CHƯƠNG 5: QUẢN LÝ KIỂM THỬ

MỤC TIÊU THỰC HIỆN:

Sau khi học xong chương này, sinh viên có khả năng:

- Trình bày được quy trình tổng quát cũng như các công việc trong quá trình quản lý kiểm thử.
- Tự tổ chức và quản lý được hoạt động kiểm thử một dự án nhỏ theo quy định.
- Viết báo cáo và trình bày kết quả kiểm thử đã thực hiện được của nhóm theo hướng dẫn.
- Tham gia một cách chủ động và tích cực vào các công việc được giao.

5.1 Tổ chức kiểm thử

5.1.1 Tổ chức kiểm thử độc lập

Hiệu quả của việc tìm lỗi có thể được cải thiện nâng cao nhờ có những Tester độc lập.

Có thể có những loại hình kiểm thử độc lập như sau:

- Không có các đội ngũ kiểm thử độc lập. Các lập trình viên tự kiểm thử Code của chính họ. Các lập trình viên vừa lập trình vừa viết Unit Test để kiểm thử những chức năng mình viết.
- Tester độc lập bên trong nhóm phát triển phần mềm. Trong một dự án phần mềm sẽ có các lập trình viên và có một hoặc vài nhân viên kiểm thử theo xuyên xuốt dự án để kiểm thử dự án phần mềm đó. Loại hình này có thuận lợi là nhân viên kiểm thử và lập trình viên trao đổi công việc với nhau thuận lợi, hiểu rõ yêu cầu phần mềm, hạn chế việc hiểu sai yêu cầu vì kiểm thử và lập trình viên nằm chung nhóm phát triển nên việc truyền thông cũng hiệu quả hơn. Tuy nhiên, vấn đề khách quan trong kiểm thử cũng cần phải được quan tâm trong trường hợp này.
- Nhóm kiểm thử độc lập trong tổ chức hoặc công ty. Trong trường hợp này, bên trong tổ chức hoặc công ty sẽ có một đội kiểm thử riêng làm nhiệm vụ kiểm thử cho các dự án và nghiên cứu phát triển kiểm thử để mang lại hiệu quả kiểm thử cao nhất. Khi có dự án phần mềm, trưởng dự án có thể nhờ đội kiểm thử hỗ trợ kiểm thử dự án, khi đó trưởng nhóm kiểm thử có thể cử nhân viên qua hỗ trợ, một nhân viên kiểm thử có thể kiểm thử một hoặc nhiều dự án một lúc. Loại hình này hiện nay được các tổ chức và công ty áp dụng nhiều vì tính chuyên nghiệp và khách quan của nó.

- Những đội ngũ kiểm thử độc lập được thuê từ bên ngoài công ty. Có những công ty chuyên làm nhiệm vụ kiểm thử thuê cho các công ty khác hoặc cung cấp giải pháp kiểm thử phần mềm cho các công ty khác. Loại hình này đem lại tính chuyên nghiệp và khách quan cao, tuy nhiên vấn đề cần quan tâm là việc truyền thông giữa đội ngũ phát triển phần mềm trong công ty và đội ngũ kiểm thử bên ngoài cần phải như thế nào để các đặc tả yêu cầu phần mềm được hiểu rõ và thống nhất cao.

Ngoài ra còn có những tùy chọn kiểm thử độc lập khác như:

- Các đội ngũ kiểm thử từ công ty của khách hàng hoặc từ cộng đồng người sử dụng phần mềm.
- Các chuyên gia hoặc các tổ chức kiểm thử độc lập như các tổ chức kiểm thử bảo mật, kiểm thử tính khả dụng của phần mềm, hoặc các tổ chức kiểm tra chứng nhận.

Lợi ích của kiểm thử độc lập bao gồm:

- Các Tester độc lập có thể nhìn thấy được nhiều loại lỗi khác nhau ở mọi ngóc ngách của phần mềm một cách khách quan và không thiên vị.
- Một Tester độc lập có thể xác minh các giả định mà mọi người đưa ra trong quá trình đặc tả và triển khai hệ thống. Các Tester có thể đóng vai trò là người dùng trong suốt quá trình kiểm thử phần mềm để hiểu người dùng mong muốn gì từ đó sẽ có hướng kiểm thử phù hợp hơn, đáp ứng đầy đủ các yêu cầu của người dùng hơn.

Ngoài những lợi ích khi có đội ngũ kiểm thử độc lập, chúng ta cần phải quan tâm đến những vấn đề sau:

- Một vấn đề vừa là điểm mạnh và cũng là khó khăn đối với đội ngũ kiểm thử độc lập đó là sự cô lập với nhóm phát triển phần mềm. Các tester độc lập có ưu điểm là kiểm thử khách quan, không thiên vị, tuy nhiên một vấn đề gặp phải là vấn đề truyền thông giữa Tester độc lập và các lập trình viên, đội ngũ thiết kế,..để hiểu rõ yêu cầu phần mềm cũng như những thay đổi yêu cầu từ phía khách hàng, nếu truyền thông không tốt có thể dẫn đến hiểu sai yêu cầu, yêu cầu chưa kịp cập nhật,... Do đó cần có các buổi review và các buổi daily scrum thường xuyên để cập nhật tiến độ cũng như những thay đổi yêu cầu từ phía khách hàng.
- Các tester độc lập có thể là nút cổ chai như là điểm kiểm tra cuối cùng. Dự án trước khi bàn giao cho khách hàng thì phải qua đội ngũ kiểm thử để đảm bảo chất

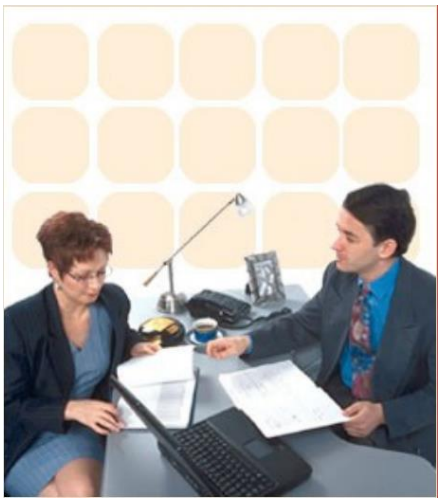
lượng phần mềm, tuy nhiên nếu các developer lập trình không tốt dẫn đến phần mềm kém chất lượng, có nhiều lỗi sẽ dẫn đến quá trình kiểm thử và sửa lỗi sẽ tốn nhiều thời gian, dẫn đến khả năng chậm trễ bàn giao cho khách hàng.

- Các developer có thể mất ý thức trách nhiệm về chất lượng. Vì nghĩ rằng đã có đội ngũ kiểm thử phần mềm, các developer có thể sa vào tình trạng code nhanh, code ẩu mà không unit test kỹ càng các hàm, các chức năng dẫn đến khi chuyển phần mềm sang đội kiểm thử thì phát hiện ra nhiều lỗi, dẫn đến chậm tiến độ bàn giao cho khách hàng. Do đó, tư duy chất lượng phải có ở mỗi người trong đội phát triển phần mềm.

