Nhập môn Công nghệ phần mềm

Biên tập bởi:

Phạm Thị Quỳnh

Nhập môn Công nghệ phần mềm

Biên tập bởi:

Phạm Thị Quỳnh

Các tác giả:

Phạm Thị Quỳnh

Phiên bản trực tuyến:
http://voer.edu.vn/c/fb584480

MỤC LỤC

Phần mềm là gì?

Khái niệm

Phần mềm là các chương trình máy tính và những tài liệu liên quan đến nó như: các yêu cầu, mô hình thiết kế, tài liệu hướng dẫn sử dụng… Do đó, chúng ta thấy rằng đặc điểm của phần mềm là trừu tượng và vô hình.

Các sản phẩm phần mềm được chia thành 2 loại:

- Sản phẩm đại trà (Generic Product): được phát triển để bán ra ngoài thị trường, đối tượng người sử dụng là tương đối đa dạng và phong phú. Những sản phẩm phần mềm thuộc loại này thường là những phần mềm dành cho máy PC.

- Sản phẩm theo đơn đặt hàng (Bespoke Product hoặc Customised Product): được phát triển cho một khách hàng riêng lẻ theo yêu cầu. Ví dụ: Những hệ thống phần mềm chuyên dụng, hỗ trợ nghiệp vụ cho một doanh nghiệp riêng lẻ …

Một phần mềm mới có thể được tạo ra bằng cách phát triển các chương trình mới, thay đổi và điều chỉnh các hệ thống phần mềm đại trà hoặc tái sử dụng lại các phần mềm đã tồn tại.

Công nghệ phần mềm là gì?

Công nghệ phần mềm là những quy tắc công nghệ (engineering discipline) có liên quan đến tất cả các khía cạnh của quá trình sản xuất phần mềm.

Các kỹ sư phần mềm nên tuân theo một phương pháp luận có hệ thống và có tổ chức trong công việc của họ. Đồng thời, họ nên sử dụng các công cụ và kỹ thuật thích hợp với vấn đề cần giải quyết, các ràng buộc và tài nguyên sẵn có.

Sự khác biệt giữa công nghệ phần mềm và khoa học máy tính?

Khoa học máy tính đề cấp tới lý thuyết và những vấn đề cơ bản; còn công nghệ phần mềm đề cập tới các hoạt động xây dựng và đưa ra một phần mềm hữu ích.

Khi sự phát triển của phần mềm trở lên mạnh mẽ thì các lý thuyết của khoa học máy tính vẫn không đủ để đóng vai trò là nền tảng hoàn thiện cho công nghệ phần mềm.

Sự khác biệt giữa công nghệ phần mềm và công nghệ hệ thống?

Đặt vấn đề

- Học viên đã bao giờ nghe nhắc tới Công nghệ hệ thống hay chưa?

- Hãy trình bày một số vấn đề có liên quan đến Công nghệ hệ thống.

- Công nghệ phần mềm có phải là Công nghệ hệ thống không?

Công nghệ hệ thống (hay còn gọi là kỹ nghệ hệ thống) liên quan tới tất cả các khía cạnh của quá trình phát triển hệ thống dựa máy tính bao gồm: phần cứng, phần mềm, và công nghệ xử lý. Công nghệ phần mềm chỉ là một phần của quy trình này, nó có liên quan tới việc phát triển hạ tầng phần mềm (software infrastructure), điều khiển, các ứng dụng và cơ sở dữ liệu trong hệ thống.

Kỹ sư hệ thống phải thực hiện việc đặc tả hệ thống, thiết kế kiến trúc hệ thống, tích hợp và triển khai.

Quy trình phần mềm là gì?

Đặt vấn đề

- Hãy cho biết để sản xuất một phần mềm, người ta phải thực hiện những công việc nào?

Quy trình phần mềm là một tập hợp các hành động mà mục đích của nó là xây dựng và phát triển phần mềm. Những hành động thường được thực hiện trong các quy trình phần mềm bao gồm

- Đặc tả: đặc tả những gì hệ thống phải làm và các ràng buộc trong quá trình xây dựng hệ thống.

- Phát triển: xây dựng hệ thống phần mềm.

- Kiểm thử: kiểm tra xem liệu phần mềm đã thoả mãn yêu cầu của khách hàng.

- Mở rộng: điều chỉnh và thay đổi phần mềm tương ứng với sự thay đổi yêu cầu.

Những loại hệ thống khác nhau sẽ cần những quy trình phát triển khác nhau. Ví dụ, hệ thống thời gian thực yêu cầu phải hoàn thành đặc tả hệ thống trước khi chuyển sang giai đoạn xây dựng nó. Nhưng với hệ thống thương mại điện tử, chúng ta có thể vừa đặc tả vừa xây dựng chương trình một cách đồng thời.

Tuy nhiên, nếu chúng ta không sử dụng một quy trình phát triển hệ thống thích hợp thì có thể làm giảm chất lượng của hệ thống và tăng chi phí xây dựng.

Mô hình quy trình phát triển phần mềm là gì?

Mô hình quy trình phát triển phần mềm là một thể hiện đơn giản của một quy trình phần mềm, và nó được biểu diễn từ một góc độ cụ thể.

Sau đây là một số ví dụ về mô hình quy trình phát triển phần mềm:

- Mô hình luồng công việc (workflow): mô tả một chuỗi các hành động cần phải thực hiện.

- Mô hình luồng dữ liệu (data-flow): mô tả luồng thông tin.

- Mô hình Vai trò/Hành động (Role/action): chỉ ra vai trò của những người liên quan trong quy trình phần mềm và nhiệm vụ của từng người.

Ngoài ra, còn có một số mô hình quy trình chung cũng được đề xuất như:

- Mô hình thác nước (waterfall)

- Mô hình phát triển lặp lại (Iterative development)

- Mô hình công nghệ phần mềm dựa thành phần (Component-based software engineering).

Các chi phí trong công nghệ phần mềm

Đặt vấn đề

- Để xây dựng một hệ thống phần mềm chúng ta phải đầu tư cho những hạng mục nào?

- Tất cả các hệ thống phần mềm có cùng các hạng mục chi phí hay không? Tại sao?

Để xây dựng một hệ thống phần mềm, chúng ta thường phải đầu tư một khoản ngân sách khá lớn. Theo thống kê cho thấy, chi phí cho việc xây dựng phần mềm chiếm một phần đáng kể của GNP ở tất cả các nước phát triển.

Chi phí phần mềm thường chiếm phần lớn chi phí của cả hệ thống máy tính. Chi phí phần mềm trên máy PC thường lớn hơn chi phí phần cứng. Chi phí phần mềm dành cho việc bảo trì phần mềm thường lớn hơn chi phí xây dựng phần mềm. Đối với những hệ thống hoạt động trong thời gian dài, thì chi phí bảo trì thường lớn gấp nhiều lần so với chi phí xây dựng.

Xấp xỉ 60% chi phí là chi phí xây dựng và 40% là chi phí kiểm thử. Đối với những phần mềm làm theo yêu cầu của khách hàng, chi phí mở rộng thường vượt quá chi phí xây dựng.

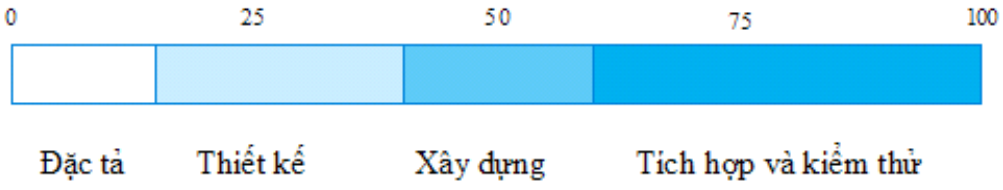
Chi phí biến đổi tuỳ thuộc vào từng loại hệ thống được xây dựng và các yêu cầu về đặc điểm của hệ thống như: hiệu năng và độ tin cậy của hệ thống.

Hình 1.1: Chi phí sản xuất phần mềm

Việc phân bổ chi phí cũng phụ thuộc vào mô hình phát triển hệ thống được sử dụng. Sau đây là bảng so sánh chi phí của 3 mô hình phổ biến nhất, thường được sử dụng:

Mô hình thác nước:

Chi phí của các pha đặc tả, thiết kế, cài đặt, tích hợp và kiểm thử được xác định một cách riêng rẽ.



Hình 1.2: Phân bổ chi phí trong mô hình thác nước

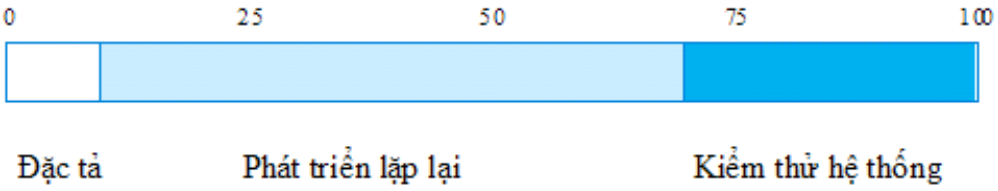
Mô hình phát triển lặp lại

Không thể phân biệt rõ chi phí cho từng pha trong quy trình.

Chi phí đặc tả giảm vì đây là đặc tả ở bậc cao.

Tại mỗi bước lặp, các pha trong quy trình xây dựng hệ thống được thực hiện lại nhằm thực hiện các yêu cầu hệ thống khác nhau ở từng bước lặp.

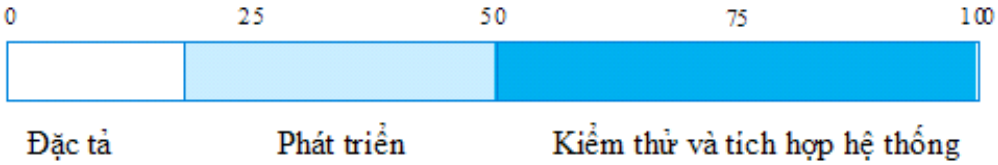
Sau khi đã thực hiện hết các bước lặp, phải có chi phí kiểm thử toàn bộ hệ thống.



Hình 1.3: Phân bổ chi phí trong mô hình phát triển lặp lại

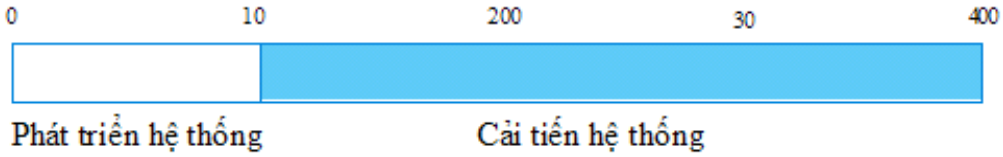
Mô hình công nghệ phần mềm hướng thành phần

Chi phí phụ thuộc nhiều vào việc tích hợp và kiểm thử hệ thống.



Hình 1.4: Phân bổ chi phí trong công nghệ phần mềm hướng thành phần

Ngoài chi phí xây dựng, chúng ta còn phải để một phần lớn chi phí phục vụ cho việc thay đổi phần mềm sau khi nó đã được đưa vào sử dụng. Chi phí cải tiến phần mềm thay đổi phụ thuộc vào từng loại phần mềm.



Hình 1.5: Phân bổ chi phí trên các hệ thống có chu kỳ sống dài

Các phương pháp công nghệ phần mềm là gì?

Phương pháp công nghệ phần mềm bao gồm các mô hình hệ thống, các ký pháp, quy tắc, hướng dẫn thiết kế và quy trình để xây dựng phần mềm một cách dễ dàng, đảm bảo chất lượng cao và chi phí hiệu quả.

Một số phương pháp công nghệ phần mềm đã được đề xuất như: Phân tích hướng cấu trúc - tập trung vào việc xác định các chức năng cơ bản của hệ thống; phương pháp hướng đối tượng - tập trung vào việc định nghĩa các đối tượng và sự cộng tác giữa chúng ...

CASE (Computer-Aided Software Engineering)

Các hệ thống CASE thường được sử dụng để hỗ trợ các hoạt động trong quy trình xây dựng phần mềm. Có hai loại CASE:

- Upper-CASE: công cụ để hỗ trợ các hoạt động đầu tiên như đặc tả yêu cầu và thiết kế.

- Lower-CASE: công cụ để hỗ trợ các hoạt động sau như lập trình, gỡ lỗi và kiểm thử.

Thế nào là một phần mềm tốt?

Đặt vấn đề

- Bạn có thường xuyên sử dụng phần mềm không?

- Theo bạn, thế nào là một phần mềm tốt?

Phần mềm phải đáp ứng các chức năng theo yêu cầu, có hiệu năng tốt, có khả năng bảo trì, đáng tin cậy, và được người sử dụng chấp nhận.

- Khả năng bảo trì: phần mềm phải được điều chỉnh và mở rộng để thoả mãn những yêu cầu thay đổi.

- Mức độ tin cậy: phần mềm phải được tin cậy, bảo mật và chính xác.

- Hiệu quả: phần mềm không nên sử dụng lãng phí tài nguyên của hệ thống.

- Khả năng được chấp nhận: người sử dụng phải chấp nhận phần mềm. Điều đó có nghĩa là nó phải dễ hiểu, sử dụng được và tương thích với các hệ thống khác.

Thách thức đối với công nghệ phần mềm?

Đặt vấn đề

- Nếu quan tâm đến sự phát triển của công nghệ phần mềm, bạn hãy cho biết những thách thức mà công nghệ phần mềm phải đối mặt.

Công nghệ phần mềm trong thế kỷ 21 phải đối mặt với rất nhiều thách thức to lớn. Với mỗi thách thức này, chúng ta phải có những giải pháp cụ thể.

- Không đồng nhất: phát triển các kỹ thuật xây dựng phần mềm để giải quyết sự không đồng nhất về môi trường thực hiện và nền tảng hạ tầng.

- Chuyển giao: phát triển các kỹ thuật nhằm dẫn tới việc chuyển giao phần mềm tới người sử dụng nhanh hơn.

- Độ tin cậy: phát triển các kỹ thuật để chứng minh rằng phần mềm được người sử dụng nó tin tưởng.

Vấn đề về tính chuyên nghiệp và đúng quy tắc

Đặt vấn đề

- Theo bạn, để đáp ứng được các yêu cầu công việc một kỹ sư phần mềm phải thoả mãn được những yêu cầu gì?

Quy trình xây dựng phần mềm được thực hiện trong một môi trường chuyên nghiệp và đòi hỏi tuân thủ các nguyên tắc một cách chính xác. Do đó, những kỹ sư phần mềm phải coi công việc của họ là trách nhiệm to lớn, chứ không đơn thuần chỉ là việc ứng dụng kỹ thuật.

Kỹ sư phần mềm phải ứng xử trung thực và cách làm của họ phải rất chuyên nghiệp và đúng quy tắc. Trong phần này, chúng ta sẽ trình bày một số nguyên tắc cần thiết mà một kỹ sư phần mềm phải thực hiện.

- Sự tin cẩn: kỹ sư phần mềm phải tạo được sự tin cẩn từ phía nhân viên và khách hàng.

- Năng lực: kỹ sư phần mềm không nên trình bày sai khả năng của mình, không nên nhận những công việc vượt quá khả năng của mình.

- Các quyền về tài sản trí tuệ: kỹ sư phần mềm nên quan tâm về các tài sản trí tuệ được bảo hộ như: bằng sáng chế, quyền tác giả … để đảm bảo rằng tất cả tài sản trí tuệ của nhân viên và khách hàng đều được bảo hộ.

- Lạm dụng máy tính: kỹ sư phần mềm không nên sử dụng các kỹ năng của mình để gây ảnh hưởng tới người khác. Lạm dụng máy tính có thể được hiểu là những việc tầm thường (Ví dụ: chơi điện tử trên máy tính của người khác) đến những vấn đề nghiêm trọng (Ví dụ: phát tán virus).

Vấn đề về tính chuyên nghiệp và đúng quy tắc đối với kỹ sư phần mềm quan trọng tới mức một số tổ chức ở Mỹ đã hợp tác để phát triển bản Code of Ethics gồm 8 quy tắc liên quan đến ứng xử và cách ra quyết định của các kỹ sư phần mềm chuyên nghiệp.

Một số mô hình phát triển phần mềm

Một số mô hình

Giới thiệu

Mô hình phát triển phần mềm là một thể hiện trừu tượng của quy trình phần mềm. Nó biểu diễn các đặc tả về quy trình từ những khía cạnh cụ thể; do đó, nó chỉ cung cấp một phần thông tin về quy trình phần mềm.

Phần sau đây sẽ trình bày năm mô hình phát triển phần mềm phổ biến thường được sử dụng:

- Mô hình thác nước

- Mô hình xây dựng tiến triển

- Công nghệ phần mềm dựa thành phần

- Mô hình phát triển lặp lại, tăng thêm

- Mô hình xoắn ốc

Mục tiêu

- Phải hiểu rõ năm mô hình phát triển phần mềm cơ bản.

- Phân biệt được sự khác nhau giữa các mô hình; ưu và nhược điểm của từng mô hình.

- Biết rõ đối với loại hệ thống nào thì nên áp dụng mô hình phát triển nào cho phù hợp.

Mô hình thác nước

Các pha của mô hình thác nước bao gồm:

- Phân tích và xác định các yêu cầu

- Thiết kế hệ thống và phần mềm

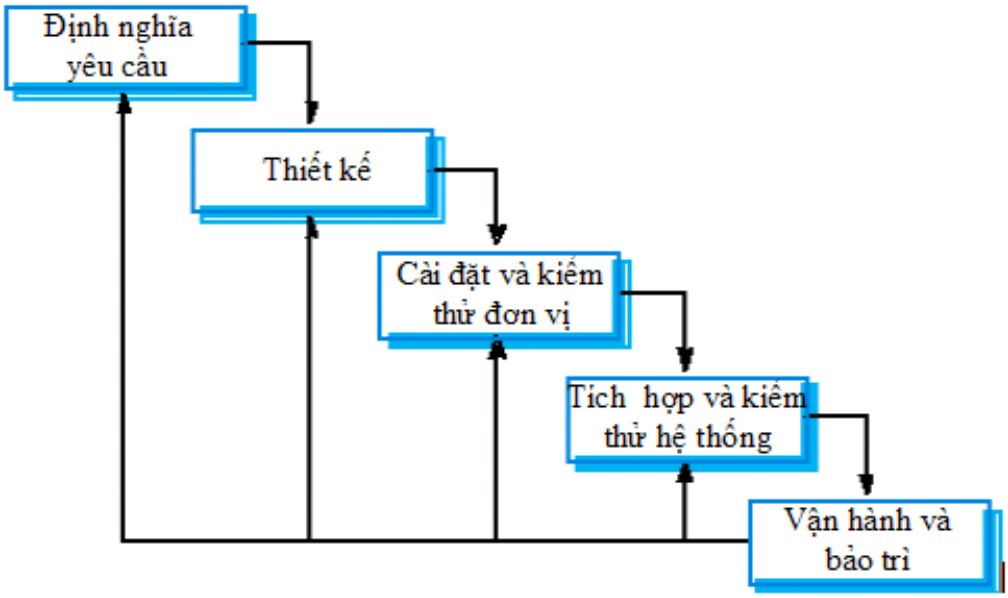
- Cài đặt và kiểm thử đơn vị

- Tích hợp và kiểm thử hệ thống

- Vận hành và bảo trì.

Trong mô hình thác nước, năm pha trên phải được thực hiện một cách tuần tự; kết thúc pha trước, rồi mới được thực hiện pha tiếp theo. Do đó, nhược điểm chính của mô hình thác nước là rất khó khăn trong việc thay đổi các pha đã được thực hiện. Giả sử, pha phân tích và xác định yêu cầu đã hoàn tất và chuyển sang pha kế tiếp, nhưng lúc này lại có sự thay đổi yêu cầu của người sử dụng; thì chỉ còn cách là phải thực hiện lại từ đầu.

Cho nên, mô hình này chỉ thích hợp khi các yêu cầu đã được tìm hiểu rõ ràng và những thay đổi sẽ được giới hạn một cách rõ ràng trong suốt quá trình thiết kế. Tuy nhiên, trong thực tế có rất ít những hệ thống nghiệp vụ có các yêu cầu ổn định.



Hình 2.1: Mô hình thác nước

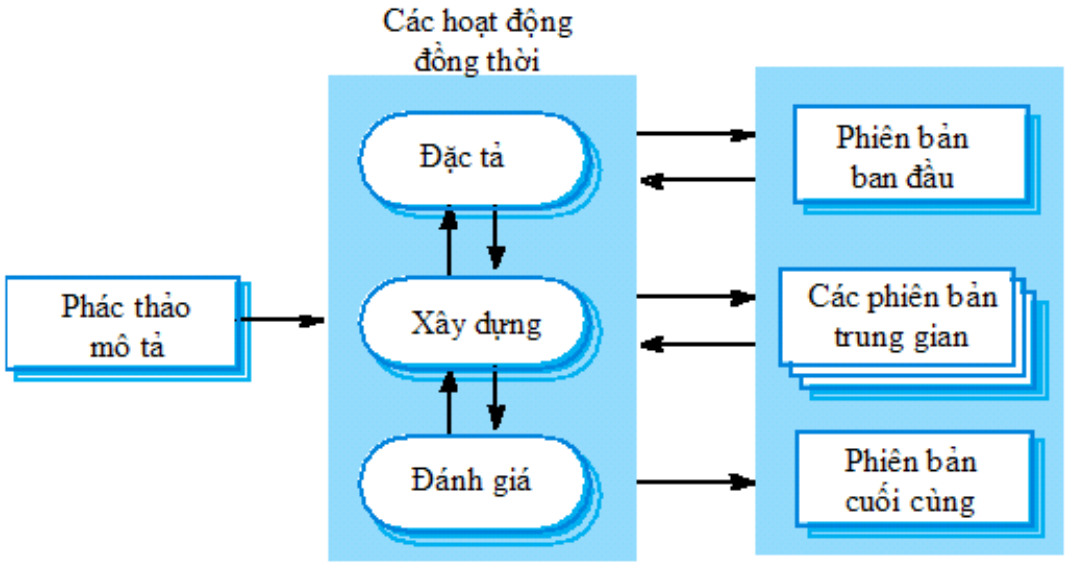
Mô hình xây dựng tiến triển

Mô hình xây dựng tiến triển dựa trên ý tưởng xây dựng một mẫu thử ban đầu và đưa cho người sử dụng xem xét; sau đó, tinh chỉnh mẫu thử qua nhiều phiên bản cho đến khi thoả mãn yêu cầu của người sử dụng thì dừng lại.

Có hai phương pháp để thực hiện mô hình này:

- Phát triển thăm dò: mục đích của nó là để làm việc với khách hàng và để đưa ra hệ thống cuối cùng từ những đặc tả sơ bộ ban đầu. Phương pháp này thường bắt đầu thực hiện với những yêu cầu được tìm hiểu rõ ràng và sau đó, bổ sung những đặc điểm mới được đề xuất bởi khách hàng. Cuối cùng, khi các yêu cầu của người sử dụng được thoả mãn thì cũng là lúc chúng ta đã xây dựng xong hệ thống.

- Loại bỏ mẫu thử: mục đích là để tìm hiểu các yêu cầu của hệ thống. Phương pháp này thường bắt đầu với những yêu cầu không rõ ràng và ít thông tin. Các mẫu thử sẽ được xây dựng và chuyển giao tới cho người sử dụng. Từ đó, ta có thể phân loại những yêu cầu nào là thực sự cần thiết và lúc này mẫu thử không còn cần thiết nữa. Như vậy, mẫu thử chỉ có tác dụng để làm sáng tỏ yêu cầu của người sử dụng.



Hình 2.2: Mô hình xây dựng tiến triển

Tuy nhiên, nhược điểm của mô hình xây dựng tiến triển là: thiếu tầm nhìn của cả quy trình; các hệ thống thường hướng cấu trúc nghèo nàn; yêu cầu các kỹ năng đặc biệt (Ví dụ: các ngôn ngữ để tạo ra mẫu thử nhanh chóng).

Mô hình xây dựng tiến triển chỉ nên áp dụng với những hệ thống có tương tác ở mức độ nhỏ hoặc vừa; trên một phần của những hệ thống lớn; hoặc những hệ thống có thời gian chu kỳ tồn tại ngắn.

Công nghệ phần mềm dựa thành phần

Mô hình này dựa trên kỹ thuật tái sử dụng một cách có hệ thống; trong đó hệ thống được tích hợp từ nhiều thành phần đang tồn tại hoặc các thành phần thương mại COTS (Commercial-off-the-shelf).

