2-2009

GIÁO TRÌNH
PHÂN TÍCH THIẾT KẾ HỆ THỐNG

Lưu Hành Nội Bộ

LỜI NÓI ĐẦU

Tài liệu môn phân tích thiết kế hệ thống được chọn lọc theo đề cương môn học ***Phân tích thiết kế hệ thống*** *của Trung Tâm Công Nghệ Thông Tin – trường Đại Học Công Nghiệp Tp. HCM. Tài liệu bao gồm hai nội dung chính****: phân tích thiết kế hệ thống hướng chức năng và phân tích thiết kế hệ thống hướng đối tượng với UML****. Tuy nhiên phần phân tích thiết kế hướng đối tượng chỉ đề cập một số phần chính để sinh viên làm quen, đọc hiểu được các hệ thống hướng đối tượng đã được phân tích, thiết kế. Nếu sinh viên có nhu cầu tham khảo thêm về phần thiết kế hướng đối tượng có thể liên hệ giáo viên giảng dạy môn này để được cung cấp tài liệu.*

TÀI LIỆU THAM KHẢO

1. *Tài liệu Phân tích và thiết kế hệ thống,* trường Đại Học Sư Phạm Hà Nội.
2. *Tài liệu Phân tích và thiết kế hệ thống,* Th.S Trần Đắc Phiến , trường Đại Học Công Nghiệp Tp.HCM.
3. *Giáo trình Phân tích và thiết kế Hệ thống thông tin với UML*, TS. Dương Kiều Hoa – Tôn Thất Hòa An, trường Đại Học Cần Thơ.
4. *Tài liệu Phân tích và thiết kế hệ thống hướng đối tượng*, Trung Tâm đào tạo lập trình viên quốc tế FPT.

NỘI DUNG TÀI LIỆU

PHẦN 1. ĐẠI CƯƠNG VỀ HỆ THỐNG THÔNG TIN

Nội dung

Phần 1 trình bày các khái niệm cơ bản về hệ thống thông tin và quá trình phát triển một hệ thống thông tin.

CHƯƠNG 1. TỔNG QUAN VỀ PHÂN TÍCH THIẾT KẾ HỆ THỐNG

Mục tiêu học tập:

Chương này nhằm cung cấp các khái niệm cơ bản về phân tích và thiết kế hệ thống. Đồng thời đưa ra một quy trình phát triển hệ thống đơn giản.

1.1. Khái niệm hệ thống thông tin

Thông tin là một loại tài nguyên của tổ chức, phải được quản lý chu đáo giống như mọi tài nguyên khác. Việc xử lý thông tin đòi hỏi chi phí về thời gian, tiền bạc và nhân lực. Việc xử lý thông tin phải hướng tới khai thác tối đa tiềm năng của nó.

Hệ thống thông tin (Information System - IS) trong một tổ chức có chức năng thu nhận và quản lý dữ liệu để cung cấp những thông tin hữu ích nhằm hỗ trợ cho tổ chức đó và các nhân viên, khách hàng, nhà cung cấp hay đối tác của nó. Ngày nay, nhiều tổ chức xem các hệ thống thông tin là yếu tố thiết yếu giúp họ có đủ năng lực cạnh tranh và đạt được những bước tiến lớn trong hoạt động. Hầu hết các tổ chức nhận thấy rằng tất cả nhân viên đều cần phải tham gia vào quá trình phát triển các hệ thống thông tin. Do vậy, phát triển hệ thống thông tin là một chủ đề ít nhiều có liên quan tới bạn cho dù bạn có ý định học tập để trở nên chuyên nghiệp trong lĩnh vực này hay không.

Hệ thống thông tin là một hệ thống bao gồm con người, dữ liệu, các quy trình và công nghệ thông tin tương tác với nhau để thu thập, xử lý, lưu trữ và cung cấp thông tin cần thiết ở đầu ra nhằm hỗ trợ cho một hệ thống.

Hệ thống thông tin hiện hữu dưới mọi hình dạng và quy mô.

Phân loại hệ thống thông tin:

Các hệ thống thông tin có thể được phân loại theo các chức năng chúng phục vụ.

Hệ thống xử lý giao dịch (Transaction processing system – TPS) là một hệ thống thông tin có chức năng thu thập và xử lý dữ liệu về các giao dịch nghiệp vụ.

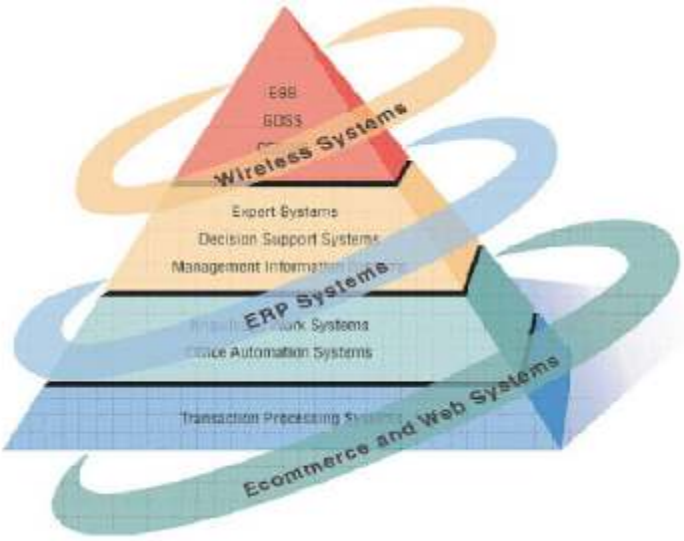
Hệ thống thông tin quản lý (Management information system - MIS) là một hệ thống thông tin cung cấp thông tin cho việc báo cáo hướng quản lý dựa trên việc xử lý giao dịch và các hoạt động của tổ chức.

- **Hệ thống hỗ trợ quyết định** (Decision support system – DSS) là một hệ thống thông tin vừa có thể trợ giúp xác định các thời cơ ra quyết định, vừa có thể cung cấp thông tin để trợ giúp việc ra quyết định.
- **Hệ thống thông tin điều hành** (Excutive information system – EIS) là một hệ thống thông tin hỗ trợ nhu cầu lập kế hoạch và đánh giá của các nhà quản lý điều hành.
- **Hệ thống chuyên gia** (Expert System) là hệ thống thông tin thu thập tri thức chuyên môn của các chuyên gia rồi mô phỏng tri thức đó nhằm đem lại lợi ích cho người sử dụng bình thường.
- **Hệ thống truyền thông và cộng tác** (Communication and collaboration system) là một hệ thống thông tin làm tăng hiệu quả giao tiếp giữa các nhân viên, đối tác, khách hàng và nhà cung cấp để củng cố khả năng cộng tác giữa họ.
- **Hệ thống tự động văn phòng** (Office automation system) là một hệ thống thông tin hỗ trợ các hoạt động nghiệp vụ văn phòng nhằm cải thiện luồng công việc giữa các nhân viên.

Các công nghệ mới:

Các công nghệ mới đang được tích hợp vào các hệ thống truyền thống:

- Thương mại điện tử (e-commerce) sử dụng Web để thực hiện các hoạt động kinh doanh.
- Lập kế hoạch khai thác nguồn tài nguyên doanh nghiệp (ERP-Enterprise Resource Planning) có mục đích tích hợp các hệ thống thông tin khác nhau trong một tổ chức.
- Các thiết bị cầm tay và không dây, bao gồm thương mại di động (m-commerce).
- Phần mềm mã nguồn mở



Hình 1-1 Các công nghệ mới tác động tới tất cả các hệ thống

Phân tích và thiết kế hệ thống là cách tiếp cận có hệ thống tới:

- Việc xác định các vấn đề, cơ hội và mục tiêu
- Việc phân tích các luồng thông tin trong các tổ chức.
- Việc thiết kế các hệ thống thông tin trên máy tính để giải quyết vấn đề

Tài liệu này đề cập tới hai nội dung chính, một là "phân tích" những yêu cầu nghiệp vụ cho các hệ thống thông tin và hai là "thiết kế" các hệ thống thông tin đáp ứng những yêu cầu đó. Nói một cách khác, sản phẩm của quá trình phân tích và thiết kế hệ thống chính là một hệ thống thông tin.

1.2. Một quy trình phát triển hệ thống đơn giản

Trên đây, bạn đã được giới thiệu về các loại hình hệ thống thông tin khác nhau, một số xu hướng công nghệ có ảnh hưởng tới sự phát triển của các hệ thống thông tin. Trong mục này, bạn sẽ học một khía cạnh nữa về hệ thống thông tin, đó là "quy trình" phát triển một hệ thống thông tin.

Hầu hết các quy trình phát triển hệ thống của các tổ chức đều hướng theo cách tiếp cận giải quyết vấn đề (problem-solving). Cách tiếp cận này thường kết hợp các bước giải quyết vấn đề nói chung sau:

1. Xác định vấn đề
2. Phân tích và hiểu vấn đề
3. Xác định các yêu cầu giải pháp
4. Xác định các giải pháp khác nhau và chọn cách "tốt nhất"
5. Thiết kế giải pháp đã lựa chọn

6. Cài đặt giải pháp đã lựa chọn
7. Đánh giá kết quả (nếu vấn đề vẫn không được giải quyết thì quay lại bước 1 hoặc 2)

Nhằm mục đích đơn giản, chúng tôi sẽ trình bày cách tiếp cận giải quyết vấn đề ban đầu gồm bốn giai đoạn hoặc pha cần phải được hoàn thành đối với bất kỳ một dự án phát triển hệ thống nào – đó là pha khởi đầu hệ thống, phân tích hệ thống, thiết kế hệ thống và cài đặt hệ thống. Bảng dưới đây thể hiện quan hệ giữa các bước giải quyết vấn đề nói chung và quy trình được trình bày.

Quy trình phát triển hệ thống đơn giản hóa	**Các bước giải quyết vấn đề nói chung**
Khởi đầu hệ thống	1. Xác định vấn đề. (Đồng thời lập kế hoạch cho giải pháp của vấn đề).
Phân tích hệ thống	1. Phân tích và hiểu vấn đề . 2. Xác định các yêu cầu giải pháp.
Thiết kế hệ thống	1. Xác định các giải pháp khác nhau và chọn cách "tốt nhất" 2. Thiết kế giải pháp đã lựa chọn
Cài đặt hệ thống	1. Cài đặt giải pháp đã lựa chọn 2. Đánh giá kết quả. (Nếu vấn đề vẫn không được giải quyết thì quay lại bước 1 hoặc 2).

Bảng 1-1 Quy trình phát triển hệ thống

Cần lưu ý là bất cứ quy trình phát triển hệ thống nào cũng phải được quản lý trên cơ sở dự án. Phải có ít nhất một nhân sự nhận trách nhiệm làm người quản lý dự án để đảm bảo rằng hệ thống được phát triển đúng thời gian, trong giới hạn ngân sách cho phép và có chất lượng chấp nhận được. Hoạt động quản lý một dự án được gọi là quản lý dự án

Quản lý dự án *(Project Management) là hoạt động xác định, lập kế hoạch, điều khiển, kiểm soát một dự án để phát triển một hệ thống chấp nhận được trong khoảng thời gian và ngân sách được giao*

Quản lý quy trình *(Process Management) là hoạt động liên tục nhằm xác định, cải thiện và kết hợp việc sử dụng phương pháp luận mà tổ chức đã lựa chọn ("quy trình") với các tiêu chuẩn đối với mọi dự án phát triển hệ thống.*

1.2.1. Khởi đầu hệ thống

Các dự án hệ thống thông tin thường phức tạp. Chúng đòi hỏi sự đầu tư, nỗ lực và thời gian đáng kể. Các vấn đề cần giải quyết thường được phát biểu một cách mơ hồ, có nghĩa rằng giải pháp được hình dung ban đầu có thể còn chưa hoàn thiện. Vì vậy, các dự án hệ thống phải được lập kế hoạch cẩn thận. Giai đoạn khởi đầu hệ thống hình thành phạm vi dự án và kế hoạch giải quyết vấn đề. Do đó, pha khởi đầu hệ thống thiết lập phạm vi dự án, mục tiêu, lịch biểu và ngân sách cần thiết để giải quyết vấn đề.

Phạm vi dự án xác định lĩnh vực nghiệp vụ được hướng đến của dự án và các mục tiêu cần đạt được. Phạm vi và mục tiêu về cơ bản đều ảnh hưởng tới các đảm bảo về tài nguyên, cụ thể là lịch biểu và ngân sách, những nhân tố cần được thực hiện để hoàn thành dự án. Bằng việc thiết lập một ngân sách và lịch biểu dựa vào phạm vi và mục tiêu ban đầu, bạn cũng sẽ thiết lập được một ranh giới mà dựa vào đó tất cả các nhân sự đều có thể chấp nhận thực tế là bất cứ thay đổi nào trong tương lai đối với phạm vi hoặc mục tiêu cũng sẽ tác động tới lịch biểu và ngân sách.

Người quản lý dự án, người phân tích hệ thống và người sở hữu hệ thống là những nhân lực chủ yếu trong pha khởi đầu hệ thống.

Khởi đầu hệ thống *(System Initiation) là việc lập kế hoạch ban đầu cho một dự án để xác định phạm vi nghiệp vụ, mục tiêu, lịch biểu và ngân sách ban đầu.*

1.2.2. Phân tích hệ thống

Bước tiếp theo trong quy trình phát triển hệ thống mà chúng tôi trình bày là giai đoạn phân tích hệ thống. Pha này nhằm cung cấp cho đội dự án hiểu biết thấu đáo hơn về vấn đề và nhu cầu của dự án. Hiểu một cách đơn giản, lĩnh vực nghiệp vụ (phạm vi của dự án – như đã xác định trong pha khởi đầu hệ thống) có thể được nghiên cứu và phân tích để thu được những hiểu biết chi tiết hơn. Pha phân tích hệ thống yêu cầu làm việc với người sử dụng hệ thống để xác định rõ các yêu cầu nghiệp vụ đối với hệ thống sẽ được mua hoặc phát triển.