5.1.2 Nhiệm vụ của Test Leader và Tester

Nhiệm vụ của Test leader:

- Lập kế hoạch, theo dõi và kiểm soát tiến độ kiểm thử.
- Đưa ra các mục tiêu kiểm thử, chính sách kiểm thử, chiến lược kiểm thử và kế hoạch kiểm thử.
- Ước lượng kiểm thử và tài nguyên dành cho việc kiểm thử.
- Lập kế hoạch kiểm thử tự động nếu cần thiết.
- Lập lịch, hướng dẫn, theo dõi, đo lường và đánh giá chất lượng kiểm thử.
- Đảm bảo việc quản lý cấu hình.
- Báo cáo tiến độ kiểm thử, báo cáo tóm tắt kết quả của quá trình kiểm thử.
- Đưa ra hành động để khắc phục các vấn đề phát sinh trong quá trình kiểm thử.

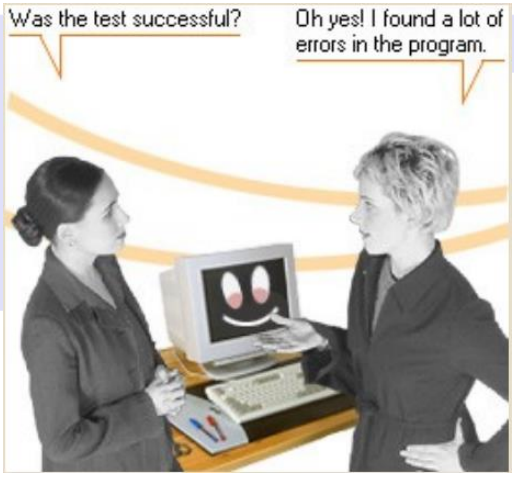


Nhiệm vụ của Tester:

- Xem xét và đóng góp cho kế hoạch kiểm thử.
- Đọc hiểu, phân tích, nhận xét bản đặc tả yêu cầu phần mềm (SRS), bản đặc tả thiết kế phần mềm, đặt câu hỏi để làm rõ yêu cầu của khách hàng, đóng góp xây

dựng phần mềm dựa trên những hiểu biết của bản thân qua những phần mềm đã kiểm thử trước đây.

- Viết Test Cases cho các chức năng mình phụ trách.
- Thiết lập môi trường dùng cho việc kiểm thử, ví dụ: thiết lập phiên bản Windows, thiết lập phiên bản Ios, phiên bản Android, phiên bản trình duyệt Web,... giống như môi trường sử dụng của khách hàng để kiểm thử được chính xác theo yêu cầu.
- Thực thi chạy Test Cases để kiểm thử phần mềm và ghi lại các lỗi tìm được, đánh giá kết quả và ghi nhận lại các trường hợp gặp phải trong quá trình kiểm thử để cải tiến, bổ sung Test Cases hoặc cải tiến hướng kiểm thử để tốt hơn trong những dự án tương lai.
- Thực hiện kiểm thử tự động nếu cần.
- Nhận xét các Test Cases được phát triển bởi những người khác.



5.2 Lập kế hoạch và ước lượng kiểm thử

5.2.1 Lập kế hoạch kiểm thử

- Lập kế hoạch là hoạt động cần thiết để dự án đạt được thành công.
- Một kế hoạch mô tả một phương pháp quản lý, nếu được tuân theo, sẽ đảm bảo rằng dự án đạt được các mục tiêu của nó.

- Các hoạt động quản lý dự án nhằm mục đích đảm bảo rằng dự án đáp ứng được những điều sau đây:
  - Dự án được hoàn thành trong thời gian dự kiến.
  - Dự án được hoàn thành trong chi phí cho phép.
  - Dự án tạo ra được một sản phẩm chất lượng tốt, đáp ứng yêu cầu của khách hàng.

Hình 5.1: Các yêu cầu đối với một dự án thành công

- Test Plan là một tài liệu mô tả phạm vi kiểm thử, cách tiếp cận kiểm thử, những tài nguyên và lịch trình của những hoạt động kiểm thử dự định.
- Quản lý dự án bao gồm các hoạt động lập kế hoạch, giám sát và kiểm soát dự án để đảm bảo dự án thành công.

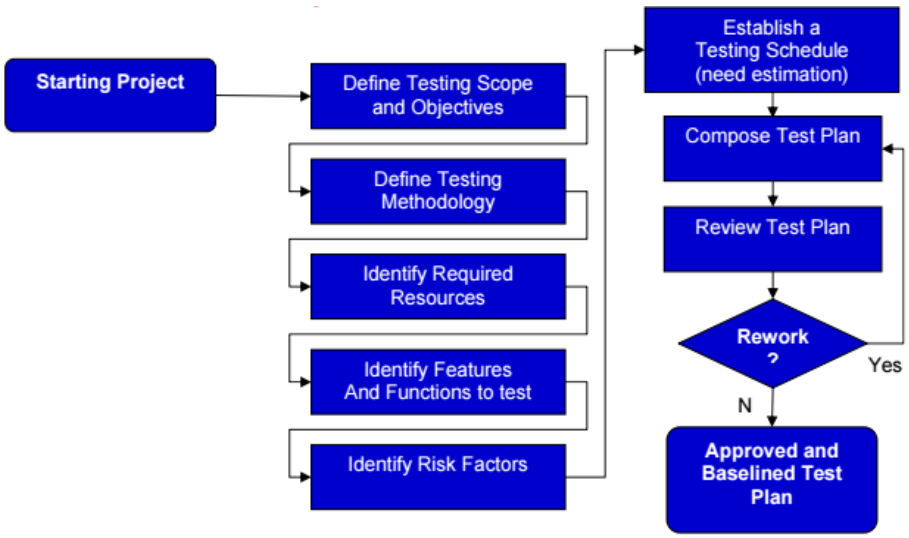
5.2.2 Những hoạt động lập kế hoạch kiểm thử

Kế hoạch kiểm thử thường do Test Manager hoặc Test Leader xây dựng và thường bao gồm những hoạt động sau:

- Xác định các chiến lược được dùng để kiểm thử và đảm bảo rằng sản phẩm thỏa mãn đặc tả thiết kế phần mềm và các yêu cầu khác về phần mềm.
- Xác định các mục tiêu và phạm vi kiểm thử.
- Dự kiến các phương pháp mà đội kiểm thử sẽ dùng để thực hiện công việc kiểm thử.
- Dự kiến những phần cứng, phần mềm và những tiện ích cần cho việc kiểm thử
- Xác định các chức năng sẽ được kiểm thử.
- Xác định và dự kiến những rủi ro có thể gây nguy hại cho việc kiểm thử.
- Lập lịch kiểm thử và phân phối công việc cho mỗi thành viên tham gia.