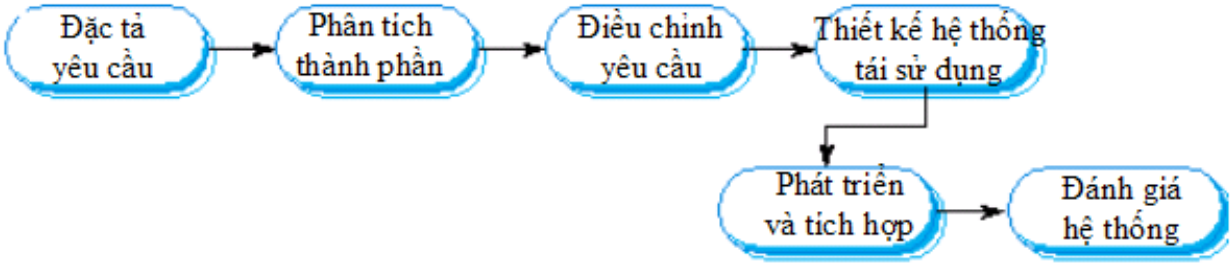
Các trạng thái chính của quy trình bao gồm:

- Phân tích thành phần sẵn có

- Điều chỉnh yêu cầu

- Thiết kế hệ thống với kỹ thuật tái sử dụng

- Xây dựng và tích hợp hệ thống

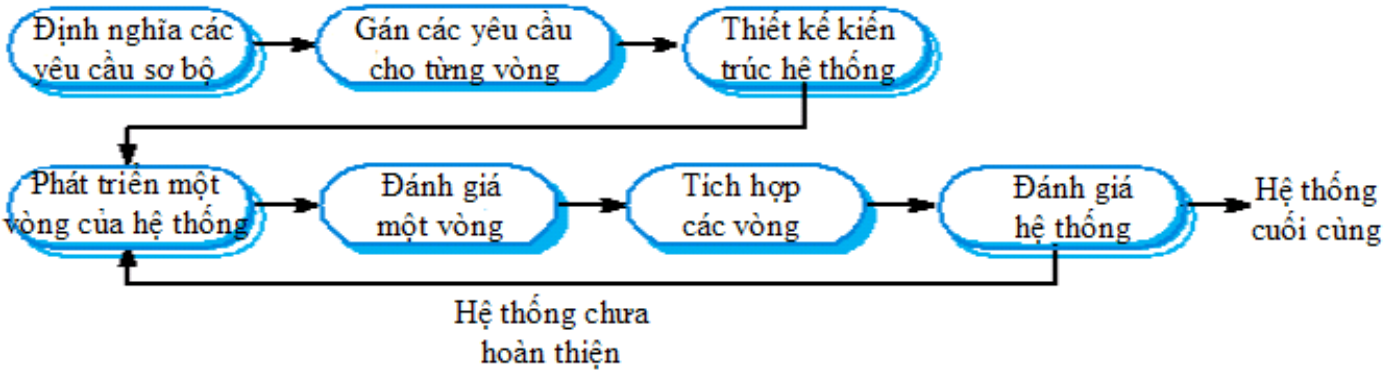


Hình 2.3: Công nghệ phần mềm hướng thành phần

Mô hình phát triển lặp lại, tăng thêm

Mô hình này được đề xuất dựa trên ý tưởng thay vì phải xây dựng và chuyển giao hệ thống một lần thì sẽ được chia thành nhiều vòng, tăng dần. Mỗi vòng là một phần kết quả của một chức năng được yêu cầu.

Các yêu cầu của người sử dụng được đánh thứ tự ưu tiên. Yêu cầu nào có thứ tự ưu tiên càng cao thì càng ở trong những vòng phát triển sớm hơn.



Hình 2.4: Kết quả tăng dần

Từ đó, chúng ta có thể thấy rõ một số ưu điểm của mô hình phát triển tăng vòng:

- Sau mỗi lần tăng vòng thì có thể chuyển giao kết quả thực hiện được cho khách hành nên các chức năng của hệ thống có thể nhìn thấy sớm hơn.

- Các vòng trước đóng vai trò là mẫu thử để giúp tìm hiểu thêm các yêu cầu ở những vòng tiếp theo.

- Những chức năng của hệ thống có thứ tự ưu tiên càng cao thì sẽ được kiểm thử càng kỹ.

Mô hình xoắn ốc

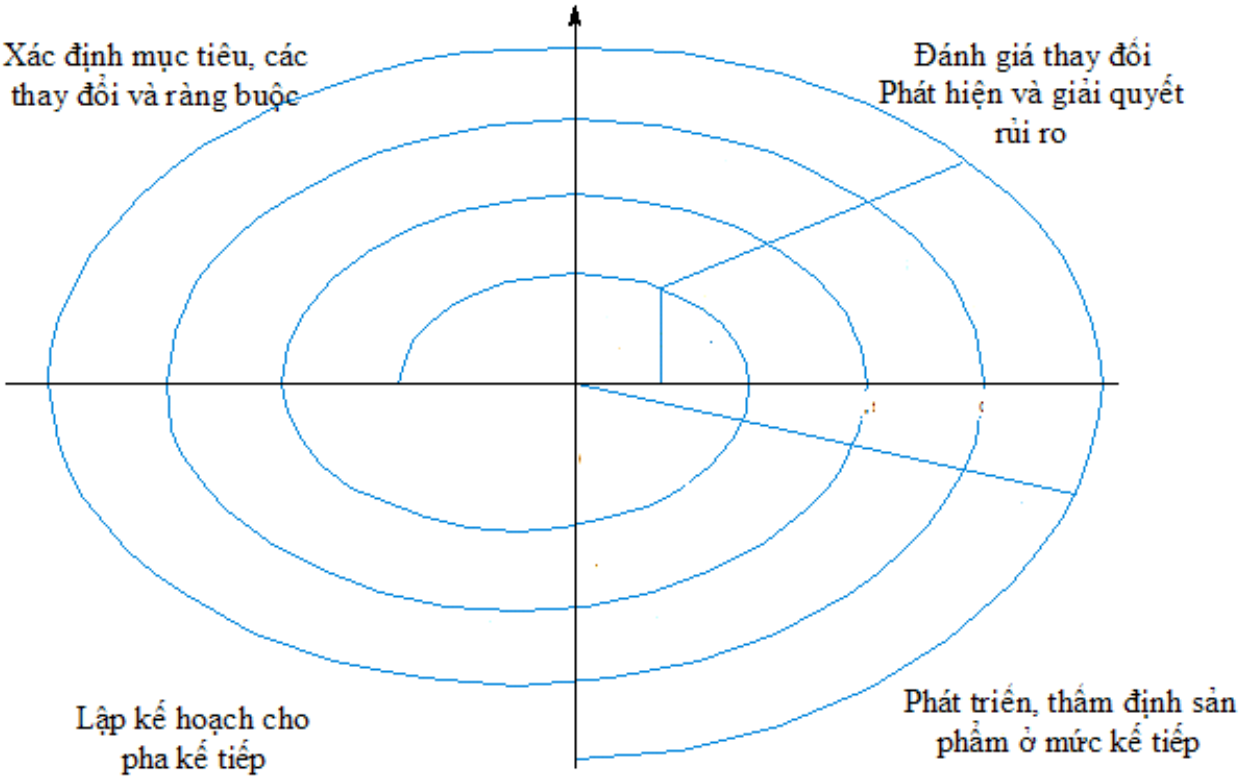
Trong mô hình xoắn ốc, quy trình phát triển phần mềm được biểu diễn như một vòng xoắn ốc. Các pha trong quy trình phát triển xoắn ốc bao gồm:

- Thiết lập mục tiêu: xác định mục tiêu cho từng pha của dự án.

- Đánh giá và giảm thiểu rủi ro: rủi ro được đánh giá và thực hiện các hành động để giảm thiểu rủi ro.

- Phát triển và đánh giá: sau khi đánh giá rủi ro, một mô hình xây dựng hệ thống sẽ được lựa chọn từ những mô hình chung.

- Lập kế hoạch: đánh giá dự án và pha tiếp theo của mô hình xoắn ốc sẽ được lập kế hoạch.



Hình 2.5: Mô hình xoắn ốc

Các hoạt động trong quy trình phần mềm

Các hoạt động

Giới thiệu

Trong quy trình phần mềm gồm 4 hoạt động cơ bản. Những hoạt động này bao gồm:

- Đặc tả: các chức năng của hệ thống và những ràng buộc khi vận hành hệ thống cần phải được xác định một cách đầy đủ và chi tiết.

- Thiết kế và cài đặt: phần mềm được xây dựng phải thoả mãn đặc tả của nó.

- Đánh giá: phần mềm phải được đánh giá và thẩm định để đảm bảo rằng nó đã thoả mãn tất cả các yêu cầu.

- Cải tiến: phần mềm cần phải cải tiến và điều chỉnh để phù hợp với những thay đổi về yêu cầu hệ thống.

Với mỗi mô hình khác nhau thì các hoạt động này cũng được tổ chức theo các cách khác nhau. Ví dụ, trong mô hình thác nước, chúng được tổ chức một cách tuần tự. Trong mô hình tiến triển, các hoạt động này có thể gối lên nhau. Trong các phần tiếp sau đây, chúng ta sẽ nghiên cứu cụ thể từng hoạt động.

Mục tiêu

- Xác định rõ những công việc nào cần phải làm trong quy trình phát triển phần mềm.

- Từng công việc đó được thực hiện cụ thể ra sao

- Phải nhớ một điều rằng: khi xây dựng bất kỳ phần mềm nào, chúng ta đều phải thực hiện bốn công việc trên. Tuy nhiên, với việc sử dụng các mô hình phát triển phần mềm khác nhau thì trình tự thực hiện các công việc trên cũng khác nhau.

Đặc tả phần mềm

Đặt vấn đề

- Công việc đầu tiên cần phải làm trong quá trình xây dựng phần mềm là gì?

- Tầm quan trọng của việc đặc tả phần mềm.

Đặc tả phần mềm (hay còn gọi là kỹ thuật xác định yêu cầu) là quy trình tìm hiểu và định nghĩa những dịch vụ nào được yêu cầu và các ràng buộc trong quá trình vận hành và xây dựng hệ thống.

Quy trình xác định yêu cầu bao gồm bốn pha chính:

- Nghiên cứu khả thi: Nghiên cứu khả thi giúp xác định những yêu cầu của người sử dụng có thoả mãn những công nghệ hiện tại hay không. Về góc độ kinh doanh, nghiên cứu khả thi nhằm xác định hệ thống đưa ra có mang lại lợi nhuận không. Việc nghiên cứu khả thi nên được thực hiện một cách nhanh chóng và không quá tốn kém. Kết quả của việc nghiên cứu khả thi sẽ xác định có nên tiếp tục xây dựng hệ thống nữa hay không.

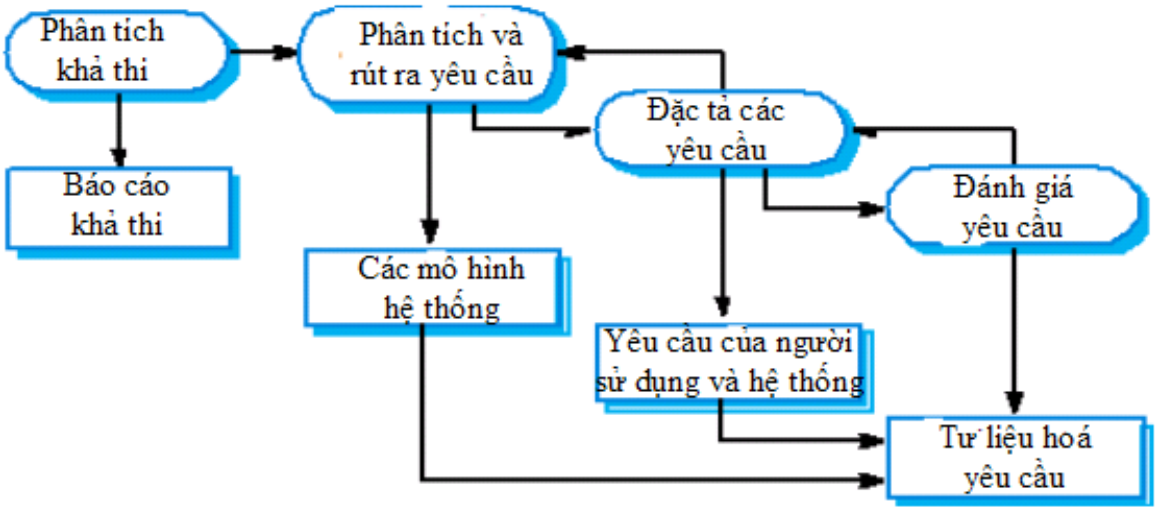
- Phân tích và rút ra các yêu cầu: đây là quy trình đưa ra các yêu cầu hệ thống thông qua một số phương pháp như: quan sát hệ thống hiện tại, phỏng vấn và thảo luận với người sử dụng, phân tích nhiệm vụ, phân tích tài liệu hoặc hệ thống cũ … Trong pha này, chúng ta có thể phải xây dựng một hoặc nhiều mô hình hệ thống và các mẫu thử.

- Đặc tả yêu cầu: Pha này sẽ tư liệu hoá những thông tin thu thập được. Có hai loại yêu cầu cần được xác định:

* Yêu cầu của người sử dụng: là những yêu cầu bằng ngôn ngữ tự nhiên bổ sung thêm cho các biểu đồ của các dịch vụ mà hệ thống cung cấp và các ràng buộc vận hành của nó. Kiểu yêu cầu này được viết bởi người sử dụng.

* Yêu cầu hệ thống: là những tài liệu có cấu trúc mô tả chi tiết về các chức năng, dịch vụ và các ràng buộc vận hành của hệ thống. Yêu cầu hệ thống sẽ định nghĩa những gì cần phải xây dựng, cho nên nó có thể trở thành bản hợp đồng giữa khách hàng và nhà thầu.

- Đánh giá yêu cầu: pha này sẽ kiểm tra lại các yêu cầu xem chúng có đúng thực tế hay không, có thống nhất không, có đầy đủ không. Nếu phát hiện ra lỗi thì ta phải chỉnh sửa các lỗi này.



Hình 2.6: Quy trình xác định yêu cầu

Thiết kế phần mềm và cài đặt

Đặt vấn đề

- Bỏ qua giai đoạn thiết kế, sau khi đặc tả và phân tích yêu cầu, có thể thực hiện cài đặt hệ thống ngay được không?

- Vai trò của bản thiết kế đối với giai đoạn cài đặt là gì?

Thiết kế phần mềm là quá trình thiết kế cấu trúc phần mềm dựa trên những tài liệu đặc tả. Hoạt động thiết kế bao gồm những công việc chính sau:

- Thiết kế kiến trúc: Các hệ thống con cấu thành lên hệ thống cần xây dựng và mối quan hệ giữa chúng được xác định và tư liệu hoá.

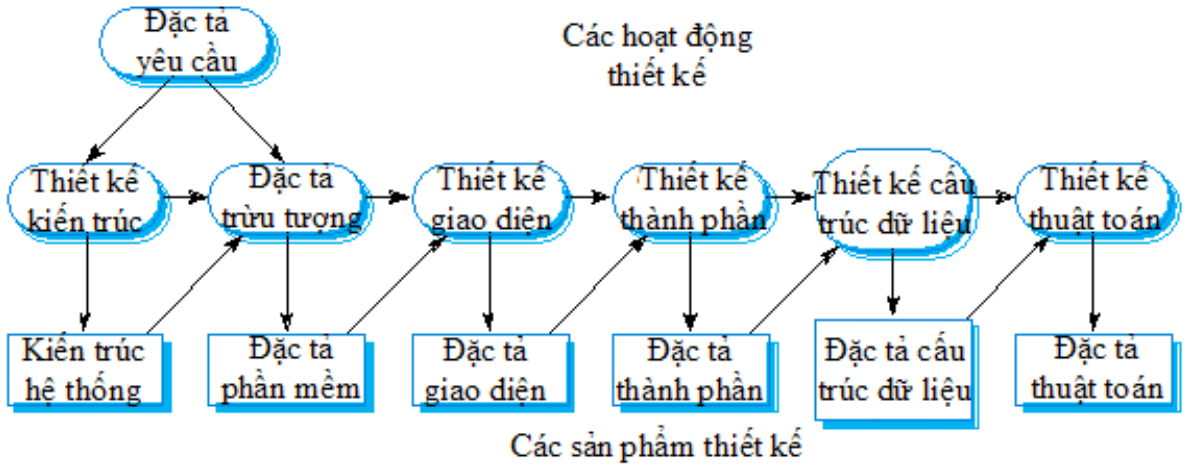
- Đặc tả trừu tượng: với mỗi hệ thống con, phải có một bản đặc tả về các dịch vụ của nó và những ràng buộc khi nó vận hành.

- Thiết kế giao diện: với mỗi hệ thống con, các giao diện của nó với những hệ thống con khác phải được thiết kế và tư liệu hoá.

- Thiết kế thành phần: các dịch vụ cung cấp cho các thành phần khác và các giao diện tương tác với chúng phải được thiết kế.

- Thiết kế cấu trúc dữ liệu: cấu trúc dữ liệu được sử dụng để cài đặt hệ thống phải được thiết kế một cách chi tiết và cụ thể.

- Thiết kế thuật toán: Các thuật toán được sử dụng để cung cấp các dịch vụ phải được thiết kế chi tiết và chính xác.



Hình 2.7: Mô hình chung của quy trình thiết kế

Cài đặt là quy trình chuyển đổi từ tài liệu đặc tả hệ thống thành một hệ thống thực, có thể vận hành được và phải loại bỏ các lỗi của chương trình.

Lập trình là một hành động cá nhân, không có quy trình lập trình chung. Người lập trình phải thực hiện một số kiểm thử để phát hiện ra lỗi trong chương trình và loại bỏ nó trong quy trình gỡ lỗi.

Đánh giá phần mềm

Đặt vấn đề

- Sau khi cài đặt phần mềm, chúng ta có thể chuyển giao ngay cho người sử dụng được không?

- Vai trò của việc đánh giá phần mềm là gì?

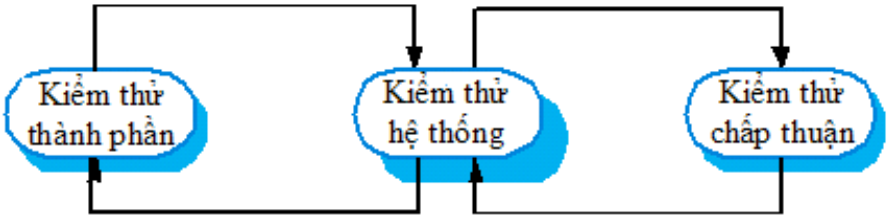
Đánh giá phần mềm hay còn gọi là thẩm tra và đánh giá (V&V - Verification and validation) được sử dụng để chỉ ra rằng hệ thống đã thực hiện theo đúng các đặc tả và thoả mãn mọi yêu cầu của khách hàng.

Đánh giá phần mềm bao gồm các công đoạn: kiểm tra, xem xét lại, và kiểm thử hệ thống. Kiểm thử hệ thống tức là cho hệ thống thực hiện trên những trường hợp có dữ liệu thật được lấy từ tài liệu đặc tả hệ thống. Quy trình kiểm thử gồm các pha sau:

- Kiểm thử thành phần (đơn vị): các thành phần được kiểm thử một cách độc lập, thành phần có thể là một chức năng hoặc một đối tượng hoặc một nhóm các thực thể gắn kết với nhau.

- Kiểm thử hệ thống: kiểm thử toàn bộ hệ thống.

- Kiểm thử chấp thuận: kiểm thử trên dữ liệu của khách hàng để kiểm tra hệ thống có đáp ứng tất cả các yêu cầu của khách hàng hay không.



Hình 2.8: Quy trình kiểm thử

Khi chuyển giao hệ thống cho khách hàng thì quy trình kiểm thử beta sẽ được thực hiện. Khách hàng sẽ thông báo các lỗi cho đội dự án. Những lỗi này sẽ được chỉnh sửa và tiếp tục kiểm thử beta hoặc chuyển giao thực sự cho khách hàng.

Cải tiến phần mềm

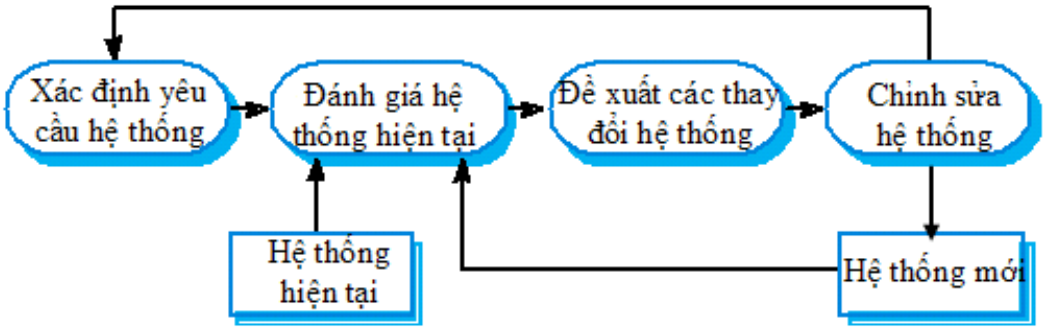
Đặt vấn đề

- Sau khi chuyển giao phần mềm cho khách hàng, thì mọi công việc đã kết thúc chưa?

- Cải tiến phần mềm để làm gì?

- Tại sao không xây dựng hệ thống mới mà lại cải tiến hệ thống cũ?

Khi các yêu cầu hệ thống thay đổi theo sự thay đổi của các yêu cầu nghiệp vụ thì phần mềm phải cải tiến và thay đổi để hỗ trợ khách hàng. Thông thường chi phí để bảo trì và cải tiến thường đắt hơn nhiều so với chi phí xây dựng phần mềm.



Hình 2.9: Cải tiến hệ thống

Quản lý dự án

Định nghĩa về quản lý dự án

Đặt vấn đề

- Để xây dựng được một hệ thống phần mềm thành công, cần thiết phải quản lý dự án

- Dựa trên hiểu biết của mình, hãy cho biết quản lý dự án là gì?

- Quản lý dự án phần mềm có giống quản lý các dự án khác không?

Quản lý dự án phần mềm là một phần quan trọng của công nghệ phần mềm. Nếu quản lý tốt thì chưa chắc dự án đã thành công, nhưng nếu quản lý tồi thì chắc chắn dự án sẽ thất bại. Dự án thất bại khi phần mềm chuyển giao chậm hơn so với kế hoạch, chi phí lớn hơn dự tính, và không thoả mãn các yêu cầu đề ra.

Quản lý dự án phần mềm có liên quan tới những hoạt động nhằm đảm bảo chuyển giao phần mềm đúng thời hạn, đúng kế hoạch và phù hợp với các yêu cầu của tổ chức phát triển phần mềm.

Quản lý dự án phần mềm có một số đặc trưng khác biệt so với các loại dự án khác:

- Sản phẩm là vô hình. Sản phẩm có khả năng thay đổi linh động.

- Công nghệ phần mềm không được thừa nhận như một quy tắc công nghệ có trạng thái chuẩn mực như các ngành công nghệ khác.

- Quy trình phát triển phần mềm không được chuẩn hoá.

- Nhiều dự án phần mềm là những dự án chỉ làm một lần.

Quản lý dự án là một yêu cầu cần thiết vì phát triển phần mềm luôn phải thoả mãn các ràng buộc về kế hoạch và chi phí đã được xác định bởi tổ chức phát triển phần mềm. Người quản lý dự án phải chịu trách nhiệm lập kế hoạch và theo dõi quá trình thực hiện dự án.

Các hoạt động quản lý

Đặt vấn đề

- Những công việc phải làm trong Quản lý dự án là gì?

- Hãy thảo luận về những công việc đó. Cho biết công việc nào là quan trọng nhất.

Các hoạt động quản lý dự án bao gồm:

- Viết kế hoạch dự kiến: Đây là một công việc khá phức tạp. Nó mô tả mục tiêu của dự án, phương pháp thực hiện, ước lượng thời gian và chi phí …

- Lập kế hoạch dự án: liên quan đến việc xác định các hành động, các mốc thời gian và các sản phẩm được tạo ra.

- Tính chi phí dự án

- Điều hành và xem xét lại dự án: người quản lý phải giám sát quy trình thực hiện dự án, so sánh quy trình và chi phí thực tế với kế hoạch đã định. Nếu điều hành tốt, người quản lý dự án có thể phát hiện và khắc phục được những rủi ro tiềm tàng.

- Lựa chọn và đánh giá cá nhân. Việc lựa chọn nhân viên thích hợp cho một dự án là rất khó khăn. Khi lựa chọn đội dự án, người quản lý dự án có thể gặp phải một số vấn đề sau: ngân sách của dự án không đủ để trả cho những nhân viên có mức lương cao, không có được những nhân viên có kinh nghiệm và trình độ thích hợp, tổ chức muốn chỉ định một số nhân viên mới tham gia vào dự án ...

- Viết báo cáo và trình bày.

Tuy nhiên, ngày nay chúng ta có rất nhiều kỹ thuật và công cụ được sử dụng để hỗ trợ cho việc quản lý dự án phần mềm.

Lập kế hoạch dự án

Mục tiêu

- Hiểu rõ tầm quan trọng của việc lập kế hoạch dự án

- Phải biết rằng ứng với mỗi hoạt động trong quá trình phát triển phần mềm, chúng ta sẽ phải có một bản kế hoạch riêng.

- Nắm được cấu trúc của một bản kế hoạch dự án phát triển hệ thống phần mềm.

Lập kế hoạch dự án có thể là hoạt động tốn nhiều thời gian nhất trong quá trình quản lý dự án. Nó liệt kê các hành động từ pha khởi tạo cho đến khi đưa ra được hệ thống. Kế hoạch phải được theo dõi thường xuyên, nhất là khi có những thông tin hoặc những yêu cầu mới xuất hiện.

Trong quá trình thực hiện dự án, chúng ta có nhiều loại kế hoạch được xây dựng để hỗ trợ cho kế hoạch chính của dự án phần mềm như: kế hoạch chất lượng, kế hoạch thẩm tra, kế hoạch quản lý cấu hình, kế hoạch bảo trì, kế hoạch phát triển nhân sự …

Cấu trúc của bản kế hoạch dự án gồm:

- Phần giới thiệu: mô tả các mục tiêu của dự án và các ràng buộc gây ảnh hưởng tới việc quản lý dự án.

- Tổ chức dự án: mô tả cách tổ chức của đội dự án, bao gồm những ai và những nhiệm vụ gì.