Sự hoàn thiện của pha phân tích hệ thống thường thể hiện kết quả ở nhu cầu cập nhật các kết quả đã có trước đó ở pha khởi đầu hệ thống. Việc phân tích có thể phát hiện yêu cầu phải xét lại phạm vi hoặc mục tiêu của dự án – ví dụ có thể cảm thấy phạm vi của dự án quá lớn hoặc quá nhỏ. Cuối cùng, tính khả thi của bản thân dự án trở nên đáng ngờ. Dự án có thể bị hủy bỏ hoặc có thể chuyển sang giai đoạn tiếp theo.

Người quản lý dự án, người phân tích hệ thống và người sử dụng hệ thống là những nhân lực cơ bản trong pha phân tích hệ thống.

Phân tích hệ thống *(System Analysis) là việc nghiên cứu lĩnh vực vấn đề nghiệp vụ để đề xuất các cải tiến và xác định các yêu cầu nghiệp vụ cũng như thứ tự ưu tiên cho giải pháp.*

1.2.3. Thiết kế hệ thống

Sau khi đã có hiểu biết về các yêu cầu nghiệp vụ của một hệ thống thông tin, ta có thể tiến hành pha thiết kế hệ thống. Trong giai đoạn này, trước tiên cần xem xét các giải pháp công nghệ khác nhau. Hiếm khi chỉ có một giải pháp cho một vấn đề.

Một khi một giải pháp đã được lựa chọn và chấp nhận, pha thiết kế hệ thống phát triển các bản đặc tả và thiết kế chi tiết được yêu cầu để cài đặt giải pháp cuối cùng. Các bản đặc tả và thiết kế chi tiết đó sẽ được dùng để cài đặt cơ sở dữ liệu, chương trình, giao diện người dùng và mạng cho hệ thống thông tin. Trong trường hợp ta lựa chọn mua phần mềm thay vì xây dựng nó thì bản thiết kế chi tiết sẽ xác định cách thức phần mềm đó được tích hợp vào hoạt động nghiệp vụ và các hệ thống thông tin khác.

Nhắc lại về các định hướng công nghệ đã trình bày ở trên, các định hướng đó sẽ ảnh hưởng chủ yếu tới quy trình thiết kế hệ thống và ra quyết định. Nhiều tổ chức xác định một kiến trúc công nghệ thông tin chung dựa trên các định hướng công nghệ đó. Nếu vậy, tất cả các pha thiết kế hệ thống cho hệ thống thông tin mới đều phải tuân theo kiến trúc công nghệ thông tin chuẩn.

Người quản lý dự án, người phân tích hệ thống và người thiết kế hệ thống là những nhân lực chính trong pha thiết kế hệ thống.

Thiết kế hệ thống *(System Design) là quá trình xác định và xây dựng giải pháp kỹ thuật dựa trên máy tính cho các yêu cầu nghiệp vụ được xác định trong pha phân tích hệ thống.*

1.2.4. Cài đặt hệ thống

Bước cuối cùng trong quy trình phát triển hệ thống đơn giản mà chúng tôi trình bày là cài đặt hệ thống. Pha cài đặt hệ thống xây dựng hệ thống thông tin mới và đưa nó vào hoạt động. Trong giai đoạn này, các phần cứng và phần mềm được cài đặt và sử dụng. Các phần mềm ứng dụng được mua và cơ sở dữ liệu được cài đặt và cấu hình. Các phần mềm tùy biến và cơ sở dữ liệu được xây dựng dựa trên các bản đặc tả và thiết kế chi tiết được phát triển ở pha thiết kế hệ thống.

Khi các thành phần hệ thống đã được xây dựng hoặc cài đặt thì chúng phải được kiểm thử riêng rẽ. Sau đó, toàn bộ hệ thống cũng phải được kiểm thử để đảm bảo rằng nó hoạt động chính xác và đáp ứng được các yêu cầu của người dùng. Một khi hệ thống

đã được kiểm thử đầy đủ, nó phải được đưa vào hoạt động. Dữ liệu từ hệ thống trước đó có thể phải được chuyển đổi hoặc nhập vào cơ sở dữ liệu khởi đầu và người sử dụng hệ thống phải được đào tạo để sử dụng hệ thống một cách chuẩn xác. Cuối cùng, một số kế hoạch chuyển tiếp từ quy trình nghiệp vụ và hệ thống thông tin cũ có thể phải được tiến hành.

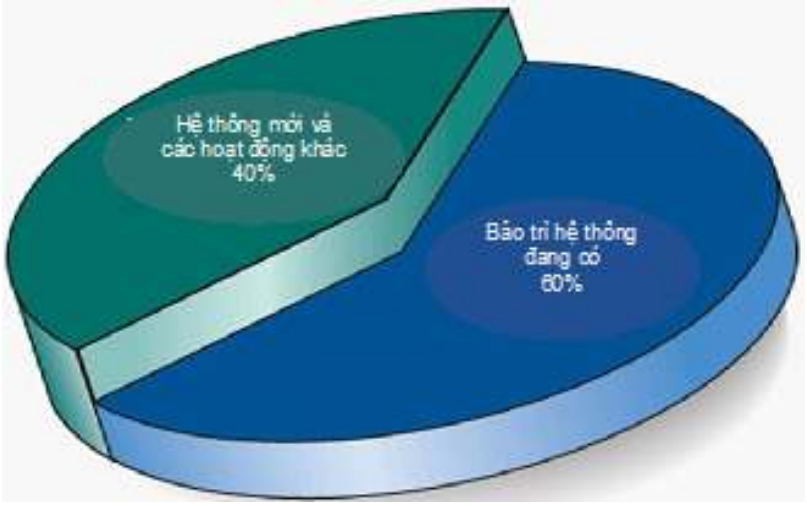
Người quản lý dự án, người phân tích hệ thống và người xây dựng hệ thống là những nhân lực chủ yếu trong giai đoạn cài đặt hệ thống.

Cài đặt hệ thống *(System Implementation) là giai đoạn xây dựng, cài đặt, kiểm thử và triển khai một hệ thống.*

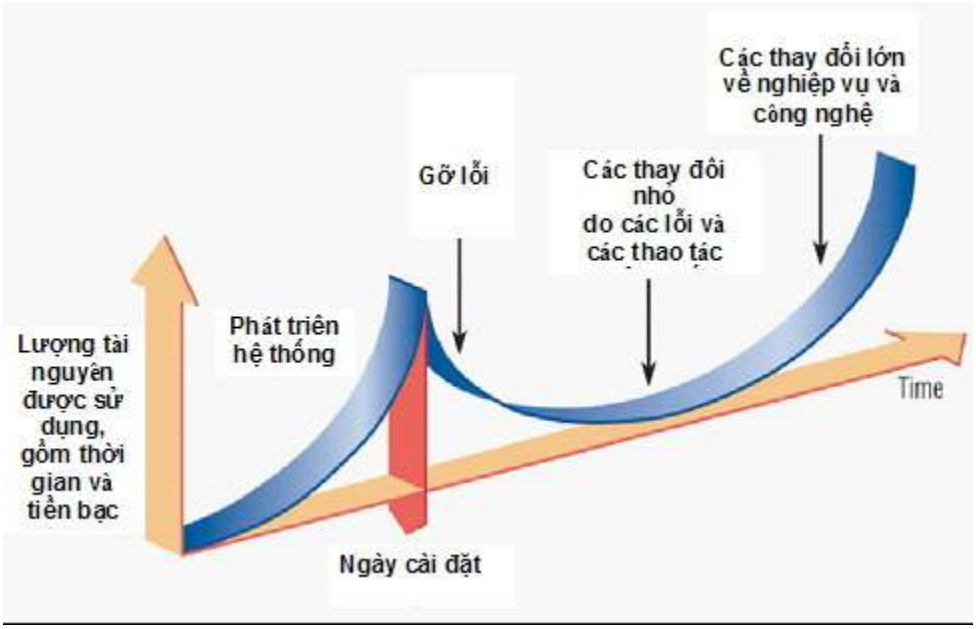
1.2.5. Hỗ trợ hệ thống và cải thiện không ngừng

Sẽ thật thiếu xót nếu không khẳng định rằng việc cài đặt hệ thống thông tin sẽ dẫn tới việc phải đối mặt với sự tồn tại của giai đoạn hỗ trợ và cải thiện không ngừng. Các hệ thống thông tin được cài đặt rất hiếm khi hoàn hảo. Những người sử dụng sẽ tìm thấy lỗi và thỉnh thoảng bạn sẽ tìm thấy những sai sót trong thiết kế và cài đặt cần được sửa chữa. Ngoài ra, các yêu cầu nghiệp vụ và của người dùng thay đổi không ngừng. Do đó, sẽ có nhu cầu cải thiện không ngừng bất kỳ hệ thống thông tin nào tới khi nó lỗi thời.

Hỗ trợ và cải thiện hệ thống thuộc về một dự án khác, thường được gọi là dự án bảo trì và nâng cấp. Một dự án như vậy cần tuân theo cùng cách tiếp cận giải quyết vấn đề như đã được xác định với bất kỳ dự án nào khác. Điểm khác biệt duy nhất là nỗ lực và ngân sách cần để hoàn thành dự án. Nhiều pha sẽ được hoàn thành nhanh hơn nhiều, đặc biệt là nếu nhân lực ban đầu đã tài liệu hóa một cách đúng đắn hệ thống ngay từ giai đoạn đầu. Lẽ tất nhiên, nếu họ không làm như vậy thì dự án cải thiện hệ thống có thể tiêu tốn nhiều thời gian, nỗ lực và tiền bạc hơn.



Hình 1-2 Tỉ lệ thời gian cho việc bảo trì hệ thống



Hình 1-3 Mức sử dụng tài nguyên trong quy trình phát triển hệ thống

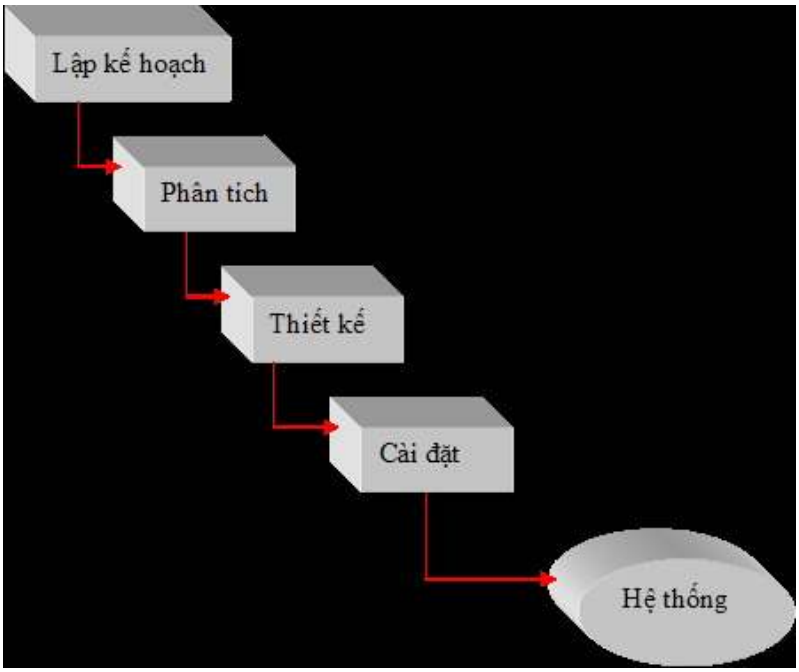
1.2.6. Phát triển tuần tự và phát triển lặp

Tất cả nội dung trình bày ở các mục trên có thể khiến bạn kết luận rằng phát triển hệ thống là một quy trình tuần tự một cách tự nhiên. Trước tiên, bạn khởi đầu dự án, rồi phân tích, thiết kế và cuối cùng là triển khai hệ thống. Điều này không phải là luôn đúng đắn. Có các chiến lược hoặc cách tiếp cận khác nhau để thực hiện quy trình phát triển hệ thống nói chung.