Kế hoạch kiểm thử cần phải được xây dựng sớm trong mỗi chu kì phát triển phần mềm để có thể:

- Tập hợp và tổ chức các thông tin kiểm thử cần thiết.
- Cung cấp thông tin về quy trình kiểm thử sẽ xảy ra trong quá trình phát triển phần mềm.
- Giúp cho mỗi thành viên trong nhóm kiểm thử có hướng đi đúng.
- Gán các trách nhiệm rõ ràng cụ thể cho mỗi thành viên trong nhóm kiểm thử.
- Có lịch biểu làm việc rõ ràng từ đó các thành viên có thể làm việc và phối hợp với nhau tốt nhất.



Hình 5.2: Quy trình xây dựng kế hoạch kiểm thử

5.2.3 Tiêu chí đầu vào

Tiêu chí đầu vào là một tập hợp các điều kiện hoặc yêu cầu, được yêu cầu phải được đáp ứng hoặc đạt được để tạo ra một điều kiện phù hợp và thuận lợi để kiểm thử. Được

hoàn thành và quyết định sau khi phân tích kỹ lưỡng các yêu cầu nghiệp vụ và phần mềm, tiêu chí đầu vào đảm bảo tính chính xác của quy trình kiểm thử. Bỏ qua tiêu chí đầu vào có thể ảnh hưởng đến chất lượng của cả quy trình. Một số tiêu chí đầu vào thường được sử dụng để đánh dấu sự bắt đầu kiểm thử như sau:

- Code hoàn chỉnh hoặc code cho một phần chức năng hoàn chỉnh.
- Requirements được xác định và phê duyệt.
- Có đủ test data.
- Test cases đã được viết và sẵn sàng triển khai.
- Môi trường kiểm thử đã được thiết lập và tất cả các tài nguyên cần thiết khác như công cụ và thiết bị đều khả dụng. Cả hai giai đoạn phát triển và kiểm thử đều được sử dụng làm nguồn để xác định tiêu chí đầu vào cho quy trình kiểm thử phần mềm, ví dụ như sau:

- Giai đoạn / quy trình phát triển cung cấp thông tin hữu ích liên quan đến phần mềm, thiết kế, chức năng, cấu trúc và các tính năng có liên quan khác, hỗ trợ trong việc quyết định các tiêu chí đầu vào chính xác như yêu cầu chức năng và kỹ thuật, thiết kế hệ thống, v.v.
- Từ giai đoạn kiểm thử, các đầu vào sau được xem xét: Test Plan; Test Strategy; Test Data và Test Tools; Môi trường kiểm thử (Test Environment).

Các tiêu chí đầu vào chủ yếu được xác định cho bốn cấp độ kiểm thử cụ thể: Unit Testing, Integration Testing, System Testing, Acceptance Testing. Mỗi cấp độ kiểm thử này yêu cầu các tiêu chí đầu vào riêng biệt để xác định mục tiêu của chiến lược kiểm thử và để đảm bảo đáp ứng các yêu cầu sản phẩm:

Tiêu chí đầu vào của giai đoạn Unit Testing

- Giai đoạn lập kế hoạch đã hoàn thành.
- Thiết kế hệ thống, thiết kế kỹ thuật và các tài liệu liên quan khác được xem xét, phân tích và phê duyệt hợp lý.
- Yêu cầu nghiệp vụ và chức năng được xác định và phê duyệt.
- Testable Code hoặc Units có sẵn.
- Môi trường kiểm thử đã sẵn sàng.

Tiêu chí đầu vào của giai đoạn Integration Testing

- Hoàn thành Unit Testing .
- Priority bugs được tìm thấy trong Unit Testing đã được sửa và đóng.

- Kế hoạch tích hợp và môi trường kiểm thử để thực hiện kiểm thử tích hợp đã sẵn sàng.
- Mỗi mô-đun đã trải qua kiểm thử đơn vị trước quá trình tích hợp.

Tiêu chí đầu vào của giai đoạn System Testing

- Hoàn thành quá trình kiểm thử tích hợp.
- Priority bugs được tìm thấy trong các hoạt động kiểm thử trước đó đã được sửa và đóng.
- Có sẵn môi trường kiểm thử hệ thống.
- Test Cases có sẵn để thực hiện.

Tiêu chí đầu vào của giai đoạn Acceptance Testing

- Hoàn thành giai đoạn kiểm thử hệ thống.
- Priority bugs được tìm thấy trong các hoạt động kiểm tra trước đó đã được sửa và đóng.
- Yêu cầu chức năng và nghiệp vụ đã được đáp ứng.
- Môi trường kiểm thử chấp nhận đã sẵn sàng.

Xác định tiêu chí đầu vào cho quá trình kiểm thử là một bước cần thiết, nó giúp Tester hoàn thành các nhiệm vụ kiểm thử trong thời hạn quy định mà không ảnh hưởng đến chất lượng, chức năng và hiệu quả của phần mềm.

5.2.4 Tiêu chí xuất (Exit criteria)

Mục đích của tiêu chí xuất là để xác định khi nào thì dừng việc kiểm thử và bàn giao sản phẩm cho khách hàng.

Thông thường tiêu chí xuất có thể dựa vào những yếu tố sau:

- Đo lường kỹ lưỡng, như độ bao phủ code, độ bao phủ chức năng hoặc rủi ro. Điều này có nghĩa là để xác định khi nào có thể dừng việc kiểm thử lại để bàn giao sản phẩm cho khách hàng, chúng ta có thể dựa vào các thông số như: độ bao phủ câu lệnh, độ bao phủ quyết định, độ bao phủ chức năng, số lượng các tình huống quan trọng đã được kiểm thử,...độ bao phủ càng cao thì chứng tỏ nỗ lực kiểm thử càng nhiều, hạn chế được nhiều rủi ro của phần mềm.
- Dựa vào việc ước lượng mật độ lỗi hoặc đo lường độ tin cậy của phần mềm.
- Dựa vào chi phí thực hiện của dự án.
- Dựa vào những rủi ro còn tồn tại, ví dụ như những lỗi chưa được sửa hoặc thiếu kiểm thử bao phủ trong những khu vực nhất định. Thay vì nhìn vào số lượng những nỗ lực kiểm thử đã thực hiện, ta có thể nhìn ở khía cạnh khác để quyết

định khi nào ngừng kiểm thử, ví dụ nếu các test cases cho phần mềm đã được chạy hết, thay vì kết luận phần mềm đã tốt có thể bàn giao thì ta có thể đi xác định những rủi ro còn tồn tại, những trường hợp mới mà các test cases chưa bao quát tới, những chức năng quan trọng đã test kỹ hay chưa (vì các chức năng chính của phần mềm chiếm 20% thường gây ra tới 80% lỗi) từ đó mới quyết định có dừng việc kiểm thử lại và bàn giao phần mềm hay không. Kiểm thử dựa trên kinh nghiệm (đoán lỗi, kiểm thử khám phá) rất hữu ích trong trường hợp này.

- Dựa vào lịch trình marketing quy định của công ty, ví dụ: thời gian nào phải tung ra thị trường thì đạt hiệu quả bán sản phẩm cao nhất, từ đó sẽ quyết định khi nào thì dừng việc kiểm thử lại để hoàn chỉnh các thủ tục trước khi đưa ra thị trường.