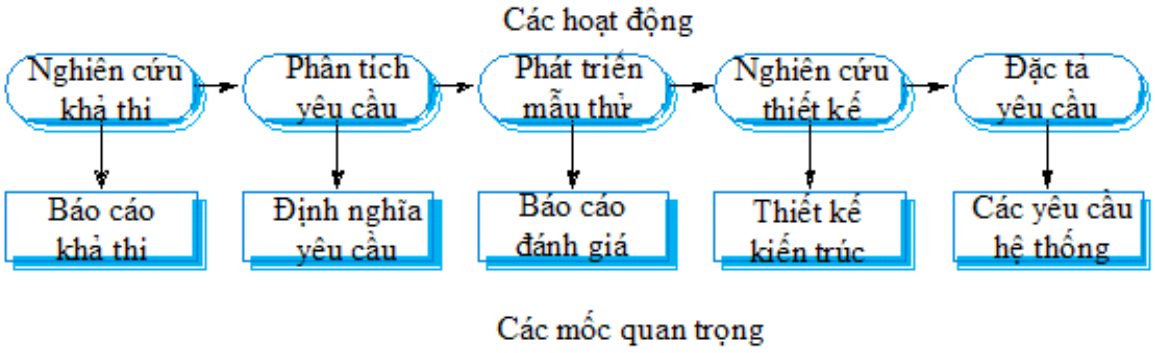
- Phân tích rủi ro: mô tả những rủi ro có thể xảy ra, dự báo khi nào chúng xảy ra và đề xuất chiến lược giảm rủi ro.

- Các yêu cầu về tài nguyên phần cứng và phần mềm: xác định những phần cứng và phần mềm nào cần thiết cho quá trình thực hiện dự án.

- Bảng thống kế công việc: xác định các công việc, từng mốc thời gian và kết quả của từng công việc.

- Lịch biểu của dự án: lịch biểu cho thấy sự phụ thuộc giữa các hành động, thời gian ước tính để đạt tới mốc và phân công công việc cho từng người. Mốc là điểm cuối của một hành động trong quy trình. Ví dụ, trong mô hình thác nước cho phép ta định nghĩa các mốc của tiến trình một cách rõ ràng.

- Các kỹ thuật điều hành và báo cáo



Hình 3.1: Các mốc trong quy trình xác định yêu cầu

Lịch biểu của dự án

Mục tiêu

- Phải hiểu rõ lập lịch biểu dự án là làm gì?

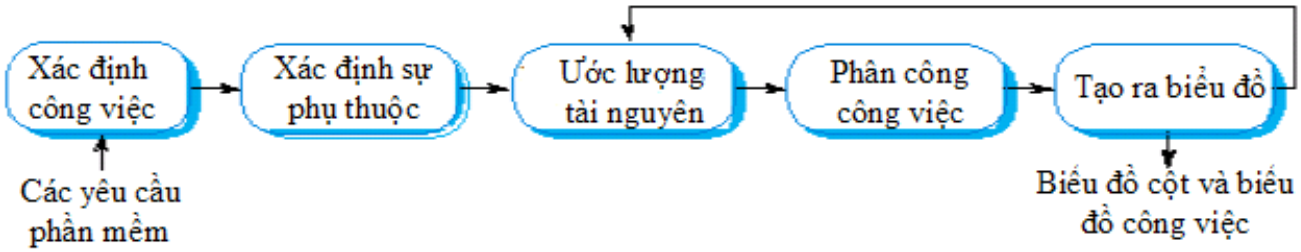
- Nắm được một số quy tắc và các gợi ý khi lập lịch biểu

- Có khả năng áp dụng biểu đồ cột, sơ đồ mạng để xây dựng lịch biểu

- Có thể sử dụng một số công cụ hỗ trợ lập lịch biểu như: M.Excel, M.Project …

Lập lịch biểu dự án là một trong những công việc khó khăn nhất đối với người quản lý dự án. Người quản lý phải chia dự án thành nhiều nhiệm vụ, ước lượng thời gian và tài nguyên cần thiết để hoàn thành từng nhiệm vụ.

Khi lập lịch biểu, người quản lý nên tổ chức các công việc song song để sử dụng tối ưu lực lượng lao động và tối thiểu hoá sự phụ thuộc lẫn nhau giữa các nhiệm vụ để tránh sự chậm trễ khi một nhiệm vụ phải đợi nhiệm vụ khác hoàn thành.



Hình 3.2: Quy trình lập lịch biểu dự án

Chất lượng của lịch biểu phụ thuộc vào hiểu biết và kinh nghiệm của người quản lý. Tuy nhiên, khi lập lịch biểu chúng ta phải chú ý tới các vấn đề sau:

- Việc ước lượng mức độ khó của một vấn đề nào đó và xác định chi phí để giải quyết nó là rất khó khăn.

- Khả năng sản xuất không tương ứng với số lượng người làm việc trong một nhiệm vụ.

- Bổ sung thêm người vào dự án sẽ làm cho nó chậm hơn vì giao tiếp trong dự án trở lên quá tải.

- Những sự việc xảy ra ngoài mong đợi.

Chúng ta sử dụng các ký pháp đồ hoạ để minh hoạ cho lịch biểu của dự án. Sử dụng biểu đồ giúp ta thấy rõ cách chia dự án thành nhiều nhiệm vụ. Các nhiệm vụ không nên quá nhỏ, chúng nên được thực hiện trong vòng một hoặc hai tuần.

Ví dụ

Giả sử có một loạt các hoạt động T_i, thời gian thực hiện từng hoạt động và sự phụ thuộc lẫn nhau giữa các hoạt động được liệt kê như bảng dưới đây. Hãy thực hiện các yêu cầu sau:

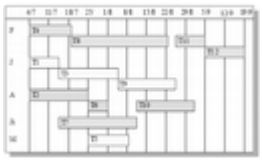
1.Xây dựng mạng các hoạt động

2. Xây dựng biểu đồ nhằm biểu diễn các hoạt động theo dòng thời gian

3. Biểu đồ phân công công việc

Hoạt động	Thời gian (ngày)	Phụ thuộc
T1	8	
T2	15	
T3	15	T1 (M1)
T4	10	
T5	10	T2, T4 (M2)
T6	5	T1, T2 (M3)
T7	20	T1 (M1)
T8	25	T4 (M5)
T9	15	T3, T6 (M4)
T10	15	T5, T7 (M7)
T11	7	T9 (M6)
T12	10	T11 (M8)

Kết quả thực hiện ví dụ

 Mạng hoạt động	 Theo thời gian	 Theo công việc

Quản lý rủi ro

Mục tiêu

- Phải hiểu được rủi ro luôn luôn có thể xảy ra trong dự án và không thể lường trước được

- Nắm được một số loại rủi ro thường gặp

- Biết được các hoạt động cần thực hiện trong quản lý rủi ro.

Quản lý rủi ro liên quan tới việc xác định rủi ro và lập ra các kế hoạch để tối thiểu hoá ảnh hưởng của chúng tới dự án.

Sau đây là một số loại rủi ro thường gặp trong quá trình phát triển hệ thống phần mềm:

- Rủi ro của dự án có ảnh hưởng tới lịch biểu và tài nguyên của dự án

- Rủi ro của sản phẩm ảnh hưởng tới chất lượng hoặc hiệu năng của phần mềm sẽ được xây dựng.

- Rủi ro thương mại sẽ ảnh hưởng tới tổ chức xây dựng phần mềm.

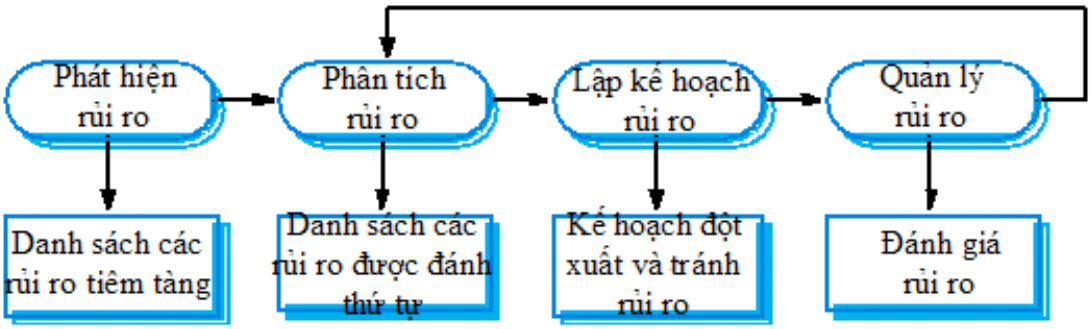
Để quản lý rủi ro, chúng ta cần phải thực hiện các hoạt động sau:

- Phát hiện rủi ro: Phát hiện các loại rủi ro có liên quan đến: công nghệ, con người, tổ chức, các yêu cầu, ước lượng.

- Phân tích rủi ro: Đánh giá các khả năng xảy ra rủi ro và tính nghiêm trọng của nó nếu nó xảy ra.

- Lập kế hoạch rủi ro: Xem xét từng rủi ro và phát triển chiến lược để quản lý nó. Bao gồm các chiến lược như: phòng tránh - giảm khả năng xảy ra rủi ro, tối thiểu hoá - giảm ảnh hưởng của rủi ro, kế hoạch bất ngờ - kế hoạch này để dành cho khi rủi ro xảy ra.

- Kiểm soát rủi ro: Đánh giá từng rủi ro đã được xác định một cách thường xuyên để xác định khả năng nó có thể xảy ra hay không và cũng đồng thời đánh giá mức độ ảnh hưởng của nó. Những rủi ro chính nên được thảo luận tại các cuộc họp quản lý tiến trình.



Hình 3.3: Quy trình quản lý rủi ro

Một số yêu cầu về nhập môn công nghệ phần mềm

Yêu cầu hệ thống

Giới thiệu

Các yêu cầu của hệ thống phần mềm thường được chia thành ba loại: yêu cầu chức năng, yêu cầu phi chức năng và yêu cầu miền ứng dụng. Tuy nhiên, trong thực tế chúng ta rất khó phân biết ba loại yêu cầu này một cách rõ ràng.

Mục tiêu

- Tìm hiểu về các yêu cầu hệ thống và đặc điểm của chúng

- Phương pháp xác định các yêu cầu hệ thống

- Nắm rõ các kỹ thuật đặc tả yêu cầu hệ thống và biết cách áp dụng những kỹ thuật này một cách thích hợp.

Ví dụ

Trong chương này, chúng ta sẽ sử dụng một ví dụ về hệ thống thư viện để xác định các loại yêu cầu.

Hệ thống thư viện (LIBSYS) cung cấp một giao diện đơn giản để lưu CSDL về các bài báo trên các thư viện khác nhau. Người sử dụng có thể tìm kiếm, download và in những tài liệu này.

Yêu cầu chức năng

Đặt vấn đề

- Để xây dựng một hệ thống có thể dùng được thực sự, trước hết nó phải đạt được yêu cầu gì?

- Yêu cầu chức năng có phải là quan trọng nhất không?

- Nếu ta không xác định đầy đủ, rõ ràng các yêu cầu chức năng thì sẽ xảy ra chuyện gì?

Yêu cầu chức năng mô tả hệ thống sẽ làm gì. Nó mô tả các chức năng hoặc các dịch vụ của hệ thống một cách chi tiết.

Đặc điểm của yêu cầu chức năng:

- Tính mập mờ, không rõ ràng của các yêu cầu: Vấn đề này xảy ra khi các yêu cầu không được xác định một cách cẩn thận. Các yêu cầu mập mờ có thể được người xây dựng và người sử dụng hiểu theo nhiều cách khác nhau.

- Tính hoàn thiện và nhất quán: Về nguyên tắc, yêu cầu phải chứa tất cả các mô tả chi tiết và không có sự xung đột hoặc đối ngược giữa các yêu cầu. Tuy nhiên, trong thực tế rất khó có thể đạt được điều này.

Xác định các yêu cầu chức năng của LYBSYS

- Người sử dụng có thể tìm kiếm tất cả CSDL hoặc một tập con của CSDL.

- Hệ thống sẽ cung cấp những giao diện thích hợp để người sử dụng đọc tài liệu.

- Tất cả những hoá đơn mà người sử dụng đăng ký để in sao tài liệu có một mã duy nhất.

Yêu cầu phi chức năng

Đặt vấn đề

- Nếu hệ thống chỉ thoả mãn những yêu cầu chức năng thì đã đủ chưa?

- Ví dụ hệ thống không tiện dụng đối với người sử dụng thì sao?

- Yêu cầu phi chức năng bao gồm những vấn đề gì?

Yêu cầu phi chức năng không đề cập trực tiếp tới các chức năng cụ thể của hệ thống. Yêu cầu phi chức năng thường định nghĩa các thuộc tính như: độ tin cậy, thời gian đáp ứng, các yêu cầu về lưu trữ …và các ràng buộc của hệ thống như: khả năng của thiết bị vào/ra, giao diện …

Một số yêu cầu phi chức năng còn có liên quan đến quy trình xây dựng hệ thống. Ví dụ: các chuẩn được sử dụng, các công cụ CASE, ngôn ngữ lập trình …

Các yêu cầu phi chức năng có thể là hạn chế hơn những yêu cầu chức năng. Nhưng nếu nó không được thoả mãn thì hệ thống sẽ không sử dụng được.

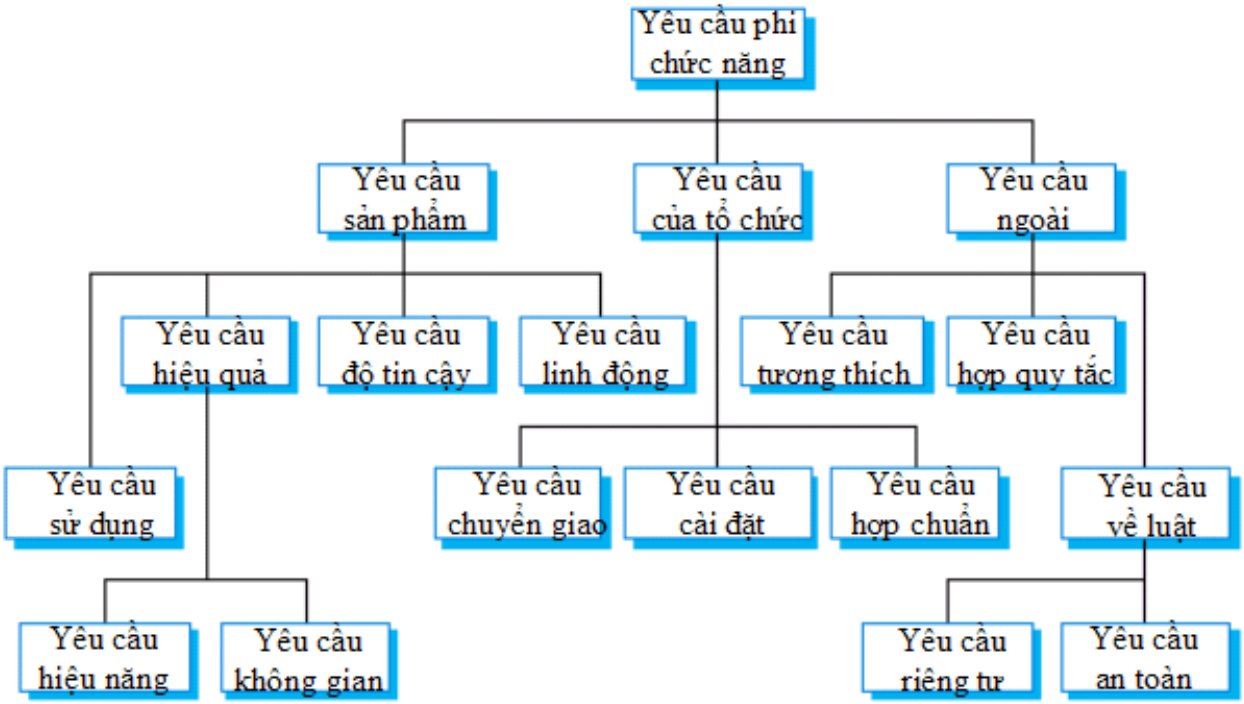
Các yêu cầu phi chức năng xuất hiện là do yêu cầu của người sử dụng, ràng buộc về ngân sách, các chính sách của tổ chức sử dụng hệ thống, yêu cầu tương thích giữa phần

cứng và phần mềm và các tác nhân ngoài khác. Do đó, chúng ta có thể phân loại các yêu cầu phi chức năng như sau:

- Các yêu cầu về sản phẩm xác định ứng xử của sản phẩm như: hiệu năng, khả năng sử dụng, độ tin cậy … của sản phẩm

- Các yêu cầu về tổ chức: các yêu cầu này được lấy từ những chính sách và quy tắc của khách hàng hoặc tổ chức sử dụng hệ thống.

- Các yêu cầu ngoài: được xác định từ các tác nhân ngoài của hệ thống.



Hình 4.1: Phân loại các yêu cầu phi chức năng

Xác định các yêu cầu phi chức năng của LIBSYS

- Yêu cầu về sản phẩm: LIBSYS phải được cài đặt bằng HTML mà không có frame hoặc Java applets.

- Yêu cầu về mặt tổ chức: Quy trình xây dựng hệ thống và các tài liệu chuyển giao phải thoả mãn các quy tắc đã được định nghĩa trong XYZCo-SP-STAN-95.

- Yêu cầu ngoài: Hệ thống không được để lộ các thông tin cá nhân của khách hàng.

Nói chung, chúng ta rất khó xác định chính xác và rất khó thẩm tra những yêu cầu phi chức năng mập mờ. Do đó, trong tài liệu đặc tả yêu cầu, người ta thường bổ sung các mục tiêu. Mục tiêu rất hữu ích đối với người phát triển hệ thống khi nó truyền tải được những mong muốn của người sử dụng hệ thống. Còn với những yêu cầu phi chức năng có thể thẩm định được là những yêu cầu có thể kiểm thử một cách khách quan.

Tuy nhiên, trong nhiều trường hợp thường xảy ra xung đột giữa các yêu cầu phi chức năng đối với những hệ thống phức tạp.

Các mục tiêu và yêu cầu phi chức năng có thể thẩm định được của LIBSYS

- Mục tiêu của hệ thống là dễ sử dụng đối với những người sử dụng có kinh nghiệm và được tổ chức để sao cho tối thiểu hoá được lỗi.

- Các yêu cầu phi chức năng có thể thẩm định được: Những người sử dụng có kinh nghiệm có thể sử dụng được tất cả các chức năng của hệ thống chỉ sau hai tiếng tập huấn. Sau khoá huấn luyện này, số lỗi chương trình gây ra bởi người sử dụng là không quá hai lỗi một ngày.

Yêu cầu miền ứng dụng

Đặt vấn đề

- Yêu cầu đối với đội phát triển hệ thống?

- Các yêu cầu về lĩnh vực ứng dụng của hệ thống thì thuộc vào loại nào trong hai loại trên đã trình bày?

Yêu cầu miền ứng dụng được xác định từ miền ứng dụng của hệ thống và phản ánh các thuộc tính và ràng buộc của miền ứng dụng. Nó có thể là yêu cầu chức năng hoặc phi chức năng.

Nếu yêu cầu miền ứng dụng không được thoả mãn thì có thể hệ thống sẽ không làm việc được.

Sau đây là một số vấn đề liên quan đến yêu cầu miền ứng dụng:

- Khả năng có thể hiểu được: các yêu cầu được biểu diễn dưới ngôn ngữ của lĩnh vực ứng dụng.

- Ẩn ý: Các chuyên gia có hiểu biết về lĩnh vực của họ nhưng họ không biết cách xây dựng những yêu cầu miền ứng dụng một cách rõ ràng, mang tính kỹ thuật.

Yêu cầu về miền ứng dụng của LIBSYS

- Giao diện người dùng chuẩn cho tất cả các CSDL đều dựa trên chuẩn Z39.50.

- Vì vấn đề bản quyền nên một số tài liệu phải xoá ngay khi vừa chuyển đến.

- Phụ thuộc vào yêu cầu của người sử dụng, những tài liệu đó có thể được in ngay trên server và chuyển đến cho người sử dụng hoặc gửi đến cho máy in mạng.

Một số kỹ thuật đặc tả yêu cầu hệ thống

Giới thiệu

Nói chung, ngôn ngữ tự nhiên thường được sử dụng để viết đặc tả yêu cầu hệ thống cũng như yêu cầu của người sử dụng. Tuy nhiên, yêu cầu hệ thống thường chi tiết hơn yêu cầu của người sử dụng nên đặc tả bằng ngôn ngữ tự nhiên thường gặp một số vấn đề sau:

- Không rõ ràng: Người đọc và người viết yêu cầu phải giải thích các từ theo cùng một nghĩa. Ngôn ngữ tự nhiên có bản chất là mập mờ nên để đạt được yêu cầu trên là rất khó khăn.

- Quá mềm dẻo: với cùng một vấn đề nhưng có nhiều cách khác nhau để đặc tả.

- Thiếu khả năng mô-đun hoá: cấu trúc của ngôn ngữ tự nhiên không tương xứng với cấu trúc của các yêu cầu hệ thống.

Vì những lý do này mà đặc tả bằng ngôn ngữ tự nhiên thường gây khó hiểu. Do đó, chúng ta có thể sử dụng một số phương pháp được trình bày trong các phần sau để đặc tả yêu cầu.

Đặc tả bằng ngôn ngữ hướng cấu trúc

Sử dụng ngôn ngữ hướng cấu trúc sẽ yêu cầu người viết đặc tả tuân theo những mẫu được định nghĩa trước. Tất cả các yêu cầu đều được viết theo chuẩn và các thuật ngữ được sử dụng có thể bị hạn chế.

Ưu điểm của phương pháp này là đạt được mức độ diễn tả cao nhất của ngôn ngữ tự nhiên nhưng mức độ đồng nhất lại bị lạm dụng trong các đặc tả.

Đặc tả dựa biểu mẫu (Form-based)

Đặc tả dựa biểu mẫu định nghĩa các chức năng hoặc thực thể, mô tả đầu vào và nơi xuất phát của nó, mô tả đầu ra và nơi nó sẽ đến. Đặc tả dựa biểu mẫu chỉ rõ những thực thể cần thiết, các điều kiện trước và sau (nếu thích hợp), các ảnh hưởng của chức năng.

Biểu đồ trình tự

Biểu đồ trình tự biểu diễn trình tự các sự kiện xảy ra khi người sử dụng tương tác với hệ thống. Nếu ta đọc biểu đồ này từ đầu đến cuối thì ta sẽ thấy được thứ tự của các hành động được thực hiện.

Yêu cầu của người sử dụng

Đặt vấn đề

- Yêu cầu của người sử dụng có được coi giống như yêu cầu hệ thống hay không?

- Yêu cầu của người sử dụng là chức năng hay phi chức năng hay cả hai?

Yêu cầu của người sử dụng nên mô tả những yêu cầu chức năng và phi chức năng để người sử dụng có thể hiểu được chúng mà không cần phải có những kiến thức về công nghệ một cách chi tiết.

Yêu cầu của người sử dụng được định nghĩa bằng cách sử dụng ngôn ngữ tự nhiên, bảng hoặc biểu đồ đơn giản. Tuy nhiên, chúng ta sẽ gặp phải một số khó khăn khi sử dụng ngôn ngữ tự nhiên:

- Không rõ ràng: Tính chính xác rất khó đạt được nếu tài liệu khó đọc.

- Yêu cầu lộn xộn: các yêu cầu chức năng và phi chức năng không rõ ràng.

- Lẫn lộn giữa các yêu cầu: các yêu cầu khác nhau có thể được diễn tả cùng với nhau.

Do đó, để viết yêu cầu của người sử dụng ta nên áp dụng một số quy tắc sau:

- Đưa ra một định dạng chuẩn và áp dụng nó cho tất cả các yêu cầu.

- Bắt buộc sử dụng ngôn ngữ một cách thống nhất

- Đánh dấu những phần quan trọng trong các yêu cầu.

- Tránh sử dụng những từ ngữ mang tính chuyên môn, kỹ thuật.

Tài liệu đặc tả yêu cầu

Tài liệu

Đặt vấn đề

- Sau khi đã xác định các yêu cầu hệ thống và yêu cầu của người sử dụng hệ thống thì tiếp tục làm gì?

- Cấu trúc của một tài liệu đặc tả yêu cầu hệ thống phải bao gồm những nội dung gì?

Tài liệu đặc tả yêu cầu là những yêu cầu chính thức về những gì cần phải thực hiện bởi đội phát triển hệ thống.

Tài liệu đặc tả yêu cầu nên bao gồm cả các định nghĩa về yêu cầu của người sử dụng và đặc tả yêu cầu hệ thống.

Tài liệu đặc tả yêu cầu không phải là tài liệu thiết kế hệ thống. Nó chỉ thiết lập những gì hệ thống phải làm, chứ không phải mô tả rõ làm như thế nào.

Tài liệu đặc tả yêu cầu dựa theo chuẩn IEEE

1. Giới thiệu

1.1. Mục đích của tài liệu yêu cầu

1.2. Phạm vi của sản phẩm

1.3. Các định nghĩa, từ viết tắt

1.4. Các tham chiếu

1.5. Tổng quan về tài liệu yêu cầu

2. Mô tả chung

2.1. Giới thiệu chung về sản phẩm

2.2. Các chức năng của sản phẩm

2.3. Đặc điểm của người sử dụng

2.4. Các ràng buộc

2.5. Giả thiết và các phụ thuộc

3. Đặc tả yêu cầu: bao gồm các yêu cầu chức năng, phi chức năng, miền ứng dụng và giao diện.