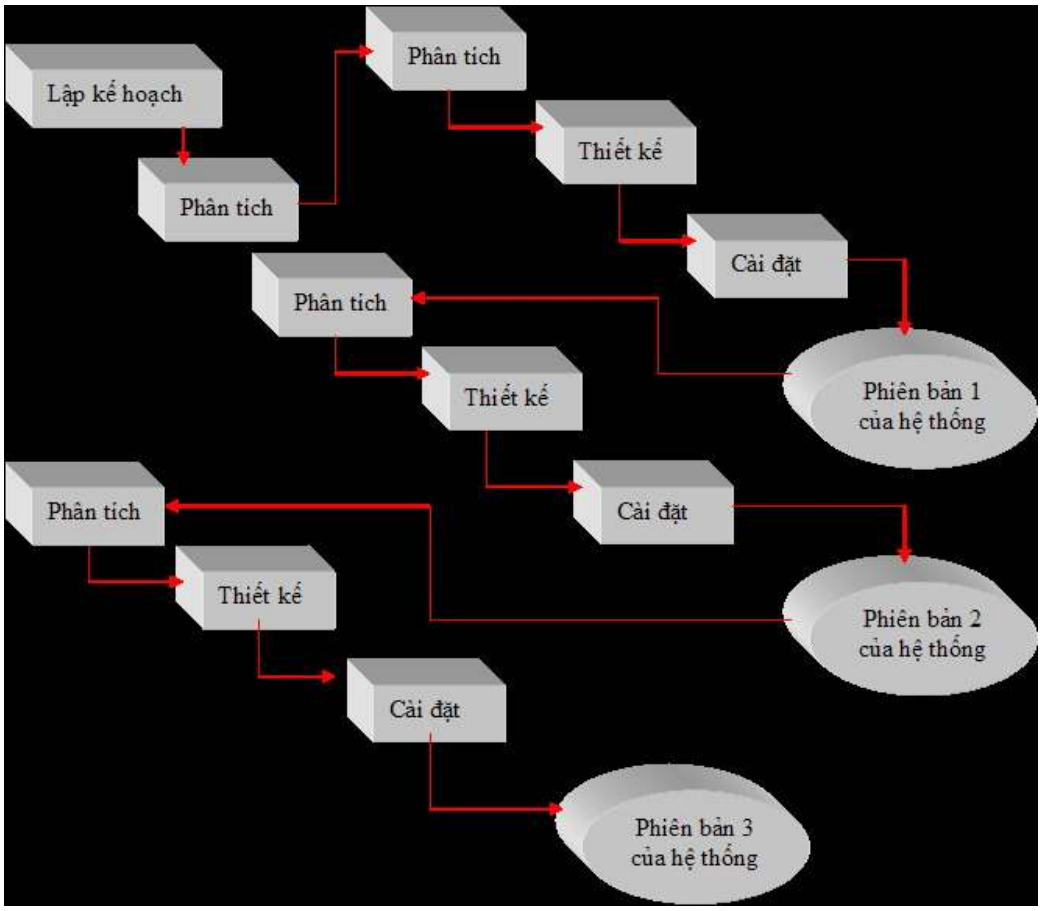
Rõ ràng các quy trình tuần tự là một trong các khả năng. Cách tiếp cận này được minh họa trong hình 1- 4. Chú ý rằng chiến lược này đòi hỏi mỗi pha phải được hoàn thành - cái này tiếp sau cái kia. Sự hoàn thành tuần tự sẽ cho kết quả trong sự phát triển một hệ thống hoàn toàn mới. Hình thức trực quan của cách tiếp cận này giống như một thác nước (waterfall) nên nó thường được gọi là quy trình "phát triển thác nước".

(Trong thực tế, các giai đoạn có thể chồng lấp lên nhau. Ví dụ phần thiết kế hệ thống có thể được bắt đầu trước khi hoàn thành giai đoạn phân tích hệ thống.).

Tuy nhiên, cách tiếp cận thác nước không còn được dùng phổ biến. Vì có một chiến lược phổ biến hơn. thể hiện trong hình 1-5, thường được gọi là quy trình phát triển lặp. Cách tiếp cận này đòi hỏi hoàn thành việc phân tích, thiết kế và cài đặt đủ để phát triển đầy đủ một phần của hệ thống mới và đưa nó vào hoạt động sớm nhất có thể. Một khi "phiên bản" đó của hệ thống được cài đặt, chiến lược tiếp theo là thực hiện thêm một số việc phân tích, thiết kế và cài đặt để tạo ra phiên bản tiếp theo của hệ thống. Quá trình lặp đi lặp lại tới khi tất cả các phần của hệ thống thông tin tổng thể được cài đặt. Sự phổ biến của quy trình lặp này có thể giải thích như sau: Người sở hữu và sử dụng hệ thống phàn nàn về thời gian quá dài cần để phát triển và cài đặt các hệ thống thông tin khi sử dụng cách tiếp cận thác nước. Trong khí đó, cách tiếp cận lặp cho phép đưa vào sử dụng các phiên bản với thời gian ngắn hơn. Điều này sẽ thỏa mãn đòi hỏi của khách hàng.



Hình 1-4 Phương pháp luận phát triển theo mô hình thác nước



Hình 1-5 Phương pháp luận phát triển lặp

Câu hỏi thảo luận

1.1. Nêu chức năng và vai trò của hệ thống thông tin trong một tổ chức.

1.2. Phân biệt hệ thống quản lý giao dịch (TPS) với hệ thống thông tin quản lý (MIS) và hệ thống thông tin điều hành (EIS).

1.3. Nêu ví dụ về hệ thống quản lý giao dịch, hệ thống thông tin quản lý và hệ thống truyền thông cộng tác.

GỢI Ý: Dựa vào các định nghĩa của các hệ thống để đưa ra ví dụ phù hợp trong thực tế.

1.4. Cho biết các xu thế công nghệ mới đang được đưa vào các hệ thống thông tin?

1.5. Nêu các giai đoạn của một quy trình phát triển hệ thống đơn giản.

GỢI Ý: Dựa vào mục 1.2, nêu tóm tắt các giai đoạn trong một quá trình đơn giản để phát triển hệ thống.

Câu hỏi trắc nghiệm

1. Một hệ thống thông tin được hình thành bởi

- a. Con người, các quy trình và dữ liệu
- b. Thiết bị, các quy trình và công nghệ
- c. Con người, dữ liệu, các quy trình và công nghệ

2. Hệ thống thông tin quản lý là:

- a. Hệ thống thông tin có chức năng thu thập và xử lý dữ liệu về các giao dịch nghiệp vụ.
- b. Hệ thống thông tin hỗ trợ nhu cầu lập kế hoạch và đánh giá của các nhà quản lý điều hành.
- c. Hệ thống thông tin cung cấp thông tin cho việc báo cáo hướng quản lý dựa trên việc xử lý giao dịch và các hoạt động của tổ chức.

Tổng kết chương 1

Chương 1 đã giới thiệu những khái niệm cơ bản về hệ thống thông tin và quy trình đơn giản để phát triển một hệ thống.

Người học cần nắm vững khái niệm về hệ thống thông tin, phân biệt được giữa các loại hệ thống thông tin và có thể đưa ra ví dụ trong thực tế:

- Hệ thống xử lý giao dịch
- Hệ thống thông tin quản lý
- Hệ thống hỗ trợ quyết định
- Hệ thống thông tin điều hành
- Hệ thống chuyên gia
- Hệ thống truyền thông và cộng tác
- Hệ thống tự động văn phòng

Ngoài ra, người học cần nắm rõ các giai đoạn cơ bản trong quá trình phát triển một hệ thống thông tin:

- Khởi đầu hệ thống.
- Phân tích hệ thống.
- Thiết kế hệ thống.
- Cài đặt hệ thống.

Hướng dẫn bài tập lớn

Bài tập lớn là tiêu chuẩn đánh giá kết quả học tập. Yêu cầu là thực hiện Phân tích và thiết kế một hệ thống thông tin quản lý. Các bạn chọn một trong các đề tài gợi ý dưới đây hoặc tự đưa ra đề tài.

STT	Đề tài	Mô tả
1	Hệ thống quản lý thư viện	Quản lý sách, độc giả, tình hình mượn trả…
2	Hệ thống quản lý nhân sự tiền lương	Quản lý hồ sơ nhân sự, tiền lương, kết quả chấm công…
3	Hệ thống quản lý vật tư	Quản lý tình hình xuất nhập vật tư…
4	Hệ thống quản lý học sinh phổ thông	Quản lý hồ sơ học sinh, kết quả học tập…
5	Hệ thống quản lý sinh viên	Quản lý hồ sơ sinh viên, kết quả học tập…

	đại học	
6	Hệ thống quản lý tuyển sinh đại học	Quản lý thông tin thí sinh, kết quả thi…
7	Hệ thống quản lý công nợ cửa hàng	Quản lý tình hình công nợ đối với khách hàng và nhà cung cấp
8	Hệ thống quản lý khách sạn	Quản lý phòng, khách thuê phòng…
9	Hệ thống quản lý hồ sơ bệnh án bệnh viện	Quản lý hồ sơ, tình trạng của bệnh nhân…
10	Hệ thống quản lý trường mẫu giáo	Quản lý hồ sơ học sinh, chỉ số thể lực, năng khiếu…

Việc làm bài tập lớn nên được tiến hành song song với quá trình học môn này và tuân theo các bước chính:

1. Khảo sát thực tế và xác lập các yêu cầu.
2. Phân tích hệ thống.
3. Thiết kế hệ thống.
4. Xây dựng và cài đặt hệ thống.

Kết quả làm bài tập lớn phải được thể hiện thông qua một báo cáo dạng file word và mã nguồn chương trình phần mềm được xây dựng.

CHƯƠNG 2. PHÁT TRIỂN HỆ THỐNG THÔNG TIN

Mục tiêu

Chương này giới thiệu khái niệm về quy trình phát triển một hệ thống thông tin, đưa ra một quy trình phát triển hệ thống. Bên cạnh đó, chương này cũng sẽ đề cập tới các hướng tiếp cận phân tích thiết kế hệ thống. Cuối cùng sẽ giới thiệu về các kỹ thuật và công cụ có thể sử dụng trong quá trình phân tích thiết kế hệ thống.

2.1. Quy trình phát triển hệ thống

2.1.1. Khái niệm

Quy trình phát triển hệ thống – một tập hợp các hoạt động, phương pháp, thực nghiệm, kết quả và các công cụ tự động hóa mà các nhân sự sử dụng để phát triển và cải thiện không ngừng hệ thống thông tin và phần mềm

Một quy trình phù hợp để phát triển hệ thống phải bảo đảm:

- *Hiệu quả* để cho phép nhà quản lý điều chuyển nguồn lực giữa các dự án
- *Tài liệu nhất quán* nhằm giảm chi phí thời gian sống để bảo trì hệ thống (bởi các đội phát triển khác) về sau
- *Chất lượng nhất quán* xuyên suốt các dự án

2.1.2. Mô hình quản lý quy trình CMM

Capability Maturity Model (CMM) là một framework chuẩn hóa để đánh giá mức độ hoàn thiện của các quy trình phát triển hệ thống thông tin, các quy trình quản lý và các sản phẩm của một tổ chức. Mục đích của CMM là để hỗ trợ cho các tổ chức cải thiện tính hoàn chỉnh của các quy trình phát triển hệ thống. Nó bao gồm 5 mức độ hoàn thiện:

- **Mức 1—Khởi đầu**: ở mức này, các dự án phát triển hệ thống không tuân theo quy trình bắt buộc nào. Mỗi đội phát triển lại có những công cụ và phương pháp riêng. Sự thành công hay thất bại thường phụ thuộc vào kỹ năng và kinh nghiệm của đội dự án.
- **Mức 2—Có thể lặp lại**: Các quy trình quản lý và thực hiện dự án được thiết lập để theo dõi chi phí dự án, lịch biểu và tính thiết thực. Các dự án đều sử dụng một quy trình phát triển hệ thống nhưng quy trình đó có thể biến đổi phù hợp với từng dự án. Đội dự án nỗ lực phối hợp để có thể lặp lại những kết quả tốt đã đạt được . Những kinh nghiệm thực tiến được áp dụng để chuẩn hóa quy trình cho mức kế tiếp.
- **Mức 3—Được định rõ**: Một quy trình phát triển hệ thống chuẩn (một "phương pháp luận") được mua về hoặc được phát triển. Tất cả các dự án sử

dụng một phiên bản của quy trình này để phát triển và bảo trì hệ thống thông tin và phần mềm. Nhờ việc sử dụng quy trình chuẩn mà mỗi dự án đều mang tính nhất quán về tài liệu và kết quả sản phẩm thu được.

- **Mức 4—Được quản lý**: Các mục tiêu đo được về chất lượng và hiệu quả phải được thiết lập. Các kết quả đo chi tiết về chất lượng quy trình phát triển hệ thống chuẩn và chất lượng sản phẩm luôn được thu thập và lưu trữ vào cơ sở dữ liệu. Đội dự án dựa vào những dữ liệu đó để cải thiện việc quản lý từng dự án.
- **Mức 5—Tối ưu**: Quy trình phát triển hệ thống chuẩn được giám sát và cải thiện không ngừng dựa trên các phép đo và phân tích dữ liệu được thiết lập trong mức 4. Có thể bao gồm việc thay đổi kỹ thuật, công nghệ để thực hiện các hoạt động được đòi hỏi trong quy trình phát triển hệ thống chuẩn, cũng như việc điều chỉnh chính quy trình.

Cần nhận thấy rằng mỗi mức CMM lại là tiền điều kiện cho mức tiếp theo. Hiện tại, trên thế giới, nhiều tổ chức đang nỗ lực để đạt được ít nhất là CMM mức 3.

