục, toàn diện và đáng tin cậy kiểm soát việc thực hiện người tham gia bên ngoài.

3.5.4. Các công cụ đảm bảo chất lượng những đóng góp của các thành viên đóng góp bên ngoài.

Chúng ta mong muốn các thành viên đóng góp bên ngoài thực hiện các phương thức SQA của chính họ, bao gồm các công cụ cần thiết để các sản phẩm phần mềm và các dịch vụ của họ đạt được mức chất lượng có thể chấp nhận được. Các công cụ được đề cập tới ở đây là những thứ mà các nhà thầu có thể áp dụng cho các thành viên đóng góp bên ngoài. Trong mục đích này, vấn đề chất lượng và thời gian là quan trọng nhất được xác định.

Các công cụ chính được áp dụng trước và trong suốt quá trình kết hợp các thành viên đóng góp bên ngoài trong một dự án phát triển phần mềm được liệt kê bên dưới.

- xem xét lại tài liệu yêu cầu.
- Đánh giá các tiêu chuẩn chọn lựa liên quan đến các thành viên đóng góp bên ngoài.
- Thành lập ủy ban điều khiển gia nhập và kết hợp của dự án.

- Sự đóng góp trong sự xem xét thiết kế.
- Sự đóng góp trong kiểm tra phần mềm.
- Cách trình bày các thủ tục đặc biệt
- Xác định các team leader của các nhà cung cấp và các thành viên.
- Chuẩn bị các báo cáo tiến trình phát triển của các hoạt động phát triển dự án.
- Xem xét lại các giao phẩm (các tài liệu) và acceptance tests

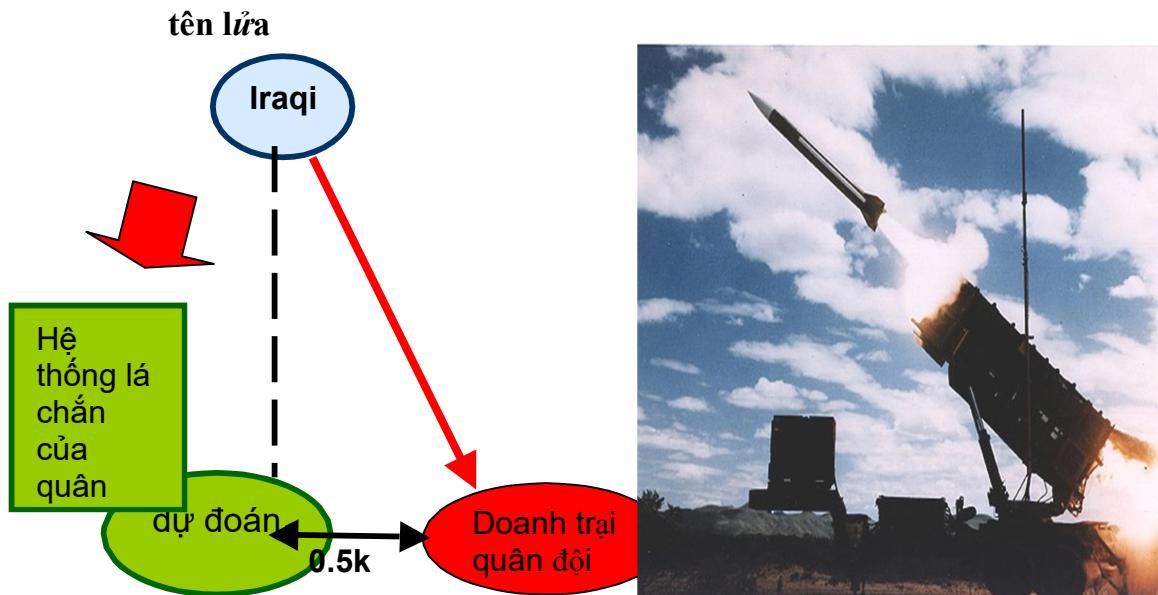
Chương 4. Kiểm thử phần mềm

4.1. Một số khái niệm cơ bản

4.1.1. Ví dụ về lỗi phần mềm

a. Vụ thất bại tên lửa Patriot:

- xảy ra tại Dharan, Saudi Arabia, vào 25 tháng 2, 1991
- làm 28 người chết
- nguyên nhân: vì lỗi rounding error (làm tròn)



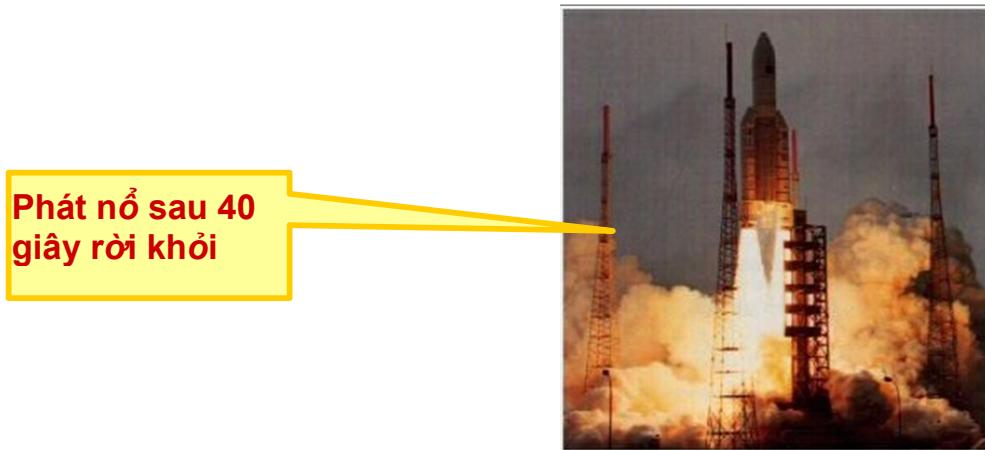
<http://www.ima.umn.edu/~arnold/disasters/patriot.html>

(biểu diễn số 1/10 bằng số phẩy tinh cơ số 2)

b. Vụ phát nổ Ariane 5:

- xảy ra tại Kourou, French Guiana, vào ngày 4 tháng sáu 1996
- thiệt hại 500 triệu \$
- nguyên nhân: vì lỗi tràn số (**Overflow error**)

(vì biểu diễn số phẩy động (floating point)
64-bit bởi số phẩy tĩnh (fixed point) 16-bit)



<http://www.ima.umn.edu/~arnold/disasters/ariane.html>

4.1.2. Đặc tả và lỗi phần mềm:

a. Đặc tả

“if you can't say it, you can't do it”

- a. Ta cần biết sản phẩm như thế nào trước khi ta có thể nói nó có lỗi.
- b. Đặc tả định nghĩa sản phẩm và:
 - c. Yêu cầu chức năng mô tả các tính năng của sản phẩm. Ví dụ, calculator:
 - d. Save, +, -, *, /, ...
- e. Yêu cầu phi chức năng là các ràng buộc về sản phẩm. Ví dụ ,
 - f. thân thiện với người dùng, hiệu năng, ...

b. Lỗi phần mềm

Phần mềm KHÔNG làm nhiệm vụ được đưa ra ở đặc tả (ví dụ, thiếu phép trừ) Phần mềm làm công việc mà đặc tả KHÔNG cho phép

Phần mềm làm công việc mà đặc tả không đề cập tới (ví dụ, tính căn bậc hai của số nguyên)

Phần mềm không làm công việc mà đặc tả không đề cập tới nhưng nên làm (ví dụ, kiểm tra divided by 0)

Phần mềm khó hiểu, khó dùng, chậm ...

c. Chi phí sửa lỗi

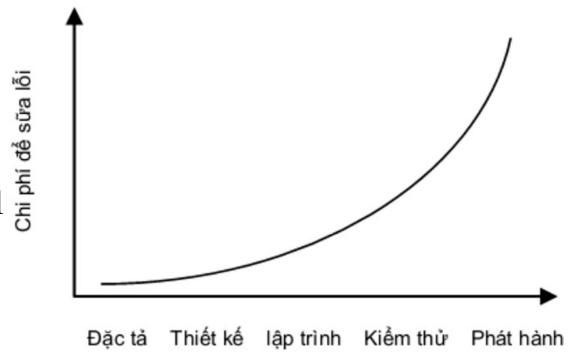
Chi phí sửa lỗi tăng theo cấp số nhân (10^x)
theo thời gian

Ví dụ, một lỗi phát hiện trong pha đặc tả tốn \$1
để sửa.

... nếu phát hiện trong pha thiết kế, tốn \$10

... nếu phát hiện trong pha cài đặt, tốn \$100

... nếu phát hiện sau khi phát hành, tốn \$1000



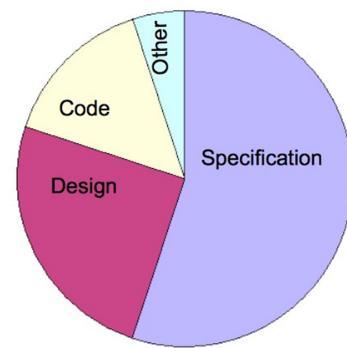
d. Nguồn lỗi

Đặc tả (Specification): đặc tả lỗi, không đầy đủ, không nhất quán.

Thiết kế (Design): lỗi cơ bản trong thiết kế phần mềm. **Cài đặt (Code):** lỗi lập trình, mã độc (malicious code). **Khác:**

Hệ thống hỗ trợ: Ngôn ngữ lập trình nghèo nàn, trình biên dịch có lỗi...

Kiểm thử không đầy đủ: kiểm thử chưa xong, kiểm chứng nghèo nàn,...



Cited from R.
Patten's
“Software
testing”, SAMS
publishing

4.1.3. Kiểm thử và tiến trình kiểm thử

a. Kiểm thử

Kiểm thử phần mềm có nhiều định nghĩa khác nhau đề xuất bởi nhiều tổ chức hay cá nhân khác nhau. Một số định nghĩa nổi bật:

Kiểm thử phần mềm là quá trình khảo sát một hệ thống hay thành phần dưới những điều kiện xác định, quan sát và ghi lại các kết quả, và đánh giá một khía cạnh nào đó của hệ thống hay thành phần đó. (Theo *Bảng chú giải thuật ngữ chuẩn IEEE của Thuật ngữ kỹ nghệ phần mềm- IEEE Standard Glossary of Software Engineering Terminology*).

Kiểm thử phần mềm là quá trình thực thi một chương trình với mục đích tìm lỗi.
(Theo “*The Art of Software Testing*” – *Nghệ thuật kiểm thử phần mềm*).

Kiểm thử phần mềm là hoạt động khảo sát thực tiễn sản phẩm hay dịch vụ phần mềm trong đúng môi trường chúng dự định sẽ được triển khai nhằm cung cấp cho người có lợi ích liên quan những thông tin về chất lượng của sản phẩm hay dịch vụ phần mềm ấy. Mục đích của kiểm thử phần mềm là tìm ra các lỗi hay khiếm khuyết phần mềm nhằm đảm bảo hiệu quả hoạt động tối ưu của phần mềm trong nhiều ngành khác nhau. (Theo *Bách khoa toàn thư mở Wikipedia*).

Có thể định nghĩa một cách dễ hiểu như sau: *Kiểm thử phần mềm là quá trình thực thi một hệ thống phần mềm để xác định xem phần mềm có đúng với đặc tả không và môi trường hoạt động có đúng yêu cầu không.*

Software testing objectives

Các mục tiêu trực tiếp

- Xác định và phát hiện nhiều lỗi nhất có thể trong phần mềm được kiểm thử
- Sau khi sửa chữa các lỗi đã xác định và kiểm tra lại, làm cho phần mềm đã được kiểm thử đến một mức độ chấp nhận được về chất lượng.
- Thực hiện các yêu cầu kiểm thử cần thiết một cách hiệu quả và có hiệu quả, trong phạm vi ngân sách và thời gian cho phép.

Các mục tiêu gián tiếp

Biên dịch một bản ghi về các lỗi phần mềm để sử dụng trong công tác phòng chống lỗi (bằng các hành động khắc phục và ngăn ngừa).

Cần lưu ý rằng thiếu sót của các mục tiêu thường xuyên “để chứng minh gói phần mềm đã sẵn sàng” không phải là ngẫu nhiên. Mục tiêu này vốn mâu thuẫn với mục tiêu đầu tiên đã được đề cập, và có thể ánh hưởng hoặc chính xác hơn, xu hướng lựa chọn của test/ hoặc các trường hợp test. Myers(1979) đã đưa ra tóm tắt gọn gàng vấn đề : “nếu mục tiêu của bạn là để hiển thị khi không có lỗi, bạn sẽ không phát hiện ra nhiều. Nếu mục tiêu của bạn là để cho thấy sự hiện diện của lỗi, bạn sẽ khám phá ra một tỷ lệ lớn lỗi”.

Cách viết mục tiêu thứ 2 phản ánh một thực tế là việc không có lỗi phần mềm là một khát vọng không tưởng. Vì vậy chúng ta thích cụm từ “mức độ chất lượng chấp nhận được”, nghĩa là tỷ lệ các lỗi nhất định mà người dùng có thể chịu đựng được, sẽ vẫn chưa xác định khi cài đặt phần mềm. Tỷ lệ này thay đổi theo từng gói phần mềm và người sử dụng, nhưng phải thấp hơn cho những gói có nguy cơ thất bại cao.

4.1.4. Các mức kiểm thử

Tuỳ thuộc vào mục tiêu của kiểm thử, ta chia ra thành các mức như sau: Mức 0: testing và debugging là giống nhau

Mức 1: Mục tiêu của kiểm thử là để chỉ ra phần mềm hoạt động

Mức 2: Mục tiêu của kiểm thử là để chỉ ra phần mềm không hoạt

Mức 3: Mục tiêu của kiểm thử là để giảm các rủi ro khi sử dụng phần mềm

Mức 4: Nhằm trợ giúp các chuyên gia CNTT phát triển các phần mềm có chất lượng cao hơn.

a. Mức 0

Testing và debuging là một

Mức này thường được thực hiện bởi các sinh viên trong các môn học lập trình. Sinh viên viết chương trình, chạy với vài đầu vào, và debug lỗi nếu có.

Mức này không phân biệt giữa hành vi không đúng và lỗi bên trong chương trình.

Mức này chỉ giúp ích đôi chút trong việc phát triển phần mềm chính xác

b. Mức 1

Nhằm để chứng minh tính đúng đắn.

Một cách phát triển tự nhiên từ mức 0, tuy nhiên ta không thể chứng minh tính đúng đắn của phần mềm. Giả sử ta chạy một tập test và không phát hiện ra lỗi nào. Vậy, kết luận là gì? Liệu ta có thể giả thiết là phần mềm chạy tốt hay tập test kém? Vì không thể chứng minh tính đúng đắn, việc kiểm thử không có giới hạn dừng cố định, cũng như không có một kỹ thuật test hình thức (formal) nào. Nếu người quản lý hỏi: còn phải thực thi bao nhiêu test nữa? Ta không có cách nào trả lời chính xác câu hỏi này.

c. Mức 2

Nhằm để chỉ ra lỗi. Tìm lỗi là một mục tiêu rõ ràng, nhưng mang tính tiêu cực. Tester có thể vui vẻ khi tìm ra lỗi, nhưng developers thì không muốn vậy - họ muốn phần mềm chạy (mức 1 là suy nghĩ tự nhiên của developers). Do đó, mức 2 đặt tester và developers vào quan hệ đối đầu. Điều này có thể ảnh hưởng xấu tới cả nhóm. Ngoài ra, câu hỏi đặt ra là nếu không tìm thấy lỗi nào thì sao? Ta có thể kết luận phần mềm chạy tốt? hoặc việc kiểm thử còn yếu?

d. Mức 3

Kiểm thử có thể chỉ ra lỗi khi nó xuất hiện, nhưng không thể chứng tỏ phần mềm không có lỗi. Nghĩa là, ta phải chấp nhận mỗi khi sử dụng phần mềm, ta có nguy cơ gặp lỗi.

Nguy cơ này có thể nhỏ, và không gây hậu quả gì, hoặc nguy cơ có thể lớn và gây ra thảm họa. Điều này chỉ ra rằng, toàn đội phát triển phần mềm có chung

mục tiêu - giảm nguy cơ gặp lỗi khi sử dụng phần mềm. Ở mức 3, cả tester và developer làm việc cùng nhau để giảm nguy cơ gặp lỗi.

e. Mức 4

Khi tester và developers có chung mục tiêu (mức 3), tổ chức có thể chuyển sang mức 4. Kiểm thử nhằm mục tiêu tăng chất lượng. Có nhiều cách để tăng chất lượng, trong đó tạo ra test có thể dẫn tới lỗi chỉ là một. Ở mức này, kỹ sư kiểm thử có thể trở thành trưởng nhóm kỹ thuật của dự án. Họ có nhiệm vụ đánh giá, cải thiện chất lượng phần mềm, và sự thẩm định của họ sẽ trợ giúp cho developers. Ví dụ như ta có 1 chương trình spell checker. Ta thường nghĩ nhiệm vụ chính của nó là để tìm những từ sai chính tả (đánh vẫn sai), nhưng thực tế, mục tiêu cao nhất của nó là để tăng khả năng viết chính tả của chúng ta. Mỗi khi spell checker tìm ra một từ sai chính tả, ta có cơ hội để học cách viết đúng.

Do vậy, spell checker là “chuyên gia” về chất lượng viết chính tả. Một cách tương tự, mức 4 hướng tới mục tiêu kiểm thử để tăng khả năng của developers trong việc phát triển các phần mềm chất lượng cao. Testers có thể đào tạo developers.

4.1.5. Một số thuật ngữ

Một số thuật ngữ liên quan tới kiểm thử lấy từ các tài liệu chuẩn (IEEE Standard Glossary of Software Engineering Terminology, DOD-STD-2167A and MIL-STD-498 from the US Department of Defense, and the British Computer Society’s Standard for Software Component Testing)

Validation (xác nhận): Tiến trình đánh giá một phần mềm ở cuối quá trình phát triển phần mềm để đảm bảo phù hợp với những dự định về sử dụng.

Verification (xác minh, kiểm chứng): tiến trình xác định xem liệu sản phẩm của một pha trong tiến trình phát triển phần mềm có đáp ứng đầy đủ yêu cầu được đưa ra trong pha trước đó.

Xác minh thường là hoạt động có tính kỹ thuật cao hơn, sử dụng những tri thức về các yêu cầu, đặc tả phần mềm. Xác nhận thường phụ thuộc vào tri thức về lĩnh vực tương ứng. Cụ thể là, tri thức về ứng dụng của phần mềm được viết. Ví dụ, xác nhận của phần mềm về máy bay yêu cầu tri thức từ kỹ sư hàng không và phi công.

Cụm từ IV & V - independent verification and validation- nghĩa là xác nhận và

xác minh độc lập. IV & V yêu cầu việc đánh giá phải được thực hiện bởi người không phải là lập trình viên. Một số trường hợp đội IV & V thuộc dự án, hoặc thuộc cùng công ty, cũng có thể thuộc một tổ chức độc lập. Vì sự độc lập của IV & V, tiến trình thường chưa bắt đầu cho tới khi dự án kết thúc và thường được thực hiện bởi chuyên gia trong lĩnh vực hơn là người phát triển phần mềm.

Hai thuật ngữ hay dùng là fault và failure. Chúng liên quan tới suy nghĩ từ mức 0 sang mức 1.

Sai sót (fault): khuyết tật trong phần mềm

Lỗi (error): một trạng thái (bên trong) không chính xác là kết quả của fault

Hỗn (failure): hành vi không chính xác (bên ngoài) so với các yêu cầu hoặc mô tả về hành vi mong đợi.

Ta hãy xem một ví dụ về bác sĩ chẩn đoán cho bệnh nhân. Bệnh nhân tới phòng khám với một danh sách failures (đó là các triệu chứng). Bác sĩ phải phát hiện ra các fault, hoặc nguồn gốc của các triệu chứng. Để trợ giúp quá trình chẩn đoán, bác sĩ có thể phải yêu cầu thực hiện các kiểm tra về sự bất thường của các trạng thái bên trong, như cao huyết áp, nhịp tim bất thường, lượng đường trong máu cao, cholesterol cao. Với thuật ngữ của chúng ta, các bất thường này gọi là errors.

Ta hãy xem xét một ví dụ về code Java

sau: public static int numZero (int[] x)

```
{ // Effects: if x == null throw NullPointerException  
// else return the number of occurrences of 0  
in x int count = 0;  
  
for (int i = 1; i < x.length;  
     i++) { if (x[i] == 0) {  
         count++; }  
     }  
  
return count;  
}
```

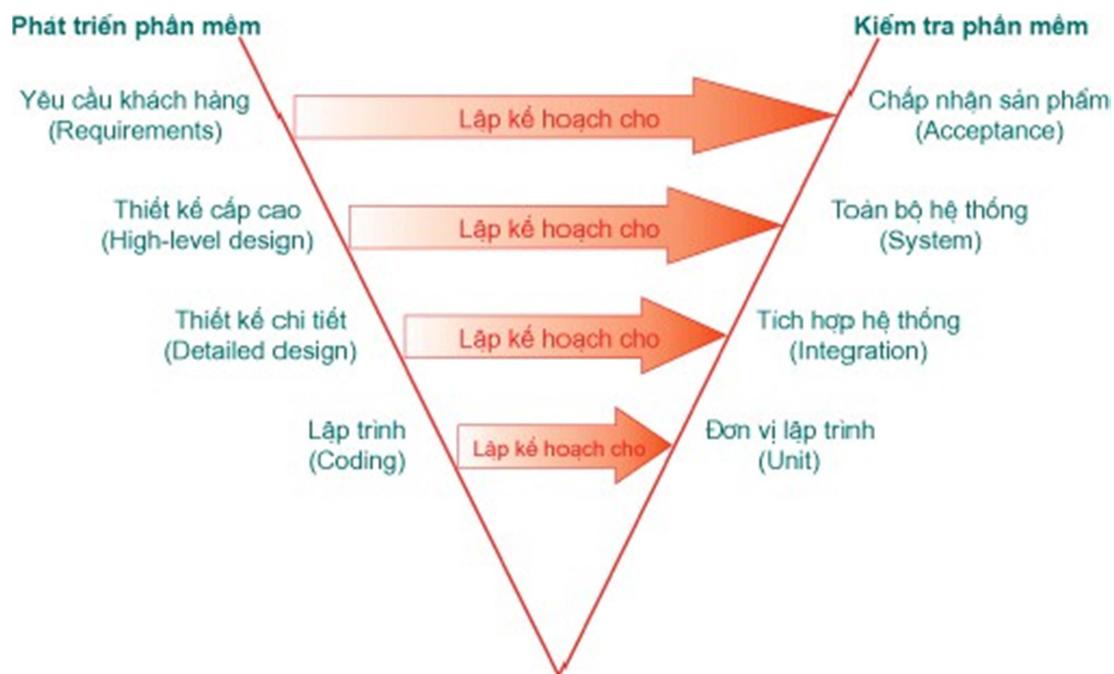
fault ở đây là chương trình bắt đầu tìm kiếm các số bằng 0 từ chỉ số 1 thay vì chỉ

số 0. Ví dụ numZero([2,7,0]) được tính chính xác là 1, còn với numZero([0,7,2]) sẽ được tính sai là 0. Trong cả 2 trường hợp, fault được thực thi. Mặc dù cả 2 trường hợp đều dẫn tới 1 lỗi, nhưng chỉ có trường hợp thứ 2 là gây ra failure. Để hiểu về trạng thái lỗi, ta cần định nghĩa trạng thái của chương trình. Trạng thái của numZero bao gồm giá trị của biến x, count, i và program counter (PC). Với ví dụ đầu tiên, trạng thái ở câu lệnh if cho vòng lặp đầu tiên là ($x = [2, 7, 0]$, count = 0, i = 1, PC = if). Lưu ý là trạng thái này là lỗi vì đáng ra $i = 0$. Tuy nhiên, trạng thái lỗi này không được lan truyền tới output và do vậy phần mềm không fail. Nói cách khác, một trạng thái là lỗi nếu nó không phải là trạng thái như mong đợi dù cho tất cả các giá trị trong trạng thái là chấp nhận được. Tổng quát hơn, nếu chuỗi trạng thái mong đợi là s0,s1,s2... trong khi chuỗi trạng thái thực tế là s0,s2,s3... thì trạng thái s2 là lỗi.

Với trường hợp thứ 2, trạng thái tương ứng là ($x = [0, 7, 2]$, count = 0, i = 1, PC = if). Trong trường hợp này, lỗi lan truyền tới kết quả trả về. Do vậy, ta thu được kết quả failure.

Định nghĩa về fault và failure cho phép ta phân biệt giữa testing và debugging.

4.2. Các cấp độ kiểm thử





Mối tương quan giữa phát triển và kiểm thử phần mềm

Các cấp độ kiểm thử phần mềm

4.2.1. Kiểm thử đơn vị - Unit Testing

Kiểm thử đơn vị là hoạt động kiểm thử nhỏ nhất

Kiểm thử thực hiện trên các hàm hay thành phần riêng lẻ. Cần hiểu biết về thiết kế chương trình và code.

Thực hiện bởi Lập trình viên (không phải kiểm thử viên)

Đơn vị: Là thành phần nhỏ nhất của phần mềm có thể kiểm thử được. Ví dụ: Các hàm, lớp, thủ tục, phương thức. Đơn vị thường có kích thước nhỏ, chức năng hoạt động đơn giản, không gây nhiều khó khăn trong việc kiểm thử, ghi nhận và phân tích kết quả do đó nếu phát hiện lỗi việc tìm kiếm nguyên nhân và sửa lỗi cũng đơn giản và tốn ít chi phí hơn.

Mục đích: Đảm bảo thông tin được xử lý đúng và có đầu ra chính xác trong mối tương quan giữa dữ liệu nhập và chức năng của đơn vị.

Người thực hiện: Do việc kiểm thử đơn vị đòi hỏi phải kiểm tra từng nhánh lệnh, nên đòi hỏi người kiểm thử có kiến thức về lập trình cũng như về thiết kế của hệ thống nên người thực hiện thường là lập trình viên.

Unit Test cũng đòi hỏi phải chuẩn bị trước các ca kiểm thử (*Test case*) hoặc kịch bản kiểm thử (*Test script*), trong đó chỉ định rõ dữ liệu đầu vào, các bước

thực hiện và dữ liệu đầu ra mong muốn. Các Test case và Test script này nên được giữ lại để tái sử dụng.

4.2.2. Kiểm thử tích hợp - Integration Testing

Kiểm thử tích hợp nhằm phát hiện lỗi giao tiếp xảy ra giữa các thành phần cũng như lỗi của bản thân từng thành phần (nếu có).

Thành phần có thể là

các module

các ứng dụng riêng lẻ

các ứng dụng client/server trên một mạng

Các loại kiểm thử tích hợp

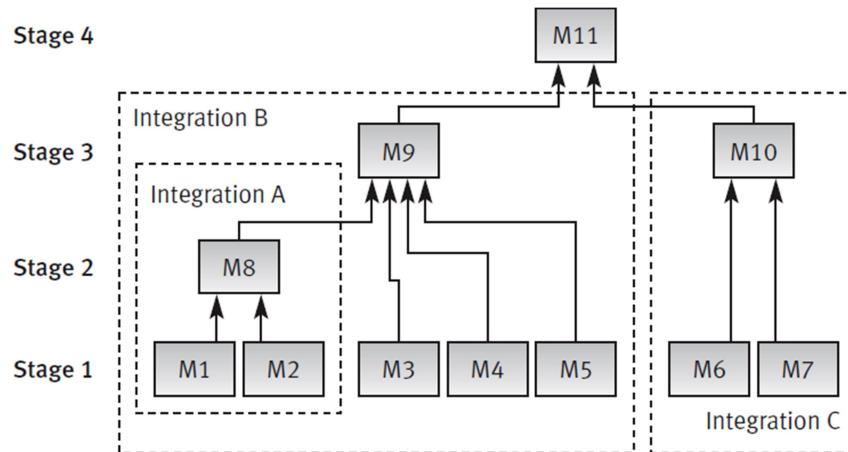
Big bang - Test toàn bộ phần mềm, một khi các gói đã hoàn thành có sẵn; có thể biết đến như là “big bang testing”.

Top-down – kiểm thử từ gốc của hệ thống phân cấp thành phần. Các thành phần được thêm theo thứ tự giảm dần của hệ thống phân cấp.

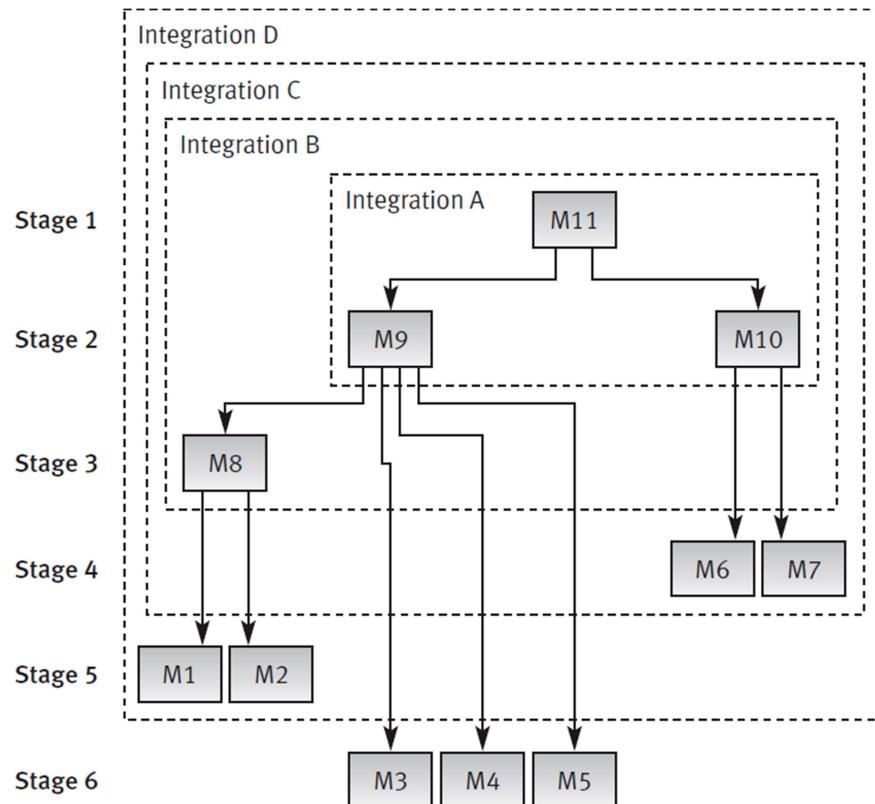
Bottom-up – kiểm thử từ đáy của hệ thống phân cấp. Các thành phần được thêm theo thứ tự tăng dần của hệ thống phân cấp.

Hình sau minh họa test top-down và test bottom-up của cùng một dự án phát triển phần mềm gồm 11 mô-đun. Hình (a), quá trình phát triển phần mềm và sự thử nghiệm tiếp thep được thực hiện theo bottom-up, trong 4 giai đoạn sau:

- Giai đoạn 1: test các mô-đun từ 1 đến 7
- Giai đoạn 2: tích hợp test A của các mô-đun 1 và 2 ,đã phát triển và test ở
giai đoạn 1, tích hợp với mô-đun 8, phát triển trong giai đoạn hiện thời.
- Giai đoạn 3: hai sự kiểm tra tích hợp riêng biệt, B, trên các mô-đun
3,4,5 và 8, tích hợp với mô-đun 9 , và C, mô đun 6 và 7 tích hợp với
mô-đun 10
- Giai đoạn 4: test hệ thống được thực hiện sau khi B và C đã tích hợp
với mô-đun 11, đã phát triển trong giai đoạn hiện thời.