4. Phụ lục

5. Chỉ mục

Phân tích khả thi

Mục tiêu

- Xác định rõ vai trò của phân tích khả thi

- Để phân tích khả thi, chúng ta phải thực hiện những công việc gì

- Nắm được một số gợi ý khi phân tích khả thi

Đối với tất cả các hệ thống mới, quy trình xác định yêu cầu thường bắt đầu bằng việc phân tích khả thi. Thông tin đầu vào để phân tích khả thi là các yêu cầu nghiệp vụ, mô tả sơ bộ về hệ thống, cách thức hệ thống hỗ trợ các yêu cầu nghiệp vụ. Kết quả của việc phân tích khả thi là một báo cáo để quyết định có nên xây dựng hệ thống đề xuất hay không.

Phân tích khả thi thường tập trung vào:

- Xác định hệ thống có đóng góp vào mục tiêu của tổ chức hay không

- Kiểm tra xem hệ thống có thể được xây dựng bằng cách sử dụng công nghệ hiện tại và ngân sách cho phép.

- Kiểm tra xem liệu hệ thống có được tích hợp với các hệ thống khác đang sử dụng hay không.

Thực hiện phân tích khả thi dựa trên việc đánh giá thông tin, lựa chọn thông tin và viết báo cáo.

Những câu hỏi thường được đặt ra để phân tích khả thi:

- Nếu hệ thống không được cài đặt thì sao?

- Vấn đề xử lý hiện tại như thế nào?

- Hệ thống đề xuất giúp đỡ được gì?

- Vấn đề về tích hợp là gì?

- Công nghệ mới cần dùng là gì? Cần có những kỹ năng gì?

- Những lợi ích mà hệ thống mang lại?

Phát hiện và phân tích yêu cầu

Mục tiêu

- Nhiệm vụ của việc phát hiện và phân tích yêu cầu là gì?

- Hiểu rõ mô hình xoắn ốc trong quy trình phát hiện và phân tích yêu cầu

- Các kỹ thuật được sử dụng để phát hiện và phân tích yêu cầu

Trong pha phát hiện và phân tích yêu cầu, nhân viên kỹ thuật và khách hàng cùng hợp tác để xác định miền ứng dụng, các dịch vụ mà hệ thống cung cấp, hiệu năng của hệ thống, các ràng buộc vận hành của hệ thống…

Ở đây, chúng ta có một khái niệm mới là stakeholder. Stakeholder là những người tham dự vào dự án xây dựng hệ thống: người sử dụng cuối, người quản lý, kỹ sư, chuyên gia lĩnh vực, … Ví dụ, trong hệ thống ATM gồm các Stakeholder sau: khách hàng của ngân hàng, đại diện của các ngân hàng khác, người quản lý ngân hầng, nhân viên ngân hàng, quản trị CSDL, quản lý bảo mật, phòng marketing, kỹ sư bảo trì phần cứng và phần mềm, người điều hành ngân hàng.

Tuy nhiên, việc phát hiện và tìm hiểu yêu cầu của stakeholder, chúng ta thường gặp khó khăn vì những nguyên nhân sau:

- Stakeholder không biết những gì mà họ thật sự mong muốn.

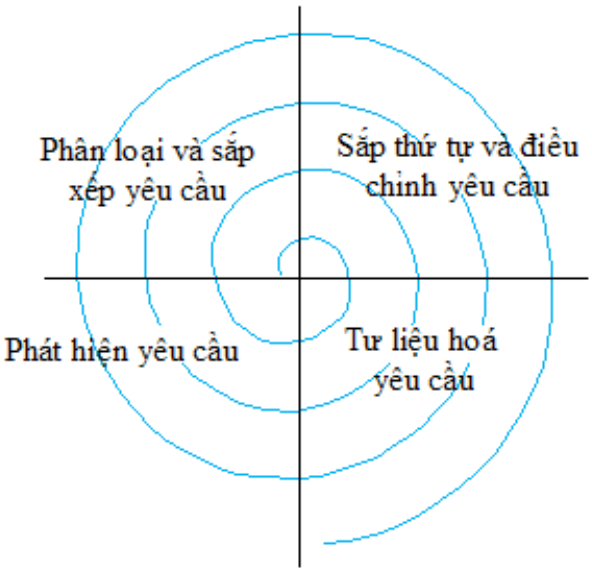
- Stakeholder mô tả các yêu cầu theo thuật ngữ của họ.

- Những stakeholder khác nhau có thể có các yêu cầu xung đột nhau

- Những yếu tố tổ chức và quyền lực có thể ảnh hưởng tới các yêu cầu hệ thống.

- Các yêu cầu có thể thay đổi trong suốt quá trình phân tích. Những stakeholder mới có thể xuất hiện và môi trường nghiệp vụ có thể thay đổi.

Do đó, người ta thường sử dụng mô hình xoắn ốc trong quy trình phát hiện và phân tích yêu cầu.



Hình 5.3: Quy trình phát hiện và phân tích yêu cầu

Trong quy trình này bao gồm các hoạt động sau:

- Phát hiện yêu cầu: tiếp xúc với các stakeholder để phát hiện ra các yêu cầu của họ. Các yêu cầu miền ứng dụng cũng được phát hiện ở bước này.

- Phân loại và sắp xếp yêu cầu: nhóm các yêu cầu có liên quan lẫn nhau và tổ chức chúng thành những nhóm gắn kết với nhau.

- Sắp thứ tự ưu tiên và điều chỉnh các yêu cầu xung đột: khi có nhiều stakeholder thì các yêu cầu của họ càng có nhiều xung đột. Hoạt động này nhằm đánh thứ tự ưu tiên của các yêu cầu, phát hiện và giải quyết xung đột giữa các yêu cầu.

- Tư liệu hóa yêu cầu: yêu cầu được tư liệu hoá và là đầu vào của vòng kế tiếp trong mô hình xoắn ốc.

Phát hiện yêu cầu là quy trình thu thập những thông tin về hệ thống được đề xuất và hệ thống đang tồn tại để xác định các yêu cầu hệ thống và yêu cầu của người sử dụng. Ta có thể lấy thông tin này từ các tư liệu, stakeholder, và bản đặc tả của những hệ thống tương tự. Chúng ta giao tiếp với stakeholder thông qua phỏng vấn hoặc quan sát và có thể sử dụng kịch bản và mẫu thử để giúp phát hiện yêu cầu.

Đánh giá yêu cầu

Mục tiêu

- Hiểu rõ tầm quan trọng của việc đánh giá các yêu cầu

- Nắm được phải kiểm tra các yêu cầu ở những khía cạnh nào

- Hiểu được một số kỹ thuật được sử dụng để đánh giá yêu cầu

Đánh giá yêu cầu có liên quan đến việc giải thích các yêu cầu đã được định nghĩa trong hệ thống. Vì chi phí cho việc giải quyết các lỗi có liên quan tới yêu cầu sẽ rất cao cho nên việc đánh giá yêu cầu là vô cùng quan trọng.

Trong quá trình đánh giá yêu cầu, chúng ta phải kiểm tra các yêu cầu ở những khía cạnh sau:

- Hợp lệ: Hệ thống có cung cấp các chức năng mà hỗ trợ tốt nhất cho các yêu cầu của người sử dụng hay không?

- Nhất quán: có yêu cầu nào xung đột nhau hay không?

- Hoàn thiện: tất cả các yêu cầu của khách hàng đã được xác định đầy đủ chưa?

- Hiện thực: các yêu cầu có thể được cài đặt với một ngân sách và công nghệ cho trước?

- Xác thực: các yêu cầu có thể được kiểm tra hay không?

Các kỹ thuật đánh giá yêu cầu sau đây có thể được sử dụng đơn lẻ hoặc hỗn hợp:

- Xem xét lại các yêu cầu: phân tích các yêu cầu một cách hệ thống.

- Mẫu thử: Sử dụng các mô hình hệ thống để kiểm tra các yêu cầu

- Tạo ra các trường hợp kiểm thử

Lập kế hoạch quản lý yêu cầu

Kế hoạch

Mục tiêu

- Nắm rõ quy trình quản lý yêu cầu

- Khi yêu cầu thay đổi, cần phải biết thực hiện những thao tác gì.

Quản lý yêu cầu là quy trình quản lý sự thay đổi của các yêu cầu trong quá trình phát hiện yêu cầu và phát triển hệ thống.

Các yêu cầu thường không đầy đủ và không đồng nhất. Đó là do một số nguyên nhân sau:

- Những yêu cầu mới xuất hiện trong quy trình khi các yêu cầu nghiệp vụ thay đổi và khi chúng ta có hiểu biết sâu hơn về hệ thống sẽ xây dựng.

- Ở các khung nhìn khác nhau sẽ có các yêu cầu khác nhau và do đó thường xuất hiện các mâu thuẫn.

- Thứ tự ưu tiên từ các khung nhìn khác nhau cung thay đổi trong suốt quá trình phát triển hệ thống.

- Môi trường nghiệp vụ và môi trương kỹ thuật của hệ thống cũng thay đổi trong có trình xây dựng.

Các yêu cầu lâu dài là những yêu cầu ổn định kế thừa từ những hành động chính của khách hàng. Và nó có thể kế thừa từ nhiều mô hình miền ứng dụng khác.

Các yêu cầu thay đổi là những yêu cầu dễ bị thay đổi trong quá trình xây dựng hoặc khi hệ thống được đưa vào sử dụng.

Quy trình lập kế hoạch quản lý yêu cầu:

- Xác định yêu cầu: cách xác định từng yêu cầu

- Quản lý thay đổi: xác định các hoạt động tiếp theo khi yêu cầu thay đổi.

- Các chính sách tìm vết: lượng thông tin về mối quan hệ giữa các yêu cầu cần phải được lưu giữ. Thông thường, nó đề cập tới quan hệ giữa tài nguyên và bản thiết kế hệ thống.

* Tìm vết nguồn: là những liên kết từ các yêu cầu tới stakeholder đưa ra những yêu cầu đó.

* Tìm vết yêu cầu: là mối liên hệ giữa các yêu cầu độc lập nhau.

* Tìm vết thiết kế: là những liên kết từ yêu cầu cho tới thiết kế.

- Hỗ trợ CASE tool: sử dụng các công cụ để hỗ trợ quản lý yêu cầu thay đổi. CASE tool thường hỗ trợ những chức năng như:

* Lưu trữ yêu cầu: các yêu cầu được quản lý một cách bảo mật và được lưu trong kho dữ liệu.

* Quản lý thay đổi: quy trình quản lý thay đổi là quy trình luồng công việc mà các trạng thái có thể được định nghĩa và luồng thông tin giữa các trạng thái là tự động.

* Quản lý vết - tự động tìm kiếm mối liên kết giữa các yêu cầu.

Chúng ta nên áp dụng tất cả các khả năng thay đổi có thể cho tất cả các yêu cầu. Các pha chính của hoạt động này bao gồm:

- Phân tích vấn đề và đặc tả thay đổi: thảo luận về các vấn đề yêu cầu và những thay đổi có thể xảy ra.

- Phân tích thay đổi và chi phí: Đánh giá ảnh hưởng của sự thay đổi trên các yêu cầu khác.

- Cài đặt thay đổi: Điều chỉnh tài liệu của các yêu cầu và những tài liệu khác để phản ánh sự thay đổi đó.

Các mô hình Quản lí

Các mô hình hệ thống

Giới thiệu

Các yêu cầu của người sử dụng thường được viết bằng ngôn ngữ tự nhiên để những người không có kiến thức về mặt kỹ thuật có thể hiểu được nó. Tuy nhiên, những yêu cầu hệ thống chi tiết phải được mô hình hoá. Mô hình hoá hệ thống giúp cho người phân tích hiểu rõ các chức năng của hệ thống.

Ta có thể sử dụng các mô hình khác nhau để biểu diễn hệ thống từ nhiều khía cạnh khác nhau.

Trong chương này, chúng ta sẽ tìm hiểu về một số mô hình hoá hệ thống.

Mục tiêu

- Hiểu được mô hình hoá hệ thống là gì? Và tại sao phải mô hình hoá hệ thống.

- Phân biệt được các mô hình hệ thống.

- Có khả năng lựa chọn và ứng dụng các mô hình hệ thống vào từng trường hợp cụ thể

Mô hình ngữ cảnh

Đặt vấn đề

- Khi xem xét một vấn đề, bao giờ chúng ta cũng muốn có cái nhìn tổng thể về vấn đề đó.

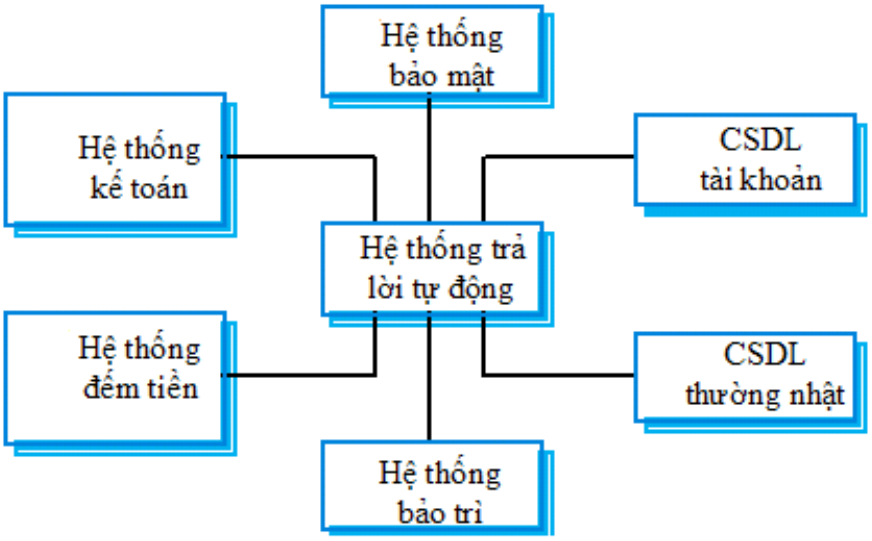
- Mô hình ngữ cảnh cho thấy những thành phần cốt lõi của hệ thống.

Trong quá trình phát hiện và phân tích yêu cầu, chúng ta nên xác định phạm vi hệ thống, tức là phân biệt cái gì là hệ thống và cái gì là môi trường của hệ thống. Điều này sẽ giúp giảm chi phí và thời gian phân tích.

Khi đã xác định phạm vi của hệ thống, hoạt động tiếp theo của quy trình phân tích là định nghĩa ngữ cảnh của hệ thống và sự phụ thuộc giữa hệ thống với môi trường của nó.

Thông thường, mô hình kiến trúc đơn giản của hệ thống sẽ được tạo ra trong bước này.

Ví dụ: mô hình ngữ cảnh của hệ thống ATM



Hình 6.1: Ngữ cảnh của hệ thống ATM

Mô hình kiến trúc mô tả môi trường của hệ thống, nhưng không chỉ ra quan hệ giữa các hệ thống khác nhau trong một môi trường. Vì vậy, người ta thường sử dụng thêm mô hình tiến trình hoặc mô hình luồng dữ liêuh để bổ trợ cho nó.

Mô hình tiến trình biểu diễn tất cả các tiến trình được hệ thống hỗ trợ. Mô hình luồng dữ liệu có thể được sử dụng để biểu diễn các tiến trình và luồng thông tin đi từ tiến trình này tới tiến trình khác.

Mô hình ứng xử và máy hệ thống

Mô hình ứng xử

Giới thiệu

Mô hình ứng xử được sử dụng để mô tả toàn bộ ứng xử của hệ thống. Có hai kiểu mô hình ứng xử là:

- Mô hình luồng dữ liệu: biểu diễn cách xử lý dữ liệu trong hệ thống và

- Mô hình máy trạng thái: biểu diễn cách đáp ứng của hệ thống với các sự kiện xảy ra.

Hai mô hình này biểu diễn những góc nhìn khác nhau, nhưng cả hai đều cần thiết để mô tả ứng xử của hệ thống.

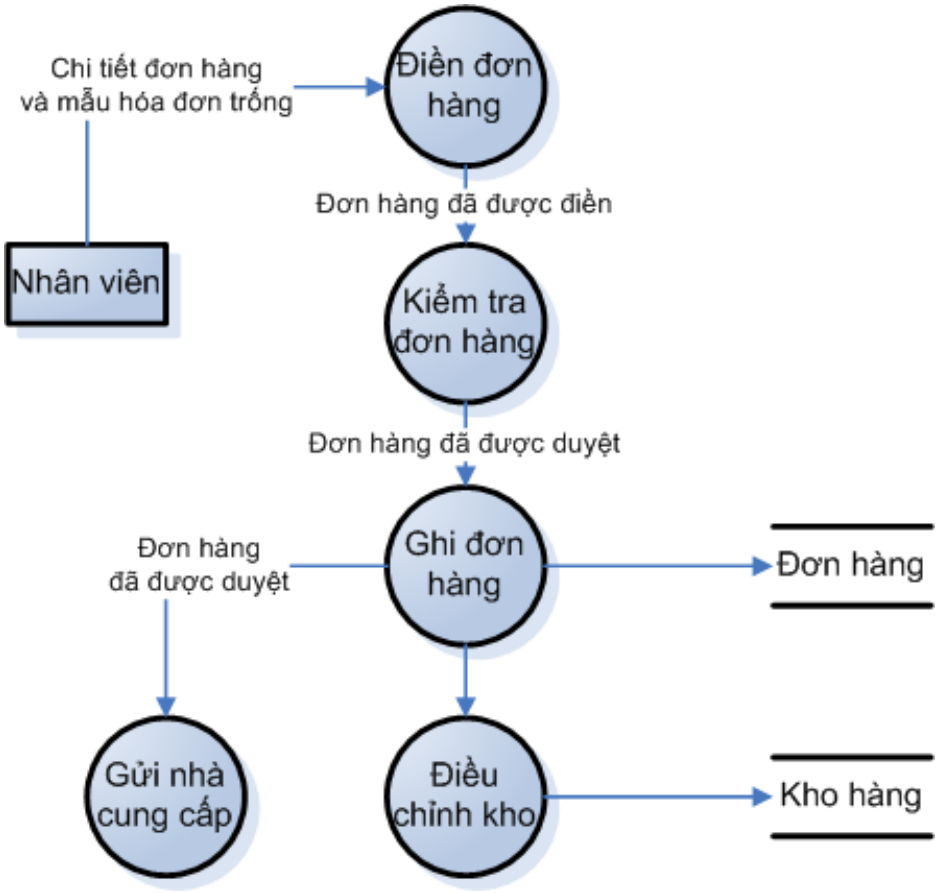
Mô hình luồng dữ liệu

Mô hình luồng dữ liệu được sử dụng để mô hình hoá quy trình xử lý dữ liệu của hệ thống. Mô hình này sẽ biểu diễn các bước mà luồng dữ liệu phải trải qua trong hệ thống từ điểm đầu tới điểm cuối.

Mô hình luồng dữ liệu mô hình hoá hệ thống từ góc độ một chức năng. Việc tìm vết và tư liệu hoá quan hệ giữa dữ liệu với một quy trình rất có ích đối với việc tìm hiểu toàn bộ hệ thống.

Mô hình luồng dữ liệu là phần cốt lõi của rất nhiều phương pháp phân tích. Nó chứa các ký pháp rất dễ hiểu đối với khách hàng.

Ví dụ: Mô hình luồng dữ liệu của chức năng xử lý đơn hàng



Mô hình máy trạng thái

Mô hình máy trạng thái mô tả đáp ứng của hệ thống với các sự kiện bên trong và bên ngoài của nó. Mô hình máy trạng thái biểu diễn các trạng thái của hệ thống và các sự kiện gây ra sự dịch chuyển trạng thái.

Mô hình máy trạng thái biểu diễn các trạng thái của hệ thống là các nút và sự kiện là các cung nối giữa các nút đó. Khi có một sự kiện xảy ra, hệ thống sẽ dịch chuyển từ trạng thái này sang trạng thái khác.

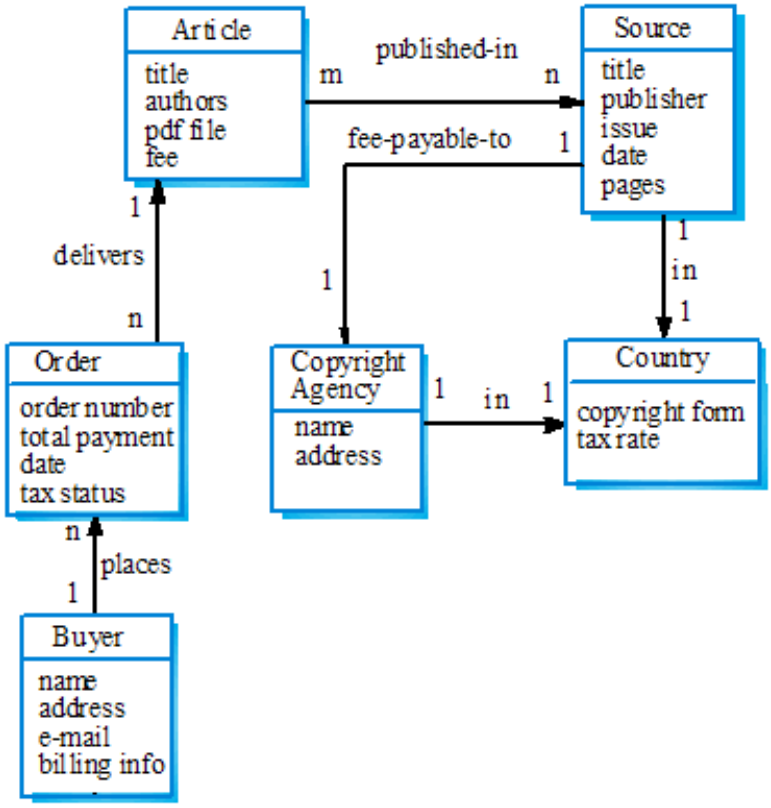
Biểu đồ trạng thái là một biểu đồ trong UML và được sử dụng để biểu diễn mô hình máy trạng thái. Biểu đồ trạng thái cho phép phân tích một mô hình thành nhiều mô hình con và mô tả ngắn gọn về các hành động cần thực hiện tại mỗi trạng thái. Ta có thể vẽ các bảng để mô tả mối quan hệ giữa trạng thái và tác nhân kích hoạt.

Mô hình dữ liệu

Giới thiệu

Mô hình dữ liệu được sử dụng để mô tả cấu trúc logic của dữ liệu được xử lý bởi hệ thống. Thông thường, chúng ta hay sử dụng mô hình thực thể - quan hệ - thuộc tính (ERA) thiết lập các thực thể của hệ thống, quan hệ giữa các thực thể và thuộc tính của các thực thể. Mô hình này được sử dụng trong thiết kế CSDL và thường được cài đặt trong các CSDL quan hệ.

Ví dụ mô hình dữ liệu của LIBSYS



Tuy nhiên, mô hình dữ liệu thường không chi tiết. Cho nên, chúng ta có thể sử dụng từ điển dữ liệu làm công cụ bổ trợ. Từ điển dữ liệu là danh sách tất cả các tên gọi được sử dụng trong các mô hình hệ thống. Đó có thể là các thực thể, quan hệ và các thuộc tính …

Ưu điểm của từ điển dữ liệu là: hỗ trợ quản lý tên và tránh trùng lặp tên, lưu trữ kiến thức một cách có tổ chức kết nối pha phân tích, thiết kế và cài đặt.

Ví dụ: từ điển dữ liệu của LIBSYS

Tên	Mô tả	Kiểu
Article	Chi tiết về bài báo có trong LIBSYS	Thực thể
Authors	Tên của tác giả viết bài báo	Thuộc tính
Buyer	Tên của người hoặc tổ chức muốn sao chép bài báo	Thực thể
Fee-payable-to	Quan hệ 1:1 giữa Article và Copyright Agency	Quan hệ
Address Buyer	Địa chỉ của người đặt mua được sử dụng để chuyển bài báo tới	Thuộc tính

Mô hình đối tượng,hệ thống,ứng xử và thừa kế

Mô hình đối tượng

Giới thiệu

Sử dụng mô hình ứng xử hay mô hình dữ liệu thường rất khó mô tả các vấn đề có liên quan đến thế giới thực. Mô hình đối tượng đã giải quyết được vấn đề này bằng cách kết hợp ứng xử và dữ liệu thành đối tượng.

Mô hình đối tượng được sử dụng để biểu diễn cả dữ liệu và quy trình xử lý của hệ thống. Nó mô tả hệ thống dựa theo thuật ngữ các lớp đối tượng và các quan hệ của nó. Một lớp đối tượng là sự trừu tượng hoá trên một tập các đối tượng có thuộc tính và phương thức chung.

Mô hình đối tượng phản ánh các thực thể trong thế giới thực được vận dụng trong hệ thống. Nếu ta càng có nhiều thực thể trừu tượng thì việc mô hình hoá càng khó khăn.

Phát hiện các lớp đối tượng là một quy trình rất khó khăn khi tìm hiểu sâu về lĩnh vực của ứng dụng. Các lớp đối tượng thường phản ảnh các thực thể liên quan tới miền ứng dụng của hệ thống.