2.1.3. Phương pháp luận phát triển hệ thống

- **Vòng đời hệ thống** – là sự phân tích vòng đời của một hệ thống thông tin thành hai giai đoạn, (1) phát triển hệ thống và (2) đưa vào hoạt động và bảo trì hệ thống
- **Phương pháp luận phát triển hệ thống** – là một quy trình phát triển chuẩn hóa xác định một tập các hoạt động, phương pháp, thực nghiệm, kết quả và các công cụ tự động hóa mà những người phát triển hệ thống và người quản lý dự án dùng để phát triển và cải thiện không ngừng các hệ thống thông tin và phần mềm
- Các phương pháp luận phát triển hệ thống
    - Phát triển ứng dụng nhanh có kiến trúc (Architected Rapid Application Development - Architected RAD)
    - Phương pháp luận phát triển hệ thống động (Dynamic Systems Development Methodology - DSDM)
    - Phát triển ứng dụng kết hợp (Joint Application Development - JAD)
    - Công nghệ thông tin (Information Engineering - IE)
    - Phát triển ứng dụng nhanh (Rapid Application Development - RAD)
    - Quy trình hợp nhất Rational (Rational Unified Process - RUP)

- **Phân tích và thiết kế hướng cấu trúc (Structured Analysis and Design) –** *đây là phương pháp được trình bày trong bài giảng này*
- Lập trình eXtreme (eXtremeProgramming - XP)

2.1.4. Các nguyên lý phát triển hệ thống

- Để người sử dựng hệ thống tham gia vào
- Sử dụng một cách tiếp cận giải quyết vấn đề
- Thiết lập các giai đoạn và các hoạt động
- Tài liệu hóa suốt quá trình phát triển
- Thiết lập các chuẩn
- Quản lý quá trình và các dự án
- Cân đối hệ thống với vốn đầu tư
- Không né tránh việc hủy bỏ hoặc sửa phạm vi
- Chia để trị
- Thiết kế hệ thống để có thể phát triển và thay đổi

Nguyên lý 1: Để người sở hữu và người sử dụng hệ thống tham gia vào tất cả các giai đoạn phát triển hệ thống

- Sự tham gia của người sử dụng sẽ tạo nên ý thức họ là người làm chủ hệ thống và dẫn đến sự chấp nhận và hài lòng của họ về hệ thống
- Có nghĩa là người sử dụng và người sở hữu hệ thống cũng "sống" trong hệ thống

Nguyên lý 2: Sử dụng một cách tiếp cận giải quyết vấn đề

- Nghiên cứu và tìm hiểu vấn đề trong ngữ cảnh của nó
- Xác định các yêu cầu của giải pháp phù hợp
- Xác định các giải pháp đề cử và chọn giải pháp tốt nhất có thể
- Thiết kế và/hoặc cài đặt giải pháp
- Quan sát và đanh giá tác động của giải pháp, và cải thiện giải pháp một cách phù hợp

Nguyên lý 3: Thiết lập các giai đoạn và các hoạt động

- Xác định phạm vi
- Phân tích vấn đề
- Phân tích yêu cầu
- Thiết kế lôgíc

- Phân tích quyết định
- Thiết kế vật lý và tích hợp
- Xây dựng và kiểm thử
- Cài đặt và đưa vào hoạt động

Các giai đoạn trên xác định các vấn đề, đánh giá, thiết kế và cài đặt giải pháp (Quy trình phát triển hệ thống)

Nguyên lý 4: Tài liệu hóa suốt quy trình phát triển hệ thống

- Là hoạt động liên tiếp để phát hiện điểm mạnh và điểm yếu của hệ thống trong suốt quy trình phát triển
- Củng cố sự truyền đạt thông tin giữa các nhân sự trong hệ thống
- Sự tán thành và giao kèo giữa người sở hữu/người sử dụng với người phân tích/người thiết kế về phạm vi, yêu cầu và tài nguyên của dự án

Nguyên lý 5: Thiết lập các chuẩn về tính nhất quán

- Các chuẩn phát triển hệ thống: tài liệu, phương pháp luận
- Các chuẩn nghiệp vụ: các quy tắc và thực tế nghiệp vụ
- Các chuẩn công nghệ thống tin: kiến trúc và cấu hình chung cho sự phát triển hệ thống nhất quán

Nguyên lý 6: Quản lý quy trình và các dự án

- **Quản lý quy trình** : hoạt động liên tiếp trong đó tài liêu hóa, quản lý, giám sát việc sử dụng và cải thiện phương pháp luận tổ chức đã lựa chọn (“quy trình”) cho việc phát triển hệ thống. Quản lý quy trình quan tâm tới các giai đoạn, các hoạt động, các kết quả và các chuẩn chất lượng nên được áp dụng nhất quán cho mọi dự án.
- **Quản lý dự án** : quy trình xác định phạm vị, lập kế hoạch, bố trí nhân sự, tổ chức, chỉ đạo và điều khiển một dự án để phát triển một hệ thống thông tin với chi phí thấp nhất, trong một khoảng thời gian cụ thể và với chất lượng có thể chấp nhận được.

Nguyên lý 7: Cân đối hệ thống với vốn đầu tư

- **Kế hoạch hệ thống thông tin mang tính chiến lược** phải phù hợp và hỗ trợ cho **kế hoạch hoạt động mang tính chiến lược** của tổ chức
- Có một vài giải pháp có thể, cái đầu tiên không nhất thiết là cái tốt nhất
- Đánh giá tính khả thi của từng giải pháp theo hai tiêu chí:

o **Hiệu quả chi phí**: phân tích chi phí/lợi ích

o **Quản lý rủi ro**: xác định, đánh giá và điều khiển những thách thức tiềm ẩn đối với sự hoàn thành một hệ thống

Nguyên lý 8: Không né tránh việc hủy bỏ hoặc sửa phạm vi

- Phạm vi của một dự án có thể tăng lên
- Quy trình phát triển có các điểm kiểm tra đối với các giai đoạn của nó:

o Hủy bỏ dự án nếu nó không khả thi (do tổ chức quyết định)

o Đánh giá lại?điều chỉnh chi phí/phạm vi nếu phạm vi mở rộng thêm (do người phân tích quyết định)

o Thu hẹp phạm vi nếu ngân sách/lịch biểu bị co lại (do người phân tích quyết định)

Nguyên lý 9: Chia để trị

- Chia một hệ thống phức tạp thành nhiều hệ thống con/thành phần đơn giản hơn
- Quy trình giải quyết vấn đề có thể được làm đơn giản hóa đối với những vấn đề nhỏ hơn
- Các hệ thống con khác nhau ứng với những loại nhân sự khác nhau

Nguyên lý 10: Thiết kế hệ thống để có thể phát triển và thay đổi

- Hệ thống cần được xây dựng sao cho mềm dẻo và dễ thích ứng để có thể thay đổi về sau

2.2. Một quy trình phát triển hệ thống

2.2.1. Động lực của một dự án phát triển hệ thống

Sự ra đời của hầu hết các dự án đều là sự kết hợp của các yếu tố thuộc 3 nhóm sau:

- **Vấn đề (**Problem) – một trạng thái khó khăn trong thực tế ngăn cản tổ chức đạt được đầy đủ mục đích, mục tiêu của nó.
- **Cơ hội** (Opportunity) – một cơ hội để cải thiện tổ chức cho dù không có vấn đề nào được xác định
- **Chỉ thị** (Directive) – một yêu cầu mới được áp đặt bởi nhà quản lý, chính phủ hoặc bộ phận có ảnh hưởng nào đó từ bên ngoài

Các dự án có kế hoạch

- Một **kế hoạch chiến lược hệ thống thông tin** xem xét toàn bộ tổ chức để xác định các dự án phát triển hệ thống, những dự án đó sẽ đem lại giá trị mang tính chiến lược dài hạn cho tổ chức.
- Việc tái cấu trúc quy trình nghiệp vụ (business process redesign) phân tích thấu đáo một chuỗi các quy trình nghiệp vụ để loại bỏ sự dư thừa, thủ tục rườm rà

đồng thời cải thiện hiệu quả và giá trị gia tăng. Khi đó, cần thiết kế lại hệ thống thông tin hỗ trợ cho các quy trình nghiệp vụ đã được thiết kế lại đó.

Các dự án không có kế hoạch

- Được kích hoạt bởi một vấn đề, cơ hội hoặc chỉ thị cụ thể xuất hiện trong khi thực hiện nghiệp vụ
- Hội đồng chỉ đạo – một bộ phận quản trị gồm người sở hữu hệ thống và ban điều hành công nghệ thông tin có trách nhiệm lựa chọn dự án phát triển hệ thống phù hợp.
- Backlog – một kho lưu trữ các đề xuất dự án không thể được cấp vốn hoặc bố trí nhân sự vì chúng có mức ưu tiên thấp hơn dự án đã được phê duyệt để phát triển hệ thống.

Cả dự án được định trước hay không định trước đều phải trải qua cùng quy trình phát triển hệ thống cơ bản, chúng ta sẽ xem xét những giai đoạn dự án đó trong phần tiếp theo sau.

2.2.2. Các giai đoạn của dự án thông thường

- **Xác định phạm vi**
- **Phân tích vấn đề**
- **Phân tích yêu cầu**
- **Thiết kế lôgíc**
- **Phân tích quyết định**
- **Thiết kế vật lý và tích hợp**
- **Xây dựng và kiểm thử**
- **Cài đặt và đưa vào hoạt động**

1. Xác định phạm vi

- Mục đích: xác định các vấn đề,cơ hội và chỉ thị (problems, opportunities, và directives - POD); đánh giá rủi ro của dự án; thiết lập phạm vi, các yêu cầu và ràng buộc sơ bộ, ngân sách và lịch biểu (*nghiên cứu sơ bộ*)
- Vấn đề: Liệu dự án có đáng để xem xét– Xác định phạm vi của dự án
- Kết quả: kế hoạch/biểu đồ dự án
- Kiểm tra tính khả thi: Hủy bỏ dự án / Phê chuẩn để tiếp tục / Thu hẹp hoặc mở rộng phạm vi phù hợp với sự thay đổi ngân sách và lịch biểu

2. Phân tích vấn đề

- Mục đích: nghiên cứu và phân tích hệ thống hiện có từ góc độ của người dùng giống như cách họ nhìn nhận dữ liệu, các quy trình và giao diện
- Vấn đề: Chi phí/lợi ích của việc xây dựng hệ thống mới để giải quyết những vấn đề đó
- Kết quả: các mục tiêu cải thiện hệ thống (các tiêu chuẩn nghiệp vụ để đánh giá hệ thống mới)
- Kiểm tra tính khả thi: Hủy bỏ dự án / Phê chuẩn để tiếp tục / Thu hẹp hoặc mở rộng phạm vi phù hợp với sự thay đổi ngân sách và lịch biểu

3. Phân tích yêu cầu

- Mục đích: tìm hiểu các nhu cầu người dùng không có trong hệ thống mới về dữ liệu, các quy trình và giao diện
- Vấn đề: Xác định các yêu cầu đối với hệ thống mới (NHỮNG GÌ CẦN THỰC HIỆN) mà không cần diễn giải các chi tiết kỹ thuật (LÀM NHƯ THẾ NÀO)
- Các lỗi và sự bỏ sót trong pha phân tích yêu cầu sẽ để lại hậu quả là sự không hài lòng của người dùng về hệ thống cuối cùng và những thay đổi hao tổn chi phí
- Kết quả: báo cáo yêu cầu nghiệp vụ

4. Thiết kế lôgíc

- Mục đích: chuyển các yêu cầu nghiệp vụ của người dùng thành mô hình hệ thống mô tả CẦN LÀM GÌ mà không xác định thiết kế kỹ thuật hoặc cài đặt cụ thể của những yêu cầu đó (*thiết kế khái niệm*)
- Vấn đề: sử dụng mô hình đồ họa của hệ thống để biểu diễn các yêu cầu của người dùng về dữ liệu, các quy trình, giao diện và để đơn giản hóa việc cải thiện sự truyền thông tin giữa các nhân sự
- Chú ý: việc mô hình hóa hệ thống quá thừa sẽ làm chậm đáng kể tiến trình hướng tới việc cài đặt giải pháp hệ thống dự định
- Kết quả: Các mô hình hệ thống lôgíc (DFD, ERD...)

5. Phân tích quyết định

- Mục đích: xác định tất cả các giải pháp đề cử, phân tích tính khả thi của từng giải pháp, tiến cử một hệ thống làm giải pháp mục tiêu
- Vấn đề: phân tích tính khả thi dưới các tiêu chí kỹ thuật, hoạt động, tính kinh tế, lịch biểu (technical, operational, economic, schedule - TOES) và rủi ro
- Kết quả: đề xuất hệ thống được phê duyệt

- Kiểm tra tính khả thi: Hủy bỏ dự án / Chấp nhận đề xuất hệ thống với sự thay đổi ngân sách và lịch biểu / Thu hẹp phạm vi của giải pháp được đề xuất với sự thay đổi ngân sách và lịch biểu
- Các giải pháp đề cử được đánh giá dưới các tiêu chí TOES và rủi ro:
- Tính khả thi kỹ thuật – Liệu giải pháp có thực tế về kỹ thuật? Liệu nhân sự có đủ thành thạo kỹ thuật để thiết kế và xây dựng giải pháp này?
- Tính khả thi hoạt động – Liệu giải pháp có đáp ứng hết các yêu cầu của người dùng? Ở mức độ nào? Liệu giải pháp có thay đổi môi trường làm việc của người sử dụng? Người dùng sẽ cảm nhận thế nào về giải pháp đó?
- Tính khả thi kinh tế – Liệu giải pháp có hiệu quả về chi phí?
- Tính khả thi lịch biểu – Hệ thống có thể được thiết kế và cài đặt trong một khoảng thời gian chấp nhận được?
- Rủi ro – Khả năng cài đặt thành công là như thế nào? (Quản lý rủi ro)

6. Thiết kế vật lý

- Mục đích: chuyển các yêu cầu nghiệp vụ thành các đặc tả thiết kế kỹ thuật cho việc xây dựng
- Vấn đề: kỹ thuật sẽ được sử dụng như thế nào để xây dựng hệ thống về mặt dữ liệu, các quy trình và giao diện
- Kết quả: các đặc tả thiết kế hệ thống (thiết kế chi tiết)
- Kiểm tra tính khả thi: Tiếp tục/ Thu hẹp hoặc mở rộng phạm vi với sự thay đổi ngân sách và lịch biểu