5.2.5 Ước lượng kiểm thử

Có hai hướng tiếp cận để ước lượng nỗ lực kiểm thử được đề cập trong giáo trình này:

- Cách tiếp cận dựa trên số liệu: Ước tính nỗ lực kiểm thử dựa trên số liệu của các dự án trước đây hoặc dựa trên số liệu của các dự án tương tự, hay dựa trên các giá trị điển hình.
- Cách tiếp cận dựa trên chuyên gia: chúng ta có thể ước lượng các nỗ lực kiểm thử (thời gian, chi phí,..) dựa trên quy định của chủ dự án (thời gian, chi phí,..) dành cho mỗi tác vụ hoặc dựa trên kinh nghiệm được tham khảo từ các chuyên gia trong cùng lĩnh vực.

Nỗ lực kiểm thử có thể phụ thuộc vào một số yếu tố như sau:

- Đặc điểm của sản phẩm.
- Đặc điểm của quy trình.
- Kết quả của kiểm thử: số lượng lỗi tìm được và số lượng làm lại được yêu cầu.

Hình 5.3: Ước lượng kiểm thử.

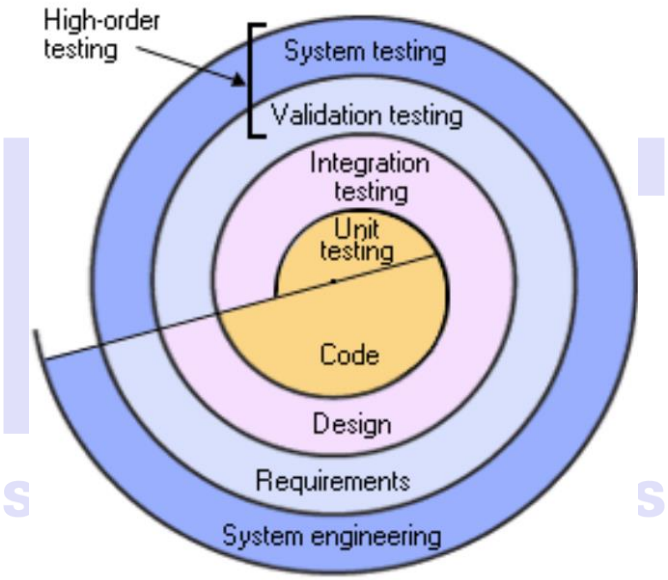
5.2.6 Hướng tiếp cận kiểm thử (chiến lược kiểm thử).

Một cách để phân loại các hướng tiếp cận kiểm thử hay chiến lược kiểm thử là dựa trên thời điểm mà phần lớn công việc thiết kế kiểm thử được bắt đầu:

- ✓ **Hướng tiếp cận phòng ngừa:** Theo cách tiếp cận này, các bộ test được thiết kế càng sớm càng tốt. Khi tiến hành dự án phần mềm, ngay từ giai đoạn làm rõ đặc tả yêu cầu của phần mềm (SRS) là đã có sự tham gia của các tester, họ sẽ dựa trên SRS để tiến hành phân tích và thiết kế các Test Cases trước khi phần mềm được thực hiện. Trong quá trình này, với kinh nghiệm đã trải qua khi kiểm thử các phần mềm trước đây, các tester có thể đặt câu hỏi để làm rõ vấn đề, bổ sung, góp ý để sản phẩm phần mềm được xây dựng theo đúng mong đợi của khách hàng hoặc thị trường, đồng thời hỗ trợ developer dự đoán được các lỗi ngầm định có thể phát sinh khi phát triển các ứng dụng phần mềm khác nhau.
- ✓ **Hướng tiếp cận phản ứng:** Theo cách tiếp cận này, các thiết kế kiểm thử được xuất hiện sau khi phần mềm hoặc hệ thống được sản xuất.

Các hướng tiếp cận hoặc chiến lược điển hình bao gồm:

- Hướng tiếp cận phân tích: phân tích những rủi ro có thể gặp của phần mềm từ đó tiến hành kiểm thử dựa trên rủi ro.
- Hướng tiếp cận dựa trên mô hình: kiểm thử ngẫu nhiên sử dụng thông tin thống kê về tỷ lệ lỗi hay gặp ở các mức kiểm thử (kiểm thử đơn vị, kiểm thử tích hợp, kiểm thử hệ thống, kiểm thử chấp nhận).
- Hướng tiếp cận dựa trên việc đoán lỗi và tấn công lỗi.
- Hướng tiếp cận tuân thủ quy trình hoặc tiêu chuẩn, ví dụ tiêu chuẩn công nghiệp.
- Hướng tiếp cận động và thực nghiệm, ví dụ: kiểm thử thăm dò.
- Phương pháp tư vấn: Bằng lời khuyên và hướng dẫn của các chuyên gia trong lĩnh vực đó.
- Hướng tiếp cận hồi quy: tái sử dụng những vật liệu hoặc tài nguyên hiện có.



Hình 5.4: Hướng tiếp cận dựa trên mô hình

Để lựa chọn một cách tiếp cận phù hợp, ta có thể căn cứ vào những yếu tố sau:

- Rủi ro thất bại của dự án, mối nguy hiểm cho sản phẩm.
- Kỹ năng và kinh nghiệm của đội ngũ, các công cụ và phương pháp thực hiện.
- Mục tiêu kiểm thử và nhiệm vụ kiểm thử.
- Các quy định nghiệp vụ của khách hàng, hợp đồng với khách hàng.
- Bản chất của sản phẩm và những nghiệp vụ mà sản phẩm phải đáp ứng, ví dụ: phần mềm y tế, phần mềm bán hàng online, hay phần mềm kế toán,...

5.3 Giám sát và điều khiển tiến độ kiểm thử

5.3.1 Giám sát tiến độ kiểm thử

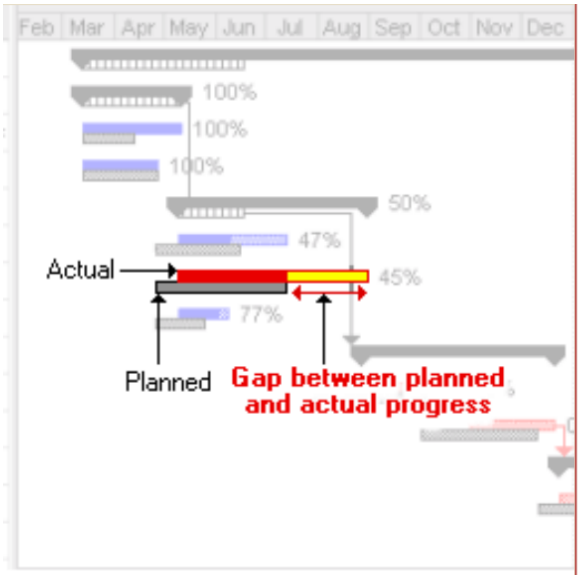
Giám sát là một nhiệm vụ của quản lý kiểm thử, giúp người quản lý kiểm thử kiểm tra định kỳ trạng thái cũng như tiến độ đạt được thực tế của dự án kiểm thử so với kế hoạch để điều chỉnh, cải tiến thời gian, nhân lực cũng như quy trình thực hiện để công việc kiểm thử dự án được diễn ra theo đúng kế hoạch đã đề ra ban đầu.