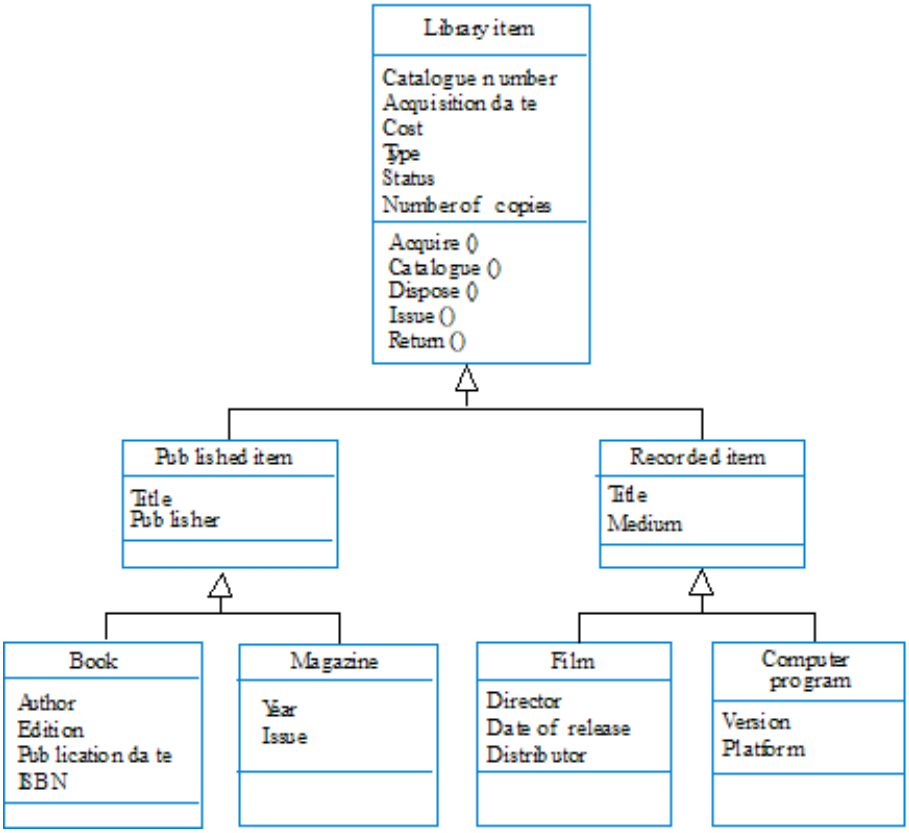
Các mô hình đối tượng bao gồm: mô hình thừa kế, mô hình kết hợp và mô hình ứng xử. Trong phần tiếp theo, chúng ta sẽ tìm hiểu từng loại mô hình này.

Mô hình thừa kế

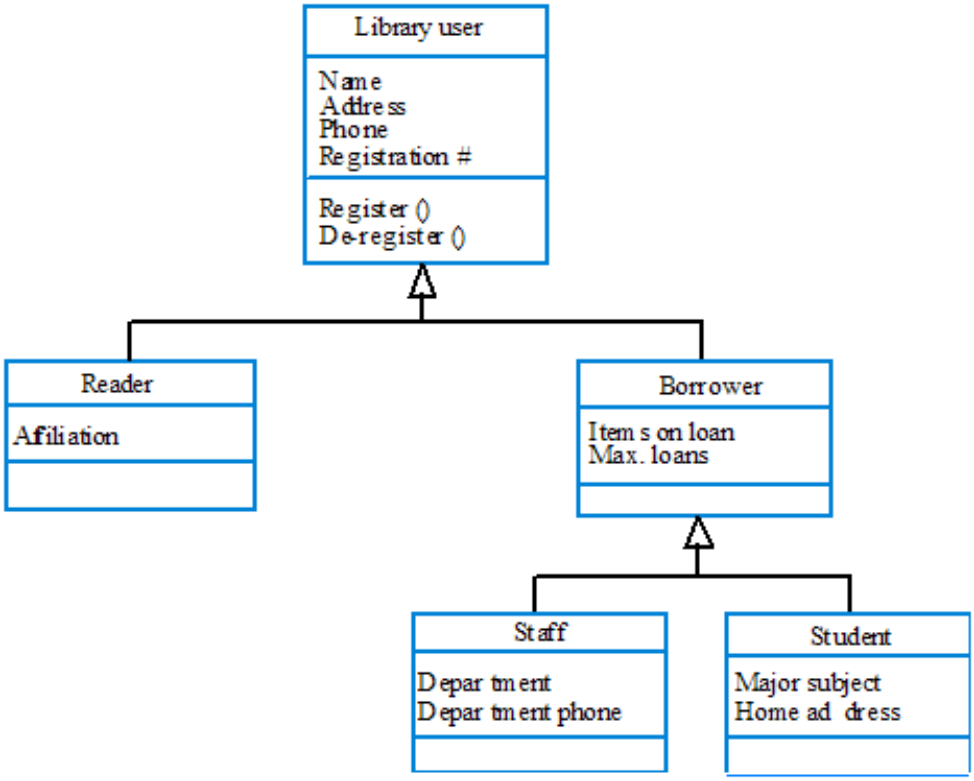
Mô hình thừa kế tổ chức các lớp đối tượng theo một cấu trúc phân cấp. Các lớp ở đỉnh của cấu trúc phân cấp phản ánh những đặc trưng chung của tất cả các lớp. Các lớp đối tượng thừa kế những thuộc tính và phương thức của các lớp cha của nó nó có thể bổ sung những đặc điểm của riêng nó.

Thiết kế lớp phân cấp là một quy trình khá phức tạp, ta nên loại bỏ sự trùng lặp giữa các nhánh khác nhau.

Ví dụ: cấu trúc phân cấp của lớp Library trong LIBSYS

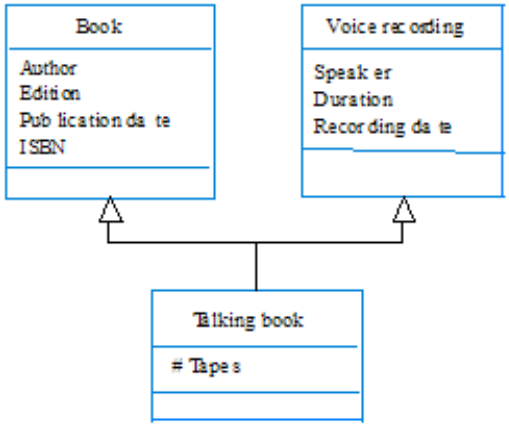


Ví dụ: cấu trúc phân cấp của lớp User trong LIBSYS



Cấu trúc đa thừa kế: lớp đối tượng có thể thừa kế từ một hoặc nhiều lớp cha. Tuy nhiên, điều này có thể dẫn tới sự xung đột về ngữ nghĩa khi các thuộc tính/phương thức trùng tên ở các lớp cha khác nhau có ngữ nghĩa khác nhau.

Ví dụ: lớp Talking book thừa kế từ hai lớp Book và Voice recording.

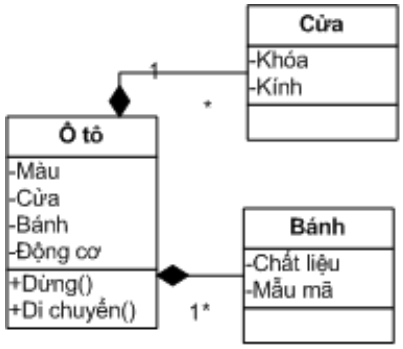


Mô hình kết hợp

Mô hình kết hợp biểu diễn cách cấu tạo của một lớp từ các lớp khác. Mô hình kết hợp tương tự như quan hệ hợp thành (part-of).

Ví dụ: Mô hình kết hợp

Đối tượng ô tô được tạo thành từ nhiều đối tượng khác như: cửa, bánh xe ...



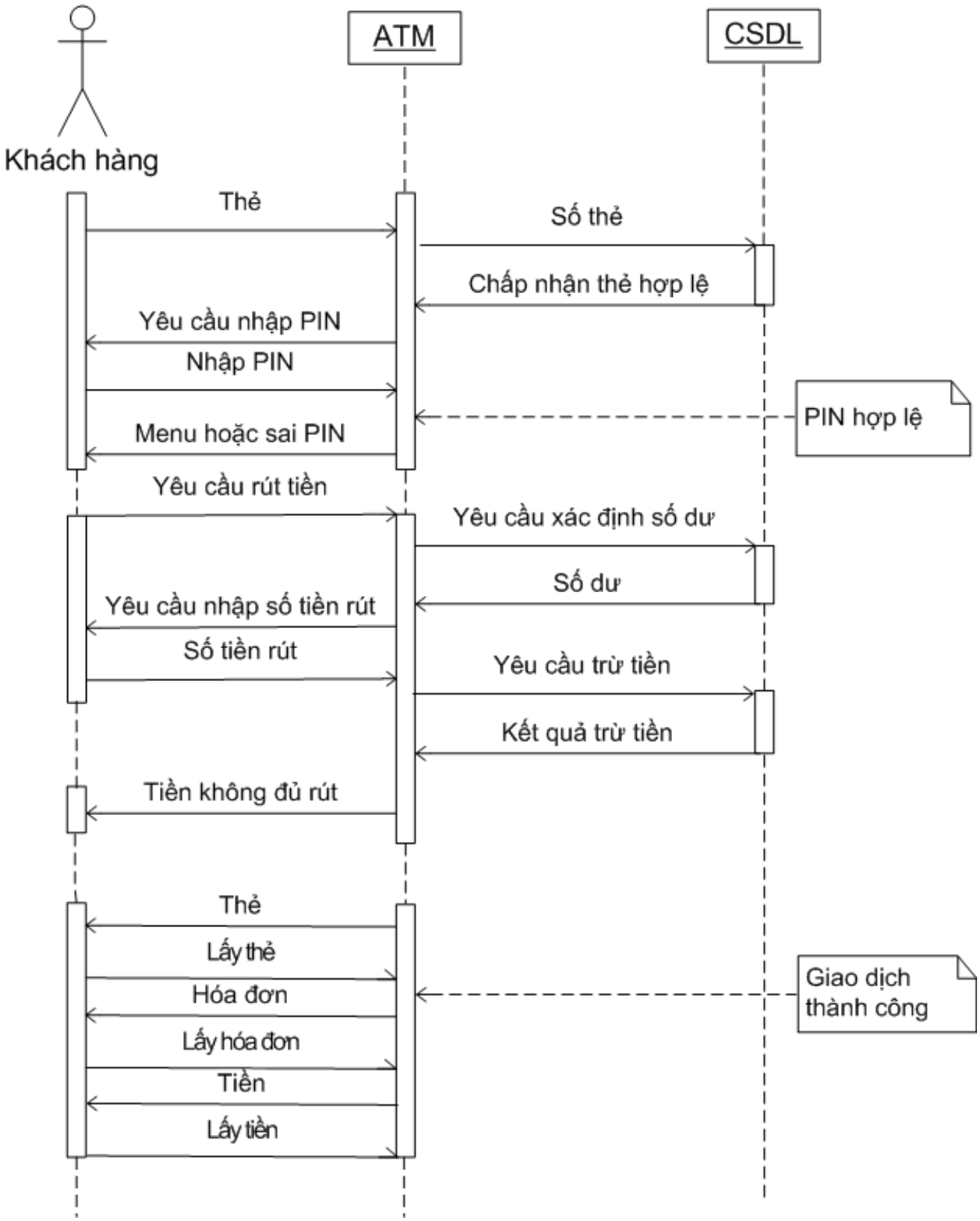
Mô hình ứng xử

Mô hình ứng xử mô tả tương tác giữa các đối tượng nhằm tạo ra một số ứng xử cụ thể của hệ thống mà đã được xác định như là một ca sử dụng.

Biểu đồ trình tự hoặc biểu đồ cộng tác trong UML được sử dụng để mô hình hoá tương tác giữa các đối tượng.

Ví dụ

Mô tả ca sử dụng Rút tiền của hệ thống ATM.


Khách hàng
ATM
CSDL
Thẻ
Số thẻ
Chấp nhận thẻ hợp lệ
Yêu cầu nhập PIN
Nhập PIN
PIN hợp lệ
Menu hoặc sai PIN
Yêu cầu rút tiền
Yêu cầu xác định số dư
Số dư
Yêu cầu nhập số tiền rút
Số tiền rút
Yêu cầu trừ tiền
Kết quả trừ tiền
Tiền không đủ rút
Thẻ
Lấy thẻ
Giao dịch thành công
Hóa đơn
Lấy hóa đơn
Tiền
Lấy tiền

Phương pháp hướng cấu trúc

Giới thiệu

Ngày nay, phương pháp hướng cấu trúc rất ít khi được sử dụng do không còn phù hợp với các hệ thống lớn. Tuy nhiên, trong giáo trình này, chúng tôi vẫn trình bày phần này để học viên có cái nhìn mang tính tổng quan về vấn đề này.

Các phương pháp hướng cấu trúc đều cung cấp framework để mô hình hoá hệ thống một cách chi tiết. Chúng thường có một tập hợp các mô hình đã được định nghĩa trước, quy trình để đưa ra các mô hình đó và các quy tắc, hướng dẫn có thể áp dụng cho các mô hình.

Tuy nhiên, các phương pháp hướng cấu trúc thường có một số nhược điểm sau:

- Không mô hình hoá được các yêu cầu hệ thống phi chức năng

- Không chứa những thông tin để xác định liệu một phương thức có thích hợp với một vấn đề đưa ra hay không.

- Tạo ra quá nhiều tài liệu

- Mô hình hoá hệ thống quá chi tiết và khó hiểu đối với người sử dụng.

CASE workbenches là tập hợp các công cụ được thiết kế để hỗ trợ các quy trình xây dựng hệ thống phần mềm như phân tích, thiết kế và kiểm thử.

CASE tools hỗ trợ mô hình hoá hệ thống là một công cụ quan trọng của phương pháp hướng cấu trúc. Sau đây là một số CASE tool thường được sử dụng:

- Soạn thảo biểu đồ

- Công cụ phân tích mô hình và kiểm tra

- Ngôn ngữ truy vấn

- Từ điển dữ liệu

- Công cụ tạo và định nghĩa báo cáo

- Công cụ định nghĩa form

- Bộ dịch

- Công cụ tạo mã lệnh tự động

Các vấn đề về thiết kế kiến trúc

Thiết kế kiến trúc

Giới thiệu

Sau khi xác định và phân tích yêu cầu hệ thống, chúng ta chuyển sang pha thiết kế và cài đặt hệ thống. Thiết kế kiến trúc hệ thống là giai đoạn sớm nhất trong quy trình thiết kế hệ thống. Thiết kế kiến trúc cung cấp cho chúng ta bản đặc tả về kiến trúc hệ thống, bao gồm những hệ thống con nào, tương tác với nhau ra sao, framework hỗ trợ điều khiển tương tác giữa các hệ thống con như thế nào …

Mục tiêu

- Thiết kế kiến trúc là gì và đặc điểm của chúng. Đồng thời, xem xét một số mô hình kiến trúc hệ thống thường được sử dụng.

- Các phương pháp tổ chức hệ thống

- Các phương pháp phân rã hệ thống con thành các mô-đun.

- Thấy được mối quan hệ giữa các phương pháp tổ chức hệ thống và các phương pháp phân rã hệ thống.

- Các chiến lược điều khiển hệ thống

- Giới thiệu một số mô hình kiến trúc tham chiếu

Thiết kế kiến trúc là gì?

Đặt vấn đề

- Công việc đầu tiên của giai đoạn thiết kế hệ thống là gì? Tại sao?

Quy trình thiết kế nhằm xác định các hệ thống con cấu tạo lên hệ thống đề xuất và framework giúp điều khiển các hệ thống con và giao tiếp giữa chúng được gọi là quy trình thiết kế kiến trúc. Kết quả của quy trình thiết kế này là bản đặc tả về kiến trúc phần mềm.

Thiết kế kiến trúc là pha sớm nhất trong quy trình thiết kế hệ thống. Thiết kế kiến trúc thường được thực hiện song song với một số hành động đặc tả. Nó bao gồm có việc phát hiện các thành phần chính của hệ thống và giao tiếp giữa chúng.

Nếu chúng ta có được bản thiết kế kiến trúc rõ ràng thì ta sẽ thấy được các ưu điểm của nó trong những hoạt động sau:

- Giao tiếp giữa các stakeholder: kiến trúc hệ thống thường được sử dụng làm tâm điểm của các buổi thảo luận giữa các stakeholder.

- Phân tích hệ thống: tức là phân tích để xác định liệu hệ thống có thoả mãn các yêu cầu phi chức năng của nó hay không.

- Tái sử dụng với quy mô lớn: kiến trúc có thể được tái sử dụng trong nhiều hệ thống.

Các đặc điểm của kiến trúc hệ thống:

- Hiệu năng: hạn chế các thao tác phức tạp và tối thiểu hoá giao tiếp.

- Bảo mật: sử dụng kiến trúc phân lớp với nhiều kiểm soát chặt chẽ ở các lớp sâu hơn.

- An toàn.

- Sẵn dùng.

- Có khả năng bảo trì.

Tuy nhiên, trong quá trình thiết kế kiến trúc có thể xảy ra các xung đột về mặt kiến trúc như sau:

- Sử dụng nhiều thành phần lớn sẽ tăng hiệu năng nhưng giảm khả năng bảo trì.

- Nếu dữ liệu bị dư thừa thì sẽ cải thiện tính sẵn dùng nhưng làm cho việc bảo mật khó khăn hơn.

- Hạn chế các thuộc tính có liên quan đến tính an toàn có nghĩa là nếu có nhiều giao tiếp thì sẽ làm giảm hiệu năng.

Thiết kế kiến trúc là một quy trình sáng tạo cho nên sự phác biệt của quy trình này phụ thuộc vào từng loại hệ thống được xây dựng. Tuy nhiên, các quy trình thiết kế đều dựa trên những quyết định sau:

- Kiến trúc ứng dụng chung có được sử dụng lại hay không?

- Hệ thống sẽ được phân tán như thế nào?

- Những phong cách kiến trúc nào là thích hợp?

- Hệ thống sẽ được phân rã thành những mô-đun nào?

- Chiến lược điều khiển nào sẽ được sử dụng?

- Cách đánh giá thiết kế kiến trúc

- Kiến trúc sẽ được tư liệu hoá như thế nào?

Hơn nữa, cũng cần phải chú ý rằng các hệ thống có cùng miền ứng dụng có thể có các kiến trúc chung để phản ánh những khái niệm liên quan đến miền ứng dụng đó. Đồng thời, các dây chuyền sản xuất phần mềm được xây dựng quanh kiến trúc nền tảng cùng với các biến đổi tuỳ thuộc vào yêu cầu của khách hàng. Do đó, khả năng tái sử dụng lại kiến trúc hệ thống là rất cao.

Sau đây là các mô hình kiến trúc cơ bản:

- Mô hình cấu trúc tĩnh: mô tả các thành phần hệ thống chính.

- Mô hình quy trình động: biểu diễn quy trình cấu trúc của hệ thống.

- Mô hình giao diện: định nghĩa tập hợp các giao diện của hệ thống con

- Mô hình quan hệ: biểu diễn quan hệ giữa các hệ thống con.

- Mô hình phân tán: biểu diễn cách cài đặt các hệ thống con trên máy tính.

Tổ chức hệ thống và các mô hình

Tổ chức hệ thống

Giới thiệu

Tổ chức hệ thống phản ánh chiến lược cơ bản được sử dụng để cấu trúc hệ thống. Trong quá trình thiết kế kiến trúc hệ thống, hoạt động đầu tiên phải thực hiện là xây dựng mô hình tổ chức hệ thống. Có 3 phương pháp tổ chức hệ thống thường được sử dụng:

- Kho dữ liệu dùng chung

- Server và các dịch vụ dùng chung (client-server)

- Phân lớp hoặc máy trừu tượng

Ở các phần tiếp theo, chúng ta sẽ tìm hiểu chi tiết ba phương pháp tổ chức hệ thống này.

Mục tiêu

- Nắm được ba phương pháp xây dựng mô hình tổ chức hệ thống

- Tìm hiểu chi tiết từng phương pháp và đánh giá ưu/nhược điểm của chúng

- Có khả năng vận dụng trên hệ thống cụ thể

Kho dữ liệu dùng chung

Các hệ thống con phải trao đổi dữ liệu và làm việc với nhau một cách hiệu quả. Việc trao đổi dữ liệu được thực hiện theo hai cách:

- Dữ liệu chia sẻ được lưu ở CSDL trung tâm hoặc kho dữ liệu và được tất cả các hệ thống con truy nhập.

- Mỗi hệ thống con bảo trì CSDL của chính nó và truyền dữ liệu một cách tường minh cho các hệ thống con khác.

Nếu số lượng dữ liệu dùng chung rất lớn thì mô hình kho dữ liệu dùng chung thường được sử dụng phổ biến nhất. Ưu điểm của mô hình này là:

- Đây là phương pháp hiệu quả để chia sẻ số lượng lớn dữ liệu.

- Các hệ thống con không cần quan tâm tới những hoạt động liên quan đến dữ liệu như: sao lưu, bảo mật,… vì đã có bộ quản lý trung tâm thực hiện nhiệm vụ này.

Tuy nhiên, việc sử dụng kho dữ liệu dùng chung cũng có một số nhược điểm sau:

- Tất cả các hệ thống con phải chấp nhận mô hình kho dữ liệu.

- Việc cải tiến dữ liệu rất phức tạp và tốn kém

- Khó phân tán một cách hiệu quả

- Không có giới hạn cho các chính sách quản lý cụ thể.

Mô hình client – server

Mô hình kiến trúc client-server là một mô hình hệ thống trong đó hệ thống bao gồm một tập hợp các server cung cấp dịch vụ và các client truy nhập và sử dụng các dịch vụ đó. Các thành phần chính của mô hình này bao gồm:

- Tập hợp các server sẽ cung cấp những dịch vụ cụ thể như: in ấn, quản lý dữ liệu…

- Tập hợp các client truy nhập đến server để yêu cầu cung cấp dịch vụ.

- Hệ thống mạng cho phép client truy cập tới dịch vụ mà server cung cấp.

Client phải biết tên của server và các dịch vụ mà server cung cấp. Nhưng server thì không cần xác định rõ client và hiện tại có bao nhiêu client. Client tạo ra một yêu cầu tới server và chờ server trả lời.

Ưu điểm của mô hình client - server là:

- Phân tán dữ liệu rõ ràng

- Sử dụng các hệ thống được kết nối mạng một cách hiệu quả và chi phí dành cho phần cứng có thể rẻ hơn.

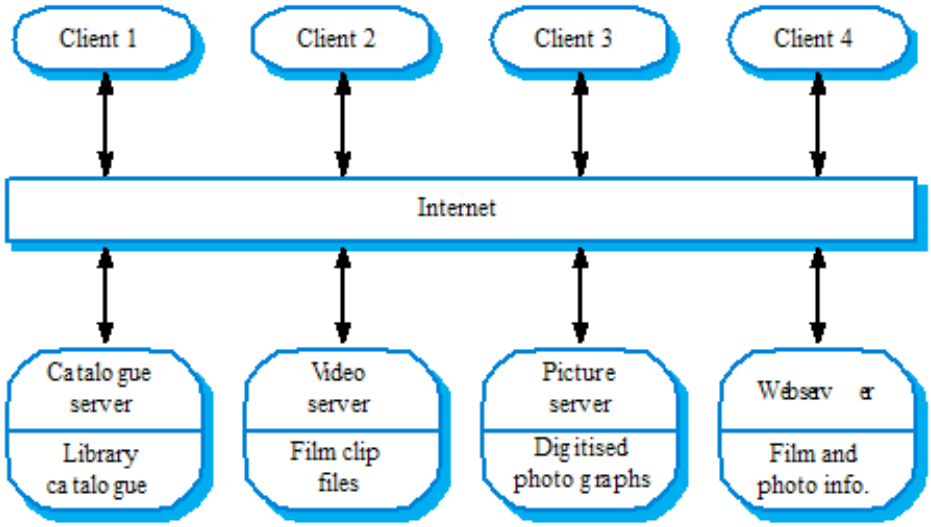
- Dễ dàng bổ sung hoặc nâng cấp server

Nhược điểm của mô hình client - server là:

- Không phải là mô hình dữ liệu dùng chung nên các hệ thống con có thể sử dụng các tổ chức dữ liệu khác nhau. Do đó, việc trao đổi dữ liệu có thể không hiệu quả.

- Quản lý mỗi server không thống nhất, dư thừa.

- Không đăng ký tên và dịch vụ tập trung. Điều này làm cho việc tìm kiếm server hoặc các dịch vụ rất khó khăn.

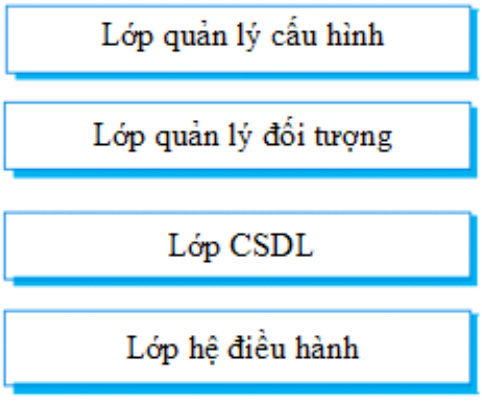


Hình 7.1: Mô hình client –server của hệ thống thư viện phim và ảnh

Mô hình phân lớp

Mô hình phân lớp tổ chức hệ thống thành nhiều lớp và mỗi lớp cung cấp một tập các dịch vụ. Mỗi lớp có thể được coi như một máy trừu tượng (abstract machine) mà ngôn ngữ của máy được định nghĩa bởi các dịch vụ mà lớp đó cung cấp. Do đó, mô hình này thường được sử dụng để mô hình hoá giao diện (interface) của hệ thống con.

Mô hình phân lớp hỗ trợ phát triển các hệ thống con theo kiểu tăng vòng ở nhiều lớp khác nhau. Khi giao diện của một lớp thay đổi thì chỉ những lớp liền kề nó mới bị ảnh hưởng.