1. Bottom-up testing



(b) Top-down testing

Trong hình trên, phát triển phần mềm và kiểm thử được thực hiện từ trên xuống qua sáu giai đoạn :

- Giai đoạn 1: test mô-đun 11(unit test)

- Giai đoạn 2: tích hợp test A của mô-đun 11 tích hợp với mô-đun 9 và 10, phát triển trong giai đoạn hiện thời
- Giai đoạn 3: tích hợp test B của A tích hợp với mô-đun 8, phát triển trong giai đoạn hiện thời.
- Giai đoạn 4: tích hợp test C của B tích hợp với mô-đun 6 và 7 , phát triển trong giai đoạn hiện thời
- Giai đoạn 5: tích hợp test D của C tích hợp với mô-đun 1 và 2, phát triển trong giai đoạn hiện thời
- Giai đoạn 6: test hệ thống của D tích hợp với mô-đun 3,4,5, phát triển trong giai đoạn hiện thời

Việc gia tăng các đường trong hình trên chỉ có hai trong số nhiều các đường. Đường dẫn trong các ví dụ là “trình tự theo chiều ngang (horizontally sequenced)” (“breadth first”), mặc dù ta có thể chọn một đường dẫn là “trình tự theo chiều dọc(vertically sequenced)” (“depth first”). Nếu ta thay đổi đường ngang của top-down trong hình (b), với một dãy dọc, kiểm thử có thể được thực hiện như sau :

- Giai đoạn 1: test đơn vị mô-đun 11
- Giai đoạn 2: tích hợp test A của tích hợp mô-đun 11 với mô-đun 9, phát triển trong giai đoạn hiện thời
- Giai đoạn 3: tích hợp test B của A với mô-đun 8, phát triển trong giai đoạn hiện thời
- Giai đoạn 4: tích hợp test C của B với các mô-đun 1 và 2 , phát triển trong giai đoạn hiện thời
- Giai đoạn 5: tích hợp test D của C với mô-đun 10, phát triển trong giai đoạn hiện thời
- Giai đoạn 6: tích hợp test E của D với các mô-đun 6 và 7, phát triển trong giai đoạn hiện thời
- Giai đoạn 7: test hệ thống được thực hiện sau khi sau khi E đã được tích hợp với các mô-đun 3,4 và 5, phát triển trong giai đoạn hiện thời.

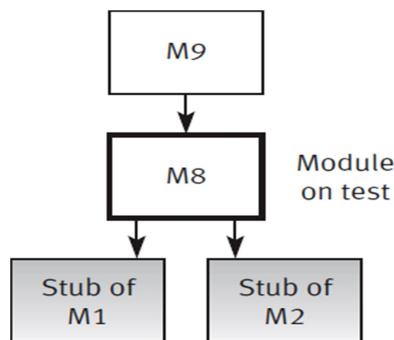
Các đường dẫn khác liên quan đến khả năng phân nhóm các mô-đun trong

một giai đoạn kiểm thử. Ví dụ, đường dẫn trong top-down ở hình 9.1(b), một trong những nhóm mô-đun có thể là 8, 1 và 2 và/hoặc các mô-đun 10, 6 và 7.

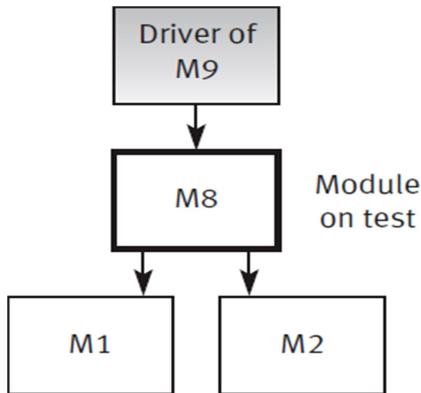
Stubs và drives để kiểm thử gia tăng

Stubs và drives là phần mềm mô phỏng thay thế cho các mô-đun không có sẵn khi thực hiện một đơn vị hoặc kiểm thử tích hợp.

Một stub (thường được cho là một “mô-đun giả (dummy module)”) thay thế một mô-đun không có sẵn ở mức thấp hơn, mô-đun phụ để kiểm thử. Các Stub được yêu cầu cho kiểm thử top-down cho các hệ thống không đầy đủ. Trong trường hợp này, stub cung cấp kết quả tính toán của mô-đun phụ, chưa được phát triển (mã hóa), được thiết kế để thực hiện. Ví dụ, ở giai đoạn 3 của ví dụ top-down trong hình (b), mô-đun 9 phía trên kích hoạt mô-đun 8, có sẵn; nó đã được kiểm thử và đúng ở giai đoạn 2. Các stub được yêu cầu để thay thế cho các mô-đun phụ 1 và 2 là những mô-đun chưa được hoàn thành. Sự thiết lập kiểm thử này được trình bày trong hình dưới đây



Giống như một Stub, nhưng một driver là một mô-đun thay thế cho các mô-đun mức trên để kích hoạt kiểm thử các mô-đun. Driver là đi từ kiểm tra dữ liệu đến kiểm thử mô-đun và chấp nhận kết quả tính toán của nó. Driver được yêu cầu kiểm thử bottom-up cho tới khi các mô-đun ở mức cao được phát triển (mã hóa). Ví dụ, ở giai đoạn kiểm thử 2 của ví dụ bottom-up trong hình (a), các mô-đun phụ 1 và 2 ở mức thấp hơn có sẵn; chúng đã được kiểm thử và đúng ở giai đoạn kiểm thử 1. Một driver được yêu cầu để thay thế cho mô-đun 9 chưa được hoàn thành. Sự thiết lập kiểm thử này được thể hiện trong hình (b).



Lưu ý

Việc duy trì một thư viện các stub và driver để tái sử dụng trong tương lai có thể tiết kiệm đáng kể nguồn tài nguyên.

- Các chiến lược Bottom-up so với Top-down

Lợi thế chính của chiến lược Bottom-up là thực hiện tương đối dễ, trong khi bất lợi của nó chính là sự chậm trễ trong trường hợp toàn bộ chương trình được theo dõi (nghĩa là, ở giai đoạn kiểm thử mô-đun cuối cùng). Lợi thế chính của chiến lược Top-down là khả năng nó cung cấp để chứng minh chức năng của toàn bộ chương trình một cách ngắn nhất ngay sau khi kích hoạt các mô-đun ở mức trên đã hoàn thành. Trong nhiều trường hợp, các đặc trưng này cho phép xác định các lỗi liên quan đến thuật toán, các yêu cầu chức năng trong phân tích và thiết kế một cách sớm nhất. Bất lợi chính của chiến lược này là tương đối khó khăn trong việc chuẩn bị các stub được yêu cầu, thường yêu cầu lập trình phức tạp. Bất lợi khác là khó khăn trong việc phân tích kết quả kiểm thử.

Các chuyên gia kiểm thử tiếp tục tranh luận xem chiến lược top-down hay bottom-up thích hợp hơn. Trong khi các quan điểm thay đổi, có vẻ như sự lựa chọn chiến lược thực sự được xác định trong hầu hết các trường hợp là sự lựa chọn phát triển – không kiểm thử - chiến lược của các nhà phát triển, nghĩa là, bottom-up hoặc top-down. Rõ ràng, những người kiểm thử nên thực hiện theo phương pháp tiếp cận của nhà phát triển vì quan trọng là việc kiểm thử sẽ được thực hiện ngay khi một mô-đun đã được code. Thực hiện một chiến lược kiểm thử khác với

chiến lược phát triển sẽ gây ra sự chậm trễ đáng kể của các cuộc kiểm thử.

Big bang so với các chiến lược khác

Trừ khi chương trình quá nhỏ và đơn giản, ứng dụng của chiến lược kiểm thử big bang chỉ ra những bất lợi nghiêm trọng. Xác định các lỗi trở nên khá rườm rà đối với số lượng lớn phần mềm. Mặc dù có nhiều nguồn lực đã đầu tư, hiệu quả của phương pháp này là tương đối khiêm tốn. Tỷ lệ nhận dạng lỗi tương đối thấp của big bang chứng minh cho kết luận này. Hơn nữa, khi đối đầu với toàn bộ gói phần mềm, việc sửa lỗi thường là nhiệm vụ nặng nề, cần phải xem xét những ảnh hưởng có thể của vài mô-đun tại cùng một thời điểm. Những ràng buộc hiển nhiên này tạo ra một nỗ lực khá mờ nhạt của việc ước lượng các nguồn lực kiểm thử cần thiết và tiến độ kiểm thử. Điều này cũng ám chỉ rằng triển vọng giữ đúng tiến độ và trong phạm vi ngân sách bị giảm đáng kể khi áp dụng chiến lược kiểm thử này.

4.2.3. Kiểm thử hệ thống - System Testing

Kiểm thử hệ thống là một mức của tiến trình kiểm thử phần mềm khi các module và tích hợp các module đã được test.

Mục tiêu của kiểm thử hệ thống là để đánh giá phần mềm có tuân thủ theo các yêu cầu đã đưa ra không.

System Test lại gồm nhiều loại kiểm thử khác nhau, phổ biến nhất gồm:

- ***Kiểm thử chức năng (Functional Test):*** Bảo đảm các hành vi của hệ thống thỏa mãn đúng yêu cầu thiết kế.
- ***Kiểm thử hiệu năng (Performance Test):*** Bảo đảm tối ưu việc phân bổ tài nguyên hệ thống (ví dụ bộ nhớ) nhằm đạt các chỉ tiêu như thời gian xử lý hay đáp ứng câu truy vấn...
- ***Kiểm thử mức độ đáp ứng (stress testing):*** thực thi hệ thống với giả thiết là các tài nguyên hệ thống yêu cầu không đáp ứng được về chất lượng, ổn định và số lượng.
- ***Kiểm thử cấu hình (Configuration Test):*** phân tích hệ thống với các thiết lập cấu hình khác nhau.
- ***Kiểm thử độ ổn định (robustness testing):*** kiểm thử dưới các điều kiện không mong đợi ví dụ như người dùng gõ lệnh sai, nguồn điện bị ngắt.

- **Kiểm thử bảo mật (Security Test):** Bảo đảm tính toàn vẹn, bảo mật của dữ liệu và của hệ thống.
- **Kiểm thử khả năng phục hồi (Recovery Test):** Bảo đảm hệ thống có khả năng khôi phục trạng thái ổn định trước đó trong tình huống mất tài nguyên hoặc dữ liệu; đặc biệt quan trọng đối với các hệ thống giao dịch như ngân hàng trực tuyến...
- **Kiểm thử quá tải (overload testing):** đánh giá hệ thống khi nó vượt qua giới hạn cho phép.
- Kiểm nghiệm chất lượng (quality testing): đánh giá sự tin tưởng, tính sẵn sàng của hệ thống. Bao gồm cả việc tính toán thời gian trung bình hệ thống sẽ bị hỏng và thời gian trung bình để khắc phục.
- **Kiểm nghiệm cài đặt (Installation testing):** Người dùng sử dụng các chức năng của hệ thống và ghi lại các lỗi tại ví trí sử dụng thật sự.

Lưu ý là không nhất thiết phải thực hiện tất cả các loại kiểm thử nêu trên. Tùy yêu cầu và đặc trưng của từng hệ thống, tùy khả năng và thời gian cho phép của dự án, khi lập kế hoạch, người Quản lý dự án sẽ quyết định áp dụng những loại kiểm thử nào.

4.2.4. Kiểm thử chấp nhận - Acceptance Testing

Kiểm thử chấp nhận là một cấp độ trong tiến trình kiểm thử phần mềm nhằm kiểm thử hệ thống về khả năng chấp nhận được.

Mục tiêu của kiểm thử này là để đánh giá sự tuân thủ của hệ thống với các yêu cầu nghiệp vụ và thẩm định xem đã có thể chấp nhận để bàn giao chưa.

Kiểm thử chấp nhận nhằm mục đích để chứng minh phần mềm thỏa mãn tất cả yêu cầu của khách hàng và khách hàng chấp nhận sản phẩm. Kiểm thử chấp nhận được khách hàng thực hiện (hoặc ủy quyền cho một nhóm thứ ba thực hiện).

Kiểm thử chấp nhận thông thường sẽ thông qua hai loại kiểm thử gọi là kiểm thử Alpha – **Alpha Test** và kiểm thử Beta – **Beta Test**. Với Alpha Test, người dùng kiểm thử phần mềm ngay tại nơi phát triển phần mềm, lập trình viên sẽ ghi nhận các lỗi hoặc phản hồi, và lên kế hoạch sửa chữa. Với Beta Test, phần mềm sẽ được gửi tới cho người dùng để kiểm thử ngay trong môi trường thực, lỗi hoặc phản hồi cũng sẽ gửi ngược lại cho lập trình viên để sửa chữa.

Thực tế cho thấy, nếu khách hàng không quan tâm và không tham gia vào quá trình phát triển phần mềm thì kết quả Acceptance Test sẽ sai lệch rất lớn, mặc dù phần mềm đã trải qua tất cả các kiểm thử trước đó. Sự sai lệch này liên quan đến việc hiểu sai yêu cầu cũng như sự mong chờ của khách hàng.

4.3. Các kỹ thuật kiểm thử

4.3.1. Kiểm thử hộp đen - Black-box Testing

Black-box testing là phương pháp kiểm thử mà không cần biết cài đặt của chương trình. Cần có một bản chương trình chạy được và đặc tả

Test cases được công thức hóa là một cặp ví dụ, (input, output mong muốn)

Một số kỹ thuật thiết kế test:

phân lớp tương đương, test biên, phân loại, kiểm thử
theo cặp Quan trọng trong công nghiệp

a. Phân lớp tương đương – Equivalence Partitioning

Phân lớp tương đương là một phương pháp kiểm thử hộp đen chia miền đầu vào của chương trình thành các lớp dữ liệu, từ đó suy dẫn ra các ca kiểm thử. Lớp tương đương biểu thị cho tập các trạng thái hợp lệ hay không hợp lệ đối với điều kiện vào.

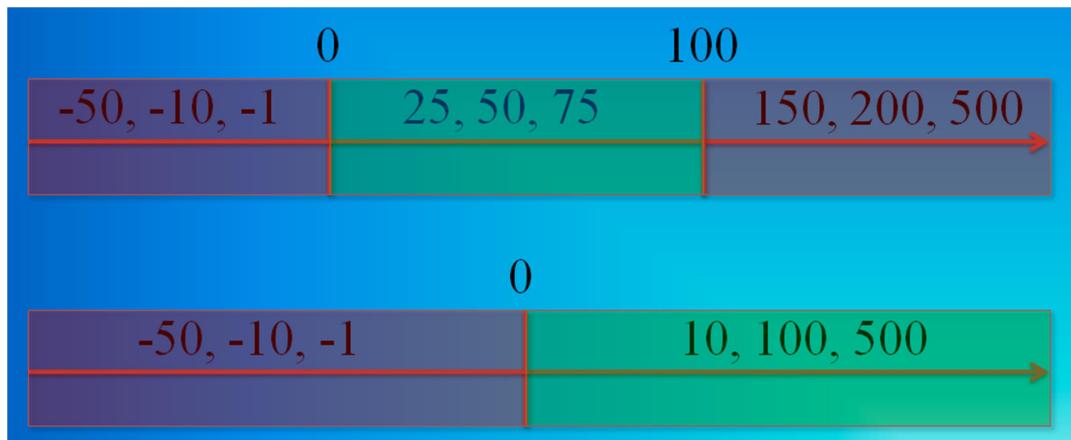
Thiết kế Test-case bằng phân lớp tương đương tiến hành theo 2 bước:

1. Xác định các lớp tương đương.
2. Xác định các ca kiểm thử.

Xác định các lớp tương đương

Các lớp tương đương được xác định bằng cách lấy mỗi trạng thái đầu vào (thường là một câu hay một cụm từ trong đặc tả) và phân chia nó thành 2 hay nhiều nhóm.

Ví dụ về các lớp tương đương:



Từ các lớp tương đương trên ta có bảng liệt kê các lớp tương đương tương ứng:

Điều kiện đầu vào	Các lớp tương đương hợp lệ	Các lớp tương đương không hợp lệ
• x lớn hơn 0 và nhỏ hơn 100.	25, 50, 75	-50, -10, -1, 150, 200, 500
• x lớn hơn 0.	10, 100, 500	-50, -10, -1

Để xác định các lớp tương đương có thể áp dụng các nguyên tắc dưới đây:

Một vùng giá trị

Điều kiện đầu vào là **một vùng giá trị**.

Ví dụ: “Giá trị x chỉ có thể dao động từ 0 đến 100”.

Chúng ta sẽ xác định được 1 lớp tương đương hợp lệ là: $0 \leq x \leq 100$

và 2 lớp tương đương không hợp lệ là: $x < 0$ và $x > 100$.

Một số giá trị

Điều kiện đầu vào được mô tả là **một số giá trị**.

Ví dụ: “Chỉ một đến sáu người có thể được đăng ký”.

Chúng ta sẽ xác định 1 lớp tương đương hợp lệ là: “*Có từ một đến sáu người đăng ký*”

Và 2 lớp tương đương không hợp lệ là: “*không người nào đăng ký*” và “*nhiều hơn sáu người đăng ký*”.

Một tập hợp các giá trị

Điều kiện đầu vào được mô tả là **một tập các giá trị**. Ta sẽ xác định mỗi giá trị trong tập đó là một lớp tương đương hợp lệ.

Ví dụ: “*Các loại xe được đăng ký là xe bus, xe khách, xe tải, xe taxi và xe máy*”.

Đối với mỗi loại xe ở trên chúng ta sẽ xác định là một lớp tương đương hợp lệ (ở đây chúng ta sẽ có 5 lớp tương đương hợp lệ) và một lớp tương đương không hợp lệ là một loại xe khác các loại xe nêu trên ví dụ như “xe đạp”.

Điều kiện đặc biệt

Điều kiện đầu vào được mô tả là **một điều kiện đặc biệt**.

Ví dụ: “*Ký tự đầu tiên phải là ký tự chữ*”

Chúng ta sẽ xác định được một lớp tương đương hợp lệ là “*ký tự đầu tiên là ký tự chữ*” và một lớp tương đương không hợp lệ là “*không phải là ký tự chữ (có thể là số hoặc ký tự đặc biệt)*”.

Xác định các ca kiểm thử

Với các lớp tương đương xác định được ở bước trên, bước thứ hai là sử dụng các lớp tương đương đó để xác định các ca kiểm thử. Quá trình này như sau:

Gán số thứ tự cho mỗi lớp tương đương đã xác định.

Viết test case cho các giá trị nằm trong các lớp tương đương hợp lệ.

Viết test case cho các lớp tương đương không hợp lệ.

b. Phân tích giá trị biên – Boundary Value Analysis

Kinh nghiệm cho thấy các ca kiểm thử mà khảo sát tỷ mỷ các điều kiện biên có tỷ lệ phần trăm cao hơn các ca kiểm thử khác. Các điều kiện biên là những điều kiện mà các tình huống ngay tại, trên và dưới các cạnh của các lớp tương đương đầu vào và các lớp tương đương đầu ra. Phân tích các giá trị biên là phương pháp thiết kế ca kiểm thử bổ sung thêm cho phân lớp tương đương, nhưng khác với phân lớp tương đương ở 2 khía cạnh:

- Phân tích giá trị biên không lựa chọn phần tử bất kỳ nào trong 1 lớp tương đương là điển hình, mà nó yêu cầu là 1 hay nhiều phần tử được lựa chọn như vậy mà mỗi cạnh của lớp tương đương đó chính là đối tượng kiểm tra.
- Ngoài việc chỉ tập trung chú ý vào các trạng thái đầu vào (không gian đầu vào), các ca kiểm thử cũng nhận được bằng việc xem xét không gian kết quả (các lớp tương đương đầu ra).

□ **Những cách phân tích giá trị biên**

• **Một vùng giá trị**

Điều kiện đầu vào được mô tả là **một vùng giá trị**.

Ta viết test case cho giá trị hợp lệ là điểm bắt đầu, kết thúc của vùng giá trị này. Test case cho giá trị không hợp lệ là giá trị ở phía ngoài của 2 điểm này.

Ví dụ: “*Giá trị x dao động từ 0 đến 100*”

Ta sẽ viết test case cho các trường hợp: **0, 100, -1, 100**.

• **Một số giá trị**

Điều kiện đầu vào được mô tả là **một số giá trị**.

Viết test case cho giá trị hợp lệ là số nhỏ nhất, lớn nhất của các giá trị này. Test case cho giá trị không hợp lệ là giá trị ở phía ngoài của 2 số này.

Ví dụ: “*Chỉ một đến sáu người có thể đăng ký*”

Ta cần viết test case cho các trường hợp: **1, 6, 0** và **7**.

• **Quan tâm đến điều kiện xuất (kết quả)**

Sử dụng cách 1 và 2 ở trên áp dụng cho điều kiện xuất.

Ví dụ: “*Màn hình hiển thị tóm tắt các tin tức mới nhất và hiển thị được nhiều nhất 4 tin*”.

Ta viết test case cho các kết quả hợp lệ là: **0, 1 và 4**

tin. Test case cho kết quả không hợp lệ là **5 tin**.

• **Danh sách có thứ tự**

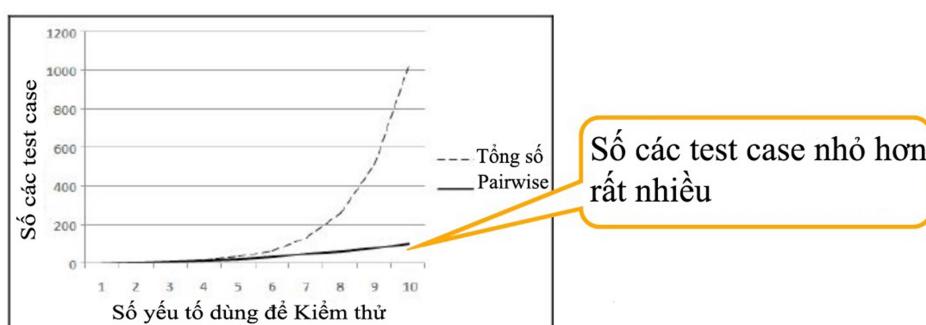
Nếu đầu vào hay đầu ra của 1 chương trình là tập được sắp thứ tự (ví dụ 1 file tuần tự hay 1 danh sách định tuyến hay 1 bảng) tập trung chú ý vào các phần

tử đầu tiên và cuối cùng của tập hợp.

Cuối cùng, tùy vào các trường hợp khác nữa, chúng ta cũng cần sự tư duy và kinh nghiệm của mình để tìm ra các biên cần test.

c. Kiểm thử theo cặp:

Thực tế cho thấy hầu hết các lỗi đều được sinh ra từ sự kết hợp giá trị của các cặp tham số đầu vào.

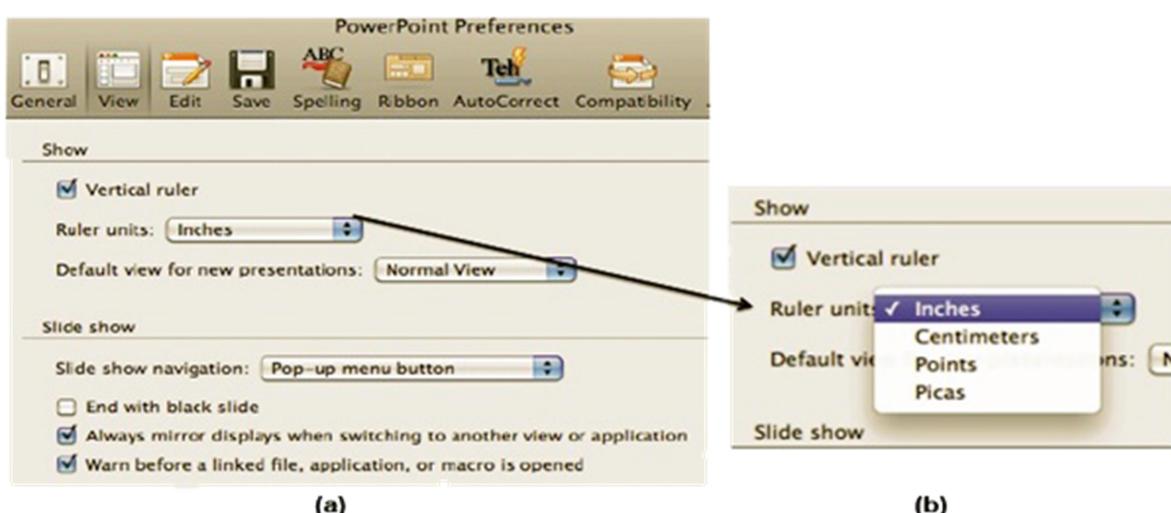


Lựa chọn tham số đầu vào và các giá trị tương ứng Lấy tổ hợp (pairwise) của các giá trị giữa 2 tham số

Xây dựng bộ test sao cho bao phủ được tất cả các cặp xác định ở trên

Ví dụ

Xét tab tùy chọn View từ một phiên bản của phần mềm Microsoft Powerpoint (xem Hình). Dữ liệu trong hình.



Vertical Ruler	Ruler Units	Default View	SS Navigation	End with Black	Always Mirror	Warn Before
Visible	Inches	Normal	Pop-up	Yes	Yes	Yes
Invisible	Centimeters	Slide	None	No	No	No

Tab View_preference gồm bảy thuộc tính, mỗi thuộc tính lại bao gồm một trong các giá trị con khác nhau: Vertical_Ruler (Visible, InVisible), Ruler_units (Inches, Centimetes, Points, Picas), Default_View (Normal, Slide, Outline), Ss_Navigator (Popup, None), End_With_Black (Yes1, No1), Always_Mirror (Yes2, No2), Warn_Before (Yes3, No3).

Ban đầu, chúng ta có tổng số $2 * 4 * 3 * 2 * 2 * 2 = 384$ test case.

Các cặp có thể có: (Visible, Inches), (Visible, Centimetes), ..., (No2, No3)

Test case (Visible, Centimetes, Nomal, Non, No1, Yes2, Yes3) bao gồm 21 cặp (Visible, Centimetes), (Visble, Nomal),..., (Yes2, Yes3)

Một bộ test case bao phủ được tất cả các cặp được cho ở bảng dưới (được tạo bởi công cụ sinh pairwise PICT

((<http://download.microsoft.com/download/f/5/5/f55484df-8494-48fa-8dbd-8c6f76cc014b/pict33.msi>)

Vertical Ruler	Ruler units	Default View	Ss	End	With	Warn
			Navigator	Black	Always	Before
Visible	Centimetes	Normal	None	No1	Yes2	Yes3
InVisible	Picas	Normal	Popup	Yes1	No2	No3
Visible	Picas	Slide	Popup	No1	No2	Yes3
InVisible	Inches	Normal	None	Yes1	Yes2	No3
InVisible	Points	Outline	None	No1	No2	No3
Visible	Centimetes	Slide	Popup	Yes1	Yes2	No3

Visible	Picas	Outline	None	Yes1	Yes2	Yes3
InVisible	Inches	Outline	Popup	No1	No2	Yes3
Visible	Points	Slide	None	Yes1	Yes2	Yes3
InVisible	Inches	Slide	None	No1	Yes2	Yes3
Visible	Inches	Normal	Popup	No1	Yes2	Yes3
Visible	Points	Normal	Popup	Yes1	Yes2	No3
InVisible	Centimetes	Outline	Popup	Yes1	No2	Yes3

d. Đoán lỗi – Error Guessing

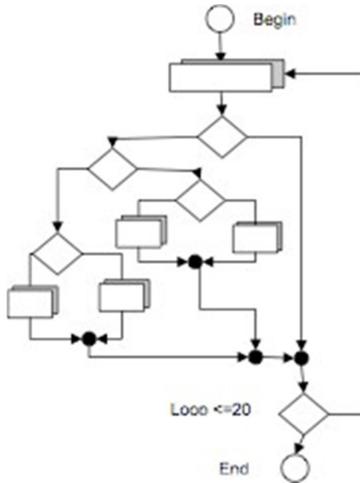
Một kỹ thuật thiết kế test-case khác là *error guessing – đoán lỗi*. Tester được đưa cho 1 chương trình đặc biệt, họ phỏng đoán, cả bằng trực giác và kinh nghiệm, các loại lỗi có thể và sau đó viết các ca kiểm thử để đưa ra các lỗi đó.

Thật khó để đưa ra một quy trình cho kỹ thuật đoán lỗi vì nó là một quy trình có tính trực giác cao và không thể dự đoán trước. Ý tưởng cơ bản là liệt kê một danh sách các lỗi có thể hay các trường hợp dễ xảy ra lỗi và sau đó viết các ca kiểm thử dựa trên danh sách đó. Một ý tưởng khác để xác định các ca kiểm thử có liên đới với các giả định mà lập trình viên có thể đã thực hiện khi đọc đặc tả (tức là, những thứ bị bỏ sót khỏi đặc tả, hoặc là do tình cờ, hoặc là vì người viết có cảm giác những đặc tả đó là rõ ràng). Nói cách khác, bạn liệt kê những trường hợp đặc biệt đó mà có thể đã bị bỏ sót khi chương trình được thiết kế.

4.3.2. Kiểm thử hộp trắng - White-box Testing (WBT)

WBT là phương pháp kiểm thử trong đó ta cần biết cấu trúc/thiết kế/cài đặt của phần mềm.

WBT ngược với black-box testing.



Với flowchart cho chương trình đơn giản (hình trên) khoảng 100 dòng với 1 vòng lặp thực hiện không quá 20 lần. Số đường có thể thực hiện lên tới 10^{14} . Giả sử để kiểm thử 1 trường hợp cần chạy trung bình 1 giây. Như vậy, để kiểm thử hết các đường của chương trình trên cần khoảng thời gian là 3170 năm. Ta có thể thấy, việc kiểm thử toàn bộ các trường hợp là bất khả thi với các hệ thống lớn. Tuy nhiên, với các module quan trọng, thực thi việc tính toán chính của hệ thống, phương pháp này là cần thiết và khả thi.

a. Kiểm tra theo câu lệnh (statement testing):

Thiết kế quá trình kiểm tra sao cho mỗi câu lệnh được thực hiện ít nhất một lần. Phương pháp này dựa trên ý tưởng: trừ phi một câu lệnh được thực hiện, nếu không ta không thể biết có lỗi xảy ra với câu lệnh đó hay không. Tuy nhiên, việc kiểm tra với một giá trị đầu vào không đảm bảo là câu lệnh sẽ đúng với mọi trường hợp

Ví dụ, đoạn chương trình

result = 0+1+...+|value| nếu result <= maxint, báo lỗi trong trường hợp ngược lại.

1 PROGRAM maxsum (maxint, value :
INT)

2 INT result := 0 ; i := 0 ;

3 IF value < 0

4 THEN value := - value ;

5 WHILE (i < value) AND (result <= maxint) 6 DO i := i + 1 ;

7 result := result + i ;

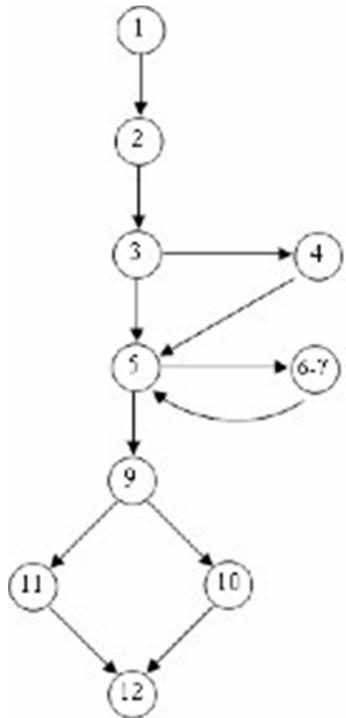
8 OD;

9 IF result <= maxint

10. THEN OUTPUT (result)

11. ELSE OUTPUT (“too large”)

12. END



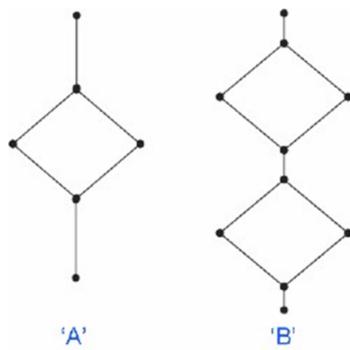
với các bộ giá trị input:

maxint = 10, value = -1 hoặc maxint = 0; value = -1

ta sẽ kiểm tra được toàn bộ các câu lệnh trong đoạn chương trình trên.

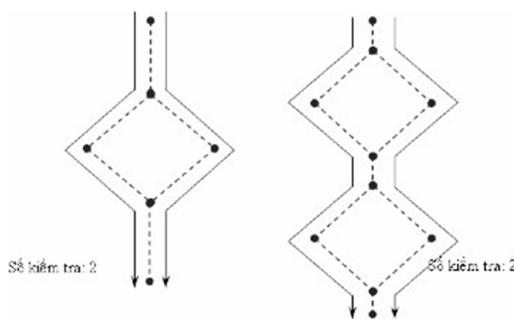
Nhận xét:

Để đánh giá phương pháp này, ta xét ví dụ sau:



với câu hỏi đầu tiên “Lược đồ nào phức tạp hơn?”, câu trả lời là B. Câu hỏi tiếp “Lược đồ nào cần nhiều bước kiểm tra hơn?”, đáp án vẫn là B.