7. Giai đoạn xây dựng

- Mục đích: xây dựng và kiểm thử hệ thống đáp ứng các yêu cầu nghiệp vụ và đặc tả thiết kế; cài đặt giao diện kết nối giữa hệ thống hiện có với hệ thống mới
- Vấn đề: xây dựng cơ sở dữ liệu, các chương trình ứng dụng giao diện người dùng/hệ thống, cài đặt phần mềm được thuê hoặc mua về
- Kết quả: hệ thống được đề xuất trong phạm vi ngân sách và lịch biểu

8. Giai đoạn cài đặt

- Mục đích: đưa hệ thống thu được vào hoạt động
- Vấn đề: huấn luyện người dùng, viết sách hướng dẫn, nạp file, tạo cơ sở dữ liệu, kiểm thử cuối cùng
- Kế hoạch chuyển đổi: từ hệ thống cũ sang hệ thống mới
- Kết quả: hệ thống sẵn sàng để hoạt động

Hoạt động và hỗ trợ

- Hỗ trợ hệ thống không ngừng tới khi hệ thống trở nên lỗi thời và bị thay thế bởi một hệ thống mới.
- Vấn đề: hỗ trợ kỹ thuật cho người dùng; sửa lỗi, kế hoạch phục hồi phù hợp với các yêu cầu nảy sinh.
- **Tóm tắt quy trình phát triển hệ thống:**
- Giai đoạn xác định phạm vi: Vấn đề nào
- Giai đoạn phân tích vấn đề: Các kết quả (Thông tin/Dữ liệu, Các quy trình, Các giao diện)
- Giai đoạn phân tích yêu cầu: Những yêu cầu của người dùng
- Thiết kế lôgíc: Mô hình khái niệm – Cần làm gì
- Giai đoạn phân tích quyết định: Giải pháp nào
- Giai đoạn thiết kế: Mô hình vật lý: Làm thế nào
- Giai đoạn xây dựng: Thực hiện
- Giai đoạn cài đặt: Sử dụng

2.2.3. Các hoạt động xuyên suốt vòng đời

Là bất kỳ hoạt động nào diễn ra tại nhiều hoặc tất cả các giai đoạn của quy trình phát triển hệ thống.

- **Tìm hiểu thực tế (Fact-finding)**

o Là quy trình sử dụng việc nghiên cứu, phỏng vấn, gặp gỡ, phiếu hỏi, mẫu và các kỹ thuật khác để thu thập thông tin về các vấn đề, yêu cầu của hệ thống.

o Rất quan trọng vào những giai đoạn đầu của dữ án, khi mà đội phát triển tìm hiểu về thuật ngữ chuyên ngành, các vấn đề, cơ hội, ràng buộc, các yêu cầu và mức ưu tiên.

- **Tài liệu hóa và trình bày**

o Tài liệu hóa – là hoạt động liên tục để ghi lại thông tin và chi tiết kỹ thuật của một hệ thống cho việc tham khảo ở hiện tại và trong tương lai

o Trình bày – là hoạt động liên tục của việc truyền đạt thông tin, tìm kiếm, đề xuất và cung cấp tài liệu để xem xét bởi người sử dụng và người quản lý

o Kho chứa – một cơ sở dữ liệu và/hoặc tệp thư mục trong đó người phát triển hệ thống lưu tất cả các tài liệu, kiến thức và các thành phần của một hoặc nhiều dự án hoặc hệ thống thông tin

- **Phân tích tính khả thi**
- **Quản lý dự án và quy trình**

2.3. Các chiến lược phát triển hệ thống

2.3.1. Chiến lược phát triển hướng mô hình

Model-driven development – một chiến lược phát triển hệ thống nhấn mạnh vào việc vẽ các mô hình hệ thống để trợ giúp việc trực quan hóa và phân tích các vấn đề, xác định các yêu cầu nghiệp vụ, và thiết kế các hệ thống thông tin.

- **Mô hình hóa chức năng** – một kỹ thuật lấy quá trình làm trung tâm được phổ biến bởi phương pháp luận **phân tích và thiết kế hướng cấu trúc**, sử dụng các mô hình yêu cầu nghiệp vụ để tạo các thiết kế phần mềm hiệu quả cho một hệ thống.
- **Mô hình hóa dữ liệu** – một kỹ thuật lấy dữ liệu làm trung tâm để mô hình hóa các yêu cầu dữu liệu nghiệp vụ và thiết kế hệ thống cơ sở dữ liệu phù hợp.
- **Mô hình hóa đối tượng** – một kỹ thuật kết nối dữ liệu và quá trình thành các cấu trúc duy nhất gọi là các đối tượng. Các mô hình đối tượng là các biểu đồ tài liệu hóa một hệ thống dưới dạng các đối tượng của nó và các tương tác giữa chúng.
- Ưu điểm:
- Kế hoạch dài hạn hơn
- Mô hình hóa hệ thống hiện tại và phân tích yêu cầu trên phạm vi rộng hơn
- Phân tích nhiều giải pháp kỹ thuật khác nhau
- Phù hợp với các hệ thống được hiểu rõ
- Nhược điểm:
- Thời gian thực hiện lâu
- Sự tham gia thụ động của người sử dụng hệ thống bởi họ không nhìn thấy sản phẩm
- Các yêu cầu trong mỗi giai đoạn cần được xác định đầy đủ: điều này không thực tế và/hoặc không mềm dẻo

2.3.2. Chiến lược phát triển ứng dụng nhanh

Rapid application development (RAD) – các kỹ thuật nhấn mạnh sự tham gia của người sử dụng trong việc xây dựng tiến hóa nhanh các bản mẫu hoạt động của một hệ thống để đẩy nhanh quy trình phát triển hệ thống đó.

- RAD được dựa trên việc xây dựng các bản mẫu, những bản mẫu này tiến hóa thành các hệ thống hoàn thiện

- Một **bản mẫu** là một mô hình hoạt động hoặc mô hình biểu diễn với tỷ lệ nhỏ hơn của các yêu cầu của người sử dụng hoặc của một thiết kế đề xuất cho một hệ thống thông tin
- Một **time box** là một khoảng thời gian không thể mở rộng, thường là 60-120 ngày mà một hệ thống đề cử phải được đưa vào hoạt động. Các cải thiện sẽ được thực hiện trong những phiên bản ra đời sau đó.
- Ưu điểm:
- Xử lý được các yêu cầu không ổn định hoặc không chính xác của người sử dụng
- Sự tham gia chủ động của người sử dụng vào việc xây dựng sản phẩm thực tế: làm tăng sự nhiệt tình, hỗ trợ của họ
- Phát hiện sớm các lỗi hoặc sự bỏ sót: trong quá trình kiểm thử và thay đổi bản mẫu
- Làm giảm rủi ro nhờ lặp đi lặp lại việc làm bản mẫu
- Nhược điểm:
- Tăng chi phí thời gian sống để hoạt động, hỗ trợ và bảo trì hệ thống (hoạt động và sửa chữa liên tục)
- Quá trình phân tích vấn đề ngắn ngủi có thể đem lại hệ quả là việc giải quyết những vấn đề sai
- Ngăn cản người phân tích xem xét các kỹ thuật khác thay vì chỉ xét tới kỹ thuật đang được dùng để làm bản mẫu

2.3.3. Chiến lược cài đặt gói ứng dụng thương mại

Commercial application package – một ứng dụng phần mềm có thể mua về và tùy biến cho phù hợp các yêu cầu nghiệp vụ của một số lượng lớn các tổ chức hoặc một ngành nghề cụ thể. Một thuật ngữ khác là *hệ thống thương mại dùng ngay* (commercial off-the-shelf (COTS) system)

- Ưu điểm
- Cài đặt nhanh hệ thống mới (nhiều chức năng tương tự nhau giữa các tổ chức khác nhau, không cần thiết phải xây dựng từ đầu)
- Không cần các chuyên gia và nhân sự cho việc phát triển
- Chi phí phát triển thấp (nhưng tốn chi phí tùy biến và cài đặt)
- Người bán chịu trách nhiệm về việc cải thiện phần mềm và sửa lỗi
- Nhược điểm

- Phụ thuộc vào người bán
- Việc tùy biến/nâng cấp trong tương lai rất tốn kém
- Một hệ thống thương mại dùng ngay hiếm khi phản ánh được hệ thống lý tưởng được tự phát triển

Phải thay đổi các quy trình nghiệp vụ hiện tại để phù hợp với hệ thống thương mại.

2.4. Các kỹ thuật và công cụ tự động hóa

2.4.1. Khái niệm CASE

Computer-Assisted Software Engineering - là các công cụ phần mềm tự động hóa hỗ trợ việc vẽ và phân tích các mô hình hệ thống và các đặc tả liên quan. Một số công cụ CASE cũng cung cấp khả năng làm bản mẫu và sinh mã.

- **Kho chứa CASE** – một cơ sở dữ liệu của người phát triển hệ thống trong đó họ có thể lưu các mô hình hệ thống, các đặc tả chi tiết và các sản phẩm khác của việc phát triển hệ thống. Cách gọi khác là *từ điển dữ liệu*.
- **Forward engineering** – một khả năng của công cụ CASE có thể sinh mã phần mềm và cơ sở dữ liệu ban đầu trực tiếp từ hệ thống.
- **Kỹ thuật đảo ngược - Reverse engineering** – một khả năng của công cụ CASE có thể sinh ra mô hình ban đầu của hệ thống từ mã cơ sở dữ liệu hoặc phần mềm.
- Bốn lý do để sử dụng công cụ CASE là:
- Tăng hiệu suất phân tích
- Làm đơn giản hóa việc giao tiếp giữa người phân tích và người sử dụng
- Cung cấp tính liên tục giữa các giai đoạn vòng đời
- Để đánh giá tác động của việc bảo trì

2.4.2. Phân loại CASE

Công cụ CASE có thể chia thành các loại sau:

- Các công cụ CASE mức cao (còn gọi là front-end CASE) dùng để thực hiện phân tích và thiết kế
- Các công cụ CASE mức thấp (còn gọi là back-end CASE) dùng để sinh mã từ thiết kế CASE đã có
- CASE tích hợp, thực hiện cả hai chức năng của CASE mức cao và mức thấp
- Các công cụ CASE mức cao:
- Tạo và thay đổi thiết kế hệ thống
- Lưu dữ liệu trong kho chứa dự án

- Kho chứa là một tập hợp các bản ghi, phần tử, biểu đồ, hình ảnh, báo cáo và các thông tin khác của dự án
- Các công cụ CASE đó mô hình hóa các yêu cầu tổ chức và xác định các đường biên của hệ thống
- Các công cụ cây mức thấp sinh mã nguồn từ thiết kế CASE đã có
- Mã nguồn thường có thể được sinh dưới dạng một số ngôn ngữ lập trình

Ưu điểm của việc sinh mã

- Giảm thời gian phát triển hệ thống mới
- Thời gian để bảo trì mã được sinh ngắn hơn thời gian bảo trì hệ thống truyền thống
- Các chương trình máy tính có thể được sinh thành nhiều ngôn ngữ
- Thiết kế CASE có thể được mua từ một nhà cung cấp thứ 3 và được điều chỉnh phù hợp với các yêu cầu tổ chức
- Việc sinh mã sẽ tránh được các lỗi về mã lập trình

Kỹ thuật đảo ngược

- Là việc sinh ra thiết kế CASE từ mã chương trình máy tính
- Mã nguồn được kiểm tra, phân tích và chuyển thành các thực thể kho chứa
- Kỹ thuật đảo ngược tạo ra (tùy thuộc vào tập công cụ được sử dụng):
- Các cấu trúc dữ liệu và các phần tử, mô tả các file, bản ghi và trường
- Các thiết kế giao diện
- Trình bày báo cáo đối với các chương trình xử lý theo khối
- Một biểu đồ thể hiện sự phân cấp của các môđun trong chương trình
- Thiết kế cơ sở dữ liệu và các quan hệ
- Kỹ thuật đảo ngược có các ưu điểm sau:
- Giảm thời gian bảo trì hệ thống
- Tài liệu chương trình được tạo ra bù cho tài liệu đã mất
- Các chương trình hướng cấu trúc có thể được sinh ra từ các chương trình phi cấu trúc đã có
- Việc bảo trì hệ thống trong tương lai dễ thực hiện hơn
- Các phần không được sử dụng của chương trình có thể được loại bỏ

2.4.3. Môi trường phát triển ứng dụng

Application development environments (ADEs) – một công cụ phát triển phần mềm tích hợp cung cấp tất cả các điều kiện cần thiết để phát triển phần mềm ứng dụng mới với chất lượng và tốc độ lớn nhất. Cách gọi khác là *môi trường phát triển tích hợp* (*integrated development environment* - IDE)

- Các thành phần ADE có thể gồm:

o Các ngôn ngữ lập trình hoặc trình dịch

o Các công cụ xây dựng giao diện

o Phần mềm trung gian

o Các công cụ kiểm thử

o Các công cụ quản lý phiên bản

o Các công cụ tạo Help

o Các liên kết tới kho chứa

2.4.4. Bộ quản lý dự án và quy trình

- **Ứng dụng quản lý quy trình** – một công cụ tự động hóa trợ giúp việc lập tài liệu và quản lý một phương pháp luận và chiến lược, các kết quả của nó và các chuẩn quản lý chất lượng. Một thuật ngữ đang nổi bật là phần mềm phương pháp – methodware (ví dụ Visio, Visible Analyst, Rational Rose…)
- **Ứng dụng quản lý dự án** – một công cụ tự động hóa trợ giúp việc lập kế hoạch các hoạt động phát triển hệ thống (tốt nhất là sử dụng phương pháp luận đã được chấp thuận), dự đoán và phân bổ nguồn lực bao gồm con người và chi phí), lập lịch biểu hoạt động và nguồn lực, giám sát tiến trình theo lịch biểu và ngân sách, điều khiển và sửa đổi lịch biểu và nguồn lực, và báo cáo tiến trình dự án (ví dụ Microsoft Project…)

Câu hỏi thảo luận

2.1. Quy trình phát triển hệ thống là gì?

2.2. Phân biệt Vòng đời hệ thống với Phương pháp luận phát triển hệ thống.

GỢI Ý: xem mục Phương pháp luận phát triển hệ thống.