Việc giám sát tiến độ kiểm thử sẽ cung cấp:

- Thông tin phản hồi về tình hình kiểm thử đang diễn ra như thế nào.
- Kết quả kiểm thử đạt được so với kế hoạch.
- Có được các dữ liệu giúp chúng ta có thể ước lượng kiểm thử tốt hơn ở các dự án trong tương lai.

Các số liệu kiểm tra phổ biến bao gồm:

- Tỷ lệ phần trăm công việc hoàn thành trong việc chuẩn bị Test Cases.
- Tỷ lệ phần trăm công việc hoàn thành trong việc chuẩn bị môi trường kiểm thử.
- Bao nhiêu Test Cases đã được thực thi, bao nhiêu Test Cases bị hoãn lại hoặc chưa được thực thi,...
- Thông tin về các lỗi tìm được.
- Độ bao phủ kiểm thử được bao nhiêu phần trăm.
- Mức độ tự tin của Tester về sản phẩm được kiểm thử.
- Các ngày đánh dấu các mốc kiểm thử.
- Chi phí kiểm thử.



Hình 5.5: Giám sát tiến độ kiểm thử

5.3.2 Báo cáo kiểm thử

Sau khi hoàn tất việc kiểm thử và bàn giao sản phẩm cho khách hàng, ta cần viết báo cáo tóm tắt về hoạt động kiểm thử đã diễn ra để trình bày những gì đã thực hiện

được và những vấn đề cần cải tiến để kiểm thử tốt hơn ở những dự án trong tương lai. Mẫu báo cáo tóm tắt kiểm thử IEEE 829 bao gồm các mục sau:

- ✓ Định danh của file báo cáo
- ✓ Tóm tắt
- ✓ Sự sai lệch
- ✓ Đánh giá toàn diện
- ✓ Tóm tắt kết quả
- ✓ Đánh giá
- ✓ Tóm tắt hoạt động
- ✓ Sự phê duyệt

Định danh của file báo cáo kiểm thử (Test summary report identifier): là một nhãn định danh duy nhất để bạn có thể tham chiếu đến tài liệu đó.

Tóm tắt (Summary): Tóm tắt những gì đã được kiểm thử và những gì đã xảy ra. Chỉ ra tất cả các tài liệu liên quan.

Sự sai lệch (Variances): Nếu có bất kỳ mục kiểm thử nào khác với đặc tả của chúng, dẫn đến quá trình kiểm thử diễn ra không đúng như kế hoạch thì chúng ta cần trình bày ra và diễn giải cụ thể tại sao lại có sự sai lệch đó.

Đánh giá toàn diện (Comprehensiveness assessment): Việc kiểm thử đã diễn ra như thế nào, những mục nào đã được kiểm thử, những mục nào còn trì hoãn, những mục nào chưa được kiểm thử đủ,...và lý do của từng trường hợp.

Tóm tắt kết quả (Summary of results): Những vấn đề nào đã được xử lý? Những vấn đề nào còn tồn tại? (ở mức tổng quát)

Đánh giá (Evaluation): Các chức năng đã được kiểm thử tốt như thế nào? Những rủi ro gì có thể ảnh hưởng đến hoạt động của các chức năng đó?

Tóm tắt hoạt động (Summary of activities): Tóm tắt những hoạt động chính đã diễn ra trong quá trình kiểm thử và những tài nguyên, nhân lực, chi phí, thời gian dành cho việc kiểm thử.

Sự phê duyệt (Approvals): Thể hiện ai có quyền phê duyệt báo cáo này và chữ ký của người phê duyệt.

5.3.3 Kiểm soát kiểm thử

Kiểm soát kiểm thử mô tả những hành động hướng dẫn hoặc khắc phục trong quá trình giám sát hoạt động kiểm thử phần mềm.

Ví dụ về các hành động kiểm soát kiểm thử như sau:

- Ra quyết định dựa trên thông tin từ việc giám sát kiểm thử.
- Ưu tiên lại những trường hợp kiểm thử.
- Thay đổi lịch kiểm thử cho phù hợp với môi trường kiểm thử được thiết lập.
- Thiết lập một tiêu chí đầu vào phù hợp.

5.4 Quản lý cấu hình

5.4.1 Tại sao cần Quản lý cấu hình?

Chắc hẳn trong chúng ta đã ít nhất một lần gặp những tình huống sau:

- Một lỗi (bug) nào đó của phần mềm đang xây dựng đã tốn nhiều công sức sửa chữa, bỗng "thình lình" xuất hiện trở lại.
- Một chức năng (function) nào đó của phần mềm đã được phát triển và kiểm tra cẩn thận bỗng thất lạc hoặc biến mất một cách khó hiểu.
- Một chương trình (program) đã được kiểm tra hết sức cẩn thận, bỗng nhiên không "chạy" được nữa.
- Một chương trình gồm nhiều module, mỗi module gồm nhiều chức năng, các chức năng được chia ra cho nhiều lập trình viên, mỗi chức năng bao gồm nhiều tập tin mã nguồn (source code) với nhiều phiên bản (version) khác nhau. Khi tích hợp hệ thống và biên dịch, trong hàng chục tập tin source code với hàng trăm version, tập tin nào, version nào là đúng và cần phải lấy để tiến hành tích hợp?

Các vấn đề trên sẽ không xảy ra nếu như trong dự án, việc quản lý cấu hình (QLCH) được thực hiện nghiêm túc và kiểm soát chặt chẽ.

Ta có thể tham khảo định nghĩa ngắn gọn sau từ CMM và ISO 15504: "Mục đích của quản lý cấu hình là để thiết lập và bảo đảm tính toàn vẹn của các sản phẩm trung

gian cũng như các sản phẩm sau cùng của một dự án phần mềm, xuyên suốt chu kỳ sống của dự án đó."

Nói cho dễ hiểu và gần gũi, quản lý cấu hình bao gồm các công việc về nhận dạng, tổ chức, và quản lý các thay đổi đối với những sản phẩm đang được xây dựng bởi một nhóm lập trình viên, từ các sản phẩm trung gian đến sản phẩm sau cùng.

Quản lý cấu hình tốt sẽ giải quyết được hàng loạt những khó khăn trong các dự án, đặc biệt trong các dự án lớn:

- Cập nhật đồng thời: Khi 2 hoặc nhiều lập trình viên làm việc cách biệt nhau nhưng trên cùng một chương trình hoặc dự án, những thay đổi mà người này thực hiện có thể sẽ phá vỡ kết quả làm việc của người khác. Ví dụ: Sản phẩm anh A sử dụng kết quả công việc của anh B, sản phẩm của anh B thay đổi có thể làm cho sản phẩm anh A không chạy được nữa.
- Chia sẻ source code: Trong các hệ thống lớn, khi các chức năng chung bị thay đổi, tất cả những người liên quan phải được biết.
- Phiên bản phần mềm (release): Hầu hết các chương trình hoặc hệ thống lớn được phát triển với nhiều release tiến hóa từ thấp đến cao. Trong trường hợp một release khách hàng đang dùng, release khác đang được kiểm tra (test), và một release khác nữa đang trong quá trình phát triển, khi có lỗi (bug) xảy ra, việc sửa lỗi phải đồng bộ giữa ba phần này, nếu không quản lý source code tốt, vấn đề đồng bộ rất khó thực hiện được. Nếu lỗi ở release khách hàng đang dùng, nó phải được sửa chữa trong tất cả các release sau đó.