Tuy nhiên, ta thấy số lần kiểm tra tối thiểu để có thể kiểm tra toàn bộ các câu lệnh cho cả



hai trường hợp đều là 2. Vì vậy, phương pháp này không tương ứng với độ phức tạp của mã lệnh.

b. Kiểm tra theo đường dẫn (path testing)

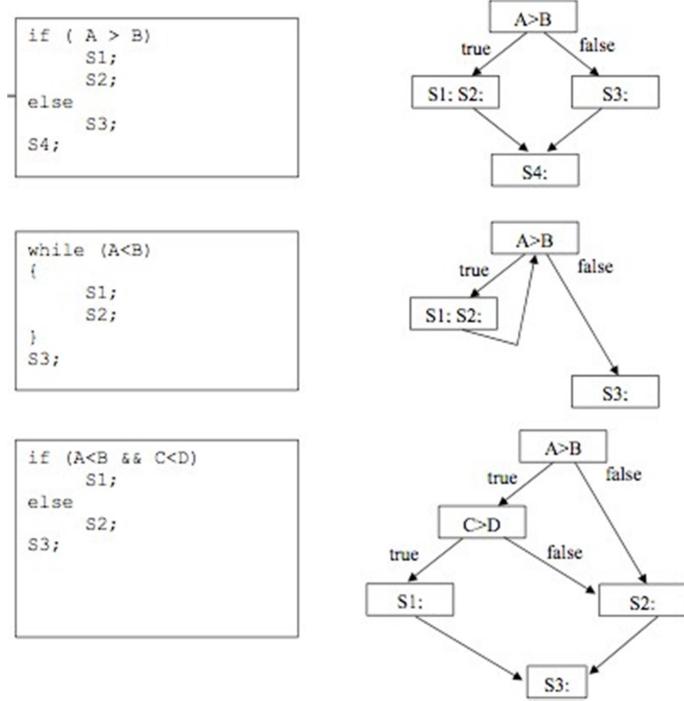
Là phương pháp kiểm tra bao trùm mọi đường dẫn của chương trình.

Nhận xét: phương pháp này phụ thuộc vào các biểu thức điều kiện. Tuy nhiên, có những trường hợp đường dẫn quá lớn (có vòng lặp). Vì vậy, thường không phải là lựa chọn thực tế để tiến hành việc kiểm tra tính đúng đắn của chương trình.

```

while (x > 0 || y > 0)
{
    x--;
    y--;
    z += x*y;
}

```



c. Kiểm tra theo điều kiện (condition testing)

Là phương pháp kiểm tra các điều kiện trên 2 giá trị true, false

```
if (x > 0 && y > 0)
    x = 1;
else
    x = 2;
```

Ví dụ:

Các bộ kiểm tra $\{(x>0, y>0), (x\leq 0, y>0)\}$ sẽ kiểm tra toàn bộ các điều kiện.

Với bộ kiểm tra $\{(x>0)\}$ sẽ kiểm tra bao trùm được các điều kiện. Tuy nhiên, ta không kiểm tra được giá trị y.

d. Kiểm tra theo vòng lặp (loop testing)

Tập trung vào kiểm tra tính hợp lệ của cấu trúc vòng lặp. Với vòng lặp đơn:

bỏ qua vòng lặp

lặp 1 lần

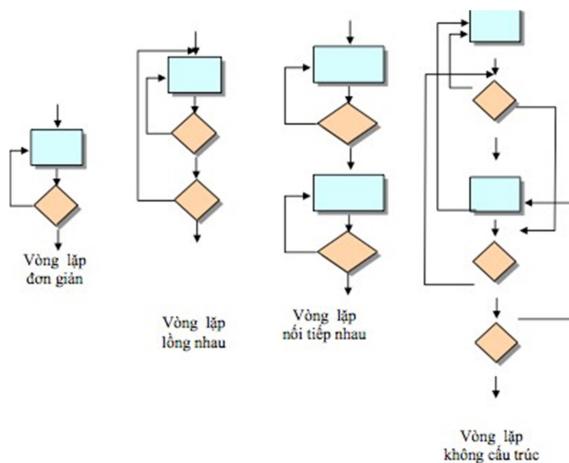
lặp 2 lần

lặp k lần ($k < n$)

lặp $n-1, n, n+1$ lần

Với n là số lần lặp tối

đa.



Với vòng lặp lồng nhau:

khởi đầu với vòng lặp trong cùng. Thiết lập các tham số lặp cho các vòng lặp bên ngoài về giá trị nhỏ nhất

kiểm tra tham số min+1, một giá trị tiêu biểu, max -1, max cho vòng lặp trong cùng khi các tham số lặp của các vòng lặp bên ngoài là nhỏ nhất

tiếp tục với các vòng lặp bên ngoài cho tới khi tất cả các vòng lặp được kiểm tra Ví dụ

```
// LOOP TESTING EXAMPLE PROGRAM import  
java.io.*; class LoopTestExampleApp {  
// ..... FIELDS .....  
public static BufferedReader keyboardInput = new  
BufferedReader(new InputStreamReader(System.in));  
private static final int MINIMUM = 1; private static final int MAXIMUM = 10;  
// ..... METHODS .....
```

```

/* Main method */ public static void main(String[] args) throws
IOException { System.out.println("Input an integer value:");

int input = new
Integer(keyboardInput.readLine()).intValue(); int
numberOfIterations=0;

for(int index=input;index >= MINIMUM && index <= MAXIMUM;index++)
{ numberOfIterations++;

} // Output and end System.out.println("Number of iterations = " + numberOfIterations);

}
}

```

Giá trị đầu vào	Kết quả
11	0 (bỏ qua vòng lặp)

10	1 (lặp 1 lần)
9	2 (lặp 2 lần)
5	6 (lặp k lần)
2	9 (lặp $n - 1$ lần)
1	10 (lặp n lần)
0	0 (bỏ qua vòng lặp)

e. Độ phức tạp của McCabe

Các số liệu đo lường phức tạp được phát triển bởi McCabe (1976) để đo độ phức tạp của một chương trình hoặc mô-đun tại cùng một thời điểm vì nó xác định số lượng tối đa các đường dẫn độc lập cần thiết để thực hiện bao phủ dòng đầy đủ của chương trình. Sự đo lường dựa trên lý thuyết đồ thị và vì thế được tính toán theo đặc điểm chương trình như giành được bằng đồ thị lưu lượng chương trình của nó.

Một đường dẫn độc lập được định nghĩa với tham chiếu để tích lũy chuỗi các đường dẫn độc lập, đó là : “bất kỳ đường dẫn nào trên đồ thị lưu lượng chương trình đều bao gồm ít nhất một cạnh mà không có trong bất kỳ đường dẫn độc lập cũ”

Một số kinh nghiệm nghiên cứu về mối quan hệ giữa số liệu đo lường phức tạp và chất lượng và các đặc điểm kiểm thử đã được tiến hành nhiều năm qua. Một số kết quả được tóm tắt bởi Jones (1996) như sau : “Các nghiên cứu thực nghiệm khám phá ra rằng các chương trình với đo lường phức tạp ít hơn 5 thường được xem là đơn giản và dễ hiểu. Đo lường phức tạp của 10 hoặc thấp hơn được coi là không quá khó; nếu 20 hoặc hơn, độ phức tạp được thấy là cao. Khi giá trị McCabe vượt quá 50, phần mềm cho mục đích thực tế trở thành không thể kiểm nghiệm”. Các báo cáo công bố khác không có thừa nhận của mối quan hệ giữa các số liệu đo lường phức tạp và chất lượng phần mềm hoặc các quan hệ đã tìm thấy đã không được hỗ trợ thống kê.

4.3.3. Kiểm thử gia tăng - Incremental Testing

Test các nhóm tích hợp các mô-đun đã được test với các mô-đun vừa hoàn thành

Được thực hiện bằng cách thêm từng thành phần một, rồi kiểm thử kết quả của hành động thêm này.

4.3.4. Thread Testing

Sử dụng trong kiểm thử tích hợp, để xác minh khả năng của các chức năng chính bằng việc kiểm thử một chuỗi các units mà thực hiện một chức năng của hệ thống.

4.3.5. Bảng tóm tắt Testing Levels/ Techniques

Testing Levels/ Techniques	White-box	Black-box	Incre-mental	Threa d
Unit Testing	X			
Integration Testing	X		X	X
System Testing		X		
Acceptance Testing		X		

4.4. Quá trình kiểm thử

4.4.1. Xác định tiêu chuẩn chất lượng phần mềm phù hợp

Mức tiêu chuẩn chất lượng được chọn cho một dự án phụ thuộc chính vào đặc trưng của phần mềm đó.

Các kiểu thiệt hại điển hình cho khách hàng và người sử dụng cũng như các nhà phát triển được liệt kê trong bảng sau:

a) Thiệt hại cho khách hàng

Kiểu Thiệt hại	Ví dụ

1. Nguy hiểm cho sự an toàn đối với cuộc sống con người	<ul style="list-style-type: none"> Các hệ thống giám sát bệnh nhân trong bệnh viện. Các hệ thống hàng không và vũ trụ. Các hệ thống vũ trang.
2. Ảnh hưởng đến sự hoàn thành của một chức năng tổ chức thiết yếu; không có khả năng thay thế hệ thống có sẵn.	<ul style="list-style-type: none"> - Kinh doanh bán hàng - Hệ thống kiểm kê nhiều cửa hàng trên toàn quốc.
3. Ảnh hưởng chức năng của firmware, nguyên nhân gây sự trực tiếp của toàn bộ hệ thống.	<ul style="list-style-type: none"> - Thiết bị gia đình - Xe ô tô - Thiết bị điện tử hóa
4. Ảnh hưởng sự hoàn thành của chức năng tổ chức thiết yếu nhưng có thay thế cái có sẵn.	<ul style="list-style-type: none"> Các hệ thống bán hàng Font-desk có thể được thay bằng kỹ thuật bằng tay.
5. Ảnh hưởng chức năng chính của các gói phần mềm cho ứng dụng nghiệp vụ.	<ul style="list-style-type: none"> - Thời gian đáp trả chậm cho việc giải quyết một POS(point -of- sale) - Vì một khuyết điểm, thông tin đó thường xuyên được cung cấp trên một màn hình được phân bố giữa 3 màn hình hiển thị khác nhau.
6. Ảnh hưởng Chức năng chính của các gói phần mềm cho một khía cạnh riêng.	<ul style="list-style-type: none"> Các máy trò chơi. Phần mềm giáo dục Bộ xử lý từ.
7. Ảnh hưởng chức năng chính của một ứng dụng firmware nhưng ngoài ảnh hưởng toàn bộ hệ thống.	<ol style="list-style-type: none"> 1. Cúp điện của bảng điều khiển của một thiết bị gia đình bên ngoài sự tồn tại của nó. 2. Hư hỏng của các hệ thống thứ hai.

<p>8. Sự phiền phức của người dùng như khôn n thàn trước g g của g hoà h kh năn hệ thống n ả g</p>	<p>1. Biến dạng nhưng không gây hiểu lầm hiển thị</p> <p>2. Không có khả năng cho đến sản phẩm đầu ra được liệt kê mặc dù thay thế các cách khác để thu thập thông tin cần thiết hoặc thực hiện các hoạt động tương tự đã có sẵn.</p>
----------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

b) Thiệt hại từ người phát triển phần mềm

Kiểu Thiệt hại	Ví dụ
<p>1. Thiệt hại về tài chính</p>	<ul style="list-style-type: none"> • Thiệt hại được trả tiền cho thương tích về thẻ chất. • Thiệt hại được trả tiền các tổ chức cho sự trực trặc của phần mềm. • Mua Chi phí hoàn trả cho khách hàng. • Phí bảo trì cao cho sự tu sửa của hệ thống lỗi .
<p>2. Các thiệt hại ngoài định lượng</p>	<ul style="list-style-type: none"> - Dự kiến sẽ ảnh hưởng đến doanh thu trong tương lai - Giảm thiểu đáng kể doanh số hiện tại

- Xác định rõ chiến lược testing phần mềm

Vấn đề đó được mô tả gồm :

- Chiến lược testing : sẽ một vụ nổ lớn hoặc chiến lược tăng testing được thông qua? Nếu tăng testing là thích hợp hơn, testing nên được thực hiện từ dưới lên hoặc từ trên xuống.
- Các phần nào của lập kế hoạch testing nên được thực hiện theo mô hình testing white box.

- Các phần nào của lập kế hoạch testing nên được thực hiện theo mô hình tự động testing.

4.4.2. Lập kế hoạch cho test

Một số câu hỏi cần trả lời trong quá trình lập kế hoạch:

- Test cái gì ?
- Ai thực hiện test?
- Thực hiện test ở chỗ nào ?
- Khi nào kết thúc test?

-Test cái gì?

Quá trình kiểm thử sẽ giới thiệu đầy đủ kế hoạch kiểm thử, từ test đơn vị (unit test) đến test tích hợp (integration test) và test hệ thống (system test). Các yếu tố quyết định sẽ liên quan đến:

- Các module nào sẽ được unit test
- Cách tích hợp cần được test.
- Xác định rõ mức độ ưu tiên để phân bổ tài nguyên test tới ứng dụng hệ thống phần mềm riêng rẽ.

+ *Dánh giá các unit, integration và các application.*

Các phương thức đánh giá cho các unit(module), integration và các ứng dụng để xác định rõ mức độ ưu tiên của chúng trong kế hoạch testing được dựa trên 2 yếu tố :

Yếu tố A: Mức thiệt hại nghiêm trọng. Độ nghiêm trọng trong trường hợp module hoặc ứng dụng thất bại.

Yếu tố B: Mức rủi ro của phần mềm. Mức độ rủi ro là xác suất xuất hiện thất bại. Để xác định mức rủi ro của một module, unit, integration hoặc ứng dụng, các vấn đề ảnh hưởng của rủi ro cần thẩm tra kỹ. Các vấn đề này có thể được phân loại như các vấn đề module/application và vấn đề người lập trình.

Các vấn đề ảnh hưởng tới mức rủi ro của phần mềm. Các vấn đề module/application

- Độ lớn

- Độ phức tạp và độ khó.
- Tỉ lệ của phần mềm gốc(với tỉ lệ phần mềm được dùng lại)

Vấn đề người lập trình

- Tính chuyên nghiệp
- Kinh nghiệm với các vấn đề của module cụ thể
- Tính khả dụng của hỗ trợ chuyên nghiệp (việc sao lưu các kiến thức và kinh nghiệm)
- Sự hiểu biết của người lập trình và năng lực về khả năng đánh giá .

- Test được thực hiện bởi ai ?

Những người thực thi kiểm thử được xác định ở giai đoạn lập kế hoạch.

- Integration tests, đặc biệt là unit tests, thường được thực hiện bởi nhóm phát triển phần mềm.
- System test thường xuyên được thực hiện bởi một nhóm test độc lập (trong nội bộ nhóm test hoặc có ván bên ngoài nhóm test).
- Trong trường hợp hệ thống phần mềm lớn, nhiều hơn một nhóm testing có thể được dùng để thực hiện các test hệ thống.
- Testing bởi nhóm phát triển khác. Mỗi nhóm phát triển sẽ coi như nhóm testing của dự án được phát triển cho các nhóm khác.

- Test được thực hiện ở đâu ?

Test đơn vị và test tích hợp được thực hiện ở phía người phát triển phần mềm. Khi test hệ thống: chúng nên thực hiện ở phía người phát triển hay phía khách hàng? Nếu test hệ thống là để được thực hiện bởi các tư vấn test bên ngoài, một lựa chọn thứ 3 phát sinh: phía người tư vấn. Lựa chọn phụ thuộc vào môi trường test hay môi trường hệ thống. Tuy nhiên, môi trường máy tính ở phía khách hàng khác có thể khác với phía nhà phát triển. Trong trường hợp này, để giảm những lo ngại của khách hàng, hệ thống sẽ được cài đặt và kiểm thử ở phía khách hàng và không cần test chấp nhận nữa.

- Khi nào dùng test?

Quyết định khi nào quá trình kiểm thử kết thúc có ý nghĩa quan trọng nhất là với test hệ thống. Có nhiều hướng lựa chọn để có thể dùng test, mỗi lựa chọn dựa trên

tiêu chí khác nhau, ví dụ như:

- Hướng hoàn tất thực thi . Theo hướng này, testing được hoàn thành khi toàn bộ kế hoạch test đã được thực hiện xong và sạch lỗi, kết quả đạt được cho tất cả các yêu cầu test. Đây là hướng lý tưởng, khi mà ta không bị giới hạn bởi ngân sách và thời gian.
- Hướng áp dụng mô hình toán học. Ở hướng này, mô hình toán học được áp dụng để ước lượng tỉ lệ lỗi không bị phát hiện trên tỉ lệ lỗi được phát hiện. Testing sẽ hoàn thành khi tỉ lệ phát hiện lỗi giảm dưới mức chấp nhận được theo một chuẩn nào đó. Những bất lợi của hướng này là mô hình tính toán được lựa chọn có thể không đại diện hoàn toàn cho đặc trưng của dự án.
- Hướng các đội test độc lập. Hai đội test thực hiện test một cách độc lập. Bằng cách so sánh danh sách các lỗi được phát hiện của mỗi đội có thể đưa ra quyết định dừng quá trình test.
- Dừng khi hết tài nguyên. Với kiểu này việc dừng test sẽ xảy ra khi ngân sách hoặc thời gian được phân bổ cho testing đã hết.

4.4.3. Thiết kế kiểm thử (test design)

Các sản phẩm của giai đoạn thiết kế test là :

- Thiết kế chi tiết và các Thủ tục cho mỗi việc test.
- Các tệp hoặc cơ sở dữ liệu của các test case.

Sau đây là một mẫu (template) kế hoạch kiểm thử phần mềm:

Mẫu Kế hoạch test phần mềm (STP)

- Phạm vi của test
 - 1. 1 Gói phần mềm được test(tên, phiên bản, rà soát)
 - 1. 2 Tài liệu cơ sở cho các kiểm thử đã lên kế hoạch (tên và phiên bản cho mỗi tài liệu)
 - Môi trường Testing
 - 2. 1 Nơi test (phía khách hàng, người phát triển hoặc ...)
 - 2. 2 Cấu hình phần sụn (firmware) và phần cứng được yêu cầu.
 - 2. 3 Các tổ chức cùng tham gia
 - 2. 4 Nhu cầu nhân lực

2. 5 Sự chuẩn bị và đào tạo nhóm kiểm thử được yêu cầu

- Chi tiết kiểm thử

3. 1 Định danh test (test identification)

3. 2 Mục tiêu test

3. 3 Các tài liệu tham khảo chéo liên quan đến tài liệu thiết kế và tài liệu yêu cầu.

3. 4 Lớp test

3. 5 Mức test (unit , integration hoặc system tests)

3. 6 Các yêu cầu test case

3. 7 Các yêu cầu đặc biệt (ví dụ , đo thời gian phản ứng , đảm bảo yêu cầu)

3. 8 Dữ liệu được ghi lại

- Lập lịch Test (cho mỗi test hoặc nhóm test) bao gồm việc ước lượng thời gian cho những công việc sau đây:

4.1 Sự chuẩn bị

4.2 Testing

4.3 Sửa lỗi

4.4 Test hồi quy

Quá trình thiết kế test được tiến hành trên cơ sở của kế hoạch test phần mềm, lập tài liệu bởi STP (software test procedure). Các thủ tục test và cơ sở dữ liệu hay tệp test case có lập tài liệu trong một tài liệu “thủ tục test phần mềm” và “ test case file” hoặc trong một tài liệu đơn gọi là “mô tả test phần mềm” (STD). Một mẫu (template) của STD được trình bày như sau:

Mẫu mô tả test phần mềm (STD)

- Phạm vi của test

1.1 Các gói phần mềm được test (name, version, và revision)

1.2 Các tài liệu cơ bản được cung cấp cho việc thiết kế test (tên và phiên bản của từng tài liệu)

- Môi trường test (của mỗi test)

2.1 Định danh test (các chi tiết test được viết tài liệu trong STP)

2.2 Mô tả chi tiết về hoạt động hệ thống và cấu hình phần cứng và yêu cầu chuyển đổi thiết lập test.

2.3 Hướng dẫn tải phần mềm.

- Tiến trình test

3.1 Hướng dẫn cho đầu vào, mô tả chi tiết mỗi bước của tiến trình đầu vào.

3.2 Dữ liệu được ghi chép trong suốt quá trình test.

- Test cases (với mỗi case)

4.1 Chi tiết định danh test case

4.2 Dữ liệu Đầu vào và thiết lập hệ thống.

4.3 Kết quả trung gian mong đợi (nếu có)

4.4 Các kết quả mong đợi (Số, thông điệp, sự kích hoạt thiết bị,...)

- Hành động trong trường hợp chương trình lỗi / kết thúc

- Thủ tục được áp dụng theo bản tóm tắt kết quả test.

- Hình dưới đây là một ví dụ về mẫu mô tả test case lấy từ công ty phần mềm Fsoft

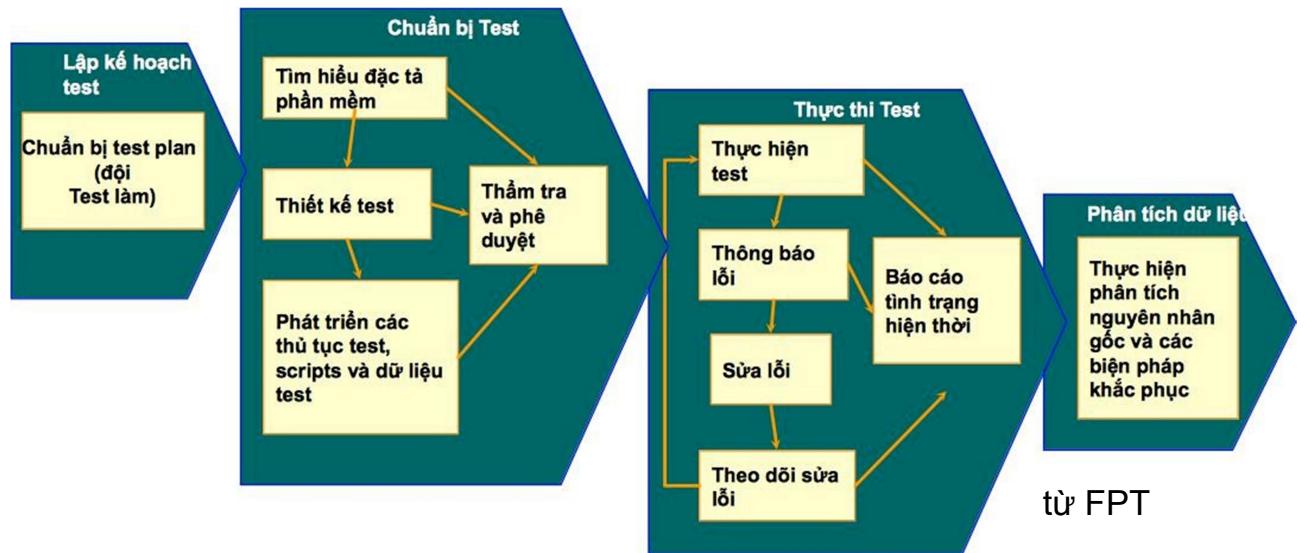
Create Test Case



Requirement *	Flow ID	Status *	Priority *
Data common		Pending	
Test Title *	Est. Time (min) *	Velocity Point	Type *
Pre-requisite	Comments		
		Test Step	Expected Result
1	Open BTA Viewer	BTA Viewer is opened successfully	
	Open data file	Data file is opened by BTA Viewer out of error	
2			
		Save	Cancel

4.4.4. Tiến trình test

Tùy vào từng tổ chức, hệ thống, ngữ cảnh, mức độ rủi ro của phần mềm mà quy trình kiểm thử phần mềm có thể gồm nhiều bước khác nhau. Nhưng nhìn chung mọi quy trình kiểm thử đều có những bước cơ bản như dưới đây:



Theo đó một quy trình kiểm thử phần mềm cơ bản gồm 4 giai đoạn:

- Lập kế hoạch kiểm thử:** Nhiệm vụ quan trọng trong phần lập kế hoạch kiểm thử là xác định được các thành phần sau:
 1. Các giai đoạn kiểm thử áp dụng cho dự án phần mềm.
 2. Các phương pháp kiểm thử.
 3. Các công cụ kiểm thử.
 4. Nguồn lực kiểm thử.
 5. Môi trường kiểm thử bao gồm tài nguyên phần cứng và phần mềm.
 6. Mốc bàn giao các tài liệu kiểm thử.
- Chuẩn bị kiểm thử:** Nhiệm vụ của giai đoạn này là:
 1. Tìm hiểu nghiệp vụ của hệ thống phải kiểm thử.

2. Xây dựng kịch bản kiểm thử.
3. Chuẩn bị dữ liệu kiểm thử.
4. Xem xét phê duyệt các tài liệu kiểm thử.

Thực thi kiểm thử:

Thực hiện kiểm thử dựa trên các kịch bản kiểm thử, test case, thủ tục, dữ liệu có sẵn từ bước chuẩn bị kiểm thử.

Thực hiện quá trình quản lý lỗi: báo lỗi, sửa lỗi.

Báo cáo và phân tích dữ liệu kiểm thử:

- Lập báo cáo kiểm thử.

- Phân tích nguyên nhân và đề xuất các hành động khắc phục.

b. Input/Output cho test

Input:

Yêu cầu của khách hàng và tiêu chí chấp nhận

Yêu cầu về thay đổi

Đặc tả yêu cầu phần mềm (Software Requirement Specification SRS)

Tài liệu thiết kế

Chương trình (Modules)

Output:

Tài liệu test: kế hoạch test, test cases và procedures, Test script, dữ liệu Test

Danh sách lỗi

Mô tả thực hiện test

Báo cáo phân tích lỗi

4.4.5. Thiết kế trường hợp kiểm thử (Test Case Design)

- Các thành phần dữ liệu Test case

Mỗi test case là một tài liệu tập hợp của các dữ liệu đầu vào và các điều kiện hoạt động được yêu cầu để thực hiện một mục test cùng với kết quả được mong đợi. Người kiểm thử mong đợi sẽ chạy một chương trình cho mỗi mục test theo tài liệu được cung cấp trong trường hợp test, và so sánh các kết quả thực tế với kết quả mong đợi được ghi lại trong các tài liệu. Nếu kết quả thu được hoàn toàn phù hợp

với kết quả mong đợi, ko có lỗi hoặc ít nhất đã được xác định. Khi một số hoặc tất cả các kết quả đều không phù hợp với các kết quả mong đợi, một lỗi tiềm tàng được xác định. Phương pháp phân vùng các lớp tương đương được áp dụng để giành được hiệu quả và xác định tính hiệu quả của test case , như thiết lập, được sử dụng trong hộp đen Test.

- **Nguồn test case**

Có 2 Nguồn cơ bản cho test case :

- Lấy mẫu ngẫu nhiên các case trong đời sống thực, ví dụ:
- Một mẫu về các hộ gia đình đô thị (test thông tin về hệ thống thuế một thành phố)
- Một mẫu về hóa đơn vận chuyển (test phần mềm billing mới)
- Một mẫu về hồ sơ kiểm soát (test phần mềm mới để kiểm soát sự sản xuất của nhà máy)
- Một mẫu ghi lại những sự kiện sẽ được “chạy” như một test case (test trực tuyến các ứng dụng của một trang web internet , và của các ứng dụng thực)
- Tổng hợp các test case (hay kịch bản test) được chuẩn bị bởi người thiết kế test. Kiểu test case này kết hợp các điều kiện hoạt động của hệ thống và các tham số (được định nghĩa bởi một tập các dữ liệu đầu vào). Sự kết hợp này được thiết kế để bao gồm tất cả sự hiểu biết về trạng thái hoạt động phần mềm hoặc ít nhất tất cả các trạng thái được mong đợi để thường xuyên sử dụng hoặc thuộc về một lớp có khả năng lỗi cao.

4.5 Phần mềm phục vụ kiểm thử

Các phần mềm phục vụ kiểm thử là các phần mềm được sử dụng trong suốt quá trình kiểm thử, bao gồm từ giai đoạn lên kế hoạch kiểm thử đến giai đoạn tổng hợp báo cáo kết quả kiểm thử. Dựa vào mục đích sử dụng, ta có thể phân loại các phần mềm phục vụ kiểm thử thành 3 nhóm chính:

- Phần mềm hỗ trợ viết tài liệu.
- Phần mềm quản lý lỗi.
- Công cụ kiểm thử tự động.

- ***Phần mềm hỗ trợ viết tài liệu***

Các phần mềm hỗ trợ viết tài liệu thường được sử dụng trong việc lên kế hoạch kiểm thử

- *Test Plan*, thiết kế và tạo ra các ca kiểm thử - *Test Case*, viết các báo cáo, các hướng dẫn sử dụng cho người dùng ...

Các phần mềm thường được sử dụng là :

- Microsoft Office Word.
- Microsoft Office Exel.
- Microsoft Office Project.
- Microsoft Office Power Point.

- ***Phần mềm quản lý lỗi***

Một lỗi phần mềm là một lỗi, thiếu sót, thất bại, hoặc lỗi trong một chương trình máy tính hoặc hệ thống sản xuất một kết quả không chính xác hoặc không mong muốn, hoặc làm cho nó hành xử theo những cách không mong muốn. Hầu hết các lỗi phát sinh từ những sai lầm và lỗi của con người trong các đoạn mã nguồn của một chương trình hoặc trong các thiết kế, và một số được gây ra bởi các trình biên dịch mã không chính xác. Chính vì thế, trong các dự án Công nghệ thông tin cần có một hệ thống theo dõi lỗi để giúp theo dõi và báo cáo các lỗi trong quá trình phát triển phần mềm.

Các thành phần chính của một hệ thống theo dõi lỗi là một cơ sở dữ liệu ghi lại những thông tin về lỗi được phát hiện như : thời gian phát hiện lỗi, mức độ nghiêm trọng của lỗi, cách gỡ lỗi ...

Dưới đây là một số loại phần mềm theo dõi lỗi phổ biến:

- Bugzilla*

Bugzilla là một phần mềm máy chủ cho phép quản lý các lỗi phát sinh trong quá trình phát triển dự án phần mềm được phát triển bởi tổ chức Mozilla. Các tính năng:

- Cho phép khai báo các lỗi mới phát hiện.

- Phân loại các lỗi theo thành phần hệ thống, độ phức tạp, mức độ ưu tiên.
- Hệ thống quản lý cho phép một người khai báo lỗi và giao trách nhiệm sửa lỗi cho một người khác.
- Cho phép quản lý quá trình hoạt động cũng như tiến độ test lỗi từng dự án.
- Cho phép nhiều user làm việc cùng lúc, dễ tìm kiếm và phân bổ công việc cho từng thành viên.
- Cập nhật thông tin cho thành viên tham gia dự án thông qua chức năng gửi thư điện tử.

Link trang chủ: <http://www.bugzilla.org/>

- Redmine*

Redmine là phần mềm nguồn mở hữu ích cho việc quản trị dự án với rất nhiều tiện ích hỗ trợ. Các tính năng:

Quản lý dự án bao gồm cả biểu đồ Gantt.

Hỗ trợ đồng thời nhiều dự án cùng với các dự án nhỏ bên trong. Kiểm soát truy cập linh hoạt theo quyền hạn.

Quản lý theo từng thời gian cụ thể.

Cho phép tùy chỉnh theo từng lĩnh vực. Quản lý file, tin tức tài liệu.

Hỗ trợ nhiều database.

Link trang chủ: <http://www.redmine.org/>

- Atlassian JIRA*

JIRA là một ứng dụng theo dõi và quản lý lỗi, vấn đề và dự án, được phát triển để làm quy trình này trở nên dễ dàng hơn cho mọi tổ chức. JIRA đã được thiết kế với trọng tâm vào kết quả công việc, có thể sử dụng ngay và linh hoạt khi sử dụng. Các tính năng:

Quản lý lỗi, tính năng, công việc, những cải tiến hoặc bất

kỳ vấn đề gì.
các phiên
bản.

thống kê thời gian thực.

hành và cơ sở dữ liệu.

Theo dõi các tệp gắn, những thay đổi, các câu phản và Bảng phân tích đồ họa có thể tùy biến và các số liệu Để dàng mở rộng và tích hợp với các hệ thống khác.

Có thể chạy trên hầu hết các nền tảng phần cứng, hệ điều

Dịch vụ Web cho phép kiểm soát hệ thống.

Link trang chủ: <http://www.atlassian.com>

- ***Công cụ kiểm thử tự động***

a. Công cụ kiểm thử tự động là gì ?

Công cụ kiểm thử tự động là các công cụ giúp thực hiện việc kiểm thử phần mềm một cách tự động.