2.3. Giải thích tại sao lại nên để người sử dụng tham gia và tất cả các giai đoạn của quá trình phát triển hệ thống?

2.4. Các nguyên nhân có thể dẫn tới việc ra đời một dự án phát triển hệ thống?

GỢI Ý: tham khảo mục Động lực của một dự án phát triển hệ thống.

2.5. Nêu các giai đoạn nói chung của một dự án phát triển hệ thống?

2.6. Nêu các hoạt động diễn ra trong suốt vòng đời phát triển hệ thống?

2.7. Nêu các ưu nhược điểm của chiến lược phát triển hệ thống hướng mô hình.

2.8. Cho biết khái niệm công cụ CASE?

2.9. Phân loại công cụ CASE.

2.10. Nêu một số ví dụ về môi trường phát triển ứng dụng.

Câu hỏi trắc nghiệm:

1. Một quy trình phù hợp để phát triển hệ thống thông tin phải đảm bảo:

- a. Hiệu quả, tài liệu nhất quán, chất lượng nhất quán
- b. Hiệu quả, đơn giản
- c. Tài liệu nhất quán, tiết kiệm chi phí

2. Chiến lược phát triển hướng mô hinh bao gồm các loại kỹ thuật:

- a. Làm bản mẫu, mô hình hóa dữ liệu, mô hình hóa đối tượng
- b. Mô hình hóa chức năng, mô hình hóa đối tượng
- c. Mô hình hóa dữ liệu, mô hình hóa đối tượng, mô hình hóa chức năng

3. Công cụ CASE mức cao có chức năng:

- a. Sinh mã từ thiết kế CASE đã có
- b. Thực hiện phân tích và thiết kế
- c. Cả hai câu trên đều đúng

Tổng kết chương 2

Chương 2 đã nêu những khái niệm liên quan tới quy trình phát triển hệ thống, đồng thời đưa ra một quy trình phát triển hệ thống. Chúng tôi cũng trình bày các hướng tiếp cận phát triển hệ thống hiện nay. Ngoài ra, chương 2 cũng đề cập tới các công cụ và kỹ thuật có thể áp dụng trong quá trình phân tích, thiết kế một hệ thống.

Người học cần nắm được các nguyên lý phát triển hệ thống:

- Để người sở hữu và người sử dụng hệ thống tham gia vào tất cả các giai đoạn phát triển hệ thống
- Sử dụng một cách tiếp cận giải quyết vấn đề
- Thiết lập các giai đoạn và các hoạt động
- Tài liệu hóa suốt quy trình phát triển hệ thống
- Thiết lập các chuẩn về tính nhất quán
- Quản lý quy trình và các dự án
- Cân đối hệ thống với vốn đầu tư
- Không né tránh việc hủy bỏ hoặc sửa phạm vi
- Chia để trị
- Thiết kế hệ thống để có thể phát triển và thay đổi

Người học cần nắm rõ các giai đoạn phát triển hệ thống:

- Xác định phạm vi
- Phân tích vấn đề
- Phân tích yêu cầu
- Thiết kế lôgíc
- Phân tích quyết định
- Thiết kế vật lý và tích hợp
- Xây dựng và kiểm thử
- Cài đặt và đưa vào hoạt động

Người học phải hiểu và phân biệt được các hướng tiếp cận phát triển hệ thống:

- Chiến lược phát triển hướng mô hình
- Chiến lược phát triển ứng dụng nhanh
- Chiến lược cài đặt gói ứng dụng thương mại

Ngoài ra, người học có thể phân biệt được các loại công cụ CASE:

- CASE mức cao

- CASE mức thấp
- CASE tích hợp

PHẦN 2. PHÂN TÍCH HỆ THỐNG

Nội dung

Phần 2 trình bày các kiến thức cần có trong quá trình phân tích hệ thống, bao gồm các nội dung:

CHƯƠNG 3. TỔNG QUAN VỀ PHÂN TÍCH HỆ THỐNG

Mục tiêu

Chương 3 tập trung giới thiệu các khái niệm liên quan tới phân tích hệ thống, cụ thể là các cách tiếp cận phân tích hệ thống, chi tiết các giai đoạn phân tích hệ thống và việc xác định yêu cầu.

3.1. Khái niệm phân tích hệ thống

- **Phân tích hệ thống:** là giai đoạn phát triển trong một dự án, *tập trung vào các vấn đề nghiệp vụ*, ví dụ như những gì hệ thống phải làm về mặt dữ liệu, các thủ tục xử lý và giao diện, *độc lập với kỹ thuật* có thể được dùng để cài đặt giải pháp cho vấn đề đó.
- **Thiết kế hệ thống:** là giai đoạn phát triển tập trung vào việc xây dựng và cài đặt mang tính kỹ thuật của hệ thống (cách thức mà công nghệ sẽ được sử dụng trong hệ thống).

3.2. Các hướng tiếp cận phân tích hệ thống

3.2.1. Các tiếp cận phân tích hướng mô hình

- Nhấn mạnh việc vẽ các mô hình hệ thống dạng đồ họa để tài liệu hóa và kiểm tra hệ thống hiện tại cũng như hệ thống được đề xuất.
- Cuối cùng thì mô hình hệ thống trở thành bản thiết kế chi tiết cho việc thiết kế và xây dựng một hệ thống được cải thiện.
- **Phân tích hướng cấu trúc** (Structured Analysis - SA)**:** thuộc kiểu phân tích hướng mô hình, là kỹ thuật lấy quá trình làm trung tâm để phân tích một hệ thống đang có và xác định các yêu cầu nghiệp vụ cho một hệ thống mới. Phân tích hướng cấu trúc là một trong các tiếp cận chính thống đầu tiên của việc phân tích hệ thống thông tin. Hiện nay, nó vẫn là một trong các cách tiếp cận được áp dụng phổ biến nhất. Phân tích hướng cấu trúc tập trung vào luồng dữ liệu luân chuyển quá các quy trình nghiệp vụ và phần mềm. Nó được gọi là "lấy quá trình làm trung tâm".

o Mô hình minh họa các thành phần của hệ thống: các quá trình (các chức năng, thao tác) và những thành phần liên quan là đầu vào, đầu ra và các file.

- **Kỹ thuật thông tin** (Inforrmation Engineering - IE): là kỹ thuật hướng mô hình và lấy dữ liệu làm trung tâm, nhưng có tính đến quá trình (rõ ràng ngữ cảnh) để lập kế hoạch, phân tích và thiết kế hệ thống thông tin. IE khác với SA ở chỗ, người phân tích sẽ vẽ mô hình dữ liệu trước. IE minh họa và đồng bộ hóa các quá trình và dữ liệu của hệ thống.
- **Phân tích hướng đối tượng** (Object Oriented Analysis - OOA): một kỹ thuật hướng mô hình tích hợp dữ liệu và quá trình liên quan tới việc xây dựng thành các đối tượng. Đây là kỹ thuật mới nhất trong số các hướng tiếp cận. OOA minh họa các đối tượng của hệ thống từ nhiều khung nhìn chẳng hạn như cấu trúc và hành vi.

3.2.2. Các tiếp cận phân tích hệ thống nhanh

Các cách tiếp cận phân tích hệ thống nhanh nhấn mạnh việc xây dựng các bản mẫu để xác định nhanh các yêu cầu nghiệp vụ và của người dùng đối với một hệ thống mới

- **Làm bản mẫu tìm hiểu** (Discovery prototyping) – một kỹ thuật dùng để xác định các yêu cầu nghiệp vụ của người dùng bằng cách để họ phản ứng với một bản cài đặt nhanh-thô của các yêu cầu đó
- Ưu điểm
    - Các bản mẫu phục vụ cho cách suy nghĩ "Ta sẽ biết cái gì mình muốn khi nhìn thấy nó", đây là đặc điểm thường gặp của nhiều người quản lý và người dùng.
- Nhược điểm
- Có thể bị chi phối bởi việc nhìn nhận và cảm giác quá vội vã
- Có thể khuyến khích sự tập trung quá sớm vào việc thiết kế
    - Người dùng có thể lầm tưởng rằng đó là hệ thống hoàn thiện có thể được xây dựng một cách nhanh chóng bằng các công cụ làm bản mẫu
- **Phân tích kiến trúc nhanh** (Rapid architected analysis) – các mô hình hệ thống dẫn xuất từ hệ thống đang có hoặc từ các bản mẫu tìm hiểu
- **Sử dụng kỹ thuật đảo ngược** (Reverse engineering) – là việc sử dụng công nghệ để đọc mã nguồn của một chương trình ứng dụng, cơ sở dữ liệu và/hoặc giao diện người dùng đang có và tự động sinh ra mô hình hệ thống tương ứng

3.2.3. Các phương pháp Agile

Agile method – sự kết hợp của nhiều cách tiếp cận của việc phân tích và thiết kế các ứng dụng được cho là phù hợp với vấn đề đang được giải quyết và hệ thống đang được phát triển.

- Hầu hết các phương pháp luận mang tính thương mại đều không áp đặt một cách tiếp cận duy nhất (phân tích hướng cấu trúc, IE hay OOA) đối với người phân tích hệ thống.
- Thay vào đó, họ tích hợp tất cả các cách tiếp cận phổ biến thành một tập hợp các phương pháp agile.
- Người phát triển hệ thống có thể lựa chọn linh động từ nhiều công cụ và kỹ thuật để hoàn thành nhiệm vụ một cách tốt nhất.

3.3. Các giai đoạn phân tích hệ thống

- Giai đoạn xác định phạm vi **WHAT PROBLEM**
- *Liệu có nên xem xét dự án và để làm gì?*
- Giai đoạn phân tích vấn đề **WHAT ISSUES**
- *Liệu có nên xây dựng một hệ thống mới và để làm gì?*
- Giai đoạn phân tích yêu cầu **WHAT REQUIREMENTS**
- *Người dùng cần gì và muốn gì từ hệ thống mới?*
- Giai đoạn thiết kế Lôgíc **WHAT TO DO**
- *Hệ thống mới cần phải làm những gì?*
- Giai đoạn phân tích quyết định **WHAT SOLUTION**
- *Giải pháp nào là tốt nhất?*

3.3.1. Giai đoạn xác định phạm vi

Bước 1.1: xác định các vấn đề, cơ hội và yếu tố chi phối theo các tiêu chí sau:

- Tính khẩn cấp: trong khoảng thời gian nào thì vấn đề cần được giải quyết hoặc cơ hội hoặc yếu tố chỉ phối cần được nhận ra?
- Tính rõ ràng: Mức độ thấy được của của một giải pháp hoặc hệ thống mới đối với khách hàng hoặc người quản lý điều hành?
- Tính hữu ích: Một hệ thống mới hoặc giải pháp có thể tăng lợi nhuận hoặc giảm chi phí hàng năm lên/xuống bao nhiêu?
- Tính ưu tiên: dựa vào những câu trả lời trên, mức ưu tiên giữa các vấn đề, cơ hội và yếu tố chi phối là như thế nào?
- Giải pháp khả thi: vào giai đoạn đầu của dự án, giải pháp khả thi có thể diễn đạt ở dạng giản đơn sau:

o Để nguyên

o Sửa nhanh

o Thay đổi đơn giản để củng cố hệ thống hiện có

o Thiết kế lại hệ thống hiện có

o Thiết kế một hệ thống mới

Bước 1.2: Thảo luận sơ bộ phạm vi

- Kết quả: Báo cáo phạm vi dự án (giới hạn của dự án)
- Những loại dữ liệu nào cần nghiên cứu
- Những quy trình nghiệp vụ nào cần đưa vào
- Hệ thống giao tiếp như thế nào với người dùng và các hệ thống khác
- Chú ý: nếu sau này phạm thay đổi thì ngân sách và lịch biểu cũng nên được thay đổi phù hợp

Bước 1.3: Đánh giá tính khả thi của dự án

- "Liệu dự án này có đáng được xem xét ?"
- Phân tích chi phí/lợi ích
- Quyết định
- Phê duyệt dự án
- Hủy bỏ dự án
- Xem xét lại phạm vi dự án (với ngân sách và lịch biểu đã được điều chỉnh)
- Bước 1.4: lập biểu và lập kế hoạch ngân sách cho dự án
- Kết quả: báo cáo dự án
- Lập kế hoạch chủ đạo cho toàn bộ dự án: lập biểu và phân bố tài nguyên
- Lập kế hoạch chi tiết và lập biểu để hoàn thiện giai đoạn kế tiếp
- Bước1.5: Trình bày dự án và kế hoạch
- Trình bày và bảo vệ dự án, kế hoạch trước hội đồng thẩm định
- Khởi đầu chính thức dự án và thông báo về dự án, các mục tiêu và lịch biểu
- Kết quả: báo cáo dự án (nhân sự, các vấn đề, phạm vi, phương pháp luận, chỉ thị về các công việc phải hoàn thành, các kết quả, các chuẩn chất lượng, lịch biểu, ngân sách)