Quản lý cấu hình được thực hiện xuyên suốt chu kỳ sống của dự án, từ lúc bắt đầu đến khi kết thúc, thậm chí vẫn còn trong giai đoạn bảo trì sản phẩm sau dự án.

5.4.2 Hoạt động quản lý cấu hình

Trước khi đi vào chi tiết, ta cần tìm hiểu 2 khái niệm rất cơ bản trong quản lý cấu hình:

Configuration Item - CI: định danh này trong quản lý cấu hình là tên gọi của các sản phẩm, sản phẩm trung gian, một tập tin (file) hoặc nhóm file, tài liệu hoặc nhóm tài liệu trong một dự án mà ta cần phải quản lý và kiểm soát. Nói chung là những "món" được tạo ra trong một dự án mà ta cần phải quản lý, ví dụ: một file source code, tài liệu về yêu cầu sản phẩm, bản thiết kế v.v.

Baseline: một điểm "mốc" được thỏa thuận bởi những người liên quan trong một dự án, sao cho sau điểm "mốc" này, mọi thay đổi phải được thông báo tới tất cả những người có liên quan.

Lập kế hoạch quản lý cấu hình (Configuration Management planning)

Thông thường, việc lập kế hoạch cho quản lý cấu hình được thể hiện trong tài liệu có tên Kế hoạch quản lý cấu hình (Configuration Manegement Plan – CMP). Bản kế hoạch này thường bao gồm:

- Ý nghĩa, mục đích và phạm vi áp dụng của bản kế hoạch.
- Vai trò và trách nhiệm của nhóm, cá nhân trong dự án thực hiện các hoạt động khác nhau liên quan đến quản lý cấu hình. Xác định cụ thể ai thực hiện (perform), ai xem xét (review), ai phê duyệt (approve) trên các CI của dự án, cũng như vai trò của khách hàng, người sử dụng đầu cuối.
- Công cụ (tool), môi trường (environment) và cơ sở hạ tầng (infrastructure). Phần này mô tả các công cụ phần mềm hoặc quy trình thủ tục được sử dụng hỗ trợ quản lý cấu hình, chẳng hạn công cụ quản lý phiên bản sản phẩm (version control). Mô tả vị trí các máy chủ, máy trạm, cấu hình hệ thống client-server,...
- Phương pháp định danh và thiết lập baseline trên các CI.
- Quy ước đặt tên trong dự án, gồm cả tên file.
- Quy trình xử lý và quản lý các thay đổi (change control process).
- Chỉ định thành viên nhóm Configuration Control Board (CCB).
- Thông tin nơi lưu trữ các CI.
- Kiểm kê và báo cáo cấu hình (configuration accounting and reporting).
- Quy trình thủ tục lưu trữ và chép dự phòng (backup and archieve).

Định danh/đánh số các CI (Identification of Configuration Items)

Định danh là một trong những hoạt động nền tảng của quản lý cấu hình. Mục đích của định danh là để xác định tính duy nhất của một CI, cũng như mối quan hệ của nó với các CI khác. Nó bao gồm việc mô tả tên, đánh số, đánh dấu đặc trưng, giúp nhận biết và phân biệt một CI với các CI hay thành phần khác.

Bạn có thể nhận thấy hình thức định danh tương tự trong đời sống thực tế. Ví dụ, người ta đánh số bàn trong nhà hàng nhằm giúp người phục vụ mang đúng thức ăn cho khách.

Trong sản xuất phần mềm, một CI có thể bao gồm một hay nhiều file. Ví dụ: một module tên ExpMod có thể được coi là một CI, module này có 2 file ExpMod.h và ExpMod.c.

Mỗi CI phải có một số định danh duy nhất

Ví dụ: PRJ001_REQB_1.0.4_draft_B cho biết:

Số ID của dự án: PRJ001

Số ID của Item : REQB

Phiên bản: 1.0.4_draft_B

Trong một dự án, thường có rất nhiều file source code, quy tắc cơ bản là: các file cùng tạo nên một khối chức năng được gom chung thành một CI.

Kiểm soát phiên bản (Version Control)

Version control là sự kiểm soát các phiên bản (version) khác nhau của một CI (bao gồm việc định danh và sự lưu trữ CI đó).

Thế phiên bản là gì? Một phiên bản là một thực thể mới của một CI sau khi đã qua một hoặc nhiều lần xem xét và thay đổi.

Hiện có nhiều công cụ trên thị trường hỗ trợ cho việc kiểm soát phiên bản, một số công cụ thông dụng là: Visual Source Safe của Microsoft, ClearCase của Rational, CVS, TortoiseSVN,....

Mỗi phiên phản sẽ có một số ID đầy đủ, và được tăng dần cho mỗi phiên bản mới.

Lưu ý rằng phiên bản của một CI khác với phiên bản của các file thành phần của CI đó. Trong ví dụ trên, phiên bản của CI "ExpMod" khác với phiên bản của file thành phần "ExpMod.h" và "ExpMod.c".

Các phiên bản quan trọng của một CI có thể được đánh dấu để nhận biết một "mốc" quan trọng trong tiến trình phát triển CI đó, phiên bản mà CI được phê duyệt hay baseline.

Quản lý baseline (Baseline Management)

Trong thực tế thường gặp các loại baseline sau:

- Chức năng (functional)
- Kế hoạch (planning)
- Yêu cầu (requirements)
- Sản phẩm (product)
- Bản phân phối (release)
- Kiểm tra (test)
- Môi trường hoạt động (environment)

Quản lý baseline bao gồm:

- Chọn các CI cho mỗi loại baseline
- Tiến hành "ghim chết" baseline tại thời điểm sau khi các thay đổi đã được chấp thuận và phê chuẩn.

Thông thường, baseline được tiến hành tại điểm kết thúc của mỗi giai đoạn hay tại các "mốc" quan trọng trong dự án. Đồng thời, trong quản lý baseline, vai trò và nhiệm vụ của những người thiết lập hoặc phê chuẩn baseline cũng phải được xác định.

Kiểm soát thay đổi (Change control)

Khi phát triển hoặc bảo trì một sản phẩm phần mềm, việc thay đổi yêu cầu là không thể tránh khỏi.

Mục đích của change control là để kiểm soát đầy đủ tất cả các thay đổi ảnh hưởng đến việc phát triển một sản phẩm. Đôi lúc chỉ một vài yêu cầu thay đổi nhỏ của khách hàng, tất cả các chặng của quy trình phát triển phần mềm từ thiết kế, đến viết code, đến kiểm tra sản phẩm đều phải thay đổi theo.