Ý nghĩa của các công cụ kiểm thử tự động:

1. Giảm bớt công sức và thời gian thực hiện quá trình kiểm thử .
2. Tăng độ tin cậy.
3. Giảm sự nhảm chán cho con người.
4. Rèn luyện kỹ năng lập trình cho kiểm thử viên.
5. Giảm chi phí cho tổng quá trình kiểm thử.

Thuận lợi và khó khăn cơ bản khi áp dụng công cụ kiểm thử tự động:

1. Không cần can thiệp của kỹ thuật viên.
2. Giảm chi phí khi thực hiện kiểm tra số lượng lớn test case hoặc test case lặp lại nhiều lần.

3. Giả lập tình huống khó có thể thực hiện bằng tay.
4. Mất chi phí tạo các script để thực hiện kiểm thử tự động.
5. Tốn chi phí dành cho bảo trì các script.
6. Đòi hỏi kỹ thuật viên phải có kỹ năng tạo script kiểm thử tự động.
7. Không áp dụng được trong việc tìm lỗi mới của phần mềm.

Quy trình kiểm thử tự động

1. Tạo kịch bản kiểm thử

Giai đoạn này chúng ta sẽ dùng test tool để ghi lại các thao tác lên phần mềm cần kiểm tra và tự động sinh ra kịch bản kiểm thử - *test script*.

2. Chính sửa kịch bản kiểm thử

Chỉnh sửa để test script thực hiện kiểm tra theo đúng yêu cầu đặt ra, cụ thể là làm theo test case cần thực hiện.

3. Chạy test script để kiểm thử tự động

Chạy kịch bản kiểm thử để kiểm tra phần mềm có đưa ra đúng như kết quả mong muốn không.

4. Đánh giá kết quả

Kiểm tra kết quả thông báo sau khi thực hiện kiểm thử tự động. Sau đó bổ sung, chỉnh sửa những sai sót.

b. Phân loại các công cụ kiểm thử

Mỗi một công cụ kiểm thử chỉ có thể hỗ trợ một khía cạnh nào đó của thử nghiệm. Các công cụ có thể được phân loại dựa trên một số tiêu chí như mục đích, thương mại / miễn phí / mã nguồn mở / phần mềm chia sẻ, công nghệ sử dụng, hoạt động của mỗi công cụ... Trong báo cáo này, tôi phân loại các công cụ kiểm thử theo mục đích của mỗi loại công cụ kiểm thử. Theo đó, ta có thể phân loại các công cụ kiểm thử tự động thành các loại sau

: Unit Testing Tools, Regression Testing Tools, Functional Testing Tools, Performance Testing Tools, ...

1.

Công cụ kiểm thử đơn vị - Unit Testing Tools

Kiểm thử đơn vị - *Unit Testing* là một cách tiếp cận kiểm tra các đơn vị cá nhân của mã nguồn và kiểm tra nếu nó phù hợp với mục đích. Kiểm thử đơn vị cho phép các lập trình viên để sửa đổi và duy trì mã hiện có và vẫn đảm bảo chắc chắn rằng các mô-đun hoạt động chính xác và đáp ứng các yêu cầu chức năng và không có chức năng dự kiến. Nó cũng giúp làm giảm sự không chắc chắn trong các đơn vị và đặc biệt hữu ích trong một cách tiếp cận từ dưới lên phong cách thử nghiệm.

Một số loại công cụ kiểm tra đơn vị:

1. JUnit.

2. Jtest: là một thử nghiệm java tự động và mã tĩnh phân tích sản phẩm được thực hiện bởi Parasoft. Nó nhằm mục đích nâng cao độ tin cậy, tính năng, bảo mật, hiệu suất, và bảo trì. Chức năng cơ bản bao gồm kiểm tra mức đơn vị, phân tích tĩnh, kiểm tra hồi quy, phát hiện lỗi thời gian chạy, và xem xét mã.

Link trang chủ: <http://www.parasoft.com>.

1. Sonar: là một nền tảng quản lý chất lượng nguồn mở Java, dành riêng cho liên tục phân tích và đo lường chất lượng mã nguồn từ các danh mục đầu tư dự án với phương thức lớp.

Link trang chủ: <http://www.sonatype.com/>

2. Công cụ kiểm thử hồi quy - Regression Testing Tools

Nhiều công cụ khác nhau có thể được sử dụng cho hồi quy. Nó giúp bạn tự động kiểm tra như các chương trình có thể được tái sử dụng.

Một số công cụ kiểm tra hồi quy :

2. Selenium IDE.

3. SilkTest: sử dụng ngôn ngữ 4Test độc quyền cho các kịch bản tự động hóa. Đây là một định hướng đối tượng ngôn ngữ tương tự như C++. Nó sử dụng các khái niệm về lớp, đối tượng, và thùng kín.

Link trang chủ: <http://www.borland.com/products/silktest/>

QA WIZARD: QA Wizard Pro tự động hóa chức năng kiểm tra và hồi quy của Web và các ứng dụng Windows.

Link trang chủ: <http://www.qawizard.com/>

3. *Công cụ kiểm thử chức năng - Functional Testing Tools*

Đây là những công cụ giúp thử nghiệm hệ thống. Một số công cụ kiểm tra chức năng:

1. **Selenium HQ.**

2. ActiWATE: mô phỏng hành động dựa trên Brower và làm cho nó có thể tạo ra các kịch bản dựa trên hành động / kết quả. Nó hỗ trợ HTTP, HTTPS & Ajax.

4. *Công cụ kiểm thử hiệu năng - Performance Testing Tools*

Kiểm thử hiệu năng được thực hiện để xác định hệ thống thực hiện một khối lượng công việc cụ thể nhanh thế nào. Nó cũng có thể dùng để xác nhận và xác minh những thuộc tính chất lượng khác của hệ thống như: khả năng mở rộng, độ tin cậy, sử dụng tài nguyên.

Load testing là khái niệm chủ yếu của việc kiểm thử mà có thể tiếp tục hoạt động ở một mức tải cụ thể, cho dù đó là một lượng lớn dữ liệu hoặc lượng lớn người sử dụng.

Volume testing là một cách kiểm tra chức năng. Stress testing là một cách để kiểm tra tính tin cậy. Load testing là một cách để kiểm tra hiệu năng. Đây là một số thỏa thuận về các mục tiêu cụ thể của load testing. Những thuật ngữ như load testing, performance testing, reliability testing, và volume testing thường sử dụng thay thế cho nhau.

Một số công cụ kiểm tra hiệu năng:

1. **Loadrunner:** là công cụ kiểm tra tải / căng thẳng cho các trang web và các ứng dụng khác, hỗ trợ một loạt các môi trường ứng dụng, nền tảng, và cơ sở dữ liệu. Link trang chủ: <http://www8.hp.com>

2. Pylot: là một công cụ mã nguồn mở được chạy thử nghiệm tải HTTP để thử nghiệm hiệu suất và khả năng mở rộng các dịch vụ web. Nó tạo ra đồng thời tải, xác minh trả lời máy chủ, và tạo ra các báo cáo với số liệu.

3. Apache JMeter : có thể được sử dụng để thử nghiệm hiệu suất cả về tài

nguyên tĩnh và động. Link trang chủ: <http://jmeter.apache.org/>

Ngoài ra còn một số các nhóm công cụ kiểm thử tự động khác như:

- o Công cụ quản lý sự cố – *Incident Management Tools*.
- o Công cụ quản lý cấu hình – *Configuration Management Tools*.
- o Công cụ bảo hiểm đo lường – *Coverage Measurement Tools*.
- o Công cụ giám sát – *Monitoring Tools*.
- o Công cụ kiểm thử cơ sở dữ liệu – *Database Testing Tools*.
- o Công cụ kiểm thử bảo mật – *Security Testing Tools*.
- o Công cụ kiểm thử tĩnh – *Static Testing Tools*.

4.6 Unit test và thư viện JUnit

• *Tổng quan về Unit Testing*

a. Định nghĩa về Unit test

Một **unit** là phần nhỏ nhất có thể test được của chương trình. Trong lập trình hướng đối tượng, một **unit** có thể là một method của một class.

Unit Test là các đoạn mã có cấu trúc giống như các đối tượng được xây dựng để kiểm tra từng bộ phận trong hệ thống. Mỗi Unit Test sẽ gửi đi một thông điệp và kiểm tra câu trả lời có đúng hay không, bao gồm:

- Các kết quả trả về mong muốn.
- Các ngoại lệ mong muốn.

Unit Test kiểm tra tính đúng đắn của các hoạt động của các thành phần đơn vị với một quy trình tách biệt với quy trình phát triển phần mềm, giúp phát hiện sai sót kịp thời. Unit Test còn có thể giúp phát hiện các vấn đề tiềm ẩn và các lỗi thời gian thực ngay cả trước khi chuyên viên kiểm định chất lượng (QA - Quality Assurance) tìm ra, thậm chí có thể sửa lỗi ngay từ ý tưởng thiết kế.

b. Đặc điểm của một Unit Test

- Đóng vai trò như là người sử dụng đầu tiên của hệ thống.
- Chỉ có giá trị khi chúng phát hiện ra các lỗi tiềm ẩn hoặc lỗi kĩ thuật.

- Các đoạn mã Unit Test hoạt động liên tục hoặc định kỳ để thăm dò và phát hiện các lỗi kỹ thuật trong suốt quá trình phát triển, do đó Unit Test còn được gọi là kỹ thuật kiểm nghiệm tự động.

- Unit Test có 3 trạng thái cơ bản:

Fail (trạng thái lỗi).

Ignore (tạm ngừng thực hiện). Pass (trạng thái làm việc).

Do việc kiểm thử đơn vị đòi hỏi phải kiểm tra từng nhánh lệnh, nên đòi hỏi người kiểm thử có kiến thức về lập trình cũng như về thiết kế của hệ thống nên người thực hiện thường là lập trình viên.

c. Quy trình thiết kế hoạt động của Unit Test

Mỗi một Unit Test đều được thiết kế theo trình tự sau:

- Chuẩn bị các ca kiểm thử (*Test case*) hoặc kịch bản kiểm thử (*Test script*), trong đó chỉ định rõ dữ liệu đầu vào, các bước thực hiện và dữ liệu đầu ra mong muốn. Các Test case và Test script này nên được giữ lại để tái sử dụng.
- Thiết lập các điều kiện cần thiết: khởi tạo các đối tượng, xác định tài nguyên cần thiết, xây dựng các dữ liệu giả lập...
- Triệu gọi các phương thức cần kiểm tra.
- Kiểm tra sự hoạt động đúng đắn của các phương thức.
- Dọn dẹp tài nguyên sau khi kết thúc kiểm tra.

Unit Test chỉ hoạt động hiệu quả khi:

- Được vận hành lặp lại nhiều lần.
- Tự động hoàn toàn.
- Độc lập với các Unit Test khác.

d. Lợi ích của Unit Test

- Tách biệt mã kiểm thử ra khỏi mã chương trình.
- Tạo ra môi trường lý tưởng để kiểm tra bất kỳ đoạn mã nào, có khả năng thăm dò và phát hiện lỗi chính xác, duy trì sự ổn định của toàn bộ phần mềm và

giúp tiết kiệm thời gian so với công việc gỡ rối truyền thống.

- Tự động hóa việc tổ chức và thi hành các bộ số liệu kiểm thử, giải phóng chuyên viên QA khỏi các công việc kiểm tra phức tạp.
- Cố lập từng phần của chương trình và đảm bảo những phần đó được test chạy đúng như yêu cầu, phát hiện các lỗi nghiêm trọng có thể xảy ra trong những tình huống rất hẹp.
- Phát hiện các thuật toán thực thi không hiệu quả, các thủ tục chạy vượt quá giới hạn thời gian
- Khi làm việc team, nếu mỗi phần do từng thành viên viết được test unit kỹ càng thì khi kết hợp lại sẽ suôn sẻ, ít gặp lỗi hơn.
- Là công cụ đánh giá năng lực của lập trình viên. Số lượng các tình huống kiểm tra (test case) chuyển trạng thái "pass" sẽ thể hiện tốc độ làm việc, năng suất của bạn.

Bên cạnh các lợi ích trên, Unit Test cũng có một số điểm hạn chế:

- Unit Test không bắt được tất cả các lỗi của chương trình, những test case chỉ kiểm lỗi những unit nhỏ nhất của chương trình, cho nên không lường trước những vấn đề có thể xảy ra khi kết hợp các module lại với nhau.
 - Có nhiều trường hợp không thể sử dụng Unit Test được, ví dụ với private class, private method, ...
- e. Chiến lược viết mã Unit Test hiệu quả
- Phân tích các tình huống có thể xảy ra đối với mã. Không được bỏ qua các tình huống tồi tệ nhất có thể xảy ra, thí dụ dữ liệu nhập làm một kết nối cơ sở dữ liệu thất bại, ứng dụng bị treo vì một phép toán chia cho không, các thủ tục đưa ra lỗi ngoại lệ sai có thể phá hỏng ứng dụng một cách bí ẩn...
 - Mọi Unit Test phải bắt đầu với trạng thái "fail" và chuyển trạng thái "pass" sau một số thay đổi hợp lý đối với mã chính.
 - Nhập một số lượng đủ lớn các giá trị đầu vào để phát hiện điểm yếu của mã theo nguyên tắc:
 - Nếu nhập giá trị đầu vào hợp lệ thì kết quả trả về cũng phải hợp lệ.

~~Nếu~~ Nếu nhập giá trị đầu vào không hợp lệ thì kết quả trả về phải không hợp lệ.

- o Sớm nhận biết các đoạn mã không ổn định và có nguy cơ gây lỗi cao, viết Unit Test tương ứng để khống chế.
- o Đòi hỏi sự nỗ lực, kinh nghiệm và sự sáng tạo như viết phần mềm.

- **Tổng quan thư viện Junit**

a. Junit là gì?

- Junit là một framework đơn giản dùng cho việc tạo các Unit Testing tự động và chạy các test case có thể lặp đi lặp lại.
- Junit được xây dựng bởi Erich Gamma và Kent Beck, hai người nổi tiếng nhất về lập trình XP.
- Junit là một phần của họ kiến trúc Xunit cho việc tạo các Unit Testing và cũng là một chuẩn trên thực tế cho Unit Testing trong Java.

b. Đặc điểm của Junit

- **Junit** là công cụ giúp ta thử nghiệm, gỡ rối chương trình Java. Với Junit bạn dễ dàng theo dõi diễn biến của chương trình, nhanh chóng dàn dựng hàng loạt phép thử (test case) để kiểm tra mọi việc có xảy ra đúng như dự định hay không.
- Các test case của JUnit là các lớp của Java, các lớp này bao gồm một hay nhiều các phương thức unit testing, và những test này lại được nhóm thành các Test Suite.
- JUnit có những đặc điểm quan trọng như sau:
 - Xác nhận (assert) việc kiểm tra kết quả được mong đợi.
 - Các Test Suite cho phép chúng ta dễ dàng tổ chức hay chạy các test.
 - Hỗ trợ giao diện đồ họa hay giao diện dòng lệnh.

Các test trong JUnit có thể là các test được chấp nhận hay thất bại, các test này được thiết kế để khi chạy mà không cần có sự can thiệp của con người. Từ những thiết kế như thế, bạn có thể thêm các bộ test vào quá trình tích hợp và xây dựng

phần mềm một cách liên tục và để cho các test chạy một cách tự động.

c. Lợi ích của Junit

- o JUnit tránh cho người lập trình phải làm đi làm lại những việc kiểm thử nhằm chán bằng cách tách biệt mã kiểm thử ra khỏi mã chương trình.
- o Tự động hóa việc tổ chức và thi hành các bộ số liệu kiểm thử.

Thoạt tiên, khi sử dụng JUnit, ta đều có thể có cảm giác là JUnit chỉ làm mất thêm thời gian cho việc kiểm thử. Thay vì phải viết thêm các lớp và phương thức mới phục vụ cho công tác kiểm thử, ta có thể soạn nhanh một bộ số liệu rồi viết ngay vào trong phương thức **main()** và quan sát ngay kết quả kiểm thử. Nhưng khi tổ chức lại chương trình cho hợp lý hơn hoặc khi phải thay đổi chương trình để phục vụ cho nhu cầu mới, các bộ số liệu kiểm thử trước đây sẽ cần được sử dụng lại để chắc chắn rằng những thay đổi trong chương trình không làm phương hại đến những thành quả trước đó, lúc này ta sẽ phải mất thời gian để tìm hiểu lại xem bộ số liệu trước đây sẽ tương ứng với kết xuất gì vì ta không thể nhớ hết mọi hoạt động kiểm thử đã diễn ra, nếu dữ liệu test lớn thì việc nhớ đó hoàn toàn bất khả thi. Và việc sử dụng Junit sẽ hoàn toàn khắc phục được những yếu điểm đó.

d. Một số phương thức trong Junit

- o **Các phương thức assertXXX()**

Các phương thức dạng assertXXX() được dùng để kiểm tra các điều kiện khác nhau. Dưới đây là mô tả các phương thức assertXXX() khác nhau có trong lớp junit.framework.Assert:

- o **Boolean assertEquals():** So sánh hai giá trị để kiểm tra bằng nhau. Phép thử thất bại nếu hai giá trị không bằng nhau.
- o **Boolean assertFalse():** Đánh giá biểu thức logic. Phép thử thất bại nếu biểu thức đúng.
- o **Boolean assertNotNull():** So sánh tham chiếu của một đối tượng với Null. Phép thử thất bại nếu tham chiếu đối tượng Null.
- o **Boolean assertNotSame():** So sánh địa chỉ vùng nhớ của hai tham chiếu hai đối tượng bằng cách sử dụng toán tử ==. Phép thử thất bại trả về nếu cả hai đều tham chiếu đến cùng một đối tượng.
- o **Boolean assertNull():** So sánh tham chiếu của một đối tượng với giá trị Null. Phép thử thất bại nếu đối tượng không là Null.

- o **Boolean assertEquals():** So sánh địa chỉ vùng nhớ của hai tham chiếu đối tượng bằng cách sử dụng toán tử `==`. Phép thử thất bại nếu cả hai không tham chiếu đến cùng một đối tượng.
- o **Boolean assertTrue():** Đánh giá một biểu thức logic. Phép thử thất bại nếu biểu thức sai.
- o **fail():** Phương thức này làm cho test hiện tại thất bại, phương thức này thường được sử dụng khi xử lý các ngoại lệ.

Tất cả các phương thức trên đều có thể nhận vào một String không bắt buộc làm tham số đầu tiên. Khi được xác định, tham số này cung cấp như một thông điệp thất bại giúp cho việc sửa lỗi được dễ dàng hơn.

Ví dụ phương thức test hai xâu có trùng nhau không:

```
@Test
public void Test(){
    String s1 = " xâu 1";
    String s2 = " xâu 2";

    assertEquals(s1, s2);
}
```

Hoặc ta thêm thông điệp “Hai xâu khác nhau” làm tham số đầu tiên của phương thức assertEquals() nhằm cung cấp thông điệp lỗi sai rõ ràng hơn.

```
@Test
public void Test(){
    String s1 = " xâu 1";
    String s2 = " xâu 2";
    assertEquals("Hai xâu khác nhau :", s1, s2);
}
```

o **setUp() và tearDown()**

Hai phương thức này là một phần của lớp junit.framework.TestCase. Khi sử dụng hai phương thức này sẽ giúp chúng ta tránh được việc trùng mã khi nhiều test cùng chia sẻ nhau ở phần khởi tạo và dọn dẹp các biến.

JUnit tuân thủ theo một dây có thứ tự các sự kiện khi chạy các test. Đầu tiên, nó tạo ra một thể hiện mới của Test Case ứng với mỗi phương thức thử. Từ đó, nếu bạn có 5 phương thức thử thì JUnit sẽ tạo ra 5 thể hiện của Test Case. Vì lý do đó, các biến thể hiện không thể được sử dụng để chia sẻ trạng thái giữa các phương thức test. Sau khi tạo xong tất cả các đối tượng test case, JUnit tuân theo các bước sau cho mỗi phương thức test:

1. Gọi phương thức setUp() của test case.
2. Gọi phương thức thử.
3. Gọi phương thức tearDown() của test case.

Quá trình này được lặp lại đối với mỗi phương thức thử trong Test Case.

Thông thường chúng ta có thể bỏ qua phương thức tearDown() vì mỗi phương thức thử riêng không phải là những tiến trình chạy tốn nhiều thời gian.

Chương 5. Các thành phần cơ bản của chất lượng phần mềm

5.1 Thủ tục, chỉ dẫn và các thiết bị hỗ trợ chất lượng

5.1.1 Các thủ tục và chỉ dẫn

Khái niệm thủ tục (procedure) và chỉ dẫn (instruction):

- Thủ tục là “một cách cụ thể để đạt được một việc nào đó”. Nói cách khác, thủ tục là những hoạt động và tiến trình chi tiết được thực hiện theo một phương thức đã đưa ra với mục đích hoàn thiện một công việc nào đó.

Những thủ tục được thực hiện trong một tổ chức được xem như liên kết các nhân viên trong tổ chức đó, nghĩa là mỗi nhân viên thực hiện tác vụ của họ theo các bước có trong tài liệu thủ tục liên quan, thường mang tên của tác vụ được chọn lựa. Những thủ tục cũng có xu hướng phổ biến trong mỗi tổ chức, nghĩa là chúng được cung cấp bất cứ khi nào tác vụ được thực thi, bất chấp cá nhân thực hiện tác vụ hay phạm vi tổ chức.

- Chỉ thị công việc được sử dụng chủ yếu trong trường hợp một phương pháp thực hiện công việc trong suốt tổ chức hoặc không thể làm được hoặc không ai muốn làm. Kết quả là, chỉ thị công việc rất rõ ràng cho nhóm hoặc các phòng, ban; chúng bổ sung các thủ tục bằng cách cung cấp các chi tiết rõ ràng chỉ phù hợp với yêu cầu của 1 nhóm, một phòng hay một đơn vị.

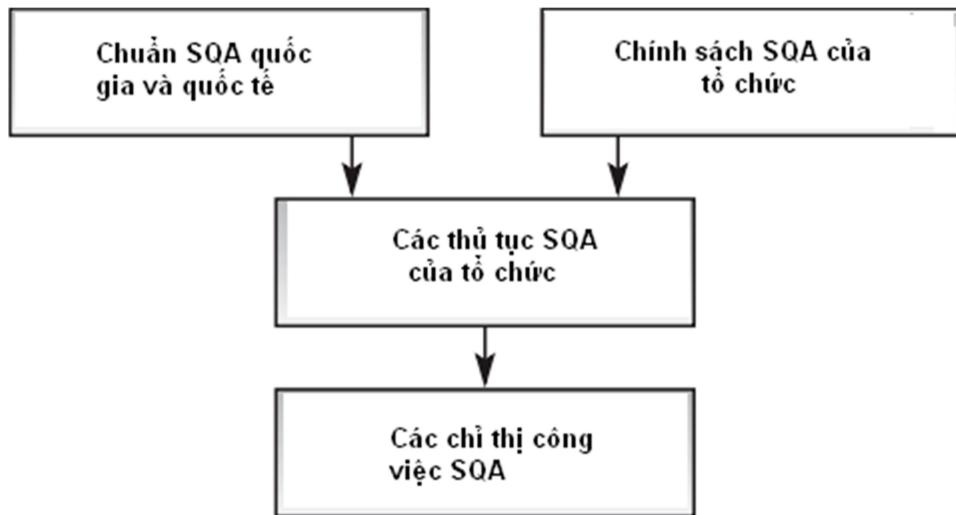
Chúng ta đặc biệt quan tâm đến các thủ tục và chỉ thị công việc đảm bảo chất lượng phần mềm bởi chúng có ảnh hưởng đến chất lượng của sản phẩm phần mềm, bảo trì phần mềm vào quản lý dự án.

- Các thủ tục SQA đã được bảo trì và phát triển một cách chuyên nghiệp đòi hỏi phải phù hợp với chính sách chất lượng của tổ chức nhưng cũng có xu hướng phù hợp với các chuẩn SQA quốc tế và quốc gia. Một điểm quan trọng cần lưu ý khi chuẩn bị chúng là các thủ tục tuân theo chuẩn SQA hỗ trợ chứng thực của hệ SQA của tổ chức. Chuẩn ISO 9000-3 (ISO, 1997; ISO/IEC, 2001) là một trong những chuẩn chứng nhận chính để hướng dẫn chuẩn bị các thủ tục.

Sự cần thiết của các thủ tục và chỉ thị công việc:

- “Tại sao chúng ta nên sử dụng thủ tục và chỉ thị công việc SQA?”
- “Nó sẽ tốt hơn không nếu mỗi chuyên gia tin vào kinh nghiệm của anh ta và thực hiện công việc một cách tốt nhất mà anh ta biết?”

- “Sẽ có lợi ích nào cho tổ chức nếu buộc tôi chỉ thực hiện các công việc theo cách mà họ lựa chọn?”



Hệ phân cấp mức khái niệm cho sự phát triển các thủ tục và chỉ thị công việc

Những câu hỏi giống như vậy thường xuyên được than phiền bởi nhân viên trong các tổ chức. Các câu trả lời mở ra thách thức phải đáp ứng các thủ tục và chỉ thị công việc: ứng dụng của kiến thực, kinh nghiệm, chuyên môn đã được tích lũy trong tổ chức.

Các thủ tục và chỉ thị công việc SQA nhằm mục đích:

- Thực hiện công việc, tiến trình hoặc các hoạt động theo cách có hiệu quả nhất mà không sai lệch yêu cầu chất lượng.
- Sự giao tiếp có hiệu quả giữa các nhân viên liên quan đến sự phát triển và bảo trì hệ thống phần mềm. Thông nhất trong quá trình thực hiện, đạt được bởi tuân theo các thủ tục và chỉ thị công việc, sẽ giảm thiểu sự hiểu lầm dẫn đến lỗi phần mềm.
- Đơn giản hóa sự phối hợp giữa các công việc và hoạt động được thực hiện bởi các bộ phận của tổ chức. Phối hợp tốt nghĩa là ít lỗi hơn.

Các thủ tục và sổ tay thủ tục:

Thủ tục cung cấp tất cả những chi tiết cần thiết để thực hiện các tác vụ theo các phương pháp bắt buộc để hoàn thành các chức năng của tác vụ đó. Những chi tiết

này có thể được xem lại khi đáp ứng năm vấn đề, được biết đến như là Five W's:

- Những hoạt động nào phải được thực thi? (Which)
- Cách thức mà mỗi hoạt động được thực thi? (how)
- Khi nào thì hoạt động được thực thi? (When)
- Hoạt động được thực thi ở đâu? (Where)
- Ai thực thi hoạt động? (Who)

Sự chuẩn hóa - ứng dụng của cấu trúc và định dạng cố định – là nguyên lý cung cấp cho tất cả các thủ tục SQA.

Các dạng báo cáo và cung cấp tài liệu, nhưng đặc biệt là các danh sách các điều kiện xác định một chuỗi các hoạt động đan xen và bảng xác định giới hạn quyền lực(trách nhiệm), theo hướng thay đổi các đáp ứng để phát triển bên ngoài và thay đổi bên trong các sản phẩm và tác vụ. hầu hết các thay đổi không ảnh hưởng đến bất cứ sự biến đổi có hữu nào của thủ tục. Phụ lục cung cấp các chi tiết đơn giản cung cấp cách thuận tiện để đưa ra các thay đổi mà không gây phiền hà cho chính thủ tục.

Sổ tay thủ tục

Tập hợp các thủ tục SQA thường được xem như các sổ tay thủ tục SQA. Nội dung sổ tay thủ tục của bất kỳ tổ chức nào đều theo những cách khác nhau sau:

- Các kiểu hoạt động phát triển và bảo trì phần mềm thực hiện bởi tổ chức
- Phạm vi hoạt động thuộc về mỗi kiểu hoạt động
- Phạm vi khách hàng (ví dụ: khách hàng bên trong, khách hàng làm phần mềm..) và nhà cung cấp (ví dụ: tự phát triển, bảo trì, người thầu phụ, ...)
- Điều chỉnh lựa chọn phương thức được cung cấp bởi tổ chức để đạt được những mục tiêu SQA mong muốn.

Những sự khác nhau này được biểu diễn trong sổ tay thủ tục SQA như thế nào? Trong khi một tổ chức có thể yêu cầu một phạm vi rộng các thủ tục, tổ chức khác lại có thể thỏa mãn với phạm vi thủ tục giới hạn. Tuy nhiên, số lượng thủ tục và cấu trúc của nó phụ thuộc đáng kể vào các quyết định về kiểu (style) và editorial chứ không chỉ phụ thuộc vào kiểu(type) thủ tục.

Một các tiếp cận hữu ích để xác định cấu trúc bảng nội dung của các sổ tay thủ tục

SQA là sử dụng bảng nội dung của chuẩn SQA liên quan như một bộ khung.

Chỉ thị công việc và sổ tay chỉ thị công việc:

Nhu cầu cập nhật trên, chỉ thị công việc xử lý ứng dụng của các thủ tục, phù hợp với yêu cầu của một nhóm dự án khách hàng riêng biệt hoặc các bên liên quan khác.

Trong khi phương pháp luận chung được định nghĩa trong một thủ tục, các chi tiết rõ ràng cho phép ứng dụng cho một dự án hoặc một đơn vị cụ thể thường được trình bày trong một chỉ thị công việc.

Không có trường hợp nào có thể làm chỉ thị công việc mâu thuẫn với thủ tục cha của nó, mặc dù một số chỉ thị có thể kết hợp với bất kỳ thủ tục đưa ra. Điều này có nghĩa là một ai đó có thể thêm, thay đổi hoặc huỷ bỏ chỉ thị công việc mà không thay đổi thủ tục tương ứng.

5.1.2 Chuẩn bị, thực thi và cập nhật các thủ tục và chỉ dẫn

Một sổ tay thủ tục SQA “chủ động” bao gồm nhiều các hoạt động liên tiếp nhằm đảm bảo khả năng tiếp tục áp dụng của thủ tục: ví dụ: quá trình chuẩn bị của các thủ tục, quá trình cài đặt và cập nhật thường lệ. Các hoạt động liên tục này được thực hiện bởi các thành viên trong đội SQA cùng với các thành viên của các đội hoặc đơn vị thuộc tổ chức liên quan, đảm bảo các thủ tục hoàn toàn thích nghi với những thay đổi về công nghệ hoặc khách hàng và đối thủ.

5.1.2.1 Quá trình chuẩn bị cho các thủ tục mới

Các bước ban đầu để phát triển một sổ tay thủ tục SQA mới nên thực hiện với các framework có tính khái niệm và tổ chức nhằm xác định danh sách các thủ tục được đề xuất và những người sẽ chịu trách nhiệm đối với quá trình chuẩn bị, cập nhật và phê chuẩn. Framework này cũng thường được thể hiện rõ ràng như một thủ tục (thường được gọi là the procedure of procedures).

Bước tiếp theo thông thường sẽ làm với các thủ tục cụ thể. Một cách tiếp cận thông thường để chuẩn bị các thủ tục là triệu tập một ủy ban ad hoc của các chuyên gia làm việc trong các đơn vị liên quan, các thành viên đơn vị SQA và các chuyên gia giải quyết các chủ đề tương ứng. Ủy ban duyệt qua các phác thảo dự định cho tới khi đạt được một phiên bản thỏa mãn yêu cầu, và chỉ dừng lại sau khi thủ tục được phê chuẩn bởi những người được ủy quyền. Một cách tiếp cận khác đối với việc chuẩn bị sổ tay thủ tục là dựa vào sự tư vấn, ở đó một chuyên gia ở bên ngoài được giao trách nhiệm thực hiện việc chuẩn bị một thủ tục, vài thủ tục

hoặc hoàn toàn thủ công. Ưu điểm chính của việc thuê người cố vấn là giá trị gia tăng trong kinh nghiệm và chuyên môn của họ trong các tổ chức khác, giảm bớt gánh nặng đối với các chuyên gia lâu năm trong tổ chức cũng như làm giảm thời gian thực hiện công việc. Nhược điểm chính của cách tiếp cận này là làm giảm tính ứng dụng bởi vì các đặc trưng duy nhất của tổ chức.