3.3.2. Giai đoạn phân tích vấn đề

Bước 2.1: Nghiên cứu lĩnh vực vấn đề

- Tìm hiểu lĩnh vực của vấn đề và các thuật ngữ nghiệp vụ

- Dữ liệu: dữ liệu đang được lưu trữ, các thuật ngữ nghiệp vụ
- Các quá trình: các sự kiện nghiệp vụ hiện có
- Các giao diện: các vị trí và người dùng hiện tại
- Kết quả: xác định về lĩnh vực hệ thống / các mô hình của các hệ thống hiện có

Bước 2.2: Phân tích các vấn đề và cơ hội

- Nghiên cứu các nguyên nhân và hệ quả của từng vấn đề (chú ý: một hệ quả có thể lại là nguyên nhân của những vấn đề khác)
- Kết quả: các báo cáo vấn đề được cập nhật và các phân tích nguyên nhân-hệ quả của từng vấn đề và cơ hội

Bước 2.3: Phân tích các quá trình nghiệp vụ (chỉ danh cho việc tái cấu trúc quy trình nghiệp vụ)

- Đánh giá giá trị gia tăng hoặc giảm bớt của các quá trình đối với toàn bộ tổ chức
- Số lượng đầu vào, thời gian đáp ứng, các khâu đình trệ, chi phí, giá trị gia tăng, các hệ quả của việc loại bỏ hoặc hợp lý hóa quá trình
- Kết quả: các mô hình quá trình nghiệp vụ hiện tại

Bước 2.4: Xác lập các mục tiêu cải thiện hệ thống

- Xác định các mục tiêu cụ thể cải thiện hệ thống và các ràng buộc đối với mỗi vấn đề
- Các mục tiêu phải chính xác, có thể đo được
- Các ràng buộc về lịch biểu, chi phí, công nghệ và chính sách
- Kết quả: các mục tiêu cải thiện hệ thống và báo cáo đề xuất

Bước 2.5: Cập nhật kế hoạch dự án

- Cập nhật dự án:
- Thu hẹp phạm vi, chỉ giữ những mục tiêu ưu tiên cao để phù hợp với thời hạn/ngân sách
- Mở rộng phạm vi và điều chỉnh lịch biểu và ngân sách phù hợp
- Kết quả: kế hoạch dự án đã được cập nhật

Bước 2.6: trình bày các nhận xét và đề xuất

- Kết quả: các mục tiêu cải thiện hệ thống
- Quyết định: tiếp tục/điều chỉnh/hủy bỏ dự án hiện tại

3.3.3. Giai đoạn phân tích yêu cầu

Bước 3.1: xác định các yêu cầu hệ thống

- Các yêu cầu chức năng: các hoạt động và dịch vụ cung cấp bởi hệ thống: các chức năng nghiệp vụ, các đầu vào, đầu ra, dữ liệu được lưu trữ.
- Các yêu cầu phi chức năng: các đặc trưng, đặc điểm xác định một hệ thống thỏa đáng: hiệu suất, tài liệu, ngân sách, tính dễ học và sử dụng, tiết kiệm chi phí, tiết kiệm thời gian, an toàn.
- Kết quả: phác thảo các yêu cầu chức năng và phi chức năng: các mục tiêu cải thiện và đầu vào, đầu ra, các quá trình, dữ liệu được lưu trữ liên quan để đạt được mục tiêu

Bước 3.2: Phân mức ưu tiên cho các yêu cầu

- Các yêu cầu mang tính bắt buộc có ưu tiên cao hơn các yêu cầu khác
- Time boxing: đưa ra hệ thống dưới dạng một tập các phiên bản kế tiếp nhau trong một khoảng thời gian. Phiên bản đầu tiên đáp ứng cac yêu cầu thiết yếu và có mức ưu tiên cao nhất.

Bước 3.3: Cập nhật kế hoạch dự án

- Nếu các yêu cầu vượt quá phiên bản đầu tiên: thu hẹp phạm vi hoặc tăng ngân sách
- Kết quả: các yêu cầu hệ thống đã được thống nhất (các yêu cầu và mức ưu tiên đã được bổ sung)

3.3.4. Giai đoạn mô hình hóa lôgíc

Bước 4.1: Phân tích các yêu cầu mang tính chức năng

- Các mô hình hệ thống lôgíc: hệ thống phải làm gì (chứ không phải làm như thế nào)

o Việc tách biệt phần nghiệp vụ với các giải pháp kỹ thuật sẽ giúp cho việc xem xét các cách thức khác nhau để cải thiện các quá trình nghiệp vụ và các khả năng lựa chọn giải pháp kỹ thuật.

- Xây dựng các bản mẫu để xác lập các yêu cầu giao diện người dùng
- Kết quả: các mô hình dữ liệu (ERD), các mô hình quá trình (DFD), các mô hình giao diện (biểu đồ ngữ cảnh, biểu đồ Use case), các mô hình đối tượng (các biểu đồ UML) của hệ thống được đề xuất.

Bước 4.2: Kiểm tra các yêu cầu mang tính chức năng

- Kiểm tra tính đầy đủ, xem xét lại, thực hiện các thay đổi và bổ sung đối với các mô hình hệ thống và các bản mẫu để đảm bảo rằng các yêu cầu đã được xác định thỏa đáng.
- Liên kết các yêu cầu phi chức năng với các yêu cầu mang tính chức năng.

3.3.5. Giai đoạn phân tích quyết định

- Là giai đoạn chuyển tiếp giữa phân tích hệ thống và thiết kế hệ thống

Bước 5.1: xác định các giải pháp đề cử

- Xác định tất cả các giải pháp đề cử có thể có
- Kết quả: ma trận các hệ thống (giải pháp) đề cử

Bước 5.2: Phân tích các giải pháp đề cử

- Việc phân tích tính khả thi được thực hiện với từng đề cử mà không quan tâm tới tính khả thi của các đề cử khác

o Các tính khả thi về kỹ thuật, tính sẵn sàng hoạt động, tính kinh tế, lịch biểu

- **Phân tích tính khả thi:**

o Tính khả thi về kỹ thuật. Liệu giải pháp có phù hợp với thực tế công nghệ? Liệu đội ngũ dự án có chuyên gia kỹ thuật để thiết kế và xây dựng giải pháp?

o Tính khả thi về hoạt động. Liệu giải pháp có thực hiện được yêu cầu của người dùng? Ở mức độ nào? Giải pháp sẽ thay đổi môi trường làm việc của người dùng như thế nào? Người dùng sẽ cảm thấy như thế nào về giải pháp như vậy?

o Tính khả thi về kinh tế. Liệu giải pháp có chi phí hiệu quả?

o Tính khả thi lịch biểu. Liệu giải pháp có thể được thiết kế và xây dựng trong một khoảng thời gian chấp nhận được hay không?

Bước 5.3: So sánh các giải pháp đề cử

- Chọn giải pháp đề cử có sự kết hợp "toàn diện tốt nhất" của các tính khả thi về kỹ thuật, hoạt động, kinh tế và lịch biểu
- Ma trận tính khả thi
- Kết quả: giả pháp được đề xuất

Bước 5.4: Cập nhật kế hoạch dự án

- Đầu vào: giải pháp đề xuất
- Xem xét và cập nhật lịch biểu mới nhất của dự án và phân bố tài nguyên
- Kết quả: cập nhật kế hoạch dự án

Bước 5.5: đề xuất một giải pháp

- Kết quả: đề xuất dự án

3.4. Xác định các yêu cầu của người dùng

3.4.1. Giới thiệu

- **Vai trò của việc xác định yêu cầu:**

o **Yêu cầu hệ thống** (yêu cầu nghiệp vụ) là một mô tả các nhu cầu và mong muốn đối với một hệ thống thông tin. Một yêu cầu có thể mô tả các chức năng, đặc trưng (thuộc tính) và các ràng buộc.

o **Các yêu cầu mang tính chức năng:** các chức năng hoặc đặc trưng có thể có trong một hệ thống thông tin để nó thỏa mãn nhu cầu nghiệp vụ và có thể chấp nhận được đối với người dùng

o **Các yêu cầu phi chức năng:** các đặc trưng, đặc điểm và thuộc tính của các hệ thống cũng như bất kỳ các ràng buộc nào có thể giới hạn ranh giới của giải pháp được đề xuất.

- Hậu quả của yêu cầu không chính xác:

o Hệ thống có thể tốn nhiều chi phí hơn

o Hệ thống có thể hoàn thiện muộn hơn thời gian đã định

o Hệ thống có thể không phù hợp với những gì người dùng mong muốn và sự hài lòng đó có thể khiến họ không sử dụng nó

o Chi phí bảo trì và củng cố hệ thống có thể quá cao

o Hệ thống có thể không chắc chắn và dễ có lỗi và ngừng hoạt động

o Uy tín của các chuyên gia trong đội dự án có thể bị giảm sút bởi bất kỳ thất bại nào, cho dù là do ai gây ra thì cũng sẽ bị xem là lỗi của cả đội dự án

- **Các tiêu chuẩn xác định yêu cầu hệ thống:**

o **Nhất quán** – các yêu cầu không mâu thuẫn hay nhập nhằng lẫn nhau.

o **Toàn diện**– các yêu cầu mô tả mọi đầu vào và đáp ứng có thể có của hệ thống.

o **Khả thi** – các yêu cầu có thể được thoả mãn dựa trên các tài nguyên và ràng buộc sẵn có.

o **Cần thiết** – các yêu cầu là thực sự cần thiết và đáp ứng mục đích của hệ thống.

o **Chính xác** – các yêu cầu được phát biểu chính xác.

o **Dễ theo dõi** – các yêu cầu ánh xạ trực tiếp tới các chức năng và đặc trưng của hệ thống.

o **Có thể kiểm tra** – các yêu cầu đã được vạch rõ nên

3.4.2. Quy trình xác định yêu cầu

- Phân tích yêu cầu:

o Phân tích các yêu cầu để giải quyết các vấn đề về:

- Các yêu cầu bị thiếu
- Các yêu cầu mâu thuẫn nhau
- Các yêu cầu không khả thi
- Các yêu cầu trùng lặp
- Các yêu cầu mơ hồ

o Chính thức hóa các yêu cầu:

- Lập tài liệu xác định các yêu cầu
- Truyền đạt tới các nhân sự tham gia
- Lập tài liệu yêu cầu - một tài liệu xác định yêu cầu bao gồm:

o Các chức năng, dịch vụ mà hệ thống nên cung cấp.

o Các yêu cầu phi chức năng bao gồm các thuộc tính, đặc điểm và đặc trưng của hệ thống.

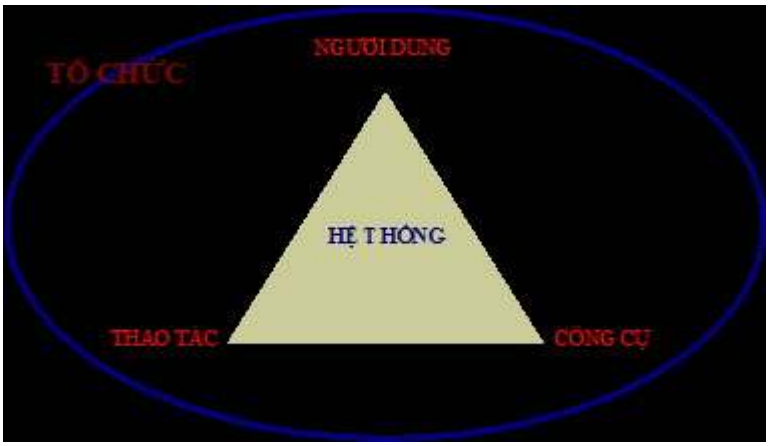
o Các ràng buộc giới hạn sự phát triển của hệ thống hoặc theo đó hệ thống phải hoạt động

o Thông tin về các hệ thống khác mà hệ thống phải giao tiếp

- Quản lý yêu cầu: là quá trình quản lý các thay đổi đối với các yêu cầu

o Trong thời gian diễn ra dự án, việc xuất hiện các yêu cầu mới hay thay đổi những yêu cầu đã có là rất phổ biến

o Các nghiên cứu cho thấy rằng có đến 50% hoặc hơn thế các yêu cầu sẽ biến đổi trước khi hệ thống được hoàn thiện



Hình 3-1 Ngữ cảnh yêu cầu đối với một hệ thống thông tin

3.4.3. Các phương pháp tìm hiểu thực tế

o Lấy mẫu của các cơ sở dữ liệu, biểu mẫu và tài liệu hiện có

o Nghiên cứu và thăm địa điểm của tổ chức.

o Quan sát môi trường làm việc

o Lập phiếu hỏi

o Phỏng vấn

o Làm bản mẫu thăm dò

o Lập kế hoạch yêu cầu kết hợp

- **Làm bản mẫu thăm dò** (Discovery prototyping) – là hoạt động xây dựng một mô hình làm việc hoặc mô hình minh họa quy mô nhỏ đối với các yêu cầu của người dùng để phát hiện hoặc kiểm tra các yêu cầu đó
- **Lập kế hoạch yêu cầu kết hợp** (Joint requirements planning - JRP) – một quá trình trong đó các cuộc họp nhóm làm việc được tổ chức chặt chẽ (có chương trình rõ ràng và những đại diện quan trọng) nhằm mục đích phân tích các vấn đề và xác định các yêu cầu.

o JRP là một tập con của kỹ thuật phát triển ứng dụng kết hợp (Joint Application Development – JAD) bao gồm toàn bộ quá trình phát triển hệ thống.