Hình 7.2: Mô hình phân lớp của hệ thống quản lý phiên bản

Phân rã hệ thống và phân rã đối tượng

Phân rã hệ thống

Giới thiệu

Sau khi cấu trúc hệ thống đã được lựa chọn, ta cần phải xác định phương pháp phân rã các hệ thống con thành các mô-đun.

Hệ thống con là một hệ thống có thể vận hành một cách độc lập, có thể sử dụng một số dịch vụ được cung cấp bởi các hệ thống con khác hoặc cung cấp dịch vụ cho các hệ thống con khác sử dụng.

Mô-đun là một thành phần hệ thống cung cấp các dịch vụ cho các thành phần khác, nhưng nó thường không được coi như là một hệ thống riêng rẽ, độc lập.

Có hai cách để phân rã các hệ thống con thành các mô-đun:

- Phân rã hướng đối tượng: hệ thống được phân rã thành các đối tượng tương tác với nhau.

- Pipeline hướng chức năng hoặc luồng dữ liệu: hệ thống được phân rã thành các mô-đun chức năng chịu trách nhiệm chuyển đổi thông tin đầu vào thành kết quả đầu ra.

Mục tiêu

- Phân biệt được hai khái niệm: hệ thống con và mô-đun

- Nắm được hai phương pháp phân rã hệ thống và đánh giá ưu/nhược điểm của từng phương pháp.

Phân rã hướng đối tượng

Mô hình kiến trúc hướng đối tượng cấu trúc hệ thống thành một tập hợp các đối tượng gắn kết lỏng dựa trên các giao diện đã được định nghĩa.

Phân rã hướng đối tượng liên quan tới việc xác định lớp đối tượng, các thuộc tính và phương thức của nó. Khi cài đặt lớp, các đối tượng sẽ được tạo ra từ các lớp này và có một số mô hình điều khiển được sử dụng để kết hợp các phương thức của đối tượng.

Ưu điểm của mô hình hướng đối tượng:

- Đối tượng được gắn kết lỏng nên khi thay đổi cách cài đặt chúng có thể không ảnh hưởng tới các đối tượng khác.

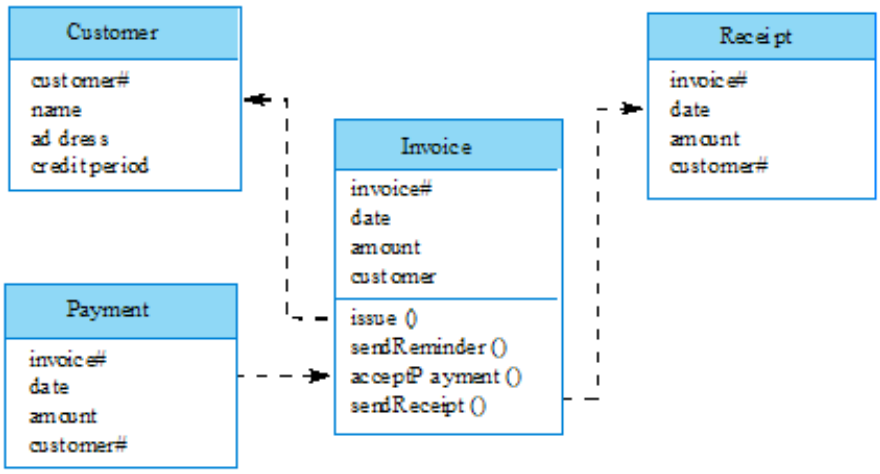
- Đối tượng phản ánh thực thể trong thế giới thực.

- Các ngôn ngữ lập trình hướng đối tượng được sử dụng rộng rãi.

<

Tuy nhiên, khi giao diện của đối tượng thay đổi có thể gây ra những vấn đề hết sức khó khăn và rất khó biểu diễn các thực thể phức tạp trong thế giới thực như là các đối tượng.

Ví dụ: các đối tượng trong hệ thống xử lý hoá đơn



Pipeline hướng chức năng

Mô hình pipeline hướng chức năng hoặc mô hình luồng dữ liệu là quy trình chuyển đổi thông tin đầu vào thành kết quả đầu ra. Việc chuyển đổi thông tin đầu vào thành kết quả đầu ra có thể được thực hiện tuần tự hoặc song song. Dữ liệu được xử lý trong quy trình có thể là riêng lẻ hoặc theo lô.

Ưu điểm của mô hình:

- Hỗ trợ tái sử dụng quy trình chuyển đổi

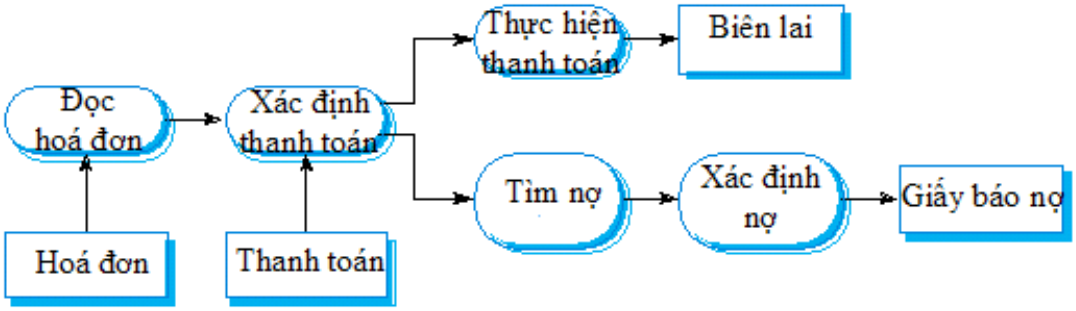
- Cung cấp tài liệu để giao tiếp với stakeholder

- Dễ dàng bổ sung thêm quy trình chuyển đổi mới.

- Dễ dàng thực hiện, kể cả với hệ thống tuần tự hoặc song song.

Tuy nhiên, mô hình này yêu cầu phải có định dạng dữ liệu chung để truyền qua các pipeline và rất khó hỗ trợ cho các tương tác hướng sự kiện.

Ví dụ: Mô hình luồng dữ liệu của hệ thống xử lý hoá đơn



Các chiến lược điều khiển

Giới thiệu

Các mô hình cấu trúc hệ thống có liên quan tới cách phân rã hệ thống thành nhiều hệ thống con. Để hệ thống làm việc tốt, ta phải điều khiển được các hệ thống con; cho nên, các dịch vụ của chúng phải được thực hiện đúng chỗ và đúng thời điểm.

Có 2 loại chiến lược điều khiển:

- Điều khiển tập trung: một hệ thống con chịu trách nhiệm kiểm soát, khởi tạo hoặc dừng các hệ thống con khác.

- Điều khiển hướng sự kiện: mỗi hệ thống đáp ứng với các sự kiện xảy ra từ các hệ thống con khác hoặc từ môi trường của hệ thống.

Mục tiêu

- Giải thích được tại sao phải thiết lập chiến lược điều khiển hệ thống?

- Nắm được các chiến lược điều khiển hệ thống và đánh giá chúng.

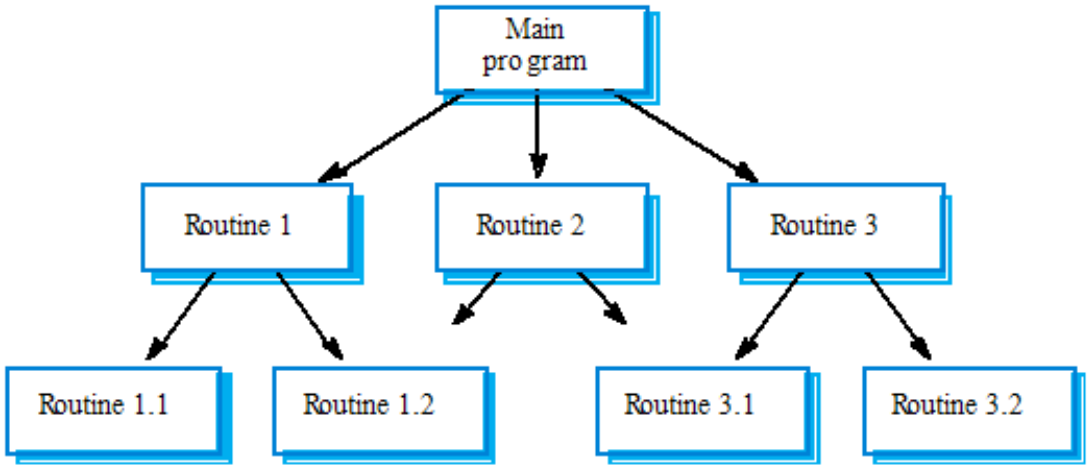
Điều khiển tập trung

Hệ thống con điều khiển chịu trách nhiệm quản lý việc thực hiện của các hệ thống con khác.

Chiến lược điều khiển tập trung gồm 2 loại mô hình:

Mô hình gọi - trả lời (call-return)

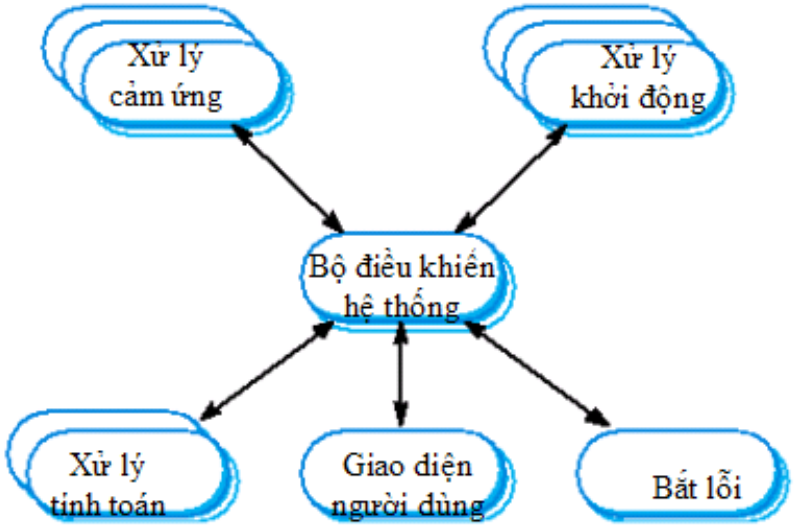
Gồm các thủ tục con được sắp xếp phân cấp, thủ tục điều khiển nằm ở đỉnh của cấu trúc phân cấp và di chuyển dần xuống dưới. Mô hình này thường được áp dụng cho các hệ thống tuần tự.



Hình 7.3: Mô hình gọi - trả lời

Mô hình quản lý

Thường áp dụng cho các hệ thống song song. Một thành phần hệ thống điều khiển việc khởi tạo, ngừng, hoặc cộng tác với các quy trình hệ thống khác.



Hình 7.4: Mô hình điều khiển tập trung của hệ thống thời gian thực

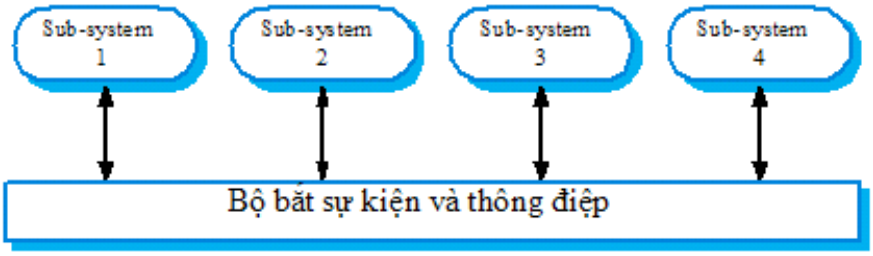
Điều khiển hướng sự kiện

Mô hình lan truyền (Broadcast)

Trong mô hình lan truyền, sự kiện được lan truyền tới tất cả các hệ thống con. Bất kỳ hệ thống nào nếu có thể bắt được sự kiện này thì sẽ xử lý nó.

Mô hình này có hiệu quả đối với việc tích hợp các hệ thống con trên nhiều máy tính khác nhau trong cùng một mạng.

Các hệ thống con phải đăng ký những sự kiện mà nó có thể bắt. Khi những sự kiện này xảy ra, điều khiển sẽ được truyền cho hệ thống con có thể bắt được sự kiện đó. Những quy tăc điều khiển không được gắn với sự kiện và bộ bắt sự kiện. Các hệ thống phải quyết định sự kiện nào là sự kiện mà nó đã đăng ký. Nhưng nó không cần phải biết khi nào sự kiện sẽ được bắt.



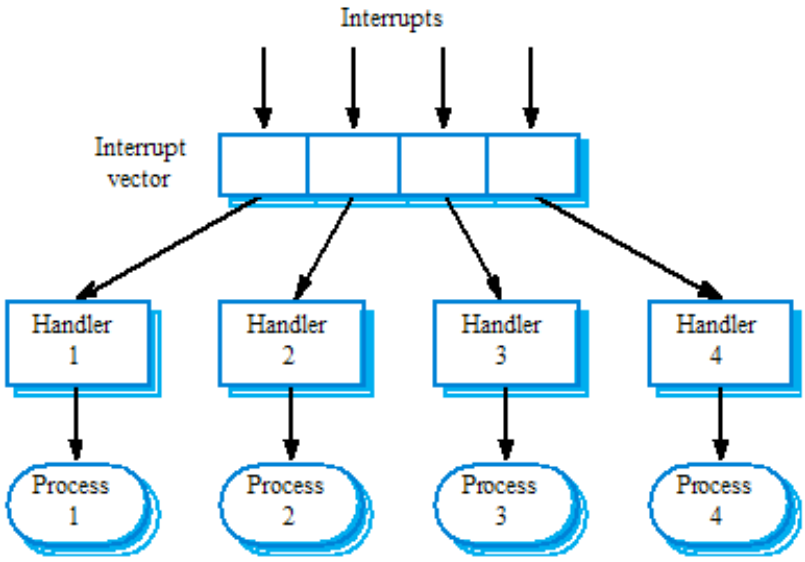
Hình 7.5: Mô hình điều khiển lan truyền

Mô hình hướng ngắt (Interrupt-driven)

Mô hình hướng ngắt được sử dụng trong các hệ thống thời gian thực trong đó các ngắt được phát hiện bởi bộ bắt ngắt (interrupt handler) và được truyền cho một số các thành phần khác để xử lý.

Các kiểu ngắt và bộ bắt tương ứng được định nghĩa trước. Mỗi kiểu ngắt được gắn với một vị trí nhớ và một bộ chuyển mạch để đưa ngắt tới bộ bắt tương ứng của nó.

Mô hình này cho phép đáp ứng rất nhanh, nhưng lập trình khá phức tạp và khó đánh giá.



Hình 7.6: Mô hình điều khiển lan truyền

Các kiến trúc tham chiếu

Giới thiệu

Mặc dù các hệ thống có cùng một miền ứng dụng có khác nhau về chi tiết nhưng kiến trúc chung của nó khá giống nhau. Do đó, khi xây dựng một hệ thống mới, ta có thể sử dụng lại kiến trúc của hệ thống tương tự đã tồn tại.

Có 2 loại mô hình kiến trúc cho một miền ứng dụng cụ thể:

- Mô hình chung: là sự trừu tượng hoá từ một số các hệ thống thực và bao hàm các thuộc tính quan trọng của các hệ thống này. Mô hình chung là mô hình bottom-up.

- Mô hình tham chiếu: là mô hình trừu tượng hoá và lý tưởng. Các mô hình tham chiếu thường kế thừa từ những nghiên cứu về miền ứng dụng hơn là từ các hệ thống đang tồn tại. Mô hình tham chiếu là mô hình top-down. Mô hình tham chiếu được sử dụng như một phần cơ bản để cài đặt hệ thống hoặc để so sánh với các hệ thống khác. Nó đóng vai trò là một mô hình chuẩn để đánh giá các hệ thống khác.

Thiết kế giao diện người dùng

Thiết kế

Giới thiệu

Chúng ta phải luôn nhớ một nguyên tắc quan trọng khi xây dựng một hệ thống phần mềm, đó là: người sử dụng không quan tâm đến cấu trúc bên trong của hệ thống, đơn giản hay phức tạp; cái mà họ có thể đánh giá được và cảm nhận được chính là giao diện tương tác giữa hệ thống và người sử dụng. Nếu người sử dụng cảm thấy giao diện không thích hợp, khó sử dụng thì rất có thể họ sẽ không sử dụng cả hệ thống; cho dù hệ thống đó có đáp ứng tất cả các chức năng nghiệp vụ mà họ muốn. Và như vậy, dự án của chúng ta sẽ thất bại.

Vì tầm quan trọng của giao diện người dùng, nên chúng ta có cả một chương để nói về chúng. Trong chương này, chúng ta sẽ nghiên cứu những vấn đề sau:

- Các yếu tố liên quan đến giao diện người dùng

- Quy trình xây dựng giao diện người dùng

Giao diện người dùng

Giới thiệu

Giao diện người dùng cần phải được thiết kế sao cho phù hợp với kỹ năng, kinh nghiệm và sự trông đợi của người sử dụng nó.

Người sử dụng hệ thống thường đánh giá hệ thống thông qua giao diện hơn là chức năng của nó. Giao diện của hệ thống nghèo nàn có thể khiến người sử dụng tạo ra các lỗi hết sức nghiêm trọng. Đó là lý do tại sao nhiều hệ thống phần mềm không bao giờ được sử dụng.

Mục tiêu

- Nắm được sự ảnh hưởng của người sử dụng tới giao diện

- Một số nguyên tắc khi thiết kế giao diện người dùng

- Phân loại các khả năng tương tác giữa người và máy để thiết kế giao diện cho phù hợp

- Biết cách biểu diễn thông tin cho phù hợp với người sử dụng

Tác nhân con người trong thiết kế giao diện

Một nhân tố quan trọng ảnh hưởng tới quá trình thiết kế giao diện đó chính là người sử dụng hệ thống. Do đó, chúng ta phải tìm hiểu một số đặc điểm của người sử dụng có liên quan đến giao diện hệ thống:

- Khả năng nhớ tức thời của con người bị hạn chế: con người chỉ có thể nhớ ngay khoảng 7 loại thông tin. Nếu ta biểu diễn nhiều hơn 7 loại, thì có thể khiến người sử dụng không nhớ hết và gây ra các lỗi.

- Người sử dụng có thể gây ra lỗi: khi người sử dụng gây ra lỗi khiến hệ thống sẽ hoạt động sai, những thông báo không thích hợp có thể làm tăng áp lực lên người sử dụng và do đó, càng xảy ra nhiều lỗi hơn.

- Người sử dụng là khác nhau: con người có những khả năng khác nhau. Những người thiết kế không nên chỉ thiết kế giao diện phù hợp với những khả năng của chính họ.

- Người sử dụng thích các loại tương tác khác nhau: một số người thích hình ảnh, văn bản, âm thanh …

Các nguyên tắc thiết kế giao diên

Thiết kế giao diện phải phụ thuộc vào yêu cầu, kinh nghiệm và khả năng của người sử dụng hệ thống.

Người thiết kế cũng nên quan tâm đến những giới hạn vật lý và tinh thần của con người và nên nhận ra rằng con người luôn có thể gây ra lỗi.

Không phải tất cả các nguyên tắc thiết kế giao diện đều có thể được áp dụng cho tất cả các giao diện. Sau đây là các nguyên tắc thiết kế giao diện:

- Sự quen thuộc của người sử dụng: giao diện phải được xây dựng dựa trên các thuật ngữ và các khái niệm mà người sử dụng có thể hiểu được hơn là những khái niệm liên quan đến máy tính. Ví dụ: hệ thống văn phòng nên sử dụng các khái niệm như thư, tài liệu, cặp giấy … mà không nên sử dụng những khái niệm như thư mục, danh mục …

- Thống nhất: hệ thống nên hiển thị ở mức thống nhất thích hợp. Ví dụ: các câu lệnh và menu nên có cùng định dạng …

- Tối thiểu hoá sự bất ngờ: nếu một yêu cầu được xử lý theo cách đã biết trước thì người sử dụng có thể dự đoán các thao tác của những yêu cầu tương tư.

- Khả năng phục hồi: hệ thống nên cung cấp một số khả năng phục hồi từ lỗi của người sử dụng và cho phép người sử dụng khôi phục lại từ chỗ bị lỗi. Khả năng này bao gồm cho phép làm lại, hỏi lại những hành động như xoá, huỷ …

- Hướng dẫn người sử dụng: như hệ thống trợ giúp, hướng dẫn trực tuyến …

- Tính đa dạng: hỗ trợ nhiều loại tương tác cho nhiều loại người sử dung khác nhau. Ví dụ: nên hiển thị phông chữ lớn với những người cận thị.

Tương tác giữa người sử dụng và hệ thống được chia thành 5 loại sau:

- Vận hành trực tiếp

- Lựa chọn menu

- Điền vào biểu mẫu (Form)

- Ngôn ngữ ra lệnh

- Ngôn ngữ tự nhiên

Biểu diễn thông tin

Biểu diễn thông tin có liên quan tới việc hiển thị các thông tin trong hệ thống tới người sử dụng. Thông tin có thể được biểu diễn một cách trực tiếp hoặc có thể được chuyển thành nhiều dạng hiển thị khác như: dạng đồ hoạ, âm thanh …

Thông tin cần biểu diễn được chia thành hai loại:

- Thông tin tĩnh: được khởi tạo ở đầu của mỗi phiên. Nó không thay đổi trong suốt phiên đó và có thể là ở dạng số hoặc dạng văn bản.

- Thông tin động: thay đổi trong cả phiên sử dụng và sự thay đổi này phải được người sử dụng quan sát.

Các nhân tố ảnh hưởng tới việc hiển thị thông tin:

- Người sử dụng thích hiển thị một phần thông tin hay quan hệ dữ liệu?

- Giá trị của thông tin thay đổi nhanh như thế nào? Sự thay đổi đó có cần phải thể hiện ngay lập tức hay không?

- Người sử dụng có phải thực hiện các hành động để đáp ứng với sự thay đổi không?

- Có phải là giao diện vận hành trực tiếp không?

- Thông tin ở dạng văn bản hay dạng số? Các giá trị quan hệ có quan trọng không?

- Biểu diễn digital hay analogue?

Nếu chúng ta cần hiển thị số lượng lớn thông tin thì nên trực quan hoá dữ liệu. Trực quan hoá có thể phát hiện ra mối quan hệ giữa các thực thể và các xu hướng trong dữ liệu. Ví dụ: thông tin về thời tiết được hiển thị dưới dạng biểu đồ, trạng thái của mạng điện thoại nên được hiển thị bởi các nút có liên kết với nhau.

Chúng ta thường sử dụng màu trong khi thiết kế giao diện. Màu bổ sung thêm một chiều nữa cho giao diện và giúp cho người sử dụng hiểu được những cấu trúc thông tin phức tạp. Màu có thể được sử dụng để đánh dấu những sự kiện ngoại lệ.

Tuy nhiên, khi sử dụng màu để thiết kế giao diện có thể gây phản tác dụng. Do đó, chúng ta nên quan tâm tới một số hướng dẫn sau:

- Giới hạn số lượng màu được sử dụng và không nên lạm dụng việc sử dụng màu.

- Thay đổi màu khi thay đổi trạng thái của hệ thống

- Sử dụng màu để hỗ trợ cho những nhiệm vụ mà người sử dụng đang cố gắng thực hiện.

- Sử dụng màu một cách thống nhất và cẩn thận.

- Cẩn thận khi sử dụng các cặp màu.

Khi người sử dụng tương tác với hệ thống, rất có thể xảy ra lỗi và hệ thống phải thông báo cho người sử dụng biết lỗi gì đã xảy ra hoặc đã có chuyện gì xảy ra với hệ thống. Do đó, thiết kế thông báo lỗi vô cùng quan trọng. Nếu thông báo lỗi nghèo nàn có thể làm cho người sử dụng từ chối hơn là chấp nhận hệ thống.

Vì vậy, thông báo lỗi nên ngắn gọn, xúc tích, thống nhất và có cấu trúc. Việc thiết kế thông báo lỗi nên dựa vào kỹ năng và kinh nghiệm của người sử dụng.

Ví dụ: Giao diện thông báo lỗi

Trong hệ thống quản lý bệnh viện, y tá phải nhập hồ sơ bệnh nhân. Trong khi nhập, y tá quên tên bệnh nhân.

1. Thiết kế giao diện thống báo lỗi.

2. Cho biết các tiêu chuẩn đánh giá một giao diện tốt?

Quy trình thiết kế giao diện người dùng

Quy trình thiết kế

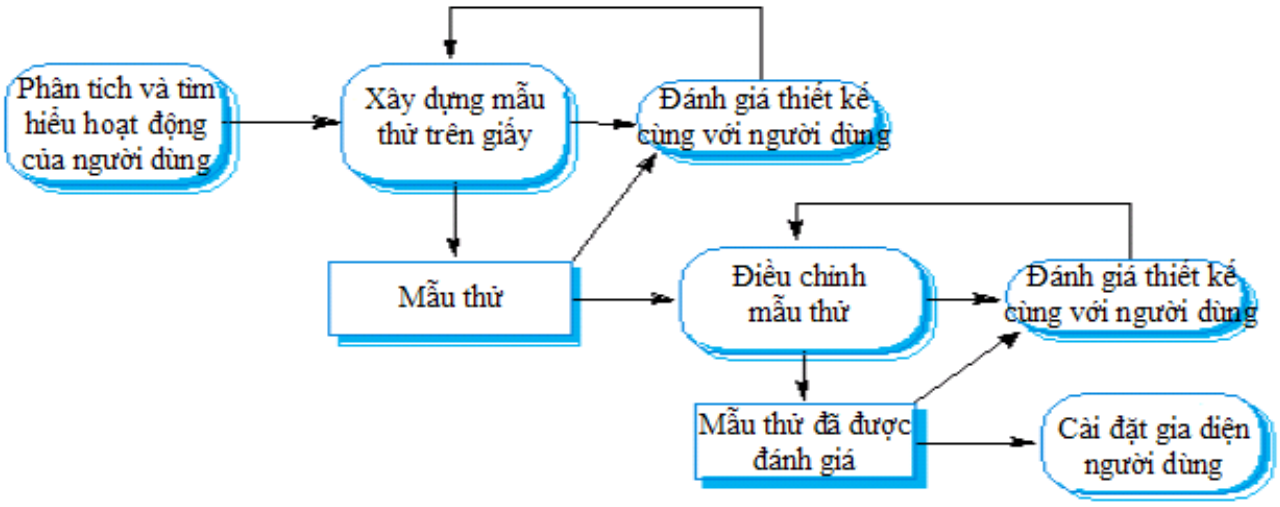
Giới thiệu

Thiết kế giao diện người dùng là một quy trình lặp lại bao gồm sự cộng tác giữa người sử dụng và người thiết kế. Trong quy trình này gồm 3 hoạt động cơ bản:

- Phân tích người sử dụng: tìm hiểu những gì người sử dụng sẽ làm với hệ thống.