5.1.2.2 Thực thi các thủ tục mới hoặc các thủ tục được sửa đổi

Sự phê chuẩn một thủ tục mới hoặc được sửa đổi làm dễ dàng hơn đối với quá trình thực thi thủ tục đó, nó tách biệt và thông thường khó đưa ra. Trong nhiều trường hợp, sự phân tán các tài liệu dạng giấy in hoặc email và các chỉ dẫn của đội hoặc đơn vị là không đầy đủ để đảm bảo đủ hoặc gần đủ sự phù hợp. Thực tế là các thành viên của một đội hoặc một ban trong quá trình chuẩn bị thủ tục thuyết phục đồng nghiệp của họ tuân theo các yêu cầu mới nhưng thường không được thỏa đáng. Các chỉ thị tiếp theo và riêng lẻ của những người này thường thiếu hoặc xem nhẹ các thủ tục mới như sự ủy nhiệm cho việc tích hợp các thủ tục bên trong các công việc thường ngày.

5.1.2.3 Cập nhật thủ tục

Động lực cho việc cập nhật các thủ tục hiện hành dựa trên những điều sau:

- Sự thay đổi công nghệ của các công cụ phát triển, phần cứng, thiết bị truyền thông ...
- Thay đổi các lĩnh vực hoạt động của tổ chức
- Các yêu cầu của người dùng tăng dần
- Quá trình phân tích thành công cũng như thất bại
- Đề xuất cải tiến bắt đầu bởi các báo cáo kiểm tra nội bộ
- Sự học hỏi từ kinh nghiệm của các tổ chức khác
- Kinh nghiệm của đội SQA

Khi sự cần thiết của việc cập nhật được nhận ra, một kỹ thuật tương tự như khi áp dụng quá trình chuẩn bị các thủ tục mới có thể được thực hiện: một đội ad hoc được triệu tập để chuẩn bị phiên bản cập nhật, theo sau bởi những người được ủy quyền và các hoạt động thực thi. Điều này ngụ ý rằng việc cập nhật nên được xem xét như là một quá trình được áp dụng trong toàn bộ khâu kiểm tra chất lượng sản phẩm, quan trọng như là việc chuẩn bị các thủ tục mới.

5.1.3 Khuôn mẫu (templates)

Template là một mẫu hoặc một khuôn được sử dụng như một sự chỉ dẫn cho hình dạng của một thứ gì đó sẽ được tạo ra (Từ điển Cao đẳng Webster). Nhưng khi được áp dụng trong công nghệ phần mềm, thuật ngữ template ám chỉ một định dạng (đặc biệt là các bảng khái niệm) được tạo ra bởi các tổ chức, được áp dụng khi soạn thảo một bản báo cáo hoặc viết tài liệu. Việc áp dụng các khuôn hình có thể là bắt buộc đối với một vài tài liệu và không bắt buộc đối với các tài liệu khác; trong một vài trường hợp chỉ một phần của khuôn hình (ví dụ các chương cụ thể hoặc cấu trúc chung) được yêu cầu.

Ví dụ về mẫu:

- Kế hoạch kiểm tra phần mềm (STP)
- Mô tả kiểm tra phần mềm (STD)
- Báo cáo kiểm tra phần mềm (STR)
- Yêu cầu thay đổi phần mềm (SCR)
- Tài liệu về cấu hình phần mềm

5.1.3.1 Đóng góp của các khuôn hình đối với chất lượng phần mềm

Sử dụng khuôn hình tạo nhiều thuận lợi cho các đội phát triển và đội xem xét lại. Đối với đội phát triển, sử dụng khuôn hình có các ưu điểm:

- Làm cho dễ dàng hơn trong việc chuẩn bị tài liệu bằng cách tiết kiệm thời gian và năng lượng được yêu cầu để dựng lên cấu trúc của bản báo cáo.
- Đảm bảo các tài liệu được chuẩn bị bởi các nhà phát triển đầy đủ hơn bởi tất cả các chủ đề trong tài liệu đã được định nghĩa và được xem xét lại nhiều lần bởi các chuyên gia thông qua việc sử dụng khuôn hình. Các lỗi thông thường, như là bỏ qua một chủ đề thì khó xảy ra hơn.
- Dễ dàng tích hợp các thành viên mới trong đội thông qua sự quen thuộc. Cấu trúc tiêu chuẩn của tài liệu, được chuẩn bị theo các khuôn hình mà có thể các thành viên mới đã biết từ trước khi làm việc trong các đội hoặc tổ chức khác, làm cho việc tìm kiếm thông tin dễ dàng hơn. Nó cũng giải quyết ổn thỏa việc chuẩn bị tài liệu tiếp theo khi mà các phần tài liệu đã được chuẩn bị bởi thành

viên trong đội khác có thể rời đi hoặc không.

- Dễ dàng xem xét lại tài liệu bằng cách loại bỏ sự cần thiết nghiên cứu cấu trúc của một tài liệu và xác nhận sự đầy đủ của nó, khi mà tài liệu được dựa trên khuôn hình thích hợp. Nó cũng làm đơn giản việc xem xét lại tài liệu đầy đủ vì cấu trúc của nó là chuẩn và những người xem xét lại đã quen với nội dung trong đó (các chương, đoạn và phụ lục). Kết quả của tính nhất quán này là việc xem xét lại được cho là kỹ lưỡng và tốn ít thời gian hơn.
- Đối với các đội bảo trì phần mềm, việc sử dụng khuôn hình có ưu điểm:
- Làm cho dễ dàng xác định thông tin được yêu cầu cho các nhiệm vụ thực thi bảo trì.

5.1.3.2 Framework của tổ chức cho việc chuẩn bị, cài đặt và cập nhật các khuôn mẫu

Các tổ chức đều có khuynh hướng tiết kiệm các tài nguyên nội tại, điều đó có nghĩa là những báo cáo thành công trong công việc thì được chuẩn bị cho 1 bộ phận hoặc với mục đích sẽ để nó trở thành những khuôn mẫu cho toàn bộ tổ chức. Do đó, nếu những báo cáo của ông Brown hoặc ông Johnson đã có tiếng là toàn diện và có tính chuyên nghiệp cao thì các kiểu mẫu nội dung của những báo cáo đó có thể được sử dụng như là khuôn mẫu cho những đồng nghiệp của họ. 1 điều bất lợi của việc này đó là không phải ai được hưởng thuận lợi từ những khuôn mẫu này cũng có thể nhận ra được sự tồn tại của chúng. 1 điều bất lợi khác là sự cải tiến của những khuôn mẫu, đã được hoàn thiện qua sự kiểm duyệt(review) của những đội ngũ chuyên nghiệp, có thể bị cản trở.

Đơn vị SQA thường có trách nhiệm cho việc chuẩn bị những khuôn mẫu có tính chuyên nghiệp chọn từ những kiểu thông dụng hơn của các báo cáo và tài liệu đã được yêu cầu của các nhân viên trong tổ chức. Những sự bắt đầu không chính thức từ 1 lĩnh vực có thể khuyến khích đơn vị SQA hoạt động, nhưng việc phát triển cơ sở hạ tầng chung cho việc sử dụng các khuôn mẫu, chủ đề của phần này, thì thuộc về nhiệm vụ của

đơn vị SQA.

5.1.3.3 Sự chuẩn bị cho những khuôn mẫu mới

Sự phát triển của 1 cơ sở hạ tầng cho khuôn mẫu hiển nhiên được đặt vào vị trí trung tâm trong công việc của nhóm chuyên nghiệp được dành riêng cho nhiệm vụ này. Nhóm này (có thể là 1 ủy ban) nên thêm vào những nhân viên lâu năm, những

người miêu tả những đường lối phát triển phần mềm khác nhau, trưởng kĩ sư phần mềm và các thành viên đơn vị SQA. Những người phát triển không chính thức của “các dịch vụ khuôn mẫu” cũng nên được khuyến khích tham gia vào nhóm

Một trong những nhiệm vụ đầu tiên của nhóm là biên soạn 1 danh sách các khuôn mẫu mục tiêu để phát triển. Khi danh sách được chấp nhận, thứ tự mức ưu tiên sẽ được thiết lập. Những cái có mức ưu tiên cao được cho vào loại khuôn mẫu của những tài liệu thông thường nhất đã được chuẩn bị, cũng có thể được đưa vào những khuôn mẫu không chính thức đã sẵn sàng sử dụng (nó được ước lượng khi chỉ cần công sức tối thiểu cho việc hoàn thiện và phân quyền). Các ủy ban con được giao nhiệm vụ chuẩn bị những bản phác thảo đầu tiên. 1 thành viên đơn vị SQA có thể được xem xét giao nhiệm vụ làm leader, nhưng ngoại lệ cơ hội cũng có thể dành cho 1 thành viên của ủy ban. Bất kể người trưởng nhóm là ai, anh ấy (hoặc chị ấy) phải nhận thấy được sự phân bố các bản thảo khuôn mẫu trong số các thành viên, việc tổ chức buổi họp và các việc kế tiếp được tạo ra bởi các ủy ban con có nhiệm vụ chuẩn bị khuôn mẫu. Sự phân bố của các bản thảo khuôn mẫu trong các team leader cho việc bình luận của họ có thể tạo ra những sự phát triển quan trọng và cùng 1 lúc đẩy mạnh sử dụng các khuôn mẫu trong tương lai.

Các nguồn thông tin thông thường nhất được sử dụng trong việc chuẩn bị khuôn mẫu được liệt kê như sau:

- Các khuôn mẫu không chính thức đã được sử dụng trong tổ chức
- Các ví dụ khuôn mẫu được tìm thấy trong sự công bố của các chuyên gia
- Các khuôn mẫu đã được sử dụng bởi những tổ chức giống nhau

5.1.3.4 Áp dụng các khuôn mẫu

Vài quyết định cơ bản được đưa vào sự cài đặt những khuôn mẫu mới hoặc những khuôn mẫu đã được cập nhật:

- Những kênh nào nên được sử dụng để quảng bá những khuôn mẫu này
- Bằng cách nào các khuôn mẫu được tạo có thể sẵn sàng cho những “khách hàng” nội tại của tổ chức
- Những khuôn mẫu nào sẽ là bắt buộc và làm sao có thể áp dụng được chúng

Tất cả các phương pháp liên lạc chuyên nghiệp nội bộ có thể được sử dụng để quảng bá các khuôn mẫu bên trong tổ chức : truyền đơn, e-mail, mạng SQA nội bộ cũng như những bài trình bày ngắn trong các buổi họp

Một trong số các phương pháp có hiệu quả nhất của việc làm cho các khuôn mẫu luôn sẵn sàng cho tổ chức là mạng nội bộ, được dùng nhiều hơn là các kênh truyền trên giấy tờ. Sự phân bố qua mạng nội bộ đảm bảo cho người sử dụng lựa chọn những phiên bản được cập nhật của các khuôn mẫu cần thiết và cùng 1 lúc lưu lại những cái chính (dành cho các khuôn mẫu dựa trên giấy tờ) của các bảng biểu nội dung của tài liệu.

Những chỉ thị bắt buộc phải sử dụng trong các khuôn mẫu đặc biệt thường được tìm thấy trong thủ tục hoặc hướng dẫn công việc của tổ chức. Người đứng đầu đội kĩ sư phần mềm hoặc các nhân viên lâu năm thường được ủy quyền để xác định danh sách các khuôn mẫu có tính bắt buộc và phù hợp với những thủ tục đã được chọn, mặc dù chúng ta có thể mong đợi rằng khuôn mẫu được tập hợp lại để đưa ra danh sách được đề cử của chính nó.

5.1.3.5 Cập nhật các khuôn mẫu

Việc quyết định cập nhật 1 khuôn mẫu đã có có thể được xem như là 1 tiêu chuẩn đánh giá lại (reactive measure), nó phụ thuộc vào bất kì yếu tố sau:

- Ý kiến và đề xuất của người dùng
- Những sự thay đổi bên trong lĩnh vực hoạt động của tổ chức
- Những đề xuất đã được khởi xướng bởi việc xem xét lại thiết kế (design review) và đội kiểm tra dựa trên sự xem xét lại các tài liệu đã chuẩn bị tùy theo từng khuôn mẫu
- Sự phân tích thất bại cũng như thành công
- Kinh nghiệm khác của tổ chức
- Đề xuất của đội SQA

Các công việc tiếp theo của việc cập nhật khuôn mẫu khá giống với của việc chuẩn bị khuôn mẫu.

5.1.4 Danh mục kiểm tra (Checklists)

Checklists được sử dụng bởi các nhà phát triển phần mềm để cập đến danh sách

khoản mục đặc biệt được xây dựng cho từng loại tài liệu, hay một menu chuẩn bị các việc cần hoàn thành trước khi thực hiện một hoạt động (ví dụ như, cài đặt một gói phần mềm tại trang web của khách hàng). Checklists được lên kế hoạch toàn diện nếu công việc chưa hoàn thành. Thông thường, Checklists có xu hướng được coi là một công cụ tùy chọn cơ sở hạ tầng, phụ thuộc chủ yếu vào danh sách của thuộc tính chuyên nghiệp, sự hiểu biết của người dùng với danh sách và tính khả dụng.

Một số checklists có 2 mục đích: trong khi cung cấp một danh sách đầy đủ các khoản mục được xác nhận, nó cũng cung cấp không gian cho việc tìm kiếm tài liệu.

Tiếp theo, chúng ta cần thấy được sự đóng góp của checklists vào chất lượng phần mềm và những nỗ lực cần thiết để thành lập, duy trì và áp dụng các checklists đó.

5.1.4.1 Những đóng góp của checklists để phần mềm có chất lượng

Cũng giống như các mẫu, checklists cung cấp nhiều lợi ích để phát triển team, duy trì phần mềm của team và chất lượng của tài liệu.

Những lợi thế để phát triển các nhóm như sau:

■ **Giúp các nhà phát triển thực hiện tự kiểm tra các tài liệu hoặc code** của phần mềm trước khi tài liệu hoặc code của phần mềm hoàn thành và đưa vào thiết kế chính thức. Checklists được dùng để giúp đỡ nhà phát triển khám phá không đầy đủ các phần, cũng như phát hiện các lỗi mà không nhận thấy được. Checklists cũng được dự kiến sẽ đóng góp vào chất lượng của các tài liệu hoặc code của phần mềm để xem xét các vấn đề như chất lượng sẽ được khảo sát bằng cách xem xét lại các vấn đề đã được liệt kê trong checklist

■ **Giúp phát triển chuẩn bị cho công việc của họ** như: cài đặt phần mềm tại trang web của khách hàng, thực hiện kiểm tra chất lượng tại các trang web của người thầu phụ hoặc ký kết hợp đồng với nhà cung cấp các mô-đun có thể sử dụng lại của phần mềm. Checklists dự kiến để giúp các nhà phát triển trang bị tốt hơn cho công việc hiệu quả.

Những lợi thế để rà soát nhóm là:

■ **Đảm bảo đầy đủ các tài liệu để xem lại bằng cách xem xét thành viên trong nhóm**

cũng như tất cả các khoản mục có liên quan xuất hiện trên danh sách.

■ **Tạo điều kiện cải thiện hiệu quả của các phiên review**, xem như là một môn học và trật tự các cuộc thảo luận được định nghĩa và cũng được biết trước.

5.1.4.2 Cấu trúc framework cho việc chuẩn bị, thực thi và cập nhật các checklist

Mặc dù được đánh giá cao, nhưng việc sử dụng các checklist vẫn mang tính tùy ý. Sự chuẩn bị và cập nhật checklist, cũng như quá trình sử dụng vẫn thường được áp định cho “SQA unit”. Một “checklist group”, đứng đầu bởi 1 thành viên của “SQL unit”, có thể đảm trách công việc liên tục thu thập những danh sách đã được cập nhật. Đội ngũ khác bao gồm những người quan tâm đến việc xúc tiến việc sử dụng các checklist tinh nguyện tham gia vào trong nhóm.

Tuy nhiên trong 1 số trường hợp thì việc trợ giúp của 1 cố vấn SQL là cần thiết. Ở phần còn lại, chúng ta sẽ miêu tả những quy trình cần thiết để duy trì cơ sở hạ tầng của 1 checklist như là sự chuẩn bị cho các checklist mới, quá trình xúc tiến sử dụng và cập nhật các checklist đó.

5.1.4.3 Sự chuẩn bị cho những checklist mới:

Một trong những công việc đầu tiên chuẩn bị cho “checklist group” là việc biên dịch những danh sách những checklist được nghiên cứu và phát triển theo sự định nghĩa của tiêu chuẩn chung cho tất cả những checklist được nhóm đưa ra.

Những checklist được nhóm chọn thường là những checklist bình thường đã được sử dụng bởi một số thành viên của nhóm phát triển và những nhà phê bình (reviewer). Trong hầu hết các trường hợp, 1 chút thay đổi và sự chỉnh sửa của những checklist này là đủ để thỏa mãn các tiêu chuẩn về nội dung hay định dạng mà nhóm đã đưa ra. Sự chuẩn bị cho những checklist mới cũng như sự cải tiến những checklist thông thường được hỗ trợ từ những nguồn thông tin sau:

- Những checklist thông thường đã được sử dụng trong quá trình tổ chức.
- Những checklist mẫu được tìm ra trong sách vở và những ấn phẩm chuyên ngành khác.
- Các checklist được sử dụng bởi những tổ chức tương tự.

Quá trình chuẩn bị những checklist mới cũng tương tự như việc chuẩn bị cho các mẫu.

5.1.4.4 Xúc tiến sử dụng checklist

Do việc sử dụng checklist này hiếm khi là bắt buộc, cho nên việc xúc tiến sử dụng đều dựa trên lợi ích của việc quảng cáo và bảo hành. Những kênh thông tin nội địa có thể được sử dụng cho việc giới thiệu những checklist này đến người sử dụng như: tờ rơi, e-mail, mạng SQA cũng như những cuộc họp cấp cao. Tuy nhiên mạng lưới nội địa vẫn là phương thức thích hợp hơn và có hiệu quả cao nhất giúp cho việc làm cho các checklist này đến với những tổ chức khách hàng nội địa.

5.1.4.5 Cập nhật các checklist:

Giống như các mẫu và các thủ tục, việc chủ động cập nhật các mẫu checklist đã có nhìn chung theo các nguồn sau:

- Sự đề xuất và gợi ý của người sử dụng.
- sự thay đổi về công nghệ, lĩnh vực hoạt động và nhóm khách hàng.
- Những đề nghị truyền thụ với ý định tổng quan và những đội kiểm tra Phát ra từ những tổng quan tài liệu.
- Việc phân tích thất bại cũng như thành công.
- Những kinh nghiệm tổ chức khác
- Năng lực của đôi ngũ SQA

Quy trình cập nhật các checklist khá giống với sự chuẩn bị chúng.

5.2 Đào tạo đội ngũ và cấp chứng chỉ

Người ta luôn nói rằng muốn nhân viên theo kịp với những kiến thức nghề nghiệp mới có gần đây nhất là chìa khóa để đạt chất lượng trong phát triển và bảo trì phần mềm. Tuy nhiên, có một điều thường được chấp nhận rằng đào tạo chuyên nghiệp, đào tạo lại hoặc cập nhật là bắt buộc (mandatory) để khe hở giữa kiến thức nghề nghiệp được yêu cầu và kiến thức hiện tại càng hẹp càng tốt. Chứng chỉ bên trong (internal certification)(dưới đây chỉ “chứng chỉ”) của nhân viên giữ vị trí chính trong phát triển và bảo trì phần mềm là loại khác – một công cụ để đảm bảo chất lượng nghề nghiệp.

Điều quan trọng của việc đào tạo nghề nghiệp như là một phần của bất kỳ một hệ thống đảm bảo chất lượng (SQA) nào, được nhấn mạnh trong chuẩn ISO 9000-3 cũng như trong CMM Guidelines (có thể tìm thấy ISO, 1997; ISO/IEC,

2001; Paulk et al., 1995). Phát triển mô tả nghề nghiệp và chương trình đào tạo cho toàn bộ cán bộ công nhân viên SQA đã được Mendis (1999) thảo luận. Chương trình cấp giấy chứng nhận cho các kỹ sư chất lượng phần mềm cũng được **American Society for Quality** (ASQ) quan tâm, được mô tả bởi Hamilton (1999) và một cuốn sổ ASQ (ASQ, 1999).

5.2.1 Mục tiêu của đào tạo và cấp chứng chỉ

Mục tiêu của việc đào tạo và cấp chứng chỉ đó là:

- Để phát triển sự hiểu biết và kỹ năng cho nhân viên mới cần để thực hiện các công việc phát triển và bảo trì phần mềm ở mức độ hiệu quả và có hiệu lực. Sự huấn luyện như vậy tạo điều kiện thuận lợi cho việc các nhân viên mới hòa nhập với đội dự án.
- Để đảm bảo sự phù hợp với những tiêu chuẩn của tổ chức cho những sản phẩm phần mềm (tài liệu và code) bởi những kiểu truyền và cấu trúc các thủ tục với những chỉ dẫn của công việc.
- Để cập nhật sự hiểu biết và kỹ năng của những nhân viên từng trải trong việc đáp lại những phát triển của tổ chức, và đảm bảo hiệu quả và việc thực hiện tốt các tác vụ cũng như sự phù hợp với kiểu của tổ chức và cấu trúc thủ tục và các chỉ dẫn công việc.
- Để truyền những tri thức của những thủ tục đảm bảo chất lượng phần mềm.
- Để đảm bảo rằng những ứng cử viên cho những vị trí chính trong phát triển và bảo trì phần mềm được xác định đầy đủ.

5.2.2 Tiến trình đào tạo và cấp chứng chỉ

Hoạt động của một hệ thống đào tạo và cấp chứng chỉ thành công yêu cầu những hành động sau phải được thực hiện một cách đều đặn:

- Xác định yêu cầu về sự hiểu biết nghề nghiệp với mỗi vị trí.
- Xác định nhu cầu đào tạo và cập nhật kiến thức nghề nghiệp (professional)
- Lập kế hoạch cho chương trình đào tạo chuyên nghiệp
- Lập kế hoạch cho việc cập nhật kiến thức nghề nghiệp
- Xác định chứng chỉ yêu cầu của từng vị trí
- Lập kế hoạch cho quy trình cấp chứng chỉ

- Thực hiện những việc tiếp theo của những nhân viên đã được đào tạo và cấp chứng chỉ.

Tất cả những hoạt động này được hội tụ vào trong một quá trình tổng hợp trong đó những phản hồi từ những hoạt động trước và những thông tin về những sự phát triển nghề nghiệp kích thích một chu trình của việc đào tạo, cấp chứng chỉ và sự thích nghi với những thay đổi về yêu cầu chất lượng.

Các hoạt động đào tạo và cấp chứng chỉ có nghĩa là để lấp đầy những nhu cầu của những nhân viên từng trải và những nhân viên mới. Tiếp theo những kết quả của chương trình hiện thời cũng như là theo dõi những sự phát triển trong nghề nghiệp được yêu cầu để đảm bảo rằng chương trình được cập nhật một cách đầy đủ. Thảo luận chi tiết của mỗi hoạt động này được thể hiện trong những đoạn tiếp theo.

5.2.3 Xác định yêu cầu kiến thức chuyên môn và sự cần thiết của đào tạo và cập nhật

Các vị trí thông thường nhất trong một tổ chức phát triển và bảo trì phần mềm là: những người phân tích hệ thống, lập trình viên, trưởng nhóm phát triển phần mềm, trưởng nhóm lập trình, kỹ thuật viên bảo trì phần mềm, nhân viên kiểm thử phần mềm, trưởng nhóm kiểm thử phần mềm. Hầu hết các tổ chức đều đặt ra các yêu cầu về giáo dục và kinh nghiệm cho mỗi vị trí trên. Các thành viên của phòng ban là những người đáp các yêu cầu đào tạo tại chỗ cần phải bổ sung những hiểu biết và kỹ năng về “local” hoặc “internal”, có liên quan đến việc phát triển rõ ràng và các thủ tục bảo trì. Những hiểu biết về chuyên môn có thể được nhóm lại vào 2 loại:

- Kiến thức và kỹ năng về chủ đề kỹ nghệ phần mềm, thí dụ như các công cụ phát triển phần mềm, các phiên bản ngôn ngữ lập trình, các phiên bản CASE tool ứng dụng bởi tổ chức riêng biệt hoặc đơn vị. Các thủ tục và các chỉ dẫn công việc thích hợp này được biên soạn để thực hiện chúng đó cũng là bốn phần của mục này.
- Hiểu biết về chủ đề SQA, những thủ tục gắn liền với rất nhiều hoạt động phát triển và bảo trì. Sự phân công được thực hiện bởi một người nắm giữ một chức vụ đặc biệt.

5.2.4 Xác định những nhu cầu đào tạo và cập nhật (updating)

Những nhu cầu đào tạo và cập nhật được xác định bằng việc so sánh kiến thức hiện tại của nhân viên với những yêu cầu về kiến thức được cập nhật. Kiểu đào tạo này được điều chỉnh để phù hợp với 3 nhóm nhân viên riêng biệt:

- Đào tạo: Cho những nhân viên mới, theo vị trí họ được phân công.
- Đào tạo lại: Cho những nhân viên được phân công đến vị trí mới hoặc tiếp nhận những nhiệm vụ mới.
- Cập nhật: Cho đội ngũ nhân viên như yêu cầu bởi vị trí của họ.

Nhu cầu cập nhật nhân viên được đánh giá thường xuyên để thuận tiện lập kế hoạch cho những chương trình yêu cầu.

Cuối cùng , việc tiếp theo là đánh giá nhân viên trong các phòng đào tạo và cập nhật cung cấp đầu vào chính để sử dụng cho việc định nghĩa lại nhu cầu đào tạo.

5.2.5 Lên kế hoạch đào tạo và chương trình cập nhật

Từ thực tế thấy rằng, 2 chương trình cơ bản phải được đề ra - 1 cho chủ đề kỹ nghệ phần mềm và 1 cho chủ đề SQA.

Lập kế hoạch chương trình đào tạo và cập nhật cho chủ đề kỹ nghệ phần mềm:

Sự tính toán về thời gian của những hoạt động đào tạo và đào tạo lại không thể được xác định sớm bởi vì những cán bộ công nhân mới được tuyển mộ và và những nhân viên kỳ cựu được chuyển vị trí thường xuyên sau khi được thông báo trước tương đối ngắn. Mặc dù, các hoạt động cập nhật có thể có thể được sắp xếp trước dễ dàng , với nội dung hoàn tất đóng kín đến thời điểm thực hiện chúng. Bất kể dù những chương trình này được tiến hành trong tổ chức hay bởi một tổ chức outsourcing, nhân viên cấp cao, ví dụ như kỹ sư trưởng phần mềm, thường xuyên tham gia chuẩn bị chúng.

Lập kế hoạch những chương trình đào tạo và cập nhật cho chủ đề SQA:

Những chương trình đào tạo cho các chủ đề SQA bao gồm đào tạo cho nhân viên mới cũng như cập nhật cho những nhân viên kỳ cựu. Đặc điểm chung của chương trình đào tạo SQA cho phép chúng được tổ chức theo định kỳ, một hoặc 2 tháng một lần, và phân phối đến tất cả nhân viên mới tuyển mộ trong thời gian đó. Đặc thù của chương trình cập nhật SQA là được thực hiện một lần một năm, hoặc cứ sáu tháng một lần, tùy thuộc vào tốc độ của sự thay đổi. Đơn vị SQA hoặc

những người khác chịu trách nhiệm về kết quả SQA trong tổ chức thường tổ chức những chương trình đào tạo và cập nhật này.

5.2.6 Định nghĩa các vị trí yêu cầu cấp chứng chỉ

Thông thường việc chỉ định nhân viên vào vị trí chủ chốt trong phát triển phần mềm và các tổ chức bảo trì đòi hỏi phải hết sức thận trọng. Một trong những thủ tục được sử dụng để đảm bảo sự phù hợp của các ứng cử viên là giấy chứng nhận.

5.2.7 Lên kế hoạch các tiến trình cấp chứng chỉ

Cấp giấy chứng nhận là nhằm cung cấp một framework cho kiểm tra toàn diện khả năng của một ứng cử viên và một biểu hiện của kiến thức chuyên môn và kỹ năng. Những chi tiết của quá trình cấp giấy chứng nhận duy nhất cho các tổ chức. Chúng phản ánh những tính năng đặc biệt, lĩnh vực chuyên môn, phát triển phần mềm và công cụ bảo trì, khách hàng... Bởi vì các quá trình này hướng về những điều cần thiết và quyết định của các tổ chức cụ thể, nội bộ cấp giấy chứng nhận không thể tự động thay thế bởi các giấy chứng nhận chung mà công nhận bởi những người chuyên nghiệp và nhà cung cấp hàng đầu của công cụ phát triển và phần mềm mạng lưới thông tin liên lạc hoặc tương tự.

Việc cấp giấy chứng nhận trong từng chi tiết và mỗi vị trí, đòi hỏi phải chấp nhận như được định nghĩa trong các thủ tục cấp giấy chứng nhận.

Các thủ tục cấp giấy chứng nhận điển hình

Đối với những người đã từng được cấp giấy chứng nhận, một quá trình cấp giấy chứng nhận tiêu biểu đòi hỏi phải đáp ứng một số hoặc là tất cả các yêu cầu sau đây:

- Giáo dục chuyên môn : học tập hay kỹ thuật cao và một số trường hợp cấp giấy chứng nhận bởi một tổ chức chuyên nghiệp hoặc bởi một người lãnh đạo sản xuất phần mềm thương mại.
- Các khoá đào tạo nội bộ
- Chuyên gia kinh nghiệm trong tổ chức (có thể là một phần hoặc hoàn toàn thay thế bằng kinh nghiệm trong tổ chức khác)
- Đánh giá về những thành tựu và khả năng như đã nêu trong định kỳ thực thi việc đánh giá

- Ước lượng bởi các cấp cao hơn của ứng cử viên (thường được hoàn thành bởi một câu hỏi đặc biệt)
- Biểu hiện của kiến thức và kỹ năng nghĩa là một cuộc thử nghiệm hay dự án.
- Sự giám sát của người nhiều kinh nghiệm cho một giai đoạn nhất định về thời gian.

Chức năng của uỷ ban cấp giấy chứng nhận

Tương tự mô hình đề nghị cho chương trình đào tạo và tái đào tạo, cá nhân hay là uỷ ban chịu trách nhiệm về cấp giấy chứng nhận thường có thẩm quyền trong phát triển phần mềm và duy trì đội ngũ. Trách nhiệm của những người cấp giấy chứng nhận bao gồm:

- Thực hiện quá trình cấp giấy chứng thực trên cơ sở thực hiện các yêu cầu cá nhân hay đơn vị và chấp nhận cấp giấy chứng nhận cho những người đủ điều kiện.
- Tiếp tục cấp giấy chứng nhận (như cố vấn) thực hiện bởi những người khác.
- Cập nhật các yêu cầu cấp giấy chứng nhận trong hướng ứng sự phát triển trong tổ chức tốt như các chuyên gia.

Thay đổi danh sách các vị trí yêu cầu cấp giấy chứng nhận.

5.2.8 Phân phối các chương trình đào tạo và cấp chứng chỉ

Đào tạo và cập nhật có thể bao gồm các chủ đề như kỹ nghệ phần mềm, bảo đảm chất lượng phần mềm và kỹ năng quản lý (trong khuôn khổ cấp giấy chứng nhận hoặc cho các thông tin chung), tất cả đều được phối hợp với nhu cầu của tổ chức hay doanh nghiệp. Làm thế nào để đào tạo và cập nhật mang lại sự thay đổi phù hợp. Các khoá học có thể truyền trong các định dạng mà bài giảng và thuyết minh ngắn, thường chỉ kéo dài nửa ngày, để tổ chức các khoá học dài một vài tuần hoặc tháng. Những khoá học này có thể thực hiện tại nhà bởi các đơn vị đào tạo của tổ chức, hoặc ra bên ngoài, bởi chương trình hướng nghiệp hoặc cơ sở giáo dục học tập đặt chương trình nền móng cho các yêu cầu của tổ chức.

Thông tin về tổ chức và cung cấp đào tạo và chương trình cấp giấy chứng nhận có thể tìm thấy trong quản lý nhân lực.

5.2.9 Những công việc tiếp theo của việc đào tạo và cấp chứng chỉ

Những người quản lý và chuyên gia phần mềm thường ngờ về những tác

động của đào tạo và chứng chỉ trong công tác đào tạo và cấp giấy chứng nhận nói chung hoặc của một trong những hoạt động liên quan. Họ luôn đòi hỏi cho dù những nguồn nhân lực và nỗ lực đầu tư trong đào tạo thực sự có giá trị. Để kiểm định những nghi ngờ, hệ thống theo dõi là thực sự cần thiết để cung cấp thông tin phản hồi tới những đơn vị chuyên nghiệp. Những thông tin phản hồi này cho biết những nỗ lực đào tạo được điều chỉnh tại cùng thời điểm nó đảm bảo sự phát triển tiếp diễn của những hoạt động đào tạo và cấp chứng chỉ. Những thông tin cung cấp bởi hệ thống theo dõi liên hệ tới:

- Tất cả những hoạt động đào tạo và thủ tục cấp chứng chỉ tiến hành – những bản ghi về hiệu suất của những người tham gia chương trình.
- Thông tin về trường hợp đặc biệt của các hoạt động đào tạo là minh chứng được đánh giá cao thành công hay thất bại trong việc cải thiện hiệu suất của nhân viên.
- Thông tin về những trường hợp của những thất bại của nhân viên đạt chứng chỉ trong việc thực thi rõ ràng không đạt những yêu cầu chứng chỉ.