Nếu các thay đổi này không được kiểm soát chặt chẽ sẽ dẫn đến rất nhiều sai sót. Xét ví dụ sau: 5 lập trình viên cùng làm trong một dự án, nhưng chỉ có 3 người được thông báo về việc thay đổi thiết kế. Kết quả là khi tích hợp, hệ thống sẽ không vận hành được.

Yêu cầu trong kiểm soát thay đổi là mọi sự thay đổi phải được thông báo đến tất cả những người hoặc nhóm làm việc có liên quan.

Các bước cơ bản của kiểm soát thay đổi bao gồm:

- Nghiên cứu, phân tích
- Phê chuẩn hoặc không phê chuẩn
- Thực hiện việc thay đổi
- Kiểm tra việc thay đổi
- Xác lập baseline mới

Trong kiểm soát thay đổi, ta cũng thường gặp khái niệm "nhóm kiểm soát thay đổi" gọi tắt là CCB (Change Control Board), nhóm này được thành lập trong từng dự án. CCB thông thường bao gồm:

- Người quản lý cấu hình (Configuration Manager)
- Người quản lý dự án (Project Manager)
- Trưởng nhóm (Technical Lead)
- Trưởng nhóm kiểm soát chất lượng (Test Lead)
- Kỹ sư chất lượng (Quality Engineer)
- Và những ai bị ảnh hưởng bởi các thay đổi

Nhiệm vụ của CCB bao gồm:

- Bảo đảm tất cả các thay đổi đều được các bộ phận liên quan nhận biết và tham gia.
- Xem xét, phê chuẩn hoặc phủ quyết các thay đổi trên các baseline.

- Kiểm tra, xác nhận các thay đổi.
- Phê chuẩn các bản phân phối sản phẩm (release) đến khách hàng

Báo cáo tình trạng cấu hình (Configuration Status Accounting)

Công việc này bao gồm việc ghi nhận và báo cáo tình trạng của các CI cũng như yêu cầu thay đổi, tập hợp số liệu thống kê về CI, đặc biệt là các CI góp phần tạo nên sản phẩm. Nó trả lời cho câu hỏi như: có bao nhiêu file bị ảnh hưởng khi sữa chữa một lỗi phần mềm nào đó?

Kết quả của công việc này được ghi nhận trong một báo cáo mang tên Configuration Status Accounting Report (CSAR). Báo cáo này thường làm rõ những điểm sau:

- Liệt kê tất cả baseline và CI thành phần hoặc có liên quan.
- Làm nổi bật các CI đang được phát triển hoặc vừa bị thay đổi.
- Liệt kê các thay đổi còn đang dang dở hay đang hoàn thành, và các baseline bị ảnh hưởng (bởi sự thay đổi đó).

Việc báo cáo này được làm thường xuyên và định kỳ, xuyên suốt dự án.

Auditing

Audit là một thuật ngữ rất thường dùng, cho nhiều ngành nghề khác nhau, tuy nhiên trong lĩnh vực sofware, nó có ý nghĩa gần với “kiểm tra” và “xem xét”. Có 3 loại audit thường được thực hiện:

- CSAR Audit: Thường được làm sau mỗi lần một CSAR được tạo ra, việc kiểm tra bao gồm:

- Bảo đảm các baseline mới nhất được liệt kê trong CSAR
- Bảo đảm tất cả CI tạo nên một baseline được liệt kê
- Kiểm tra các CI đã bị thay đổi từ lần baseline trước đó, so sánh chúng với các yêu cầu thay đổi để khẳng định rằng sự thay đổi trên CI là hợp lý.

- Physical configuration audit (PCA): nhằm mục đích khẳng định xem những gì khách hàng yêu cầu có được hiện thực hay không. Gồm 2 việc:

- Kiểm tra vết để phản ánh tính 2 chiều (traceability) giữa yêu cầu khách hàng và việc hiện thực code trong dự án.
- Xác định những gì sẽ được phân phối cho khách hàng (executable files, source code, tài liệu đi kèm...) có đáp ứng yêu cầu khách hàng hay không.

- Functional configuration audit (FCA): nhằm mục đích khẳng định những gì khách hàng yêu cầu có được kiểm tra chặt chẽ trên sản phẩm tạo ra trước khi giao cho khách hàng hay không.

Quản lý release (Release management)

Trong thực tế, có nhiều định nghĩa khác nhau về khái niệm "release". Về cơ bản, chúng ta có thể hiểu: Quá trình phát triển một phần mềm thường qua nhiều lần tích hợp, kết quả của mỗi lần tích hợp là một bản "build", trong rất nhiều bản "build" đó, một số bản đáp ứng một số yêu cầu hoặc kế hoạch đã định (theo yêu cầu khách hàng) sẽ được gởi cho khách hàng để kiểm tra hoặc đánh giá. Các bản build này được gọi là "release". Công việc tạo ra và phân phối các bản release được gọi là công việc "release". Theo cách hiểu này, sản phẩm sau cùng cũng là một bản release, đôi khi được gọi là "final release".

Trong quá trình release, việc quản lý đòi hỏi phải thực hiện các công việc sau:

- Baseline môi trường phát triển sản phẩm và các file, tài liệu (sẽ release)
- Thực hiện báo cáo CSAR
- Thực hiện các audit: PCA và FCA
- Đóng gói file và tài liệu sẽ release
- Xác nhận đã nhận bản release từ khách hàng

Lưu trữ và chép dự phòng (Backup & archive)

Lưu trữ và chép dự phòng là một hoạt động của quản lý cấu hình và là một trong những hoạt động quan trọng phải có của sản xuất phần mềm. Nó giúp khắc phục các trường hợp rủi ro bị mất mát dữ liệu do thao tác sai, virus, hoặc sự cố phần cứng/ phần mềm. Ở khía cạnh khác, nó hỗ trợ cho hoạt động version control trong trường hợp ta muốn sử dụng những version khác nhau.

Lưu trữ và chép dự phòng đòi hỏi toàn bộ sản phẩm và sản phẩm trung gian của dự án phải được định kỳ chép dự phòng trên những thiết bị hoặc những nơi khác một cách an toàn.

5.5 Kiểm thử và rủi ro

5.5.1 Rủi ro của dự án

Rủi ro của dự án là những rủi ro xoay quanh việc dự án có được chuyển giao đúng tiến độ và đầy đủ như mục tiêu đề ra hay không, bao gồm những vấn đề có thể gặp phải như sau:

- Vấn đề về nhân lực (thiếu, nghỉ việc, thay thế người khác,..), chậm trễ trong việc chuẩn bị tài nguyên và môi trường kiểm thử, nhà cung cấp thay đổi,... đều có thể gây ra những rủi ro cho dự án.
- Những thay đổi yêu cầu (change request) từ phía khách hàng có thể làm mất hiệu lực kết quả kiểm thử hoặc làm thay đổi kế hoạch ban đầu của dự án nếu chúng ta không quản lý tốt những thay đổi và điều chỉnh kịp thời.
- Môi trường kiểm thử không đầy đủ hoặc không thực tế dẫn đến kết quả kiểm thử bị sai lệch.