- Lập mẫu thử hệ thống: xây dựng một tập các mẫu thử để thử nghiệm

- Đánh giá giao diện: thử nghiệm các mẫu thử cùng với người sử dụng.



Hình 8.3: Quy trình thiết kế gia diện người dùng

Mục tiêu

- Hiểu được quy trình thiết kế giao diện người dùng

- Nắm được chi tiết từng hoạt động trong quy trình thiết kế giao diện người dùng

- Với mỗi hoạt động, chúng ta có rất nhiều cách để thực hiện. Do đó, phải có khả năng lựa chọn phương pháp nào là thích hợp nhất cho từng hoàn cảnh cụ thể.

Phân tích người sử dụng

Nếu ta không hiểu rõ những gì người sử dụng muốn làm với hệ thống, thì ta sẽ không thể thiết kế được một giao diện hiệu quả.

Phân tích người sử dụng phải được mô tả theo những thuật ngữ để người sử dụng và những người thiết kế khác có thể hiểu được.

Các ngữ cảnh mà ta mô tả thao tác ở trong đó là một trong những cách mô tả phân tích người dùng. Ta có thể lấy được rất nhiều yêu cầu của người sử dụng từ đó.

Các kỹ thuật phân tích:

- Phân tích nhiệm vụ: mô hình hoá các bước cần thực hiện để hoàn thành một nhiệm vụ.

- Phân tích nhiệm vụ phân cấp.

- Phỏng vấn và trắc nghiệm: hỏi người sử dụng về những gì mà họ làm. Khi phỏng vấn, chúng ta nên dựa trên những câu hỏi có kết thúc mở. Sau đó, người sử dụng cung cấp những thông tin mà họ nghĩ rằng nó là cần thiết; nhưng không phải tất cả các thông tin đó là có thể được sử dụng. Ngoài ra, chúng ta có thể thực hiện phỏng vấn với cả nhóm người sử dụng, điều đó cho phép người sử dụng thảo luận với nhau về những gì họ làm.

- Mô tả: quan sát người sử dụng làm việc và hỏi họ về những cách mà không được biết tới. Nên nhớ rằng có nhiều nhiệm vụ của người sử dụng thuộc về trực giác và rất khó để mô tả và giải thích chúng. Dựa trên kỹ thuật này ta có thể hiểu thêm về các ảnh hưởng xã hội và tổ chức tác động tới công việc đó.

Lập mẫu thử giao diện người dùng

Mẫu thử cho phép người sử dụng có được những kinh nghiệm trực tiếp với giao diện. Nếu không có những kinh nghiệm trực tiếp như vậy thì không thể đánh giá được khả năng có thể sử dụng được của giao diện.

Lập mẫu thử là một quy trình gồm 2 trạng thái:

- Lập các mẫu thử trên giấy.

- Tinh chỉnh mẫu thử và xây dựng chúng

Các kỹ thuật lập mẫu thử:

- Mẫu thử hướng nguyên mẫu: sử dụng công cụ như Macromedia Director để xây dựng một tập hợp các nguyên mẫu và màn hình. Khi người sử dụng tương tác với chúng thì màn hình sẽ thay đổi để hiển thị trạng thái kế tiếp.

- Lập trình trực quan: sử dụng các ngôn ngữ được thiết kế cho việc phát triển nhanh như Visual Basic.

- Mẫu thử dựa Internet: sử dụng web browser và script.

Đánh giá giao diện người dùng

Ta nên đánh giá bản thiết kế giao diện người dùng để xác định khả năng phù hợp của nó. Tuy nhiên, việc đánh giá trên phạm vi rộng tốn nhiều chi phí và không thể thực hiện được đối với hầu hết các hệ thống.

Các kỹ thuật đánh giá đơn giản:

- Trắc nghiệm lại các phản hồi của người sử dụng

- Ghi lại quá trình sử dụng mẫu thử của hệ thống và đánh giá nó.

- Lựa chọn những thông tin về việc sử dụng dễ dàng và các lỗi của người sử dụng.

- Cung cấp mã lệnh trong phần mềm để thu thập những phản hồi của người sử dụng một cách trực tuyến.

Cải tiến và bảo trì phần mềm

Cải tiến phần mềm

Giới thiệu

Thay đổi phần mềm là một điều không thể tránh khỏi vì những lí do sau:

- Những yêu cầu mới sẽ xuất hiện khi sử dụng phần mềm

- Môi trường nghiệp vụ thay đổi

- Các lỗi phần mềm cần phải sửa chữa

- Máy tính và các thiết bị mới được bổ sung vào hệ thống

- Hiệu năng hoặc độ tin cậy của hệ thống phải được cải thiện.

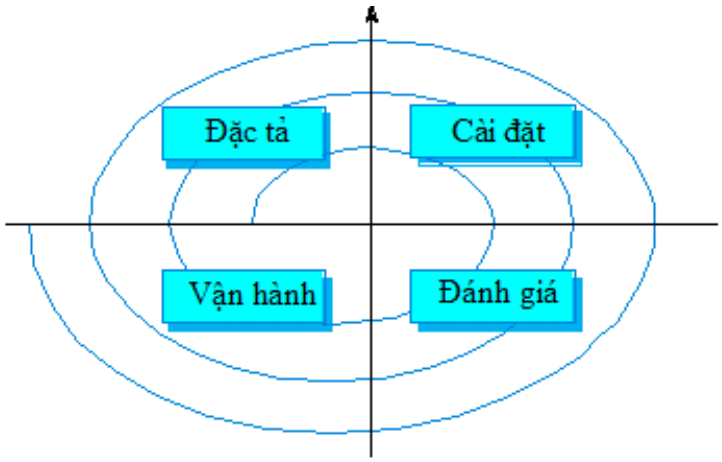
Tuy nhiên, vấn đề quan trọng là chúng ta phải thực hiện và quản lý các thay đổi đối với hệ thống phần mềm đã tồn tại. Và chúng ta phải thấy được tầm quan trọng của việc cải tiến phần mềm:

- Các tổ chức thường đầu tư một lượng vốn khá lớn vào các hệ thống phần mềm của họ. Cho nên họ có quyền đòi hỏi phải sở hữu một hệ thống hoàn hảo.

- Để bảo trì giá trị sở hữu của tổ chức, họ phải thay đổi và cải tiến hệ thống.

- Ngân sách phần mềm chính trong các công ty lớn thường dùng cho việc cải tiến các hệ thống đã tồn tại hơn là phát triển một hệ thống mới.

Người ta thường sử dụng mô hình xoắn ốc để cải tiến hệ thống phần mềm.



Hình 9.1: Mô hình xoắn ốc của quy trình xây dựng và cải tiến

Mục tiêu

- Hiểu được vai trò của việc bảo trì phần mềm

- Nắm được các vấn đề liên quan đến bảo trì: phân loại, phương pháp, chi phí bảo trì …

- Hiểu được một số quy trình và các chiến lược cải tiến phần mềm

Bảo trì phần mềm

Mục tiêu

- Bảo trì phần mềm là làm gì?

- Tại sao phải bảo trì?

- Phân biệt rõ các kiểu bảo trì phần mềm

- Nắm được các yếu tố ảnh hưởng tới chi phí bảo trì, nhằm giảm chi phí khi áp dụng trong thực tế.

Bảo trì phần mềm chính là hoạt động chỉnh sửa chương trình sau khi nó đã được đưa vào sử dụng.

Bảo trì thường không bao gồm những thay đổi chính liên quan tới kiến trúc của hệ thống. Những thay đổi trong hệ thống thường được cài đặt bằng cách điều chỉnh những thành phần đang tồn tại và bổ sung những thành phần mới cho hệ thống.

Bảo trì là không thể tránh khỏi vì:

- Các yêu cầu hệ thống thường thay đổi khi hệ thống đang được xây dựng vì môi trường thay đổi. Vì vậy, hệ thống được chuyển giao có thể không thoả mãn các yêu cầu của nó.

- Các hệ thống có gắn kết chặt chẽ với môi trường của nó. Khi hệ thống được cài đặt trong một môi trường nhất định nó sẽ làm thay đổi môi trường đó và vì vậy sẽ thay đổi các yêu cầu của hệ thống.

- Các hệ thống phải được bảo trì nếu chúng muốn là những phần hữu ích trong môi trường nghiệp vụ.

Phân loại các kiểu bảo trì:

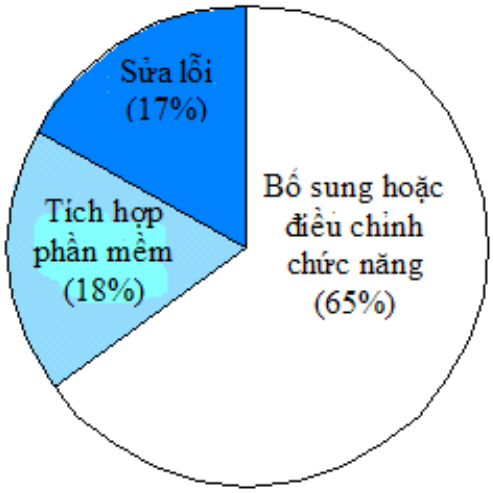
- Bảo trì sửa lỗi: thay đổi hệ thống để sửa lại những khiếm khuyết nhằm thoả mãn yêu cầu hệ thống.

- Bảo trì tích hợp hệ thống vào một môi trường vận hành khác

- Bảo trì để bổ sung hoặc chỉnh sửa các yêu cầu chức năng của hệ thống: chỉnh sửa hệ thống sao cho thoả mãn các yêu cầu mới.

Chi phí bảo trì thường lớn hơn chi phí xây dựng gấp từ 2 đến 100 lần phụ thuộc vào từng ứng dụng. Chi phí bảo trì bị ảnh hưởng bởi cả tác nhân kỹ thuật và phi kỹ thuật.

Nếu bảo trì càng nhiều, sẽ càng làm thay đổi cấu trúc phần mềm và do đó sẽ làm cho việc bảo trì càng trở lên khó khăn hơn. Phần mềm có tuổi thọ càng cao thì càng phải cần chi phí cao hơn (vì sử dụng các ngôn và chương trình dịch cũ …)



Hình 9.2: Phân bổ chi phí bảo trì

Các nhân tố ảnh hưởng đến chi phí bảo trì:

- Sự ổn định của đội dự án: chi phí bảo trì sẽ giảm nếu nhân viên trong đội dự án không thay đổi.

- Những trách nhiệm đã cam kết: người xây dựng hệ thống có thể không cam kết trách nhiệm bảo trì cho nên không có gì để bắt buộc họ phải thiết kế lại cho các thay đổi trong tương lai.

- Kỹ năng của nhân viên: nhân viên bảo trì thường không có kinh nghiệm và hiểu biết về miền ứng dụng của họ bị hạn chế.

- Tuổi thọ và cấu trúc chương trình: khi tuổi thọ và cấu trúc chương trình bị xuống cấp thì chúng càng trở lên khó hiểu và thay đổi nhiều.

Dự đoán bảo trì

Dự đoán bảo trì có liên quan tới việc đánh giá những phần nào của hệ thống có thể gây ra lỗi và cần nhiều chi phí để bảo trì.

Khả năng chịu được sự thay đổi phụ thuộc vào khả năng bảo trì của các thành phần bị ảnh hưởng bởi sự thay đổi đó. Thực hiện các thay đổi có thể làm hỏng hệ thống và giảm khả năng bảo trì của nó.

Chi phí bảo trì phụ thuộc vào số lượng các thay đổi và chi phí thay đổi phụ thuộc vào khả năng bảo trì.

Dự đoán thay đổi

Dự đoán số lượng các thay đổi có thể xảy ra và tìm hiểu mối quan hệ giữa hệ thống và môi trường của nó.

Sự thay đổi yêu cầu hệ thống có liên quan chặt chẽ tới sự thay đổi của môi trường. Trong đó, các nhân tố ảnh hưởng tới mối quan hệ này bao gồm:

- Số lượng và độ phức tạp của các giao diện hệ thống

- Số lượng các yêu cầu bất ổn định có tính phân cấp

- Các quy trình nghiệp vụ của hệ thống.

Ta có thể dự đoán bảo trì thông qua việc đánh giá độ phức tạp của các thành phần hệ thống. Độ phức tạp phụ thuộc vào:

- Độ phức tạp của cấu trúc điều khiển

- Độ phức tạp của cấu trúc dữ liệu

- Kích thước của đối tượng, phương thức và mô-đun.

Ngoài ra, ta có thể sử dụng các phép đo quy trình để đánh giá khả năng bảo trì.

- Số lượng các yêu cầu cần bảo trì sửa lỗi.

- Thời gian trung bình cần thiết để phân tích ảnh hưởng

- Thời gian trung bình để cài đặt một yêu cầu thay đổi.

- Số lượng các yêu cầu cần giải quyết.

Các quy trình cải tiến phần mềm

Các quy trình

Mục tiêu

- Nắm được một số quy trình cải tiến phần mềm cơ bản

- Biết được các chiến lược cải tiến hệ thống

- Có khả năng áp dụng các quy trình cải tiến này trên những hệ thống thực.

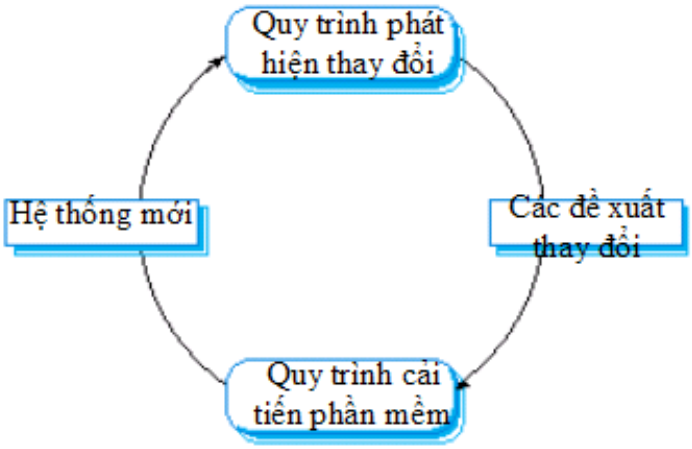
Các quy trình cải tiến phần mềm phụ thuộc vào:

- Kiểu phần mềm cần bảo trì

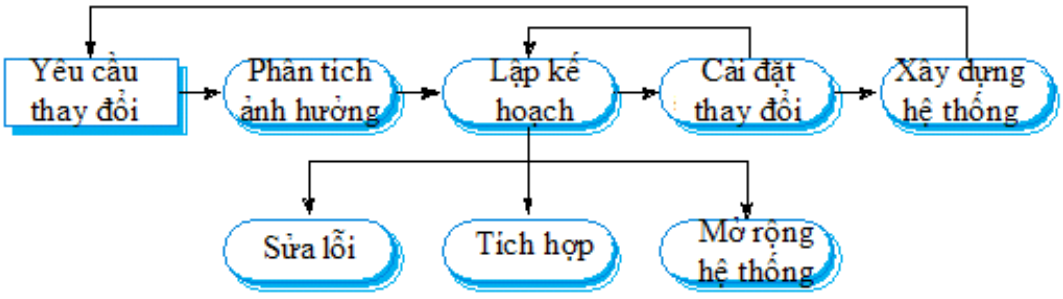
- Quy trình phát triển phần mềm đã được sử dụng

- Kỹ năng và kinh nghiệm của các stakeholder.

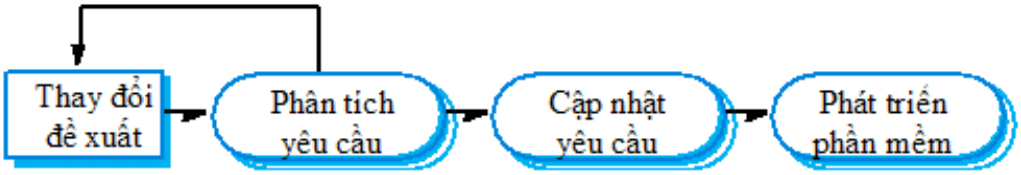
Các đề xuất thay đổi là định hướng để cải tiến hệ thống. Phát hiện thay đổi và cải tiến được thực hiện trong vòng đời hệ thống. Các hình vẽ sau đây thể hiện một cách khái quát các quy trình cải tiến hệ thống.



Hình 9.3: Phát hiện thay đổi và cải tiến



Hình 9.4: Quy trình cải tiến hệ thống



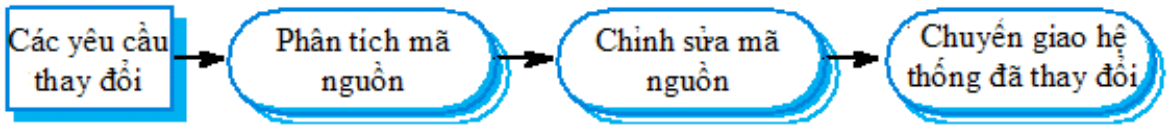
Hình 9.5: Cài đặt thay đổi

Trên đây là những quy trình cơ bản. Tuy nhiên, với các yêu cầu thay đổi khẩn cấp, ta có thể cài đặt chúng nay mà không cần phải trải qua tất cả các pha của quy trình công nghệ phần mềm. Những yêu cầu thay đổi khẩn cấp thường xảy ra khi:

- Nếu có một lỗi hệ thống nghiêm trọng xảy ra và cần phải sửa chữa.

- Nếu những thay đổi về môi trường của hệ thống gây ra những hiệu ứng không mong đợi.

- Nếu sự thay đổi về mặt nghiệp vụ yêu cầu phải có đáp ứng nhanh.



Hình 9.6:Quy trình cài đặt thay đổi khẩn cấp

Để cải tiến hệ thống hiện có, người ta đã đề xuất bốn chiến lược cơ bản:

- Tách hệ thống và chỉnh sửa các quy trình nghiệp vụ

- Tiếp tục bảo trì hệ thống

- Biến đổi hệ thống bằng cách tái kỹ nghệ để nâng cấp khả năng bảo trì của nó.

- Thay thế hệ thống bằng một hệ thống mới

Việc lựa chọn chiến lược cải tiến hệ thống phụ thuộc vào chất lượng hệ thống và giá trị nghiệp vụ của nó.

Các loại hệ thống hiện có được phân loại dựa trên tiêu chí chất lượng và giá trị nghiệp vụ mà nó mang lại như sau:

- Chất lượng thấp và giá trị nghiệp vụ thấp: những hệ thống này nền được tách ra.

- Chất lượng thấp và giá trị nghiệp vụ cao: những hệ thống này có giá trị nghiệp vụ cao nhưng chi phí bảo trì khá lớn. Ta nên tái kỹ nghệ hoặc thay thế bởi một hệ thống thích hợp

- Chất lượng cao và giá trị nghiệp vụ thấp: thay thế bằng các thành phần COTS

- Chất lượng cao và giá trị nghiệp vụ cao: tiếp tục sử dụng và bảo trì hệ thống theo cách thông thường.

Việc đánh giá giá trị nghiệp vụ được thực hiện từ nhiều khung nhìn khác nhau. Phỏng vấn các stakeholder khác nhau và đối sánh kết quả thu được. Các stakeholder thường là:

- Người sử dụng cuối

- Khách hàng của doanh nghiệp

- Người quản lý dây chuyền sản xuất

- Người quản lý công nghệ thông tin

- Người quản lý cao cấp

Đánh giá chất lượng hệ thống thông qua:

- Quy trình nghiệp vụ: quy trình nghiệp vụ đã hỗ trợ cho các mục tiêu nghiệp vụ như thế nào?

- Môi trường hệ thống: môi trường hệ thống có hiệu quả như thế nào và chi phí để bảo trì nó.

- Khả năng ứng dụng: chất lượng của ứng dụng?

Để đo hệ thống, chúng ta có thể thu thập dữ liệu định lượng để tạo ra bản đánh giá về chất lượng của hệ thống.

- Số lượng các yêu cầu thay đổi của hệ thống

- Số lượng các giao diện người dùng khác nhau

- Số lượng dữ liệu được sử dụng trong hệ thống.

Tái kỹ nghệ hệ thống (System re-engineering)

Mục tiêu

- Nắm rõ tái kỹ nghệ hệ thống là gì? Ưu và nhược điểm của nó.

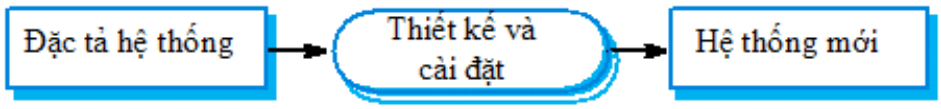
- Quy trình tái kỹ nghệ hệ thống

- Phân biệt forward-engineering và re-engineering

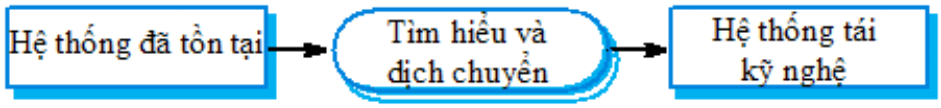
Tái kỹ nghệ hệ thống là kỹ thuật cấu trúc lại hoặc viết lại một phần hoặc toàn bộ hệ thống được thừa kế mà không thay đổi các chức năng của nó.

Tái ký nghệ giúp giảm rủi ro vì trong quá trình xây dựng phần mềm mới rủi ro có thể xảy ra là khá cao và giúp giảm chi phí.

Mô hình sau đây giúp phân biệt forward và re-engineering:



Mô hình forward-engineering



Mô hình re-engineering

Quy trình tái kỹ nghệ bao gồm các hoạt động sau:

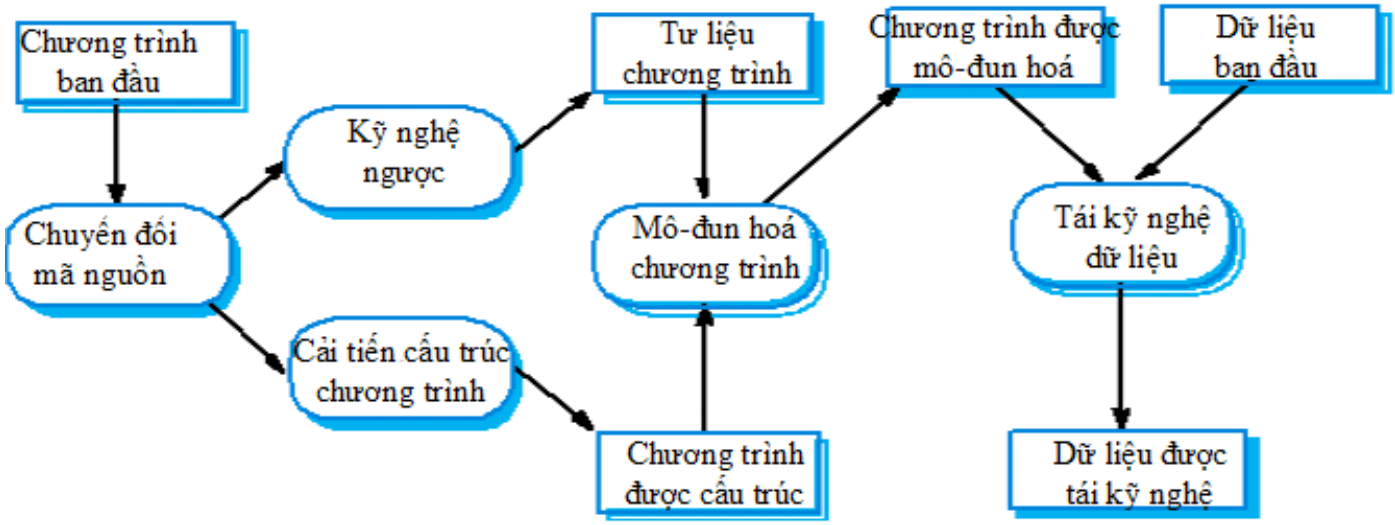
- Dịch mã nguồn: chuyển mã lệnh thành ngôn ngữ mới.

- Kỹ nghệ ngược: phân tích chương trình để tìm hiểu nó.