5.3 Các hành động sửa lỗi và phòng ngừa

Các hoạt động mang tính hệ thống mà thực thi việc nâng cao hiệu quả và hiệu suất hoạt động được gọi là các hoạt động ngăn chặn và sửa lỗi (CAPA). Có những hoạt động không phải nhằm sửa các lỗi được tìm thấy ngay lập tức mà để loại bỏ nguyên nhân của các lỗi này trong suốt quá trình phát triển phần mềm.

Bằng việc không ngừng đẩy mạnh cải tiến hiệu suất và hiệu quả, tiến trình CAPA đã trở thành một trong những công cụ chính được sử dụng để giành được mục đích hướng thực thi của SQA: vừa thỏa mãn các yêu cầu chức năng và các quản lý vừa giảm chi phí phát triển phần mềm, bảo trì và các hoạt động đảm bảo chất lượng.

Định nghĩa hoạt động sửa lỗi và phòng ngừa

Các hành động sửa lỗi: một tiến trình phản hồi được áp dụng thường xuyên bao gồm việc tập hợp thông tin về các hoạt động không tuân theo chất lượng, xác định và phân tích các nguồn không tuân theo các quy tắc cũng như việc phát triển và đồng hóa các thủ tục cải tiến, cùng việc kiểm soát việc cài đặt và đánh giá các kết quả của họ.

Các hoạt động ngăn chặn: một tiến trình phản hồi được áp dụng thường xuyên bao gồm việc tập hợp thông tin về các vấn đề chất lượng có khả năng xảy ra, xác định

và phân tích sự sai lệch về các tiêu chuẩn chất lượng, việc phát triển và và đồng hóa các thủ tục cải tiến, cùng việc kiểm soát việc cài đặt và đánh giá các kết quả của họ.

Tiến trình hành động sửa lỗi và phòng ngừa

Hoạt động thành công của 1 tiến trình CAPA bao gồm những thao tác sau

- (1) Thu thập thông tin
- (2) Phân tích thông tin
- (3) Việc phát triển các giải pháp và các phương thức cải tiến
- (4) Việc cài đặt các phương thức cải tiến
- (5) Follow-up (công việc tiếp theo)

Tiến trình này thường được làm bởi luồng thông tin từ các tài nguyên khác nhau. Để ước lượng sự thành công của tiến trình này, 1 vòng lặp phản hồi đóng được áp dụng để điều khiển luồng thông tin, sự thực thi của việc tìm kết quả thay đổi trong các tình huống thực tế và các thủ tục cùng với kích thước của các kết quả.

Thu thập, phân tích thông tin

- Thu thập thông tin

Sự đa dạng của các tài nguyên thông tin, bên trong và bên ngoài, cung cấp cho tiến trình CAPA khá rõ rệt. Theo sự phân đôi bên trong/ bên ngoài, 4 tài nguyên thông tin chính bên trong là (1) tiến trình phát triển phần mềm, (2) việc bảo trì phần mềm, (3) Cơ sở hạ tầng SQA và (4) các thủ tục quản lý chất lượng phần mềm. Tài nguyên thông tin bên ngoài chủ yếu là các thông kê những yêu cầu của khách hàng và những phản nản của họ.

- Phân tích thông tin

Thao tác thông thường của tiến trình CAPA có mục đích là tạo 1 luồng lớn các tài liệu liên quan đến 1 phạm vi rộng lớn của thông tin.

Phân tích bao gồm:

- Lọc thông tin và xác định những sự cải tiến tiềm tàng. Các tài liệu đã nhận từ nhiều nguồn thông tin khác nhau được xem xét lại bởi những người chuyên nghiệp để xác định các cơ hội tiềm tàng cho CAPA. Bước này bao gồm sự so sánh các tài liệu trong cùng 1 kiểu đã nhận được từ

các đơn vị khác nhau cũng như là việc so sánh các tài liệu của các kiểu khác nhau liên quan tới cùng 1 trường hợp giống nhau.

- Phân tích những sự cải tiến tiềm tàng. Cần cố gắng xác định:
 - Những kiểu và những mức của thiệt hại từ các lỗi được định danh
 - Lí do gây lỗi. Các lí do tiêu biểu là những chỉ dẫn và thủ tục không tuân theo công việc, kiến thức chuyên môn không đủ, quá giờ và/hoặc ngân sách quản bách do ước lượng không thực tế, và việc thiếu kinh nghiệm với những công cụ phát triển mới.
 - Việc ước lượng quy mô của tổ chức-các lỗi tiềm tàng của từng kiểu. Thông tin này cần thiết để ước lượng tổng thiệt hại và xác định mức độ ưu tiên cho từng trường hợp lỗi.
- Tạo ra phản hồi trong nội dung và sự hợp thức hóa của thông tin nhân được từ những nguồn thông tin đã được chỉ rõ.

Phát triển các giải pháp và thực thi

- Phát triển các giải pháp

Giải pháp để xác định nguyên nhân gây ra lỗi hệ thống phần mềm yêu cầu:

- Loại bỏ tái phát của các loại lỗi được phát hiện
- Đóng góp vào cải thiện hiệu quả của sản xuất và tạo năng suất cao hơn lịch trình ngắn hơn.

Một số hướng dẫn cho các giải pháp thường được chọn:

- Cập nhật các thủ tục liên quan. Các thay đổi có thể tham khảo một số thủ tục khác, từ những người cụ thể liên quan đến giai đoạn phát triển phần mềm hoặc bảo trì (ví dụ như, thay đổi trong nhận xét phần mềm, các thay đổi thủ tục hợp đồng trong giao dịch với điều khoản đề xuất cho các dự án nhỏ) để các thủ tục chung (ví dụ như các thay đổi của nhân viên tuyển dụng, các thay đổi tối đa và tối thiểu số của người tham gia nhận xét một thiết kế chính thức).
- Thay đổi trong thực hành, bao gồm cả công việc cập nhật các hướng dẫn có liên quan (nếu tồn tại).
- Chuyển đến một công cụ phát triển hiệu quả hơn và ít phát hiện ra lỗi.
- Cải tiến phương pháp báo cáo, bao gồm các thay đổi trong nội dung báo cáo,

tần số của báo cáo và công việc báo cáo. Đây là hướng cải thiện triển vọng cho việc xác định các lỗi và phần mềm hệ thống sớm phát hiện ra lỗi để mang lại kết quả đáng kể trong việc giảm thiệt hại.

- Các sáng kiến đào tạo, đào tạo lại hoặc cập nhật nhân viên. Đây là hướng thực hiện chỉ trong trường hợp khi cùng một thiếu sót đang đào tạo được tìm thấy trong một vài teams

- Thực hiện một quá trình CAPA

Việc triển khai thực hiện các giải pháp CAPA dựa vào đúng các hướng dẫn và thường xuyên đào tạo, nhưng hầu hết trên tất cả các hợp tác của các đơn vị có liên quan và riêng lẻ. Vì vậy, việc triển khai thực hiện thành công mục tiêu, đòi hỏi nhân viên, các thành viên được thuyết phục thích đáng đối với các giải pháp đề xuất. Nếu không có hợp tác, sự đóng góp của một CAPA có thể không được quyết định.

Tổ chức các hành động phòng ngừa và sửa lỗi

Sự thể hiện đúng đắn của những hoạt động CAPA này phụ thuộc vào sự tồn tại của các đơn vị tổ chức trung tâm lâu dài cũng như rất nhiều người tham gia trong đội ad-hoc. Nhân tố này nhìn chung được biết đến là Corrective Action Board (CAB), mặc dù nó có thể có những tên gọi khác nhau trong những tổ chức khác nhau, nhưng nó đều đảm nhiệm các công việc sau đây:

- Thu thập bản ghi CAPA từ các nguồn khác nhau.
- Hiển thị các thông tin đã được thu thập.
- Chỉ định toàn bộ đội CAPA ad-hoc tham gia vào công việc đã được đưa ra hoặc dẫn dắt một số các team đó.
- Xúc tiến việc thực thi CAPA trong các đơn vị, dự án....
- Liên tục thu thập thông tin, phân tích dữ liệu, quá trình được thực hiện bởi các đội ad-hoc và sự thực thi cũng như kết quả của các phương thức capa đã được cải tiến.

5.4 Quản lý cấu hình

Quản lý cấu hình phần mềm là một thành phần của quản lý chất lượng phần mềm, chịu trách nhiệm quản lý các thay đổi và trả lời chính xác toàn bộ các câu hỏi

tương tự như trên. Quản lý cấu hình phần mềm xử lí các vấn đề liên quan đến việc kiểm soát các thay đổi phần mềm, viết tài liệu phù hợp cho các thay đổi, đăng ký và lưu trữ các phiên bản phần mềm đã được phát hành, cung cấp các thông tin liên quan và bản sao các phiên bản đã được đăng ký trong suốt vòng đời của hệ thống phần mềm.

5.4.1 Các thành phần cấu hình phần mềm

Định nghĩa các thành phần cấu hình phần mềm, các phiên bản cấu hình phần mềm:

- Thành phần cấu hình phần mềm (SCI – Software Configuration Item)
hay thành phần cấu hình(CI – Configuration Item)

Một phần mã ,một tài liệu hay một mảnh phần cứng được thiết kế để quản lý cấu hình và được xem như các thực thể phân biệt trong tiến trình quản lý cấu hình phần mềm.

- Phiên bản SCI

Trạng thái của một SCI được công bố tại bất kỳ thời điểm nào trong suốt tiến trình phát triển và bảo trì

- Phiên bản cấu hình phần mềm

Một tập các phiên bản SCI trong hệ thống phần mềm được lựa chọn để phát hành hoặc viết tài liệu tại thời điểm phát hành, mô tả các hoạt động được thực hiện được điều khiển bởi quy trình quản lý cấu hình phần mềm.

5.4.2 Quản lý cấu hình phần mềm

Các nhiệm vụ của quản lý cấu hình phần mềm có thể chia làm 4 nhóm:

- Điều khiển thay đổi phần mềm
- Công bố các phiên bản SCI và cấu hình phần mềm
- Dịch vụ cung cấp thông tin quản lý cấu hình
- Thẩm định việc thực thi quy trình quản lý cấu hình phần mềm

5.4.3 Kiểm soát sự thay đổi phần mềm

Quản lý thay đổi phần mềm kiểm soát các tiến trình thay đổi bằng việc thực hiện các bước sau đây:

- Kiểm tra các yêu cầu thay đổi và phê chuẩn thực thi các yêu cầu phù hợp

- Đảm bảo chất lượng của mỗi phiên bản mới của cấu hình phần mềm trước khi nó được đưa ra sử dụng

Phé chuẩn để thực hiện các thay đổi được đề xuất

Các nhân tố ảnh hưởng đến việc quyết định có thực hiện thay đổi được đề xuất hay không:

- Mục đích mong đợi của thay đổi được đề xuất
- Sự khẩn cấp của thay đổi
- Tác động của thay đổi được đề xuất lên bảng thời gian dự án, mức độ dịch vụ...
- Sự nỗ lực đòi hỏi để thực hiện yêu cầu
- Chất lượng phần mềm được yêu cầu đảm bảo kết quả của sự nỗ lực
- Ước lượng tài nguyên và giá cả cần thiết để thực hiện thay đổi

Đảm bảo chất lượng của thay đổi phần mềm

Mục đích của đảm bảo chất lượng phần mềm là chất lượng của phiên bản mới không bị thua kém so với phiên bản trước đó.

Sự cố gắng đảm bảo chất lượng được yêu cầu ở 2 mức độ:

- Đảm bảo chất lượng của mỗi SCI thay đổi
- Đảm bảo chất lượng của toàn bộ phiên bản mới (bao gồm cả SCI thay đổi)

Đảm bảo chất lượng của các SCI thay đổi

Điều này yêu cầu phải có sự chuẩn bị cho kế hoạch review và test lại các đặc điểm thay đổi phù hợp với mức độ quan trọng của thay đổi đó. Việc test và review cần được thực hiện bởi các tester chuyên nghiệp chứ không phải là người phát triển SCI đó.

Đảm bảo chất lượng của toàn bộ phiên bản mới

Khi các SCI đã thay đổi thay thế cho các SCI trước đó, người ta thường xem xét đến một phiên bản mới cho phần mềm. Mặc dù người ta thường mong đợi nhiều ở phiên bản mới sẽ có chức năng hoàn hảo và tốt hơn phiên bản cũ, nhưng nhiều phiên bản mới, đặc biệt trong các hệ thống phức tạp, thường bị lỗi. Các hệ thống

thường bị lỗi khi kết quả ảnh hưởng tới giao diện của các SCI đã thay đổi với các SCI không thay đổi khác, bởi vì người ta không dự định trước những rủi ro khi thực hiện thay đổi. Vì vậy, toàn bộ phiên bản mới, hoặc ít nhất là toàn bộ các thành phần phần mềm có thể bị ảnh hưởng cần phải được test để xác định các nhược điểm giao diện không mong đợi.

5.4.4 Kiểm soát quản lý cấu hình phần mềm

SCM liên quan đến việc thực thi một số lượng lớn các tác vụ khác nhau bởi các authority (những người có quyền) SCM, các CCB (nhóm kiểm soát thay đổi) và các thành viên khác liên quan đến trong việc phát triển và bảo trì phần mềm. Tất cả các tác vụ tương ứng với từng thành viên được xác định trong tiến trình SCM. Kiểm tra, xem xét SCM được thực hiện bởi SCM authority và CCB để điều khiển theo đúng quy trình SCM. SCM audits có thể kết hợp với vấn đề chất lượng bên trong để khởi tạo việc cập nhật và thay đổi trong quy trình SCM. Do đó SCM audits kiểm tra xem các tác vụ này được thực hiện như thế nào bằng cách thực hiện một số ví dụ về yêu cầu thay đổi, SCI và phiên bản cấu hình phần mềm. SCM audit có thể thực hiện cho một số mẫu phát hành đã được lên kế hoạch, như được đặc tả trong SCMP. Tuy nhiên, mặc dù chúng ta mong đợi SCM audit đưa ra thông tin liên quan đến mức độ thực thi của quy trình SCM (bao gồm cả các lỗi thông thường của quy trình này) nhưng chúng không thể đáp ứng như các công cụ ép buộc thực thi

Dưới đây là danh sách các thông tin kiểm soát mà SCM audit thu thập và chuyển giao để quản lý:

- Tỷ lệ các thay đổi không được phê duyệt trong suốt quá trình phát triển cũng như hoạt động của hệ thống
- Tỷ lệ của SCO không được thực hiện theo yêu cầu và không tuân theo đầy đủ quy trình
- Tỷ lệ việc xem lại thiết kế và test lại phần mềm của các SCI đã thay đổi không được thực hiện theo quy trình liên quan
- Tỷ lệ của SCO được hoàn thành trên kế hoạch
- Tỷ lệ các trường hợp mà SCI bị tác động bởi các thay đổi không được kiểm tra, với một số thay đổi cần thiết không được thực thi

- Tỷ lệ các phiên bản cấu hình phần mềm và các SCI mới được viết tài liệu phù hợp
- Tỷ lệ các trường hợp lỗi chuyển giao tất cả các thông tin liên quan của phiên bản đến khách hàng
- Số các trường hợp được báo cáo hàng năm, nơi mà các SCI làm việc phối hợp các kỹ thuật bị lỗi (ví dụ như không ngăn chặn được các đội khác nhau đồng thời giới thiệu sự thay đổi trong cùng SCI)

5.4.5 Các công cụ máy tính quản lý cấu hình phần mềm

Các công cụ vi tính hóa cũng hoạt động theo kỹ thuật phối hợp hoạt động dựa trên thay đổi của SCI và ngăn cản các team khác nhau đồng thời giới thiệu cùng thay đổi lên một SCI

Một ưu điểm nữa khi dùng hệ thống SCM được vi tính hóa là mức độ bảo mật cao được cung cấp:

- Nó bảo vệ các phiên bản code và phiên bản file tài liệu khỏi bất cứ thay đổi, xóa hay nguy hiểm nào
- Nó hoạt động theo quy trình sao lưu được yêu cầu cho việc đảm bảo lưu trữ các file SCM

5.5 Kiểm soát tài liệu

5.5.1 Các tài liệu kiểm soát và bản ghi chất lượng

Tài liệu được kiểm soát :

Một tài liệu mà hiện tại cần thiết hoặc có thể trở nên cần thiết cho việc phát triển và bảo trì các hệ thống phần mềm cũng như cho việc quản lý các mối quan hệ hiện tại và trong tương lai với khách hàng. Do đó việc biểu diễn, lưu trữ, phục hồi và loại bỏ được kiểm soát bởi các thủ tục tài liệu. Mục đích chính cho việc quản lý các tài liệu được kiểm soát là :

- Để đảm bảo chất lượng tài liệu
- Để đảm bảo kỹ thuật đầy đủ và đúng với các thủ tục cấu trúc tài liệu và các chỉ dẫn (sử dụng các khuôn mẫu, đăng ký chính xác...)
- Để đảm bảo tính sẵn sàng sử dụng của các tài liệu trong tương lai mà có thể

được yêu cầu cho bảo trì hệ thống phần mềm, phát triển xa hơn hoặc đáp ứng các thắc mắc của khách hàng .

- Để hỗ trợ nghiên cứu các nguyên nhân làm thất bại phần mềm và để gán trách nhiệm như 1 phần của các hoạt động hiệu chỉnh và các hoạt động khác

Bản ghi chất lượng :

Là 1 kiểu tài liệu được kiểm soát đặc biệt. Nó là tài liệu nhằm tới khách hàng mà có thể được yêu cầu để chứng minh thỏa mãn các yêu cầu khách hàng và thực hiện hiệu quả hệ thống đảm bảo chất lượng phần mềm thông qua quy trình phát triển và bảo trì

Các thành phần của các thủ tục kiểm soát tài liệu

- Xác định danh sách của các loại tài liệu được kiểm soát và sự cập nhật tài liệu được kiểm soát
- Các yêu cầu chuẩn bị tài liệu
- Các yêu cầu phê chuẩn tài liệu

Các yêu cầu lưu trữ và phục hồi tài liệu, bao gồm lưu trữ các phiên bản, sự duyệt lại và việc loại bỏ các tài liệu được kiểm soát

5.5.2 Danh sách các tài liệu được kiểm soát

Vấn đề chính của việc quản lý các tài liệu được kiểm soát (bao gồm các bản ghi chất lượng) là xây dựng danh sách các kiểu tài liệu được kiểm soát. Việc xây dựng hợp lý danh sách này dựa trên việc thiết lập thẩm quyền để thực thi các khái niệm, mà quyền này gắn với một người hoặc một hội đồng người. Quyền này chịu trách nhiệm cho những vấn đề sau:

- Quyết định kiểu tài liệu nào được phân loại vào tài liệu được kiểm soát và các kiểu tài liệu được kiểm soát được phân loại như các bản ghi chất lượng
- Quyết định mức độ của việc kiểm soát là đầy đủ với mỗi kiểu tài liệu được phân loại như một tài liệu được kiểm soát
- Đưa ra sự đồng ý với danh sách các kiểu tài liệu được kiểm soát. Chủ đề này có thể được kết hợp trong kế hoạch thử nghiệm quản lý chất lượng bên

trong

- Phân tích theo việc tìm kiếm và khởi tạo các cập nhật được yêu cầu, các thay đổi, các loại bỏ và sự bổ xung cho danh sách các kiểu tài liệu được kiểm soát.

Hầu hết các kiểu tài liệu được kiểm soát là các tài liệu được tạo bên trong bởi chính tổ chức đó. Tuy nhiên số lượng thực các kiểu tài liệu bên ngoài như tài liệu hợp đồng và biên bản cuộc họp hội đồng cũng được phân vào loại này.

5.5.3 Chuẩn bị, phê chuẩn, lưu trữ và thu hồi tài liệu kiểm soát

Các yêu cầu tài liệu bao gồm việc tạo một tài liệu mới hoặc sửa lại tài liệu cũ để cải thiện tính dễ đọc và khả dụng. Các yêu cầu được thực hiện trong các tài liệu:

Cấu trúc

Phương pháp định danh

Thông tin tham khảo và sự định hướng theo chuẩn.

Phê chuẩn tài liệu.

Số tài liệu cần có sự phê chuẩn còn các tài liệu khác có thể được bỏ qua việc đánh giá kết hợp. Đối với việc phê chuẩn các tài liệu thì các thủ tục chỉ ra vị trí của người hoặc nhóm người có thẩm quyền phê chuẩn cho mỗi tài liệu và chi tiết quy trình phê chuẩn.

- Vị trí của người mà có thể phê chuẩn tài liệu hoặc kiểu tài liệu

Sự phê chuẩn có thể bởi một người hoặc một vài người, hoặc một hội đồng như hội đồng đánh giá thiết kế FDR – tùy theo kiểu tài liệu và sở thích của tổ chức. Vị trí này dành cho những người phải có kinh nghiệm và chuyên gia công nghệ.

- Quy trình phê chuẩn

Sự phê chuẩn tài liệu được yêu cầu để đảm bảo chất lượng tài liệu, sự phê chuẩn cũng nhằm mục đích phát hiện ra và ngăn ngừa việc thiếu chuyên nghiệp cùng với sự chênh lệch khuôn mẫu tài liệu. Tùy trường hợp mà sự phê chuẩn FDR được yêu cầu, thủ tục đánh giá thích hợp sẽ được áp dụng.

Việc theo dõi quy trình phê chuẩn bộc lộ thường xuyên những trường hợp sự tán thành không suy nghĩ, đó là, những tình huống mà quy trình không góp phần vào chất lượng phần mềm do sự bỏ qua hoặc thờ ơ với việc rà soát kỹ lưỡng tài liệu. Sự xác nhận việc chấp thuận hình thức làm giảm chất lượng tài liệu bởi vì những người có quyền phê chuẩn tài liệu phải chịu trách nhiệm trực tiếp với chất lượng tài liệu. Theo đó, có 2 vị trí có thể được xem xét cho các kiểu tài liệu: Các tài liệu bỏ qua sự phê chuẩn thì người chịu trách nhiệm là tác giả, hoặc tài liệu thực thi quy trình phê chuẩn thì đảm bảo việc rà soát tỉ mỉ tài liệu. Một khác, giải pháp cho việc tán thành không xem xét kĩ là chỉnh sửa lại quy trình phê chuẩn hoặc bỏ đi toàn bộ quy trình.

lưu trữ tài liệu được kiểm soát

Các yêu cầu liên quan đến việc lưu trữ và lấy tài liệu được thiết lập để đảm bảo tính bảo mật và tính sử dụng tiếp tục. Các yêu cầu áp dụng tài liệu giấy tờ cũng như các tài liệu điện tử và các tài liệu khác.

Chúng liên quan tới:

Lưu trữ tài liệu

Tính lưu thông và truy xuất của tài liệu

Bảo mật tài liệu bao gồm sự sắp xếp tài liệu

Các yêu cầu về lưu trữ tài liệu áp dụng cho số lượng các bản copy được lưu trữ, đơn vị chịu trách nhiệm lưu trữ mỗi bản copy, và phương tiện lưu trữ. Lưu trên media điện tử thường hiệu quả nhiều hơn và kinh tế hơn lưu trữ trên giấy. Tuy nhiên tài liệu giấy phù hợp với các điều khoản. Trong trường hợp này, một bản copy được lưu trữ thêm một nguyên bản giấy.

Các yêu cầu lưu thông và lấy tài liệu liên quan tới hướng dẫn thông hành một tài liệu mới, về thời gian, liên quan tới người nhận tài liệu và liên quan tới việc lấy chính xác và hiệu quả bản copy phù hợp với sự bảo mật. Thủ tục nên được áp dụng cho sự lưu thông tài liệu giấy cũng như sử dụng e-mail, intranet, internet.

Về bảo mật tài liệu, bao gồm các yêu cầu sắp xếp tài liệu: cung cấp giới hạn truy cập giới hạn vào các kiểu tài liệu, ngăn chặn sự thay đổi tài liệu trái phép, cung cấp việc phục hồi lưu trữ vào giấy cũng như các file, và xác định các giai đoạn lưu trữ. Cuối cùng của giai đoạn lưu trữ, các tài liệu có thể bị loại bỏ hoặc vứt vào kho chứa ở mức thấp hơn, một sự dịch chuyển như vậy thường làm giảm tính khả dụng

của tài liệu. Trong khi các file giấy có thể bị cháy hoặc bị nước thì những tài liệu lưu trữ điện tử hiện đại phải chịu được những rủi ro. Phương pháp được lên kế hoạch cho lưu trữ phản ánh mức độ rủi ro và mức độ quan trọng của các tài liệu.

Chương 6 . Các thành phần quản lý chất lượng phần mềm

6.1 Điều khiển tiến độ dự án

6.1.1 Các thành phần điều khiển tiến độ dự án

Quản lý tiến độ dự án có một mục tiêu tức thời : sớm phát hiện ra sự kiện bất bình thường. Việc phát hiện đó thúc đẩy thời gian bắt đầu của đáp ứng giải quyết vấn đề. Thông tin chính xác trong quản lý tiến độ cũng như những thành công và những thất bại cũng phục vụ cho những đối tượng lâu dài: Những hành động đúng đắn đầu tiên.

Thành phần chính của quản lý tiến độ dự án:

- Điều khiển những hoạt động quản lý rủi ro
- Quản lý lịch trình dự án
- Quản lý tài nguyên dự án
- Quản lý ngân sách dự án

6.1.2 Điều khiển tiến độ của các dự án nội bộ và các thành phần bên ngoài

Quản lý tiến trình được khởi tạo để cung cấp cho người quản lý cái nhìn toàn diện về tất cả hoạt động phát triển phần mềm được thực hiện trong tổ chức. Tuy nhiên, trong hầu hết các tổ chức, quản lý dự án quy định, với các lý do khác nhau, có sự giới hạn về tầm nhìn của phát triển phần mềm bên trong và thậm chí còn bị giới hạn nhiều hơn của tiến trình được tạo bởi những người tham gia bên ngoài.

6.1.3 Thực thi kiểm soát tiến độ dự án

Điều khiển tiến độ dự án thường dựa trên thủ tục được chỉ ra:

- Sự phân chia trách nhiệm cho sự thực thi của các công việc điều khiển tiến độ mà thích hợp với thành phần của dự án, bao gồm các chuẩn mực(size):
- Cá nhân hoặc đơn vị quản lý có trách nhiệm thực thi công việc điều khiển tiến độ.
 - o Tính thường xuyên của các bản báo cáo yêu cầu từ mỗi đơn vị dự án ~~lên~~ mức độ hành chính.
 - o Các tình huống yêu cầu những người chỉ đạo dự án báo cáo trực tiếp ~~đến~~ người quản lý.
 - o Các tình huống yêu cầu người quản lý thấp hơn báo cáo trực tiếp ~~tới~~ người quản lý cao hơn.
 - o Quản lý kiểm tra của tiến độ dự án, giải quyết những vấn đề chính:
 - (1) Làm ~~lên~~ nào có những báo cáo tiến độ tốt được đưa ra bởi những người chỉ đạo dự án và từ người quản lý thấp hơn đến người quản lý cao hơn, và (2) điều khiển quản lý chi tiết là tích cực để bắt đầu.

Trong những tổ chức phát triển phần mềm lớn, điều khiển tiến độ dự án có thể quản lý được những trình độ của những người quản lý khác nhau, như là quản lý lĩnh vực phần

mềm, quản lý sự phân chia phần mềm và quản lý cao nhất (top management). Mặc dù mỗi mức độ đòi hỏi định nghĩa chế độ điều khiển tiến độ dự án, cái mà phản ánh tham số đầy đủ để ước định tiến độ của dự án từ vị trí riêng biệt đó, phối hợp giữa các mức độ khác nhau là có tính bắt buộc để điều khiển tiến độ có hiệu quả.

Toàn bộ bản báo cáo sẽ truyền thông tin được lựa chọn từ mức thuộc ban quản lý thấp nhất - báo cáo tiến độ định kỳ của người chủ dự án - nó tổng kết lại trạng thái của những rủi ro dự án, lập lịch dự án và tính hữu ích của các tài nguyên, đây cũng là 3 thành phần đầu tiên của quá trình điều khiển tiến độ, Người chủ dự án dựa trên báo cáo tiến độ của mình được tập hợp từ người chủ của các nhóm.

6.1.4 Các công cụ kiểm soát tiến độ phần mềm

Tổ chức các công cụ cho việc điều khiển tiến độ dự án là một nhu cầu dễ hiểu cho việc kích thước ngày càng tăng và độ phức tạp của dự ánvà lợi ích mà chúng mang lại cho họ trên những mặt khác. Những công cụ quản lý dự án toàn diện đã có sẵn trên thị trường nhiều năm có thể phục vụ hầu hết các thành phần điều khiển

của những dự án phần mềm khá có hiệu quả. Phần lớn các gói đa dụng áp dụng cho việc phân tích PERT/CPM vì vậy mà những bản ghi kết quả giữ sự tương tác giữa các hành động và việc phê bình của mỗi hành động bên trong một tài khoản. Đây là những gói thông thường thích ứng với những trường hợp đặc biệt vì sự đa dạng lớn của những tùy chọn mà họ đưa ra.

Điều khiển ngân sách dự án:

1. Kế hoạch ngân sách dự án bởi các hoạt động và các mô đun phần mềm, cho các nhóm, các đơn vị phát triển, chỉ định những thời hạn, vv...
2. Những báo cáo sử dụng ngân sách dự án bởi các thời kỳ hoặc được tích lũy như đã chỉ rõ ở trên
3. Sự chênh lệch trong việc sử dụng ngân sách dự án bởi các thời kỳ hoặc được tích lũy như đã chỉ rõ ở trên.
4. Việc cập nhật kế hoạch ngân sách dự án phát sinh tùy thuộc vào báo cáo tiến độ và sửa chữa độ đo ứng dụng, vv...

6.2 Độ đo chất lượng phần mềm

Hai cách khác nhau được định nghĩa bổ sung(IEEE, 1990) miêu tả độ đo chất lượng phần mềm như một phạm trù của công cụ SQA.

- (1) Một sự đo lường định lượng mức độ để một khoản mục có một thuộc tính chất lượng đã quy định.
- (2) Đầu vào của một chức năng là dữ liệu phần mềm và đầu ra của nó là một giá trị số duy nhất có thể được giải thích như mức độ để các phần mềm có một thuộc tính chất lượng đã quy định.

Tức là, định nghĩa thứ hai để cập đến quá trình đưa ra các độ đo chất lượng trong khi định nghĩa đầu tiên để cập đến kết quả của quá trình nói trên.

Thông thường người ta tin rằng độ đo chất lượng phần mềm nên chứa trong phần mềm, như trong các ngành công nghiệp khác, giữa các công cụ cơ bản được dùng để giúp đỡ sự quản lý trong ba lĩnh vực cơ bản sau: *điều khiển phát triển và bảo trì phần mềm, hỗ trợ đưa ra quyết định, khởi tạo các hoạt động hiệu chỉnh*. Phân tích thống kê các dữ liệu độ đo được mong đợi để xác định (sinh động và có ý nghĩa thống kê) những thay đổi đã được khởi tạo như một kết quả của ứng dụng

của công cụ phát triển mới, thủ tục thay đổi và những sự can thiệp khác.

Phạm vi của độ đo chất lượng phần mềm đã được mở rộng đáng kể qua một vài thập niên trước đây. Chúng ta sẽ xem xét lại một số việc bảo trì và phát triển phần mềm thích hợp nhất đưa ra ở trọng tâm của chương này. Độ đo như là một công cụ đảm bảo chất lượng phần mềm, không may, không được ứng dụng ở mức độ đầy đủ trong công nghiệp phần mềm, cũng không được cung cấp lợi ích ở mức độ mong đợi. Chỉ một phần nhỏ của tổ chức phát triển phần mềm áp dụng độ đo chất lượng phần mềm một cách có hệ thống.