Ngoài ra còn có những rủi ro khác mà dự án có thể gặp phải:

- Yếu tố tổ chức: Thiếu đội ngũ có trình độ và kỹ năng đáp ứng tốt yêu cầu công việc; Các vấn đề về việc đào tạo những kỹ năng mới, chiến lược của Công ty; Vấn đề về tư duy và thái độ làm việc chuyên nghiệp.
- Yếu tố kỹ thuật: Các vấn đề trong việc xác định đúng yêu cầu; Vấn đề về chất lượng của thiết kế, lập trình và kiểm thử.
- Vấn đề nhà cung cấp.

Trên đây là những yếu tố có thể ảnh hưởng đến dự án, dẫn đến dự án có thể gặp những rủi ro như chậm tiến độ, các hạng mục không đáp ứng đúng yêu cầu khách hàng,... Kiểm soát tốt những yếu tố có thể gây ra rủi ro cho dự án giúp cho dự án được xuyên suốt, trơn tru, đáp ứng đầy đủ và đúng thời hạn theo yêu cầu của khách hàng, gia tăng uy tín của công ty, doanh nghiệp.

5.5.2 Rủi ro của sản phẩm

Rủi ro sản phẩm: Các rủi ro mà phần mềm hoặc hệ thống có thể gặp như:

- Phần mềm chuyển giao có độ tin cậy không cao, dễ bị lỗi khi chạy trong thời gian dài hoặc xử lý số lượng dữ liệu lớn mà nếu chỉ kiểm thử trong khoảng thời gian ngắn hoặc số lượng dữ liệu nhỏ thì không phát hiện ra.
- Các lỗi tiềm tàng trong phần mềm như lỗi bảo mật, lỗi khi có số lượng lớn user truy cập đồng thời,... nếu không được giả lập môi trường để kiểm thử kỹ càng thì có thể gây ra nhiều rủi ro cho sản phẩm, ảnh hưởng tới công việc của khách hàng, giảm uy tín của công ty.
- Phần mềm có những đặc điểm kém (ví dụ: kém về mặt chức năng, độ tin cậy, khả năng sử dụng và hiệu suất).
- Hiểu sai ý của khách hàng dẫn đến phần mềm không thực hiện các chức năng theo đúng mong đợi của khách hàng.

Trên đây là những rủi ro mà phần mềm hoặc hệ thống có thể gặp phải, nếu chỉ chú ý đến thực hiện dự án đúng tiến độ, với đúng tài nguyên và chi phí quy định mà không quan tâm đến chất lượng của sản phẩm làm ra, không quan tâm đến mong đợi của khách hàng, không kiểm thử đầy đủ và kỹ càng thì sản phẩm làm ra đúng hạn nhưng có thể không sử dụng được, dẫn đến lãng phí thời gian, nhân lực và tài nguyên trong quá trình phát triển phần mềm.

Các bước của mô hình quản lý rủi ro:

Mô hình quản lý rủi ro bao gồm có 5 bước như sau:

Bước 1: Xác định rủi ro

Bước 2: Phân tích rủi ro

Bước 3: Lập kế hoạch khắc phục hoặc hạn chế rủi ro

Bước 4: Theo dõi rủi ro

Bước 5: Kiểm soát rủi ro

Hình 5.6: Mô hình quản lý rủi ro

5.6 Quản lý sự cố (quản lý lỗi).

Mục tiêu của việc kiểm thử phần mềm là tìm lỗi trong phần mềm để sửa chữa, cải thiện nhằm nâng cao chất lượng của phần mềm và đáp ứng đầy đủ các yêu cầu của khách hàng.

Những khác biệt giữa kết quả thực tế và kết quả mong đợi cần phải được ghi lại dưới dạng những sự cố.

Các sự cố có thể phát sinh hoặc được phát hiện trong suốt quá trình phát triển phần mềm, xem xét các đặc tả yêu cầu phần mềm, kiểm thử hoặc sử dụng sản phẩm phần mềm.

Việc báo cáo sự cố nhằm các mục đích sau đây:

- Cung cấp cho lập trình viên và các bên liên quan những phản hồi về sự cố để có thể nhận dạng, cách ly và sửa chữa khi cần thiết.
- Cung cấp cho test leader một phương tiện để theo dõi chất lượng của sản phẩm được kiểm thử và theo dõi, kiểm soát tiến trình kiểm thử.
- Cung cấp các ý tưởng để cải tiến quy trình kiểm thử.

Mẫu Báo cáo Sự cố Kiểm thử theo Chuẩn IEEE 829

- Mã định danh của Báo cáo.
- Tóm lược báo cáo
- Mô tả sự cố (mô tả lỗi)
  - Đầu vào

- Kết quả mong đợi
- Kết quả thực tế
- Những bất thường
- Ngày và giờ
- Các bước thực hiện để bắt lỗi.
- Môi trường kiểm thử (ví dụ: Windows 10; MacOS, Android 5.0, IOS 10.0,...)
- Nỗ lực lặp lại
- Người kiểm tra và quan sát

▪ Những tác động

Một báo cáo sự cố chi tiết có thể bao gồm như sau:

- Ngày phát hành, tổ chức phát hành, và tác giả.
- Những kết quả mong đợi và kết quả thực tế.
- Xác định các mục kiểm thử và xác định môi trường kiểm thử.
- Quy trình vòng đời phần mềm hoặc hệ thống mà trong đó sự cố được quan sát.
- Mô tả về sự cố (các lỗi phần mềm) để những người phát triển có thể nhận biết và sửa chữa hoặc cải tiến, bao gồm các file báo cáo lỗi (ví dụ: defect list) và các ảnh chụp màn hình mô tả các lỗi (sự cố) đó.
- Phạm vi hoặc mức độ ảnh hưởng đến lợi ích của các bên liên quan.
- Mức độ nghiêm trọng tác động lên hệ thống.
- Mức độ khẩn cấp/ mức độ ưu tiên để sửa chữa các lỗi đó.
- Các trạng thái của lỗi như: Open (mở), Deferred (hoãn lại), Duplicate (trùng lặp), Waiting to be fixed (chờ sửa chữa), Fixed awaiting retest (đã sửa và chờ kiểm tra lại), Closed (đóng).
- Kết luận, khuyến nghị và phê duyệt.
- Các vấn đề toàn cục, chẳng hạn như những khu vực hoặc chức năng khác có thể bị ảnh hưởng bởi kết quả sửa lỗi ở một khu vực hoặc chức năng nào đó.
- Lịch sử sửa chữa, chẳng hạn như một chuỗi các hành động được thực hiện bởi các thành viên trong đội dự án liên quan đến các lỗi được báo cáo để cô lập, sửa chữa vả chuyển lỗi đó sang trạng thái “fixed“ sau khi đã sửa.
- Những tham chiếu, chẳng hạn như những lỗi đó được phát hiện từ những test case nào.