- Cải thiện cấu trúc chương trình

- Mô-đun hoá chương trình: tổ chức lại cấu trúc chương trình

- Tái kỹ nghệ dữ liệu: thu dọn và cấu trúc lại dữ liệu hệ thống



Các nhân tố ảnh hưởng tới chi phí tái kỹ nghệ:

- Chất lượng của hệ thống được tái kỹ nghệ

- Các công cụ hỗ trợ tái kỹ nghệ

- Mức mở rộng cần thiết của việc chuyển đổi dữ liệu

- Những nhân viên có kỹ năng về tái kỹ nghệ hệ thống

Kiểm thử phần mềm và quy trinh

Kiểm thử phần mềm

Giới thiệu

Kiểm thử là một pha không thể thiếu được trong quá trình phát triển hệ thống. Kiểm thử giúp cho người xây dựng hệ thống và khách hàng đều thấy được rằng hệ thống mới đã thoả mãn yêu cầu đề ra hay chưa.

Mục tiêu

- Nắm được quy trình kiểm thử phần mềm

- Tìm hiểu chi tiết về kiểm thử thành phần và kiểm thử hệ thống; các phương pháp được sử dụng.

- Có khả năng thiết kế các trường hợp kiểm thử và sử dụng các công cụ giúp tự động kiểm thử

Quy trình kiểm thử

Đặt vấn đề

- Sau khi xây dựng xong phần mềm, có thể chuyển giao ngay cho khách hàng không?

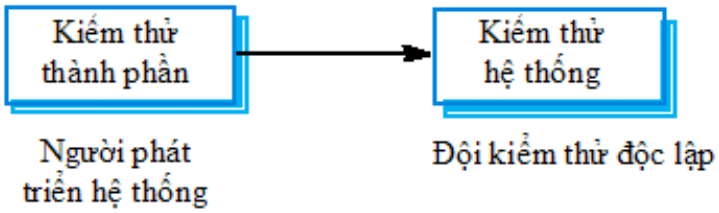
- Tại sao phải kiểm thử?

- Hãy thảo luận về quy trình kiểm thử.

Sau khi cài đặt hệ thống, chúng ta phải kiểm thử để chắc chắn rằng hệ thống đã thoả mãn tất cả các yêu cầu đề ra. Quy trình kiểm thử gồm hai pha:

- Kiểm thử thành phần: kiểm thử từng thành phần riêng biệt. Do người xây dựng thành phần tự thực hiện. Việc kiểm thử được kế thừa từ kinh nghiệm của người xây dựng nó.

- Kiểm thử hệ thống: kiểm thử một tập các thành phần được tích hợp với nhau để tạo ra hệ thống hoặc hệ thống con. Thông thường do một đội kiểm thử độc lập thực hiện. Việc kiểm thử dựa trên tài liệu đặc tả hệ thống.

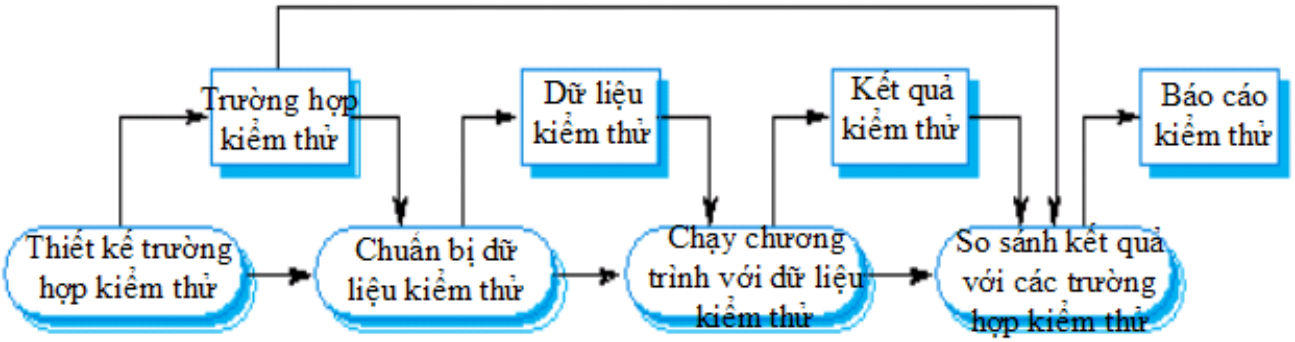


Hình 10.1: Quy trình kiểm thử

Mục đích của quy trình kiểm thử:

- Kiểm thử hợp lệ: để chứng minh cho người xây dựng và khách hàng thấy được phần mềm đã thoả mãn yêu cầu hay chưa. Kiểm thử thành công cho thấy hệ thống đã vận hành như mong đợi.

- Kiểm thử khiếm khuyết: phát hiện lỗi hoặc những khiếm khuyết của phần mềm để thấy được ứng xử của nó có chính xác hoặc phù hợp với tài liệu đặc tả của nó hay không.



Hình 10.2: Quy trình kiểm thử phần mềm

Về mặt lý thuyết, chúng ta phải kiểm thử hệ thống một cách cặn kẽ thì mới khẳng định được chương trình không còn khiếm khuyết. Tuy nhiên, trong thực tế không thể kiểm thử một cách cặn kẽ được.

Các chính sách kiểm thử định nghĩa một phương pháp thường được sử dụng để lựa chọn cách kiểm thử hệ thống:

- Tất cả những chức năng được truy nhập qua menu cần phải kiểm thử

- Các chức năng kết hợp được truy nhập thông qua cùng một menu cũng phải được kiểm thử.

- Những nơi người sử dụng phải nhập thông tin đầu vào thì tất cả các chức năng phải được kiểm thử với những đầu vào chính xác hoặc không chính xác.

Kiểm thử hệ thống ,kiểm thử tích hợp và kiểm thử độc lập

Kiểm thử hệ thống

Mục tiêu

- Nắm được quy trình kiểm thử hệ thống

- Thế nào là kiểm thử tích hợp? Các phương pháp thực hiện.

- Thế nào là kiểm thử độc lập? Các phương pháp thực hiện.

Kiểm thử hệ thống bao gồm tích hợp các thành phần tạo ra hệ thống hoặc hệ thống con; sau đó, kiểm thử hệ thống đã được tích hợp. Kiểm thử hệ thống gồm 2 pha:

- Kiểm thử tích hợp: đội kiểm thử truy nhập vào mã lệnh của hệ thống. Hệ thống cần kiểm thử được coi như các thành phần tích hợp với nhau.

- Kiểm thử độc lập: đội kiểm thử sẽ kiểm thử hệ thống đầy đủ để chuyển giao, coi hệ thống như một hộp đen.

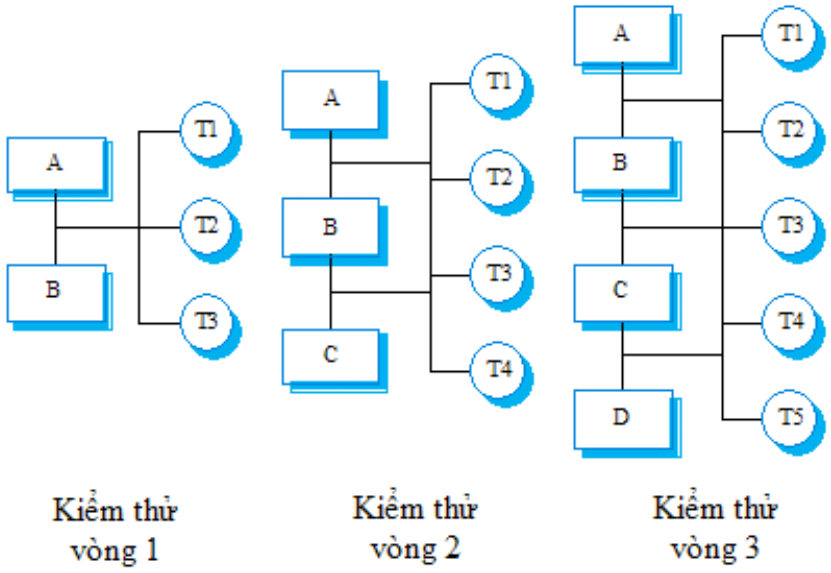
Kiểm thử tích hợp

Kiểm thử tích hợp bao gồm việc xây dựng hệ thống từ những thành phần của nó và kiểm tra xem có vấn đề gì xảy ra từ các tương tác giữa các thành phần. Có hai cách tích hợp hệ thống:

- Tích hợp từ trên xuống: xây dựng khung của hệ thống và đưa các thành phần vào trong nó.

- Tích hợp từ dưới lên: tích hợp các thành phần cơ sở, sau đó bổ sung thêm các thành phần chức năng.

Để đơn giản hoa việc xác định lỗi, hệ thống nên được tích hợp tăng vòng.



Hình 10.3: Mô hình kiểm thử tích hợp tăng vòng

Các phương pháp kiểm thử tích hợp:

- Đánh giá kiến trúc: kiểm thử tích hợp từ trên xuống thích hợp để phát hiện ra các lỗi trong kiến trúc hệ thống.

- Minh hoạ hệ thống: kiểm thử tích hợp từ trên xuống cho phép biểu hiện hệ thống một cách giới hạn ở những pha ban đầu của quá trình xây dựng hệ thống.

- Kiểm thử cài đặt: dễ dàng hơn với kiểm thử tích hợp từ dưới lên.

- Kiểm thử quan sát: các vấn đề của tất cả các phương pháp. Có thể bổ sung thêm các mã lệnh để quan sát các mẫu thử.

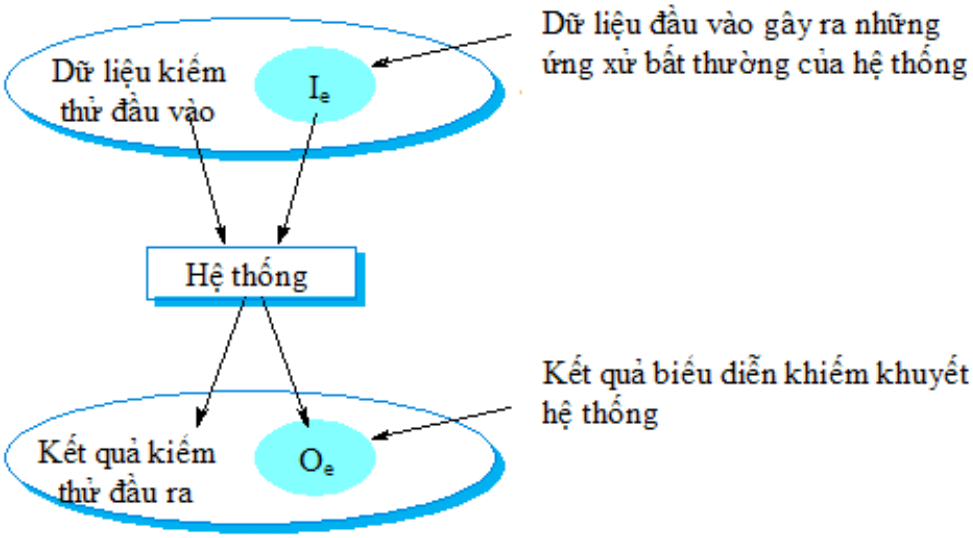
Kiểm thử độc lập

Giới thiệu

Mục đích chính của kiểm thử độc lập nhằm tăng độ tin cậy của nhà cung cấp, đảm bảo hệ thống thoả mãn các yêu cầu của nó.

Kiểm thử độc lập có thể là kiểm thử hộp đen hoặc kiểm thử chức năng; tức là chỉ dựa trên tài liệu đặc tả hệ thống, người kiểm thử không có những hiểu biết về việc cài đặt hệ thống.

Ví dụ: Kiểm thử hộp đen



Chúng ta có thể đưa ra các hướng dẫn kiểm thử cho đội kiểm thử. Hướng dẫn kiểm thử là những gợi ý cho đội kiểm thử giúp họ lựa chọn mẫu thử nhằm phát hiện ra khiếm khuyết của hệ thống.

- Lựa chọn các đầu vào sao cho hệ thống có thể đưa ra tất cả các thông báo lỗi.

- Thiết kế đầu vào sao cho vùng nhớ đệm bị tràn.

- Lặp lại nhiều lần cùng một đầu vào hoặc một chuỗi các đầu vào.

- Ép hệ thống tạo ra những kết quả không hợp lệ.

- Buộc cho các kết quả tính phải quá lớn hoặc quá nhỏ.

Ngoài ra, chúng ta có thể sử dụng ca sử dụng hoặc biểu đồ tuần tự để hỗ trợ cho quá trình kiểm thử. Ca sử dụng có thể là phần cơ bản để đưa ra những mẫu thử hệ thống. Nó giúp xác định các thao tác để kiểm thử và giúp thiết kế các ca sử dụng được yêu cầu. Kèm theo biểu đồ tuần tự tương ứng, chúng ta sẽ sử dụng các đầu ra và đầu vào của nó để tạo ra các mẫu thử.

Kiểm thử độc lập có thể bao gồm kiểm thử các thuộc tính rõ nét của hệ thống như hiệu năng và độ tin cậy.

Kiểm thử hiệu năng bao gồm việc lập kế hoạch cho một tập hợp các mẫu thử và tải trọng của nó có thể tăng lên nhanh chóng cho đến khi hiệu năng của hệ thống là không thể chấp nhận được.

Kiểm thử áp lực thử nghiệm hệ thống trên tải trọng thiết kế tối đa của nó. Áp lực hệ thống thường gây ra những khiếm khuyết của hệ thống.

Kiểm thử áp lực hệ thống xác định những ứng xử lỗi, giúp kiểm tra những lỗi không thể chấp nhận được của các dịch vụ hoặc dữ liệu. Kiểm thử áp lực thích hợp với những hệ thống phân tán.

Các phương pháp kiểm thử

Kiểm thử thành phần

Giới thiệu

Kiểm thử thành phần (hay còn gọi là kiểm thử đơn vị) là quy trình kiểm thử các thành phần riêng lẻ trong hệ thống. Đây là một quy trình phát hiện ra các khiếm khuyết. Thành phần được kiểm thử có thể là:

- Chức năng hoặc phương thức của đối tượng.

- Lớp đối tượng với những thuộc tính và phương thức.

- Thành phần kết hợp với các giao diện được định nghĩa trước để truy nhập tới các chức năng của nó.

Mục tiêu

- Nắm được các vấn đề liên quan đến kiểm thử thành phần

- Hai phương pháp kiểm thử thành phần là: kiểm thử lớp đối tượng và kiểm thử giao diện.

- Ghi nhớ các gợi ý khi kiểm thử thành phần

Kiểm thử lớp đối tượng

Kiểm thử lớp đối tượng nhằm kiểm tra mức độ hoàn thiện của lớp, bao gồm:

- Kiểm thử tất cả các thao tác được gắn với đối tượng.

- Thiết lập và kiểm tra tất cả các thuộc tính của đối tượng.

- Thực nghiệm tất cả các trạng thái có thể của đối tượng

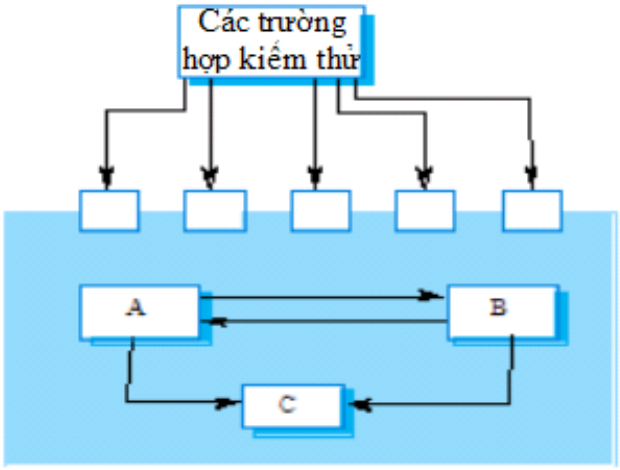
Kỹ thuật thừa kế gây khó khăn cho việc thiết kế kiểm thử lớp đối tượng vì thông tin được kiểm thử không được hạn chế.

Trong quá trình kiểm thử lớp đối tượng, chúng ta cần phải xác định các trường hợp kiểm thử đối với tất cả các phương thức của đối tượng. Đồng thời, sử dụng mô hình trạng thái để xác định chuỗi dịch chuyển trạng thái và chuỗi các sự kiện gây ra sự dịch chuyển đó.

Kiểm thử giao diện

Đặt vấn đề

Mục đích của kiểm thử giao diện là để phát hiện các lỗi của giao diện hoặc những giả thiết không hợp lý về giao diện. Kiểm thử giao diện đặc biệt quan trọng trong phát triển hướng đối tượng khi các đối tượng được định nghĩa bởi các giao diện của nó.



Hình 10.4: Kiểm thử giao diện

Giao diện gồm các loại sau:

- Giao diện tham số: dữ liệu được truyền từ thủ tục này tới thủ tục khác.

- Giao diện bộ nhớ dùng chung: các thủ tục hoặc hàm sử dụng chung khối bộ nhớ.

- Giao diện thủ tục: hệ thống con chứa một tập các thủ tục để các hệ thống con khác gọi tới.

- Giao diện truyền thông điệp: các hệ thống con yêu cầu các dịch vụ từ những hệ thống con khác.

Các loại lỗi thường xảy ra đối với giao diện bao gồm:

- Lạm dụng giao diện: một thành phần gọi tới các thành phần khác và gây ra lỗi trong khi sử dụng giao diện của nó.

- Không hiểu rõ giao diện: thành phần được gắn với các giả thiết về ứng xử của nó với thành phần được gọi, nhưng thành phần này lại sai.

- Lỗi về thời gian: các thành phần gọi và thành phần được gọi thao tác với tốc độ khác nhau và những dữ liệu cũ lại được truy nhập.

Hướng dẫn kiểm thử thành phần:

- Thiết kế các mẫu thử với những tham số gửi tới thủ tục được gọi có giá trị cận biên.

- Luôn luôn kiểm thử các tham số con trỏ với con trỏ null.

- Thiết kế những mẫu thử sao cho có thể gây ra lỗi trên thành phần.

- Thiết kế kiểm thử áp lực trên các hệ thống truyền thông điệp

- Trong những hệ thống có bộ nhớ làm chung, nên biến đổi thứ tự mà trong đó các thành phần tương tác với nhau.

Thiết kế các trường hợp kiểm thử

Mục tiêu

- Tại sao phải thiết kế các trường hợp kiểm thử?

- Có khả năng hiểu và áp dụng các phương pháp thiết kế trường hợp kiểm thử.

Thiết kế các trường hợp kiểm thử (đầu vào và đầu ra) được sử dụng để kiểm thử hệ thống. Mục đích của thiết kế trường hợp kiểm thử là tạo ra một tập hợp các mẫu kiểm thử có khả năng đánh giá hiệu quả và phát hiện khiếm khuyết.

Các phương pháp thiết kế các trường hợp kiểm thử:

- Kiểm thử dựa trên các yêu cầu: Một nguyên tắc của kỹ thuật xác định yêu cầu là những yêu cầu của hệ thống phải có khả năng kiểm thử. Kiểm thử dựa trên yêu cầu là kỹ thuật kiểm thử hợp lệ, trong đó ta phải xem xét từng yêu cầu và đưa ra một tập các mẫu thử cho những yêu cầu đó.

- Kiểm thử phân hoạch: Dữ liệu đầu vào và kết quả đầu ra thường rơi vào các lớp khác nhau, trong đó tất cả các thành viên của lớp đều có quan hệ với nhau. Mỗi lớp này thường là một phân hoạch hoặc một miền ứng dụng mà chương trình chạy theo một cách thích ứng với từng thành viên của lớp. Các trường hợp kiểm thử được lựa chọn từ những phân hoạch này.

- Kiểm thử hướng cấu trúc (hoặc kiểm thử hộp trắng): Kiểm thử hướng cấu trúc đưa ra các trường hợp kiểm thử dựa theo cấu trúc chương trình. Những hiểu biết về chương trình được sử dụng để xác định các trường hợp kiểm thử bổ sung.

- Kiểm thử đường đi: Mục tiêu của kiểm thử đường đi nhằm đảm bảo rằng tập hợp các mẫu thử trên từng đường đi qua hệ thống sẽ được thực hiện ít nhất một lần. Điểm bắt đầu của kiểm thử đường đi là biểu đồ luồng chương trình, gồm các nút biểu diễn các nhánh của chương trình và các cung biểu diễn luồng điều khiển.

Tự động kiểm thử

Giới thiệu

Kiểm thử là một pha có chi phí khá cao. Chúng ta có khá nhiều các công cụ hỗ trợ kiểm thử giúp giảm thời gian và chi phí. Phần này sẽ giới thiệu một số loại công cụ hỗ trợ tự động kiểm thử.

- Quản lý kiểm thử: giúp quản lý các chương trình kiểm thử như lưu vết dữ liệu kiểm thử, các kết quả mong muốn …

- Bộ tạo dữ liệu kiểm thử

- Oracle: tạo ra các dự đoán về kết quả kiểm thử

- Bộ so sánh file: so sánh kết quả của các chương trình kiểm thử

- Bộ tạo báo cáo

- Bộ phân tích động: bổ sung mã lệnh cho chương trình để đếm số lần thực hiện của mỗi câu lệnh.

- Bộ giả định

Tham gia đóng góp

Tài liệu: Nhập môn Công nghệ phần mềm

Biên tập bởi: Phạm Thị Quỳnh

URL: http://voer.edu.vn/c/fb584480

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Phần mềm là gì?

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/e7887c82

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Vấn đề về tính chuyên nghiệp và đúng quy tắc

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/0f087761

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Một số mô hình phát triển phần mềm

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/243ca3cb

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Các hoạt động trong quy trình phần mềm

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/7cc1ff63

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Quản lý dự án

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/eb4928a4

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Một số yêu cầu về nhập môn công nghệ phần mềm

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/65e662bd

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Yêu cầu của người sử dụng

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/aa516279

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Tài liệu đặc tả yêu cầu

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/8ec19c6e

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Phân tích khả thi

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/abc34eb9

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Phát hiện và phân tích yêu cầu

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/503a5c56

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Đánh giá yêu cầu

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/1a227435

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Lập kế hoạch quản lý yêu cầu

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/cbdbf358

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Các mô hình Quản lí

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/00d9a213

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Mô hình ứng xử và máy hệ thống

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/a904f0ba

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Mô hình dữ liệu

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/8ed90335

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Mô hình đối tượng,hệ thống,ứng xử và thừa kế

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/58c05a8f

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Phương pháp hướng cấu trúc

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/ca3686de

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Các vấn đề về thiết kế kiến trúc

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/ae88851b

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Tổ chức hệ thống và các mô hình

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/2f200196

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Phân rã hệ thống và phân rã đối tượng

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/f6cdaa3f

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Các chiến lược điều khiển

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/56047c40

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Các kiến trúc tham chiếu

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/c9416b54

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Thiết kế giao diện người dùng

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/21e8a521

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Quy trình thiết kế giao diện người dùng

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/e15854d5

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Cải tiến và bảo trì phần mềm

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/c321dc36

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Các quy trình cải tiến phần mềm

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/a15f0407

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Kiểm thử phần mềm và quy trinh

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/a3a68952

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Kiểm thử hệ thống ,kiểm thử tích hợp và kiểm thử độc lập

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/47c20c25

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Các phương pháp kiểm thử

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/5eac4108

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Thiết kế các trường hợp kiểm thử

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/355eb7f3

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Tự động kiểm thử

Các tác giả: Phạm Thị Quỳnh

URL: http://www.voer.edu.vn/m/1f842c44

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Chương trình Thư viện Học liệu Mở Việt Nam

Chương trình Thư viện Học liệu Mở Việt Nam (Vietnam Open Educational Resources – VOER) được hỗ trợ bởi Quỹ Việt Nam. Mục tiêu của chương trình là xây dựng kho Tài nguyên giáo dục Mở miễn phí của người Việt và cho người Việt, có nội dung phong phú. Các nội dung đều tuân thủ Giấy phép Creative Commons Attribution (CC-by) 4.0 do đó các nội dung đều có thể được sử dụng, tái sử dụng và truy nhập miễn phí trước hết trong trong môi trường giảng dạy, học tập và nghiên cứu sau đó cho toàn xã hội.

Với sự hỗ trợ của Quỹ Việt Nam, Thư viện Học liệu Mở Việt Nam (VOER) đã trở thành một cổng thông tin chính cho các sinh viên và giảng viên trong và ngoài Việt Nam. Mỗi ngày có hàng chục nghìn lượt truy cập VOER (www.voer.edu.vn) để nghiên cứu, học tập và tải tài liệu giảng dạy về. Với hàng chục nghìn module kiến thức từ hàng nghìn tác giả khác nhau đóng góp, Thư Viện Học liệu Mở Việt Nam là một kho tàng tài liệu khổng lồ, nội dung phong phú phục vụ cho tất cả các nhu cầu học tập, nghiên cứu của độc giả.

Nguồn tài liệu mở phong phú có trên VOER có được là do sự chia sẻ tự nguyện của các tác giả trong và ngoài nước. Quá trình chia sẻ tài liệu trên VOER trở lên dễ dàng như đếm 1, 2, 3 nhờ vào sức mạnh của nền tảng Hanoi Spring.

Hanoi Spring là một nền tảng công nghệ tiên tiến được thiết kế cho phép công chúng dễ dàng chia sẻ tài liệu giảng dạy, học tập cũng như chủ động phát triển chương trình giảng dạy dựa trên khái niệm về học liệu mở (OCW) và tài nguyên giáo dục mở (OER) . Khái niệm chia sẻ tri thức có tính cách mạng đã được khởi xướng và phát triển tiên phong bởi Đại học MIT và Đại học Rice Hoa Kỳ trong vòng một thập kỷ qua. Kể từ đó, phong trào Tài nguyên Giáo dục Mở đã phát triển nhanh chóng, được UNESCO hỗ trợ và được chấp nhận như một chương trình chính thức ở nhiều nước trên thế giới.