6.2.1 Các mục tiêu đo lường phần mềm và phân loại các độ đo

a. Mục tiêu đo lường phần mềm

Mục tiêu chính của đo lường chất lượng phần mềm

- Để thuận tiện cho việc điều khiển quản lý cũng như lập kế hoạch và thực thi ~~sự~~ can thiệp quản lý thích hợp. Đạt được mục đích này dựa trên sự tính toán của độ đo đối với:
- Độ chênh lệch giữa sự thực thi chức năng (chất lượng) thực tế từ sự thực thi ~~đã~~ lập kế hoạch.
- Độ chênh lệch của thực hiện ngân sách và thời gian biểu thực tế từ sự thực thi ~~đã~~ lập kế hoạch.
- Để xác định trạng thái (situation) yêu cầu hoặc cho phép cải tiến quy trình ~~phát~~ triển hay bảo trì dưới dạng các hoạt động ngăn ngừa và sửa đổi được đưa ra trong suốt tổ chức. Đạt mục tiêu này dựa trên:

Tích lũy thông tin về độ đo đối với sự thực hiện của cả nhóm, đơn vị,..

Việc so sánh sẽ cung cấp thực tế cơ bản cho ứng dụng quản lý của độ đo và sự tiến bộ của SQA nói chung. Độ đo được sử dụng cho việc so sánh dữ liệu thực thi với chỉ dẫn (indicator), giá trị định lượng như sau:

Xác định các chuẩn chất lượng phần mềm.

Tập hợp mục tiêu chất lượng cho tổ chức và cá nhân.

Thành tựu chất lượng trong năm trước.

Thành tựu chất lượng của dự án trước.

Mức độ chất lượng trung bình đạt được bởi các nhóm khác áp dụng cùng công ~~qi~~ phát triển trong môi trường phát triển giống nhau.

Thành tựu chất lượng trung bình của tổ chức.

Thực tiễn công nghiệp cho sự thỏa mãn yêu cầu chất lượng.

b. *Phân loại độ đo chất lượng phần mềm*

Độ đo chất lượng phần mềm có thể chia thành một số nhóm. Ở đây chúng tôi sử dụng một hệ thống 2 mức độ:

Thứ nhất, phân loại danh mục phân biệt giữa vòng đời và các pha khác nhau của hệ thống phần mềm:

- Độ đo quy trình, liên quan đến quá trình phát triển phần mềm
- Độ đo Sản phẩm, liên quan đến bảo trì phần mềm.

Thứ hai, phân loại danh mục đề cập đến các chủ đề về đo lường:

- Chất lượng.
- Thời gian biểu.
- Hiệu quả (của lỗi và xóa bỏ các dịch vụ bảo trì).
- Năng suất.

Các mục này được chia với với các nhóm riêng biệt.

Một số lượng lớn độ đo chất lượng phần mềm liên quan đến một trong hai cách đo lường cho kích thước hệ thống như sau:

KLOC - thước đo cỗ điển này đo kích cỡ của phần mềm bởi hàng ngàn dòng lệnh. Vì số lượng các dòng mã cần thiết cho lập trình một tác vụ đã quy định về cơ bản là khác nhau đáng kể với mỗi công cụ lập trình, phép đo này là cụ thể cho ngôn ngữ lập trình và công cụ phát triển được sử dụng. Việc áp dụng độ đo đó bao gồm KLOC được giới hạn trong các hệ thống phần mềm phát triển bằng cách sử dụng cùng một ngôn ngữ lập trình hay công cụ phát triển.

Điểm chức năng – một phép đo các tài nguyên phát triển (nguồn lực con người) được yêu cầu để phát triển một chương trình, dựa trên chức năng đã xác định cho hệ thống phần mềm.

Các độ đo thỏa mãn khách hàng sẽ không được trình bày ở đây, các đọc giả có thể tìm thấy phạm vi mở rộng của chủ đề này trong bài giảng marketing.

6.2.2 *Các độ đo tiến trình*

Các độ đo quy trình bao gồm:

- Các độ đo chất lượng quy trình phần mềm
 - Các độ đo thời gian biểu của quy trình phần mềm
 - Các độ đo tính hiệu quả loại bỏ lỗi
 - Các độ đo năng suất tiến trình phần mềm
- Độ đo chất lượng quy trình phần mềm bao gồm:

- Độ đo mật độ lỗi:

Code	Tên	Công thức tính
CED	Mật độ lỗi code	$CED = \frac{NCE}{KLOC}$
DED	Mật độ lỗi phát triển	$DED = \frac{NDE}{KLOC}$
WCED	Mật độ lỗi code đã đánh trọng số	$WCED = \frac{WCE}{KLOC}$
WDDED	Mật độ lỗi phát triển đã đánh trọng số	$WDDED = \frac{WDE}{KLOC}$
WCEF	Lỗi code đã đánh trọng số cho mỗi điểm chức năng	$WCEF = \frac{WCE}{NFP}$
WDEF	Lỗi phát triển đánh trọng số cho mỗi điểm chức năng	$WDEF = \frac{WDE}{NFP}$

Từ khóa:

NCE= số lỗi code được phát hiện trong code phần mềm bởi quá trình test và duyệt. Dữ liệu cho phép đo này được chọn lọc từ các báo cáo test và duyệt code.

KLOC= hàng nghìn dòng code

NDE= tổng số lỗi phát triển (thiết kế và code) được phát hiện trong quy trình phát triển phần mềm. Dữ liệu cho phép đo này được tìm ra trong rất nhiều báo cáo test và soát code và thiết kế.

WCE= lỗi code đánh trọng số được phát hiện. Nguồn dữ liệu cho độ đo này là tương tự như nguồn của NCE.

WDE= tổng lỗi phát triển (thiết kế vào code) đánh trọng số được phát hiện trong phát triển của phần mềm. Các nguồn dữ liệu cho độ đo này giống với NDE.

NFP: số điểm chức năng được yêu cầu cho phát triển phần mềm. Nguồn gốc số điểm chức năng là sự khảo sát chuyên nghiệp của phần mềm liên quan.

- Độ đo tính nghiêm trọng của lỗi

Code	Tên	Công thức tính
ASCE	Trung bình nghiêm trọng của lỗi code	$ASCE = \frac{WCE}{NCE}$
ASDE	Trung bình nghiêm trọng của lỗi phát triển	$ASDE = \frac{WDE}{NDE}$

- Độ đo thời gian biểu của quy trình phần mềm.

Code	Tên	Công thức tính
TTO	Tuân theo thời gian biểu	$TTO = \frac{MSOT}{MS}$
ADMC	Trung bình trễ hoàn thành mốc quan trọng	$ADMC = \frac{TCDAM}{MS}$

Bảng Các độ đo thời gian biểu tiến trình phần mềm

Từ khóa:

MSOT= Những mốc quan trọng(milestones) đã hoàn thành đúng thời gian

MS= tổng số mốc quan trọng

TCDAM= tổng các trễ hoàn thành (hàng ngày, hàng tuần...) cho tất cả mốc quan trọng. Để tính toán phép đo này, các trễ đã báo cáo cho tất cả các mốc quan trọng liên quan được tổng kết lại. Các mốc quan trọng hoàn thành đúng thời gian hoặc trước lịch biểu được xem là trễ ‘O’. Một số chuyên gia xem các mốc quan trọng hoàn thành trước lịch biểu như các trễ “âm”. Chúng được xét để cân bằng hiệu quả (hiệu lực) của các trễ “dương”. Trong trường hợp này, giá trị của ADMC có thể thấp hơn giá trị đạt được theo độ đo đề nghị ban đầu.

- Độ đo hiệu quả loại bỏ lỗi.

Code	Tên	Công thức tính
DERE	Hiệu quả loại bỏ lỗi phát	$DERE = \frac{NDE}{NDE + NYF}$

triển

DWERE Hiệu quả loại bỏ lỗi phát triển có đánh trọng số

$$DWERE = \frac{WDE}{WDE + WYF}$$

Từ khóa:

NYF: số thất bại(failures) phần mềm được phát hiện trong suốt 1 năm bảo trì dịch vụ. WYF: số thất bại phần mềm đã đánh trọng số trong suốt 1 năm bảo trì dịch vụ.

- Độ đo hiệu suất quy trình phần mềm.

4 độ đo năng suất tiến trình, trực tiếp và gián tiếp, được biểu diễn trong bảng 21.5.

Table 21.5: Process productivity metrics

Code	Name	Calculation formula
DevP	Development Productivity	$DevP = \frac{DevH}{KLOC}$
FDevP	Function point Development Productivity	$FDevP = \frac{DevH}{NFP}$
CRe	Code Reuse	$CRe = \frac{ReKLOC}{KLOC}$
DocRe	Documentation Reuse	$DocRe = \frac{ReDoc}{NDoc}$

Trong đó:

DevH= tổng thời gian làm việc dành cho phát triển của hệ thống phần mềm. ReKLOC = số dòng lệnh sử dụng lại trong hàng nghìn dòng code

ReDoc = số trang sử dụng lại của tài liệu

Ndoc= số trang của tài liệu

6.2.3 Các độ đo sản phẩm

Độ đo sản phẩm (product metrics) bao gồm:

- Độ đo chất lượng HD (Help Desk)
- Độ đo năng suất và hiệu quả HD
- Độ đo chất lượng bảo trì sửa chữa

- Độ đo năng suất và hiệu quả bảo trì sửa chữa phần mềm

Độ đo chất lượng HD

Các kiểu của độ đo chất lượng HD được thảo luận ở đây đề cập đến:

- Độ đo mật độ cuộc gọi HD- phạm vi của yêu cầu khách hàng cho dịch vụ HD được đo bởi số cuộc gọi.
- Độ đo tính nghiêm trọng của HD được tăng lên.
- Độ đo thành công HD - Mức độ của sự thành công trong sự phản ứng lại tới những cuộc gọi này. Thành công đạt được bằng sự hoàn thành dịch vụ yêu cầu trong thời gian nhất định của hợp đồng dịch vụ.

Độ đo năng suất và hiệu quả HD

Độ đo năng suất liên quan tới tổng tài nguyên đầu tư trong suốt một giai đoạn cụ thể, trong khi độ đo hiệu quả liên quan tới tài nguyên đầu tư trong đáp ứng lại tới 1 cuộc gọi HD.

Độ đo năng suất HD

Độ đo năng suất HD làm cho việc sử dụng của phép đo KLMC dễ dàng áp dụng cho kích thước hệ thống phần mềm bảo trì hoặc theo ước lượng điểm chức năng của hệ thống phần mềm. Hai độ đo năng suất HD được miêu tả trong hình sau:

Table 21.7: HD productivity metrics

Code	Name	Calculation formula
HDP	HD Productivity	$HDP = \frac{HDYH}{KLMC}$
FHDP	Function point HD Productivity	$FHDP = \frac{HDYH}{NMFP}$

Trong đó:

- HDYH = Tổng thời gian làm việc hàng năm đầu tư trong dịch vụ HD của hệ thống phần mềm.
- KLMC và NMFP được định nghĩa ở phần trước.

Độ đo hiệu quả HD

Độ đo trong nhóm này liên quan tới tài nguyên được đầu tư trong đáp ứng lại cuộc

gọi HD của khách hàng. Một độ đo thông dụng được miêu tả ở đây, hiệu quả HD (HDE):

$$HDE = \frac{HDYH}{NHYC}$$

Độ đo bảo trì sửa chữa phần mềm xử lý hàng loạt khía cạnh chất lượng của dịch vụ bảo trì. Một sự khác biệt cần thiết giữa thất bại hệ thống phần mềm được giải quyết bởi nhóm bảo trì và thất bại do dịch vụ bảo trì liên quan tới trường hợp bảo trì không cung cấp một sự sửa chữa thỏa mãn những tiêu chuẩn hoặc yêu cầu hợp đồng. Do đó, độ đo bảo trì phần mềm được phân loại như sau:

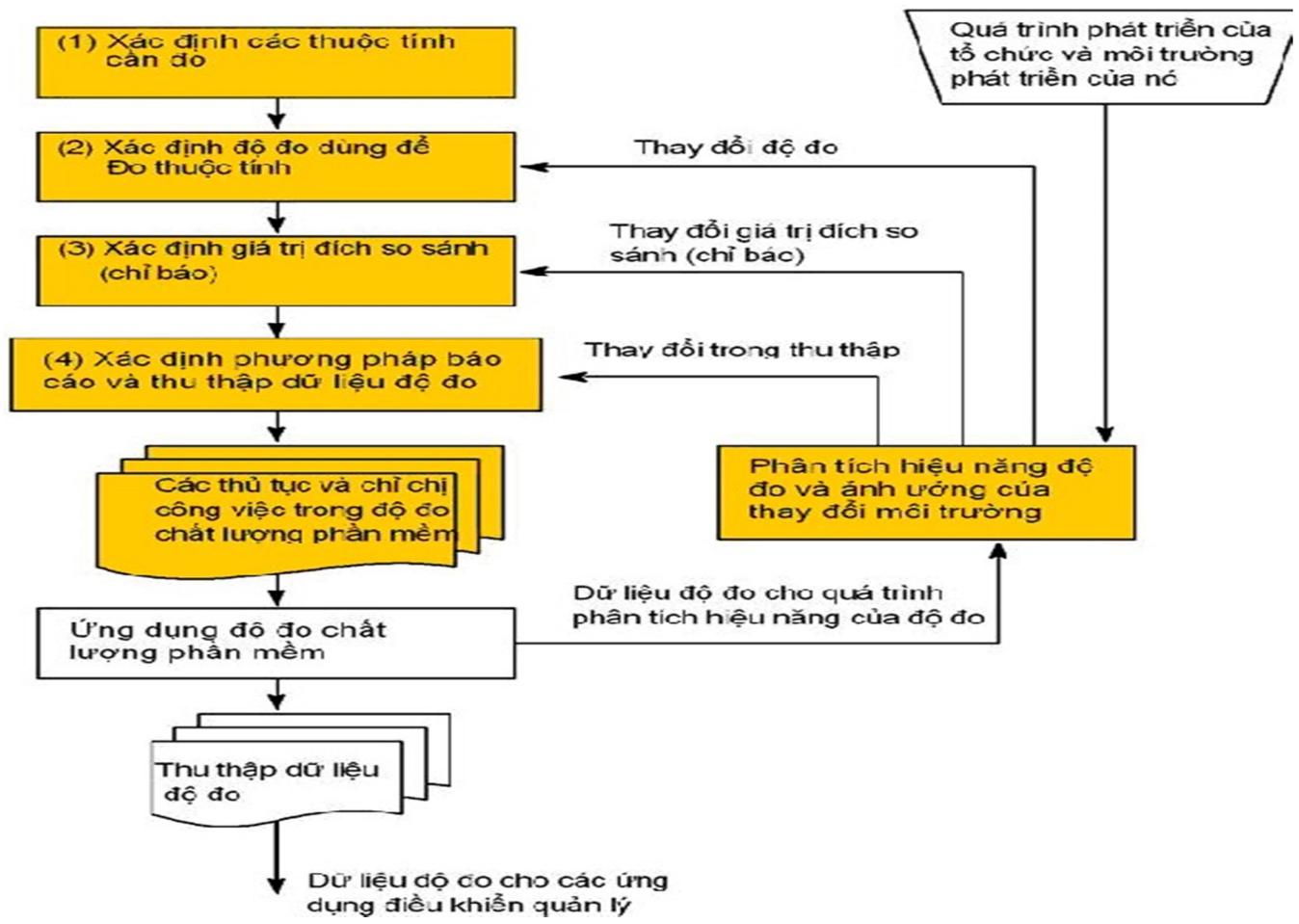
- **Độ đo mật độ thất bại hệ thống phần mềm:** Giải quyết phạm vi của yêu cầu tới bảo trì sửa chữa, dựa trên bản ghi của các thất bại được xác định trong hoạt động bình thường của hệ thống phần mềm.
- **Độ đo nghiêm trọng thất bại hệ thống phần mềm:** Giải quyết sự nghiêm trọng của thất bại hệ thống phần mềm quan tâm bởi nhóm bảo trì sửa chữa.
- **Độ đo thất bại của dịch vụ bảo trì:** Giải quyết trường hợp mà dịch vụ bảo trì không thể hoàn thành sửa chữa thất bại đúng thời gian hoặc việc sửa lỗi thất bại.
- **Độ đo sự sẵn sàng hệ thống phần mềm:** Giải quyết phạm vi của sự xáo trộn được gây ra tới khách hàng như thực hiện bởi một giai đoạn thời gian nơi dịch vụ của hệ thống phần mềm không sẵn sàng hoặc chỉ sẵn sàng một phần.

6.2.4 Thực hiện đo chất lượng phần mềm

Để thực hiện đo chất lượng phần mềm cần:

- Xác định các độ đo chất lượng phần mềm
- Áp dụng thông thường bởi đơn vị...
- Phân tích thống kê dữ liệu độ đo đã thu thập được.
- Các hành động tiếp theo bao gồm:
 - + Các thay đổi trong tổ chức và các phương pháp của các đơn vị bảo trì và phát triển phần mềm và/hoặc bất kỳ cá nhân nào thu thập dữ liệu độ đo.

- + Thay đổi về độ đo và thu thập dữ liệu độ đo.
- + Áp dụng dữ liệu và phân tích dữ liệu nhằm lên kế hoạch cho các hoạt động sửa lỗi cho tất cả các đơn vị liên quan.



Quá trình xác định các độ đo chất lượng phần mềm

6.2.5 Những giới hạn của các độ đo phần mềm

Ứng dụng của độ đo chất lượng có nhiều cản trở. Những cản trở này có thể nhóm theo sau:

- Ngân sách ràng buộc trong chỉ định tài nguyên cần thiết (nhân sự, tài chính, vv..) cho phát triển của hệ thống độ đo chất lượng và ứng dụng bình thường của nó.
- Yếu tố con người, đặc biệt là sự chống lại của người lao động để ước lượng những hoạt động này.
- Không chắc chắn về giá trị của dữ liệu, đã ăn sâu vào các bản ghi thiên vị và thành kiến

6.3 Giá thành của chất lượng phần mềm

Các mục tiêu tính giá thành các độ đo chất lượng phần mềm

Mục tiêu của các phép đo chi phí của chất lượng phần mềm

Việc áp dụng các phép đo chi phí cho đảm bảo chất lượng phần mềm cho phép có được sự quản lý tài chính đối với các hoạt động SQA và các kết quả. Các mục tiêu cụ thể là:

- Quản lý tổ chức – xây dựng chi phí để ngăn ngừa và dò tìm lỗi phần mềm
- Uớc tính thiệt hại kinh tế của thất bại phần mềm để làm cơ sở xét duyệt ngân sách cho SQA
- Uớc lượng các kế hoạch để tăng hoặc giảm các hoạt động SQA hoặc để đầu tư mới hoặc cập nhập cơ sở hạ tầng SQA dựa trên cơ sở hiệu năng kinh tế trong quá khứ

Mô hình truyền thống tính giá chất lượng phần mềm

Mô hình chi phí chất lượng phần mềm được phát triển vào đầu những năm 1950 bởi Feigenbaum và một số người khác (Xem trong Feigenbaum, 1991), cung cấp một phương pháp luận cho việc phân loại chi phí kết hợp với đảm bảo chất lượng sản phẩm từ quan điểm về kinh tế. Được phát triển để phù hợp hoàn cảnh chất lượng dựa trên các tổ chức sản xuất, mô hình đã được thực hiện một cách rộng rãi.

Mô hình phân loại chi phí liên quan đến chất lượng sản phẩm vào hai lớp chung:

- Chi phí quản lý: bao gồm chi phí phải trả cho ngăn ngừa và dò tìm lỗi phần mềm để giảm thiểu chúng đến một mức độ được chấp nhận.
- Chi phí thất bại của quản lý: bao gồm chi phí của những thất bại xảy ra bởi vì thất bại trong việc ngăn ngừa và dò tìm lỗi phần mềm.

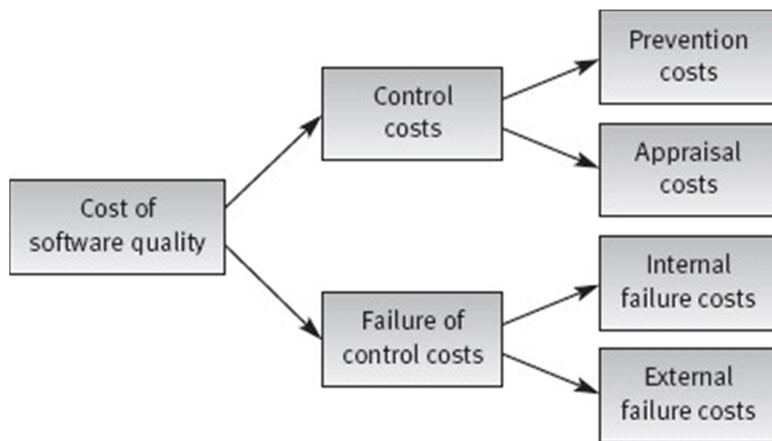
Mô hình chia nhỏ hơn nữa thành những lớp con. Chi phí quản lý được chia cho lớp con ngăn ngừa hoặc lớp con chi phí đánh giá:

- Chi phí ngăn ngừa bao gồm các khoản đầu tư vào cơ sở hạ tầng chất lượng và hoạt động chất lượng không nhằm vào một dự án hoặc hệ thống riêng biệt, là chung cho cả tổ chức.
- Chi phí đánh giá bao gồm chi phí các hoạt động được thực hiện cho những dự án cụ thể hoặc hệ thống phần mềm cho mục đích dò tìm lỗi phần mềm.

Chi phí thất bại của quản lý được chia nhỏ hơn thành chi phí thất bại bên trong và chi phí thất bại bên ngoài:

- Chi phí thất bại bên trong bao gồm chi phí sửa chữa những lỗi được tìm thấy khi xem xét lại thiết kế, kiểm tra phần mềm và kiểm tra chấp nhận (được thực hiện bởi khách hàng) và được hoàn thành trước khi phần mềm được cài đặt ở phía khách hàng.
- Chi phí thất bại bên ngoài bao gồm tất cả các chi phí sửa chữa những lỗi được tìm thấy bởi khách hàng hoặc đội bảo trì sau khi phần mềm đã được cài đặt.

Mô hình cổ điển của chi phí chất lượng phần mềm được biểu diễn ở hình 22.1. Mặc dù sự cố gắng áp dụng mô hình cổ điển vào việc phát triển và bảo trì phần mềm đã được ghi nhận là thành công nhưng vẫn không hoàn chỉnh. Nguyên nhân cho những khó khăn phải đối mặt sẽ được thảo luận phần sau của chương này. Nhưng trước khi tiếp tục, ta xem lại mô hình.



Mô hình tính chi phí truyền thống

Mô hình mở rộng tính giá chất lượng phần mềm

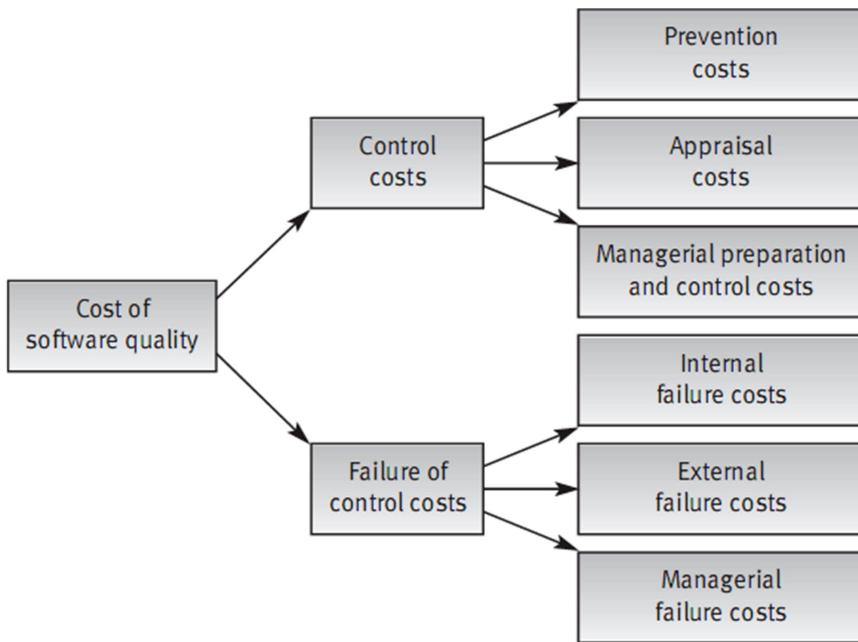
Các phân tích chi phí cho đảm bảo chất lượng phần mềm theo mô hình cổ điển cho thấy rằng có một số chi phí rất lớn đã bị bỏ qua. Những chi phí này hoặc là chỉ có trong công nghiệp phần mềm, hoặc không đáng kể trong các ngành công nghiệp khác. Ta xét ví dụ về hai chi phí tiêu biểu phải mất do các lỗi trong đảm bảo chất lượng phần mềm như sau:

- Chi phí đèn bù do hoàn thành dự án muộn, nguyên nhân xuất phát từ việc lập lịch biểu một cách không thực tế.

- Chi phí đèn bù vì hoàn thành dự án muộn do thất bại trong việc tuyển nhân viên mới có đủ khả năng.

Hai thất bại trên đây đều không phải do bất cứ một hoạt động cụ thể nào của nhóm phát triển hay do thiếu chuyên môn nghiệp vụ, mà là do việc quản lí gây ra. Do đó, những người quản lí dự án có thể thực hiện một vài hoạt động để ngăn ngừa hoặc chí ít là giảm thiểu chi phí mà các loại thất bại trên gây ra:

- Xem xét lại hợp đồng (xem xét lại các đề xuất sơ bộ và hợp đồng sơ bộ). Chi phí của các hoạt động này thường không đáng kể đối với các hợp đồng trong công nghiệp sản xuất. Tuy nhiên trong công nghiệp phần mềm, người ta cần một lượng công việc chuyên môn đáng kể để đảm bảo rằng các đề xuất của dự án được dựa trên các đánh giá cẩn thận và các ước lượng dễ hiểu (rõ ràng). Sự khác biệt về yêu cầu tài nguyên cho cùng công việc xem xét lại hợp đồng giữa công nghiệp phần mềm và công nghiệp sản xuất phát từ bản chất của sản phẩm và tiến trình sản xuất được đề cập đến trong hợp đồng. Trong khi hợp đồng trong công nghiệp sản xuất xử lý việc tạo ra nhiều lần các sản phẩm trong danh sách cố định, thì hợp đồng trong công nghiệp thường chỉ xử lí việc phát triển một hệ thống phần mềm mới, duy nhất.
- Điều khiển tiến độ dự án phần mềm một cách kĩ lưỡng và thích hợp. Trong khi việc điều khiển quá trình sản xuất trong công nghiệp sản xuất là một công việc lặp đi lặp lại và thường được thực hiện tự động bằng máy móc, thì việc điều khiển tiến độ phát triển phần mềm quản lí thiết kế tác vụ và các hoạt động coding được thực hiện bởi nhóm phát triển dự án.



Mô hình tính chi phí mở rộng

Các vấn đề trong áp dụng tính giá các độ đo chất lượng phần mềm

Ứng dụng của mô hình chi phí chất lượng phần mềm thường đi kèm với các vấn đề cần giải quyết ở bất cứ ngành công nghiệp nào. Những vấn đề này tác động lên độ chính xác và đầy đủ của dữ liệu chi phí chất lượng do:

- Sự thiếu chính xác và thiếu đầy đủ của việc xác định và phân loại các chi phí chất lượng.
- Các báo cáo cẩu thả của các thành viên nhóm và người ngoài nhóm.
- Việc báo cáo lêch lạc các chi phí phần mềm.
- Các bản ghi lêch lạc về các chi phí do thất bại bên ngoài, xuất phát gián tiếp từ các khoản đền bù được ngụy trang cho khách hàng (ví dụ: hạ giá các dịch vụ tương lai, cung cấp các dịch vụ miễn phí...). Chi phí cho các thất bại cần đền bù đó không được ghi nhận như là chi phí do thất bại bên ngoài.

Những vấn đề trên đây sinh trong ngũ cành công nghiệp phần mềm nhưng cũng có những vấn đề khác nữa. Một số trong chúng chỉ có trong sản xuất phần mềm mà thôi. Chúng ta sẽ tập trung vào các vấn đề gặp phải khi ghi nhận các chi phí chuẩn bị và điều khiển công việc quản lý cùng các chi phí do thất bại trong quản lý bởi vì những khoản mục này ảnh hưởng tới chi phí và tính dễ hiểu của tổng

chi phí cho đảm bảo chất lượng phần mềm, đặc biệt là khi áp dụng mô hình chi phí mở rộng.

Các vấn đề nảy sinh khi thu thập dữ liệu trong chuẩn bị và điều khiển việc quản lý chi phí bao gồm:

Xác định trách nhiệm khi có thất bại trong thực hiện lịch biểu. Những chi phí này có thể gán cho khách hàng (trong trường hợp khác hàng được yêu cầu bồi thường cho nhà thầu), nhóm phát triển (được xem như chi phí do thất bại bên ngoài) hay ban quản lý (được xem như chi phí cho thất bại trong quản lý). Chi phí cho thất bại lịch biểu thường được cân nhắc cho một giai đoạn đủ dài vì các nguyên nhân trực tiếp hay các đóng góp của từng thành phần trong dự án dẫn tới thất bại khó được xác định.

Chương 7 . Các chuẩn đảm bảo chất lượng

7.1 Các chuẩn quản lý chất lượng

7.1.1 Phạm vi của các chuẩn quản lý chất lượng

Các tiêu chuẩn chứng chỉ phần lớn dựa trên sự đánh giá nội dung hay mức độ quan trọng. Phạm vi của các tiêu chuẩn chứng chỉ dựa trên mục đích của các chứng chỉ .Đó là:

- Cho phép tổ chức phát triển phần mềm chứng minh năng lực phù hợp để đảm bảo sản phẩm phần mềm của mình hay các dịch vụ bảo trì tuân theo đúng yêu cầu chất lượng.
- Giống như sự giao ước cơ bản của khách hàng và nhà cung cấp nhằm đánh giá hệ thống quản lý chất lượng của nhà cung cấp , điều này có thể được hoàn thành bởi việc kiểm toán chất lượng của khách hàng .Việc kiểm toán này dựa trên các yêu cầu chứng chỉ tiêu chuẩn .
- Hỗ trợ tổ chức phát triển phần mềm để cải thiện hiệu năng của hệ thống quản lý chất lượng và nâng cao sự hài lòng của khách hàng thông qua việc tuân theo các tiêu chuẩn yêu cầu

Phạm vi của các tiêu chuẩn đánh giá cũng xác định trên mục đích của việc đánh giá, đó là:

- Các tổ chức phát triển và duy trì phần mềm sẽ sử dụng như 1 công cụ cho việc đánh giá bản thân về kỹ năng của họ trong đảm nhận các dự án phát triển phần mềm
- Như 1 công cụ để cải thiện quá trình phát triển và duy trì , các tiêu chuẩn có thể đưa ra các chỉ dẫn cho việc cải thiện tiến trình .
- Giúp các tổ chức chi trả hay đầu tư xác định được tiềm năng của nhà cung cấp
- Hướng dẫn tập luyện đánh giá bằng cách mô tả chất lượng và các khóa học lập trình.

7.1.2 ISO 9001 và ISO 9000-3

ISO 9000-3, được đưa ra bởi tổ chức ISO, trình bày cách cài đặt thực hiện phương pháp luận chung của tiêu chuẩn quản lý chất lượng ISO 9000 trong các trường hợp đặc biệt của phát triển và duy trì phần mềm . Cả ISO 9001 và ISO 9000-3 được xem xét lại và cập nhật liên tục khoảng 5 – 8 năm 1 lần , với những

các cuộc thảo luận riêng rẽ.

Giống như ISO 9000-3 dựa trên cơ sở những gì mà tiêu chuẩn ISO 9001 đưa ra , việc công bố các sửa chữa trong các nguyên tắc cũng dựa trên sự công bố các tiêu chuẩn đã được xem xét lại sau khoảng 1 vài năm . Chẳng hạn như , phiên bản 1997 của ISO 9000- 3 dựa trên phiên bản 1994 của ISO 1994 . Phiên bản 1997 của ISO 9000-3 kết hợp ISO 9001 cùng các đặc trưng của ISO 9000-3 thành tiêu chuẩn “tất cả trong một” cho ngành công nghiệp phần mềm.

8 nguyên lý hướng dẫn tiêu chuẩn ISO 9000 – 3 mới, có nguồn gốc từ tiêu chuẩn ISO9000:2000 như sau :

- Tập trung khách hàng : Tổ chức dựa trên khách hàng của họ và từ đó nêu hiếu được các yêu cầu hiện tại và trong tương lai của khách hàng
- Lãnh đạo : Những người lãnh đạo thiết lập tầm nhìn của tổ chức . Họ nên tạo ra và duy trì môi trường bên trong mà mọi người có thể phát triển nhằm đạt được mục tiêu của tổ chức qua các lộ trình định sẵn
- Sự phát triển của con người : Con người là cốt lõi của tổ chức . Tại mọi mức độ của tổ chức , sự phát triển của nhân lực cho phép kỹ năng của họ góp phần làm tăng lợi nhuận của tổ chức
- Tiếp cận tiến trình : Kết quả mong muốn giành được 1 cách hiệu quả hơn khi các hoạt động và tài nguyên được quản lý như 1 tiến trình
- Tiếp cận hệ thống để quản lý : Xác định, hiểu được và quản lý tiến trình , nếu nhìn như 1 hệ thống sẽ góp phần nâng cao hiệu năng tổ chức .
- Tiếp tục cải thiện : Liên tục cải thiện tất cả các hiệu suất nên được đề cao trong các cuộc họp của tổ chức
- Tiếp cận sự thực để ra quyết định : Các quyết định hiệu quả dựa trên việc phân tích các thông tin
- Mối quan hệ hỗ trợ lẫn nhau giữa những nhà cung cấp : 1 tổ chức và nhà cung cấp của nó là độc lập với nhau .Tuy nhiên mối quan hệ hỗ trợ lẫn nhau có thể nâng cao kỹ năng của cả 2 để tạo thêm ra giá trị gia tăng

Tiêu chuẩn ISO 9000-3 (ISO 1997) gồm 20 yêu cầu liên quan đến rất nhiều khía cạnh của hệ thống quản lý chất lượng phần mềm. Phiên bản mới hơn ISO 2001 đề nghị 1 cấu trúc mới với 22 yêu cầu được phân thành 5

nhóm cơ bản :

- Hệ thống quản lý chất lượng :
 - o Những yêu cầu chung
 - o Các tài liệu yêu cầu
- Trách nhiệm quản lý :
 - o Sự tận tâm trong quản lý
 - o Đối tượng khách hàng cần tập trung
 - o Điều khoản về chất lượng
 - o Lập kế hoạch
 - o Trách nhiệm, thẩm quyền và cách truyền đạt
 - o Xem xét lại về quản lý
- Quản lý tài nguyên :
 - o Tài nguyên cung cấp
 - o Nhân lực
 - o Các thiết bị cơ bản
 - o Môi trường làm việc
- Hiểu rõ sản phẩm :
 - o Hiểu rõ kế hoạch của sản phẩm
 - o Quá trình quan hệ với khách hàng có liên quan
 - o Thiết kế và phát triển
 - o Khả năng chi trả
 - o Dự liệu về sản phẩm và các dịch vụ
 - o Điều khiển thiết bị đo lường và kiểm tra
- Quản lý, phân tích và cải thiện :
 - o Kiểm tra và đo lường
 - o Điều khiển các sản phẩm không phù hợp

- Phân tích dữ liệu
- Cải thiện chất lượng

ISO 9001 – từ ứng dụng đến phần mềm : Sáng kiến TickIT

Được bắt đầu vào cuối thập kỷ 1980 , ngành công nghiệp phần mềm Vương quốc Anh cộng tác với Bộ công thương UK đã nâng phương pháp luận phát triển cho ISO 9001 thành các đặc điểm riêng cho công nghiệp phần mềm được biết đến như sáng kiến TickIT. Tại thời điểm nó bắt đầu, ISO 9001 đã được áp dụng thành công vào công nghiệp chế tạo tuy nhiên không có phương pháp luận nào có ý nghĩa cho các đặc điểm riêng biệt của công nghiệp phần mềm. Vài năm sau đó, TickIT cùng với sự nỗ lực nghiên cứu trong phát triển ISO 9000-3 đã đạt được thành quả này .

7.1.3 Các mô hình tăng trưởng khả năng – phương pháp đánh giá CMM và CMMI

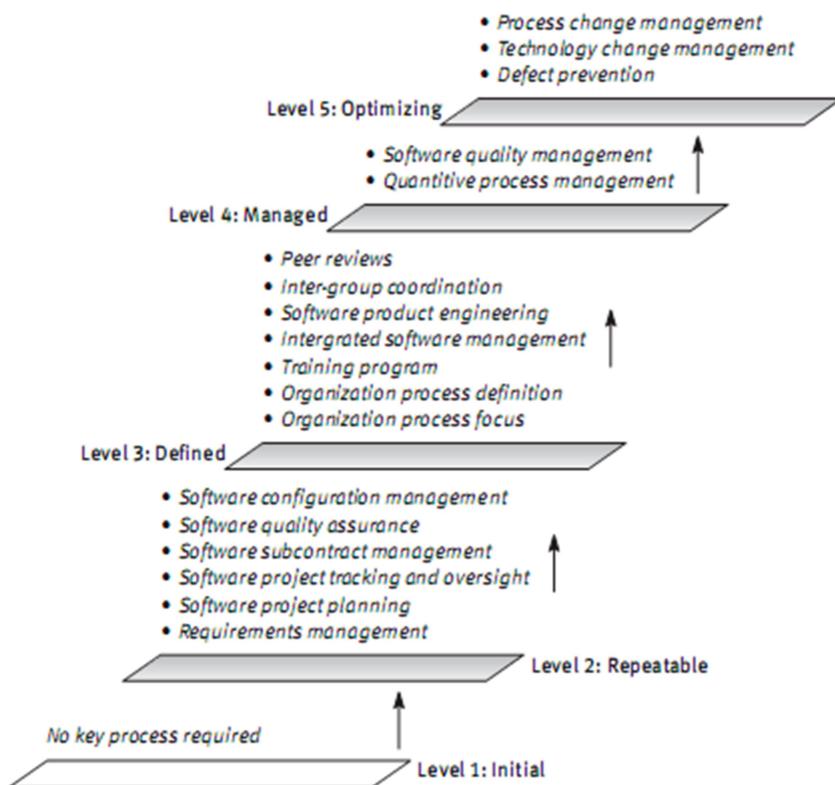
Đánh giá CMM dựa trên các khái niệm và nguyên lý sau:

- Ứng dụng của nhiều phương thức quản lý phức tạp dựa trên cách tiếp cận định lượng nhằm tăng khả năng của tổ chức để điều khiển chất lượng và cải thiện hiệu quả của quá trình phát triển phần mềm.
- Phương tiện để đánh giá việc phát triển phần mềm bao gồm 5 mức cầu mô hình đánh giá khả năng hoàn thành công việc đúng thời hạn.Mô hình cho phép một tổ chức đánh giá hiệu quả hoạt động của nó và xác định các yêu cầu cần thiết để có thể đạt tới mức tiếp theo bằng cách xác định các vùng xử lý được yêu cầu cho việc cải thiện.
- Vùng xử lý được dùng chung, chúng định nghĩa “cái gì” chứ không phải” bằng cách nào”. Đây là cách tiếp cận cho phép mô hình có thể được áp dụng cho một cho một miền rộng lớn của việc thực thi tổ chức,bởi vì:
 - Nó cho phép sử dụng một số mô hình vòng đời.
 - Nó cho phép sử dụng một số phương pháp luận,công cụ để phát triển phần mềm và ngôn ngữ lập trình.
 - Nó không chỉ rõ một tài liệu chuẩn cụ thể nào.

Mô hình CMMI cũng giống như mô hình gốc CMM bao gồm 5 mức.Các mức

của CMMI cũng giống như các mức trong bản gốc của nó ,chỉ có một sự thay đổi nhỏ ở mức 4,được đặt tên như sau:

- Capability maturity level 1: khởi tạo
- Capability maturity level 2: Lặp lại
- Capability maturity level 3: Xác định
- Capability maturity level 4: Quản lí
- Capability maturity level 5: Tối ưu hóa



Các mức của mô hình CMM và các miền tiến trình then chốt

7.2 Các chuẩn tiến trình dự án SQA

- Các chuẩn tiến trình dự án SQA tập trung vào phương pháp luận để thực hiện việc phát triển phần mềm và bảo trì phần mềm trong đó:
 - Mô tả mỗi bước của một quá trình
 - Yêu cầu kèm theo: tài liệu thiết kế, các nội dung thiết kế, xem xét lại thiết kế và các vấn đề xem xét lại, kiểm tra phần mềm và các mục

tiêu của nó...

- Các tổ chức phát triển chuẩn: IEEE, EIA, ISO....
 - Các chuẩn IEEE có thể được chia thành 3 lớp chính
 - o Các chuẩn khái niệm (conceptual standards)
 - IEEE 610.12, IEEE 1061, IEEE 1320.2, IEEE 1420.1a, IEEE/EIA 12207.0
 - o Các chuẩn yêu cầu cho sự tương thích (Prescriptive standards of conformance)
 - IEEE 828, IEEE 829, IEEE 1012, IEEE 1028, IEEE 1042.1
 - o Các chuẩn hướng dẫn (Guidance standards):
 - IEEE 1233, IEEE/EIA 12207.1, IEEE/EIA 12207.2

7.2.1 IEEE/EIA Std 12207- các tiến trình vòng đời phần mềm

Chuẩn IEEE/EIA Std 12207 cung cấp framework chung cho phát triển và quản lý phần mềm

- Tổ chức phát triển chuẩn:
 - o US Department of Defense (MIL-STD-498:1994)
 - o ANSI, IEEE và EIA (Joint Standard 016 (J-Std-016-1995))
 - o Tổ chức tiêu chuẩn quốc tế (ISO) và IEC – ISO/IEC 12207 Standard
- Chuẩn gồm 3 phần – three-part standard
 - o IEEE/EIA Std 12207.0-1996 (IEEE/EIA, 1996): bao gồm ISO/IEC 12207 gốc và các phần thêm vào
 - o IEEE/EIA Std 12207.1-1997 (IEEE/EIA, 1997a): Hướng dẫn – dữ liệu vòng đời
 - o IEEE/EIA Std 12207.2-1997 (IEEE/EIA, 1997b): Hướng dẫn – Xem xét cài đặt

Mục tiêu của IEEE/EIA Std 12207 là:

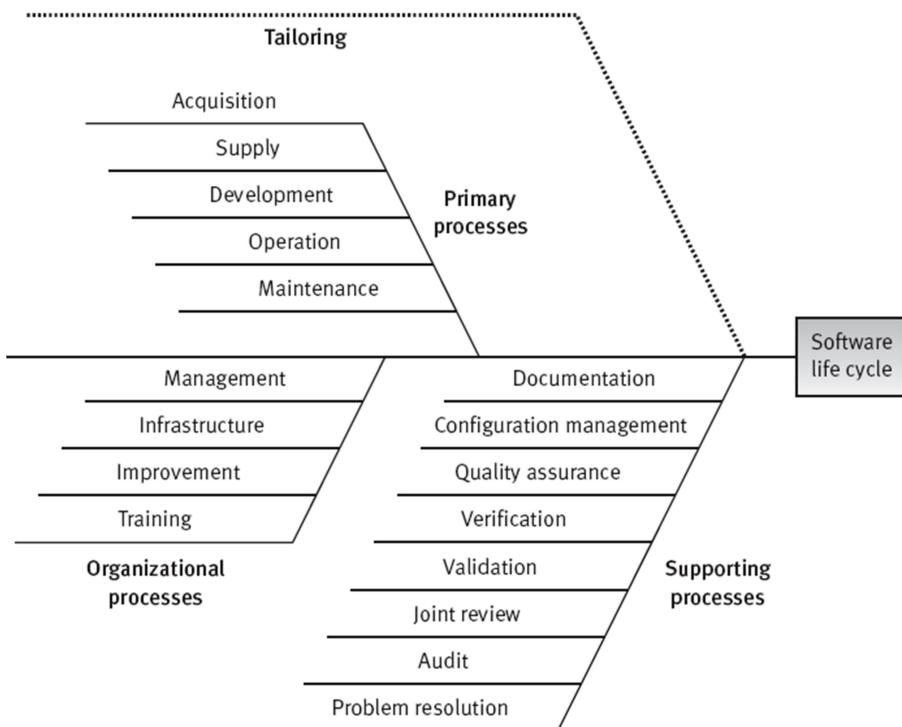
- Thiết lập mô hình các quá trình vòng đời phần mềm chung được

chấp nhận trên toàn cầu

- Đẩy mạnh sự am hiểu giữa các nhóm nghiệp vụ bằng việc áp dụng các quá trình, các hoạt động và các nhiệm vụ đã được chấp nhận chung

- **Kiến trúc vòng đời phần mềm IEEE/EIA 12207 bao gồm:**

- 4 mức
 - Process classes: Các lớp quy trình
 - Processes: Các tiến trình
 - Activities: Các hoạt động
 - Tasks: Các công việc
- 3 quy trình
 - Primary life cycle processes
 - Supporting life cycle processes
 - Organizational life cycle processes



Biểu đồ xương cá- Những tiến trình vòng đời phần mềm theo chuẩn IEEE/EIA

- **Các khái niệm chung – general concepts**

- Khả năng áp dụng chuẩn nói chung và sự thích ứng của nó bằng việc điều chỉnh cho một nhu cầu cụ thể
- Khả năng áp dụng cho tất cả các bên tham gia trong vòng đời phần mềm
- Tính mềm dẻo và tính đáp ứng với thay đổi công nghệ
- Phần mềm liên kết với hệ thống
- Tính nhất quán của quản lý chất lượng toàn phần mềm
- Không có các yêu cầu xác minh
- Baselineing

- **Các khái niệm liên quan tới công việc – task-related concepts**

- Gán trách nhiệm cho mỗi hoạt động và mỗi công việc
- Modun của các thành phần trong vòng đời phần mềm
- Các mức phù hợp được yêu cầu
- Bản chất của công việc ước lượng

7.2.2 IEEE Std 1012 – xác minh và thẩm định

Mục tiêu của chuẩn IEEE Std 1012 là:

- Thiết lập framework chung cho các hoạt động xác minh và thẩm định(V & V)
- Xác định đầu vào – đầu ra của việc V & V
- Xác định các mức toàn vẹn phần mềm, công việc V & V cho mỗi mức
- Xác định nội dung của tài liệu lập kế hoạch

V&V Các khái niệm cơ bản của IEEE 1012

- Định nghĩa rộng về các hoạt động V&V
 - Cho phép chuẩn bao quát tất cả các hoạt động V&V thực hiện xuyên suốt vòng đời phần mềm
- Tuân thủ và tương thích với các chuẩn quốc tế
- Các mức toàn vẹn của phần mềm và các yêu cầu V&V cho từng mức

Criticality	Description	Level
High	Selected function affects critical performance of the system.	4
Major	Selected function affects important system performance.	3
Moderate	Selected function affects system performance, but workaround strategies can be implemented to compensate for loss of performance.	2
Low	Selected function has noticeable effect on system performance but only creates inconvenience to the user if the function does not perform in accordance with requirements.	1

- Độc lập của các hoạt động V&V
- Tính độc lập về quản lý: nhóm V&V tự quyết định phương thức V&V nào được áp dụng.
- Tính độc lập về công nghệ: các thành viên trong nhóm V&V không liên quan đến việc phát triển phần mềm
- Tính độc lập về tài chính: quyền điều hành về ngân sách của V&V không được trao cho nhóm phát triển

Quy trình xác minh và thẩm định bao gồm:

- Quy trình quản lý
- Quy trình thu thập
- Quy trình cung cấp
- Quy trình phát triển
- Quy trình vận hành
- Quy trình bảo trì

7.2.3 IEEE Std 1028 – rà soát

- Mục tiêu của IEEE Std 1028 là

Định nghĩa các thủ tục của rà soát một cách hệ thống để:

- Thích hợp cho việc xem xét lại được thực hiện trong suốt vòng đời phần mềm
- Phù hợp với những yêu cầu việc xem xét lại được định nghĩa bởi

những tiêu chuẩn khác

Có 5 kiểu rà soát trong IEEE Std 1028:

☒ Xem xét lại quản lý.

☒ Xem xét lại kỹ thuật

☒ Kiểm tra.

☒ Walkthroughs

☒ Kiểm toán

Ba khái niệm cơ bản được đặc

trưng

- Hình thức cao - High formality
- Follow-up
- Sự tuân theo tiêu chuẩn quốc tế và IEEE Std 1028-.

1997 Phần chính của IEEE Std 1028-1997 đòi hỏi:

- Trình bày định nghĩa chi tiết của những yêu cầu của việc xem xét lại
- Phụ lục trình bày những mối quan hệ của tiêu chuẩn với những quá trình vòng đời

Thành phần của yêu cầu rà soát:

- Giới thiệu:
 - Những mục đích của mỗi kiểu xem xét lại.
 - Những ví dụ tiêu biểu của mỗi kiểu sản phẩm phần mềm
- Các trách nhiệm: Chuẩn cung cấp một danh sách những người tham gia, có thể là:
 - người tham gia ủy quyền (ví dụ: lãnh đạo, nhân viên kỹ thuật...)
 - hoặc những người tham gia tùy chọn khác (ví dụ: nhà quản lý, khách hàng...)
- Đầu vào dữ liệu được chia thành
 - những mục bắt buộc: tập trung vào mục đích của việc xem xét lại (vd: phát biểu mục tiêu, các tiêu chuẩn...)
 - tùy chọn (vd: những điều chỉnh, những bất thường)

- Tiêu chuẩn: Là những điều kiện ban đầu để cho phép và thực hiện việc xem xét lại, bao gồm:
 - Phát biểu tất cả các mục tiêu của việc xem xét lại
 - Tính sẵn sàng của dữ liệu đầu vào được yêu cầu
- Thủ tục gao gồm:
 - Chuẩn bị quản lý
 - Lập kế hoạch để xem xét lại
 - Sắp xếp thành viên trong nhóm
 - Kiểm tra sản phẩm phần mềm
 - Theo dõi các hoạt động chỉnh sửa
- Các tiêu chuẩn đã tồn tại xác định những việc phải được thực hiện trước khi việc xem xét lại kết thúc hoàn toàn, bao gồm:
 - Hoàn thành các hoạt động theo thủ tục
 - Theo dõi và phê chuẩn của các mục hành động đã được hoàn thành hoặc các hoạt động sửa chữa và ngăn chặn
 - Hoàn thành tài liệu của việc xem xét lại.
- Đầu ra: gồm:
 - Các tiêu chuẩn xác định các đầu ra cho mỗi loại xem xét lại
 - Các mục bổ sung có thể được yêu cầu bởi tổ chức, bởi các thủ tục bộ phận khác, hoặc trong các trường hợp cụ thể.
- Các khuyến nghị thu thập dữ liệu:

nhóm kiểm tra và walkthrough thu thập dữ liệu liên quan tới những bất thường đã gặp mỗi trường hợp được phân loại và xếp hạng theo mức độ nghiêm trọng.

dữ liệu này được sử dụng để

nghiên cứu tính hiệu quả của quá trình thực hiện hiện tại

chúng cũng được sử dụng để khuyến khích cải tiến các phương pháp và thủ tục.

- Cải tiến: dữ liệu kiểm tra và walkthrough sẽ được phân tích theo:
 - Hoàn thiện các thủ tục

- Cập nhật danh sách kiểm tra (checklists) được sử dụng bởi những người tham gia
- Cải thiện các quy trình phát triển phần mềm

Tài liệu tham khảo

- [1] Daniel Galin, *Software Quality Assurance – From Theory to Implementation*, Addison Wesley, 2004.
- [2] Ian Sommerville. *Software Engineering*, Sixth Edition, Addison Wesley, 2001.
- [3] Gerald D. Everett, Raymond McLeod, Jr., *Software Testing. Testing Across the Entire Software Development Life Cycle*, IEEE press, Wiley-interscience , A John Wiley&Sons, Inc, Publication, 2007.
- [4]. Glenford J. Myers. *The Art of Software Testing - Second Edition*. John Wiley & Sons, Inc, Publication, 2004.

Phụ lục

Một số lỗi thường mắc phải trong quá trình phát triển phần mềm, đặc biệt là pha cài đặt được trích từ cuốn sách “The art of software testing” cho ở dưới đây.

Lỗi tham chiếu - Data reference Error

Biến tham chiếu được gán giá trị trước khi được khởi tạo? Đây là lỗi thường xảy ra. Với các truy cập mảng, chỉ số có nằm trong biên cho phép hay không?

Với các truy cập mảng, chỉ số có nhận giá trị nguyên hay không?

Với các tham chiếu dùng con trỏ hoặc biến tham chiếu, ô nhớ tham chiếu đã được xác định chưa? Đây được gọi là “dangling reference”. Nó xảy ra khi thời gian tồn tại của con trỏ dài hơn thời gian tồn tại của ô nhớ tham chiếu. Trường hợp khác là khi con trỏ tham chiếu tới biến địa phương trong hàm, giá trị của con trỏ được gán cho một tham số bên ngoài, hoặc biến toàn cục, hàm kết thúc (giải phóng ô nhớ được tham chiếu), và sau đó chương trình dùng giá trị con trỏ.

Khi đặt nhiều tên (alias) cho một vùng nhớ với các thuộc tính khác nhau , liệu giá trị dữ liệu trong vùng nhớ này có đúng thuộc tính khi nó được tham chiếu theo một trong các tên? Ví dụ, một chương trình FORTRAN chứa một biến thực A và nguyên B; cả 2 biến đều là alias cho cùng một vùng nhớ sử dụng câu lệnh EQUIVALENCE. Nếu chương trình lưu một giá trị vào A và sau đó truy cập tới biến B; lỗi sẽ xuất hiện vì máy tính sẽ dùng biểu diễn số floating point trong bộ nhớ như một số nguyên.

Liệu giá trị của biến có kiểu hoặc thuộc tính khác với những gì trình biên dịch đặt ra? Tình huống này có thể xảy ra khi một chương trình C, C++ đọc một bản ghi vào bộ nhớ và truy cập tới nó thông qua cấu trúc, tuy nhiên biểu diễn vật lý của bản ghi khác với định nghĩa cấu trúc.

Vấn đề về địa chỉ? liệu đơn vị của vùng nhớ được cấp phát trên máy tính đang chạy có nhỏ hơn đơn vị của vùng nhớ được định nghĩa. Ví dụ, với một số môi trường, xâu bit có độ dài cố định không cần bắt đầu bằng byte biên, nhưng địa chỉ chỉ được chỉ tới byte biên. Nếu một chương trình tính địa chỉ của xâu bit và sau đó tham chiếu tới xâu thông qua địa chỉ này, ta có thể truy cập vào một vùng nhớ không chính xác. Tình huống này cũng có thể xảy ra khi truyền đối số xâu bit cho một chương trình con.

Nếu biến con trỏ hoặc biến tham chiếu được dùng, liệu vùng nhớ được tham chiếu có thuộc tính như trình biên dịch mong muốn? Một ví dụ về lỗi này là khi một con trỏ C++

trỏ tới địa chỉ của một cấu trúc dữ liệu khác với cấu trúc dữ liệu của nó.

Nếu một cấu trúc dữ liệu được tham chiếu ở nhiều hàm, thủ tục, liệu cấu trúc có được định nghĩa thống nhất trong các thủ tục không?

Khi đánh chỉ số cho một xâu/mảng, liệu có giới hạn chỉ số kết thúc xâu/mảng trong phạm vi cho phép không?

Với ngôn ngữ lập trình hướng đối tượng, liệu tất cả yêu cầu thừa kế có thỏa mãn cài đặt class?

Lỗi khai báo dữ liệu (Data-Declaration Errors)

Liệu các biến có được khai báo chính xác? Một failure trong trường hợp này có thể không phải là lỗi, nhưng nó cũng khởi nguồn cho nhiều vấn đề. Ví dụ, nếu một hàm nhận một tham số là mảng, và không định nghĩa tham số như một mảng thành công, một tham chiếu tới mảng (ví dụ C=A(I) được dịch thành một lời gọi hàm, dẫn tới máy tính cố gắng thực hiện mảng như một chương trình. Thêm nữa, nếu một biến không được định nghĩa một cách rõ ràng trong thủ tục, hoặc khỏi lệnh, liệu ta có thể hiểu là biến đó sẽ được dùng chung hay không?)

Nếu tất cả các thuộc tính của một biến không được quy định rõ ràng lúc định nghĩa, liệu giá trị mặc định có được hiểu rõ? Ví dụ, thuộc tính mặc định nhận được trong Java thường là một nguồn ngạc nhiên lớn. Khi một biến được khởi tạo bằng câu lệnh khai báo, liệu nó có được khởi tạo đúng? Trong nhiều ngôn ngữ, khởi tạo mảng và xâu đôi khi khá phức tạp và có xu hướng gây lỗi.

Liệu mỗi biến có được gán với đúng độ dài và kiểu?

Liệu khởi tạo của biến có đồng nhất với kiểu memory? Ví dụ, một biến trong hàm FORTRAN cần được khởi tạo lại mỗi lần gọi hàm, nó phải được khửoi tạo bằng một câu lệnh gán hơn là một câu lệnh DATA

Liệu có tên biến tương tự nhau (ví dụ VOLT và VOLTS)? Đây không nhất thiết là lỗi, nhưng ta nên đặt tên khác nhau phòng khi nhầm lẫn.

Lỗi tính toán (Computation Errors)

Liệu có tính toán nào sử dụng biến với kiểu dữ liệu không phù hợp?

Liệu có tính toán mixed-mode (kết hợp kiểu) nào không? Ví dụ cộng biến floating point cho biến integer. Cần hiểu rõ các luật chuyển đổi kiểu để tránh sai sót. Ví dụ, đoạn mã Java sau sẽ có lỗi rouding (làm tròn) khi thực hiện với số

integers.

```
int  
x=1;  
int  
y=2;  
int  
z=0;  
z=x/y;  
  
System.out.println  
("z="+z); OUTPUT: z = 0
```

Liệu có tính toán nào sử dụng biến có cùng kiểu nhưng độ dài khác nhau?

Liệu kiểu của biến đích trong phép gán nhỏ hơn kiểu của kết quả của biểu thức ở vé phải?

Liệu có xảy ra lỗi overflow hoặc underflow (tràn) khi tính toán biểu thức? Chẳng hạn, kết quả là giá trị hợp lệ, nhưng kết quả trung gian có thể quá lớn hoặc quá nhỏ so với kiểu cho phép của ngôn ngữ lập trình.

Liệu có được phép thực hiện chia cho 0?

Nếu máy biểu diễn biến bằng số nhị phân, liệu có chuỗi tính toán nào cho kết quả sai hay không? Chẳng hạn, 10×0.1 không bằng 1.0 khi thực hiện trên số nhị phân.

Khi đưa vào ứng dụng, liệu giá trị của biến có ra ngoài khoảng có nghĩa.

Với biểu thức chứa nhiều hơn 1 phép toán, liệu giả thiết về thứ tự ưu tiên các phép toán có chính xác?

Có phép toán không hợp lệ nào liên quan tới số integer không, đặc biệt là phép chia? Ví dụ, nếu i là biến integer, biểu thức $2*i/2 == i$ sẽ phụ thuộc vào i là số chẵn hay lẻ và phép nhân hay chia được thực hiện trước?

Lỗi so sánh (Comparison Errors)

Có phép so sánh nào giữa các biến với kiểu khác nhau không, ví dụ so sánh xâu kí tự với địa chỉ, ngày, số?

Liệu có so sánh mixed-mode nào không hoặc là so sánh giữa các biến với độ dài khác nhau? Nếu có, cần đảm bảo là luật biến đổi phải được hiểu chính xác.

Các phép so sánh có chính xác? Lập trình viên thường bị nhầm các quan hệ như

at most, at least, greater than, not less than, less than or equal.

Liệu các biểu thức Boolean có giá trị đúng như mong đợi? Lập trình viên thường mắc lỗi khi viết các biểu thức logic chứa *and, or, not*.

Liệu các toán hạng của phép toán Boolean có là giá trị Boolean? Phép so sánh và phép toán Boolean thường được dùng xen kẽ. Hành động này dễ dẫn tới lỗi. Ví dụ, ta cần xác định i có nằm trong khoảng từ 2 tới 10 hay không, biểu thức $2 < i < 10$ là sai; biểu thức đúng là $(2 < i) \&\& (i < 10)$. Nếu ta muốn xác định xem i có lớn hơn x hoặc y, $i > x \parallel y$ là sai, biểu thức đúng phải là $(i > x) \parallel (i > y)$. Nếu ta muốn so sánh 3 số có bằng nhau hay không, biểu thức $(a == b == c)$ cho kết quả khác với mong đợi. Nếu muốn kiểm tra quan hệ $x > y > z$, biểu thức đúng phải là $(x > y) \&\& (y > z)$.

Liệu có phép so sánh nào giữa số phẩy tĩnh và số phẩy động (floating point) ở hệ cơ số 2?

Đây có thể là một nguồn lỗi vì lỗi làm tròn và biểu diễn xấp xỉ cơ số 2 của số hệ cơ số 10.

Với biểu thức chứa nhiều hơn 1 phép toán boolean, quy ước về thứ tự thực hiện các phép toán có chính xác? Ví dụ, với biểu thức $(if((a == 2) \&\& (b == 2) \parallel (c == 3))$, phép and hay or sẽ được thực hiện trước cần được quy định rõ ràng.

Liệu cách thức trình biên dịch tính toán giá trị của biểu thức Boolean có ảnh hưởng tới chương trình? Ví dụ, câu lệnh :

`if((x==0 \&\& (x/y)>z)`

có thể được chấp nhận với trình biên dịch kết thúc kiểm tra điều kiện ngay khi một vế của and là false, nhưng có thể dẫn tới lỗi division-by-zero (chia cho 0) với trình biên dịch khác.

Lỗi luồng điều khiển (Control-Flow Errors)

Nếu chương trình chứa nhánh nhiều hướng, chẳng hạn phép tính GO TO, liệu biến chỉ số có ra ngoài phạm vi của nhánh? Ví dụ câu lệnh

`GO TO (200, 300, 400), i`

liệu i có nằm trong số giá trị 1, 2, 3?

Liệu vòng lặp có kết thúc? Chứng minh các vòng lặp sẽ kết thúc Chương trình, module, hàm con có kết thúc?

1. Với một vòng lặp được điều khiển bởi cả biểu thức điều kiện (ví dụ vòng lặp tìm kiếm), và iteration, what are the consequences of loop fall-through? Ví dụ, với giả mã:

DO I=1 to TABLESIZE WHILE (NOTFOUND)

chuyện gì xảy ra nếu NOTFOUND không bao giờ bằng false?

Lỗi giao diện (Interface Errors)

Liệu số tham số của module có bằng với số đối số trong các lời gọi hàm. Liệu thứ tự đối số có chính xác?

Liệu thuộc tính (chẳng hạn kiểu dữ liệu, kích thước) của từng tham số có tương thích với thuộc tính của đối số tương ứng?

Liệu đơn vị của từng tham số có tương thích với đơn vị của đối số tương ứng? Ví dụ, nếu tham số dùng đơn vị degrees, nhưng đối số lại dùng đơn vị radians?

Liệu số đối số truyền bởi module này cho module khác có bằng số tham số expected by that module?

Nếu hàm built-in được gọi, liệu số, thuộc tính và thứ tự của đối số có chính xác?

Lỗi Input/Output

Nếu files đã được định nghĩa, liệu các thuộc tính của nó có chính xác? Liệu các thuộc tính của câu lệnh OPEN file có chính xác?

Liệu định dạng file có khớp với các câu lệnh I/O? Có đủ bộ nhớ để chứa file mà chương trình sẽ đọc? Các files đã được mở trước khi dùng?

Các files đã được đóng sau khi dùng?

Điều kiện end-of-file có được phát hiện và quản lý chính xác? Các điều kiện lỗi I/O đã được quản lý chính xác?

Các xuất hiện lỗi chính tả, cú pháp ở các đoạn văn bản có được in/hiển thị ?

Các kiểm tra khác

Nếu trình biên dịch đưa ra một danh sách thuộc tính, kiểm tra các thuộc tính của từng biến để đảm bảo là không có thuộc tính mặc định không mong muốn nào đã

được gán giá trị.

Nếu chương trình được dịch thành công, nhưng máy tính đưa ra cảnh báo (“warning”), ta cần kiểm tra chúng cẩn thận. Cảnh báo ám chỉ là trình biên dịch phát hiện ra bạn đang làm việc gì đó gây nghi vấn. Các nghi vấn này cần được kiểm tra lại.

Liệu chương trình hay module có kiểm tra validity cho đầu vào.