- Ích lợi của JRP:

o JRP tập hợp những người sử dụng và người quản lý vào một dự án phát triển (khuyến khích họ trở thành "người làm chủ" dự án).

o JRP giảm lượng thời gian cần thiết để phát triển hệ thống.

o JRP kết hợp chặt chẽ những ích lợi của việc làm bản mẫu để làm phương tiện xác nhận các yêu cầu và đạt được sự phê chuẩn cho thiết kế.

Ví dụ tóm tắt kết quả khảo sát để xây dựng hệ thống quản lý bán điện:

Công ty điện X cần quản lý hệ thống bán điện cho người tiêu dùng. Khi người tiêu dùng có nhu cầu sử dụng điện năng cần phải ký hợp đồng với công ty. Hợp đồng được phân thành nhiều loại như hộ gia đình, hộ kinh doanh, hộ sản xuất …; ứng với mỗi loại có một đơn giá riêng. Đơn giá này còn phụ thuộc vào số lượng điện tiêu thụ, ví dụ: đơn giá tính theo 50 số điện đầu tiên, 100 số tiếp theo …

Hàng tháng công ty có nhân viên đi ghi số điện tại công tơ của mỗi hợp đồng mua điện. Sau đó, các thông tin này được ghi vào cơ sở dữ liệu và hệ thống tự động tạo ra các hoá đơn thanh toán của từng hợp đồng mua điện. Hoá đơn này được gửi đến từng hộ mua điện. Khi hoá đơn được thanh toán, hệ thống sẽ xác nhận và cập nhật vào cơ

sở dữ liệu. Đầu tháng, hệ thống tự động kiểm tra cơ sở dữ liệu để tìm ra những hoá đơn nào chưa thanh toán. Nếu có hoá đơn nào sau 3 tháng còn chưa được thanh toán, hệ thống tự động tạo ra một phiếu nhắc thanh toán quá hạn. Phiếu này sẽ được gửi đến hộ mua điện có hoá đơn chưa thanh toán. Nếu sau 6 tháng, hoá đơn chưa được thanh toán, hệ thống sẽ tạo ra một phiếu ngừng cung cấp điện. Công ty chỉ tiếp tục bán điện khi các hoá đơn này được thanh toán hết.

Khách hàng có thể yêu cầu tạm ngừng cung cấp điện hoặc huỷ bỏ hợp đồng mua điện với công ty. Khách hàng chỉ được huỷ bỏ hợp đồng khi đã thanh toán hết tất cả các hoá đơn.

Câu hỏi thảo luận

3.1. Phân biệt Phân tích hệ thống và Thiết kế hệ thống.

3.2. Phân biệt kỹ thuật phân tích hướng cấu trúc với phân tích hướng đối tượng.

GỢI Ý: tham khảo mục Các hướng tiếp cận phân tích hệ thống.

3.3. Kể tên các giai đoạn phân tích hệ thống.

3.4. Nêu vai trò của việc xác định yêu cầu.

3.5. Kể tên các phương pháp khảo sát thực tế.

Câu hỏi trắc nghiệm

1. Hướng tiếp cận phân tích hướng mô hình bao gồm các kỹ thuật:

- a. Phân tích hướng cấu trúc, làm bản mẫu tìm hiểu.
- b. Kỹ thuật thông tin, phân tích kiến trúc nhanh.
- c. Phân tích hướng đối tượng, phân tích hướng cấu trúc, kỹ thuật thông tin.

2. Các mô hình thu được từ giai đoạn thiết kế lôgíc là:

- a. Các mô hình dữ liệu và chức năng.
- b. Các biểu đồ ca sử dụng và các mô hình đối tượng.
- c. Cả hai câu trên.

Tổng kết chương 3

Chương 3 đã đề cập tới các tiếp cận phân tích hệ thống, các giai đoạn trong pha phân tích hệ thống và quy trình xác định yêu cầu của người dùng.

Người học cần nắm rõ và phân biệt được các chiến lược tiếp cận phân tích hệ thống:

- Các tiếp cận phân tích hướng mô hình
- Các tiếp cận phân tích hệ thống nhanh
- Các phương pháp Agile

Người học cần nắm vững các giai đoạn trong quá trình phân tích hệ thống:

- Giai đoạn xác định phạm vi
- Giai đoạn phân tích vấn đề
- Giai đoạn phân tích yêu cầu
- Giai đoạn thiết kế Lôgíc
- Giai đoạn phân tích quyết định

Người học nắm được quy trình xác định yêu cầu:

- Phân tích yêu cầu
- Lập tài liệu yêu cầu
- Quản lý yêu cầu

CHƯƠNG 4. CÁC PHƯƠNG PHÁP THU THẬP THÔNG TIN

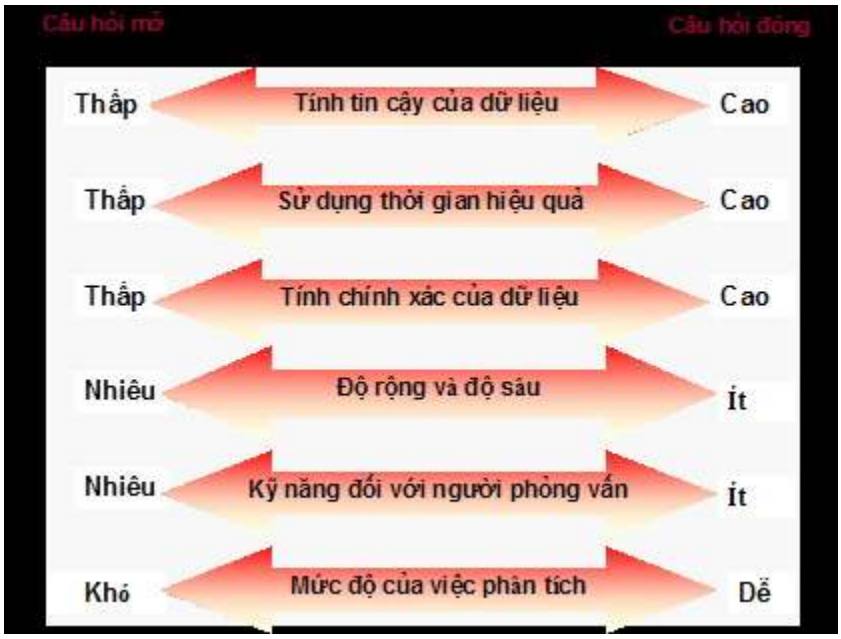
Mục tiêu

Chương 4 giới thiệu các phương pháp để khả sát thực tế nhằm thu thập các thông tin phục vụ cho việc phân tích hệ thống:

- Phỏng vấn
- Sử dụng phiếu hỏi
- Lấy mẫu
- Phân tích tài liệu định lượng/định tính
- Quan sát

4.1. Phương pháp phỏng vấn

- Phỏng vấn là một phương pháp quan trọng để thu thập dữ liệu về các yêu cầu của hệ thống thông tin
- Việc phỏng vấn nhằm phát hiện thông tin về:
- Các ý kiến của người được phỏng vấn
- Cảm giác của người được phỏng vấn
- Trạng thái hiện tại của hệ thống
- Các mục tiêu của con người và tổ chức
- Các thủ tục nghiệp vụ không chính thức
- Năm bước lập kế hoạch phỏng vấn là:
- Đọc các tài liệu cơ bản
- Thiết lập các mục tiêu phỏng vấn
- Xác định người đi phỏng vấn
- Chuẩn bị người được phỏng vấn
- Quyết định cấu trúc và kiểu câu hỏi
- Có hai kiểu câu hỏi phỏng vấn cơ bản:
- Câu hỏi mở
- Câu hỏi đóng



Hình 4-1 So sánh câu hỏi mở và câu hỏi đóng trong phỏng vấn

4.1.1. Dạng câu hỏi mở

- Các câu hỏi phỏng vấn mở cho phép những người được phỏng vấn trả lời những gì họ mong muốn và mức độ mong muốn của họ
- Các câu hỏi mở phù hợp khi người phân tích quan tâm tới độ rộng và sâu của câu trả lời
- Có tám ưu điểm:

  o Làm cho người được phỏng vấn cảm thấy thoải mái

  o Cho phép người phỏng vấn tập trung vào cách biểu đạt của người được phỏng vấn:

  o Phản ánh trình độ văn hóa, các giá trị, thái độ và niềm tin

  o Cung cấp mức độ chi tiết cao

  o Phát hiện các câu hỏi mới mà chưa được khai thác

  o Làm cho người được phỏng vấn thấy thú vị hơn

  o Cho phép tính tự nhiên cao hơn

  o Giúp người phỏng vấn dễ điều chỉnh nhịp độ hơn

  o Hữu ích khi người phỏng vấn không chuẩn bị trước
- Có năm nhược điểm:

  o Có thể thu được quá nhiều chi tiết không liên quan

  o Có thể mất đi tính điều khiển cuộc phỏng vấn

  o Có thể mất quá nhiều thời gian để thu được thông tin có ích

o Có khả năng thể hiện rằng người phỏng vấn không chuẩn bị

o Có thể gây ấn tượng rằng người phỏng vấn đang trong "cuộc hành trình đi câu"

4.1.2. Dạng câu hỏi đóng

- Câu hỏi đóng hạn chế số câu trả lời có thể có
- Câu hỏi đóng phù hợp để tạo ra dữ liệu đáng tin cậy và chính xác, dễ dàng để phân tích
- Phương pháp luận hiệu quả và đòi hỏi ít kỹ năng đối với người phỏng vấn
- Sáu ưu điểm:

  o Tiết kiệm thời gian phỏng vấn

  o Dễ dàng so sánh giữa các lần phỏng vấn

  o Dễ đạt đúng mục đích

  o Kiểm soát được cuộc phỏng vấn

  o Bảo phủ một phạm vi rộng lớn một cách nhanh chóng

  o Thu hoạch được các dữ liệu liên quan
- Bốn nhược điểm:

  o Nhàm chán đối với người được phỏng vấn

  o Khó thu được nhiều chi tiết

  o Có thể mất đi các ý tưởng chính

  o Khó tạo được mối giao tiếp tốt giữa người phỏng vấn và người được phỏng vấn

4.1.3. Các dạng câu hỏi khác

- Các câu hỏi lưỡng cực:
  - Là những câu hỏi có thể trả lời với các từ "có" hoặc "không" hoặc "đồng ý" hoặc "không đồng ý"
  - Các câu hỏi này chỉ nên dùng khi thật cần thiết
- Các câu hỏi thăm dò:
  - Các câu hỏi thăm dò gợi ra tính chi tiết hơn về câu hỏi trước đó
  - Mục đích của câu hỏi thăm dò là:
    - Thu được nhiều ý nghĩa hơn
    - Làm sáng rõ
    - Khai thác và mở rộng các quan điểm của người được phỏng vấn

4.1.4. Thứ tự đặt câu hỏi

Ba cách cơ bản để cấu trúc cuộc phỏng vấn là:

- Kim tự tháp: mở đầu với các câu hỏi đóng và tiếp tục với các câu hỏi mở
- Hình phễu: mở đầu với các câu hỏi mở và tiếp tục với các câu hỏi đóng
- Kim cương: mở đầu với các câu hỏi đóng, tiếp tục với các câu hỏi mở và kết thúc bằng các câu hỏi đóng

Cấu trúc kim tự tháp

- Mở rất chi tiết, thường là bằng các câu hỏi đóng
- Mở rộng bằng các câu hỏi mở và những câu trả lời tổng quát hơn
- Hữu ích nếu người được phỏng vấn cần được khích lệ đi vào chủ để hoặc tỏ ra không tự nguyện hướng tới chủ đề

Cấu trúc phễu:

- Mở đầu với các câu hỏi mở, mang tính tổng quát
- Kết thúc bằng cách thu hẹp các câu trả lời có thể có bằng việc sử dụng các câu hỏi đóng
- Cung cấp cách thức dễ dàng, không gây áp lực để bắt đầu một cuộc phỏng vấn
- Có ích khi người được phỏng vấn cảm thấy hứng khởi với chủ đề

Cấu trúc kim cương

- Một cấu trúc hình kim cương mở đầu theo cách rất cụ thể
- Tiếp theo các vấn đề tổng quát hơn được xem xét
- Kết thúc với các câu hỏi cụ thể
- Cấu trúc này kết hợp thế mạnh của cả cấu trúc kim tự tháp và hình phễu
- Mất nhiều thời gian hơn các cấu trúc khác

Kết thúc việc phỏng vấn:

- Luôn luôn hỏi "Liệu còn có gì khác mà bạn muốn bổ sung không?"
- Tóm tắt và cung cấp phản hồi về ấn tượng của người phỏng vấn
- Hỏi xem người tiếp theo nên phỏng vấn là ai
- Thiết lập các cuộc hẹn gặp tiếp theo
- Cảm ơn người được phỏng vấn và bắt tay

Báo cáo phỏng vấn :

- Viết càng sớm càng tốt ngay sau khi phỏng vấn
- Cung cấp một bản tóm tắt ban đầu, sau đó thì chi tiết hơn

- Xem lại báo cáo với người được phỏng vấn

4.2. Phương pháp dùng phiếu hỏi

Phiếu hỏi có ích để thu thập thông tin từ các thành viên chủ đạo trong tổ chức về:

- Thái độ
- Niềm tin
- Hành vi
- Tính cách

Phiếu hỏi có giá trị nếu:

- Các thành viên của tổ chức phân tán rộng
- Nhiều thành viên tham gia vào dự án
- Cần việc có tính thăm dò
- Các câu hỏi được thiết kế theo một trong hai kiểu

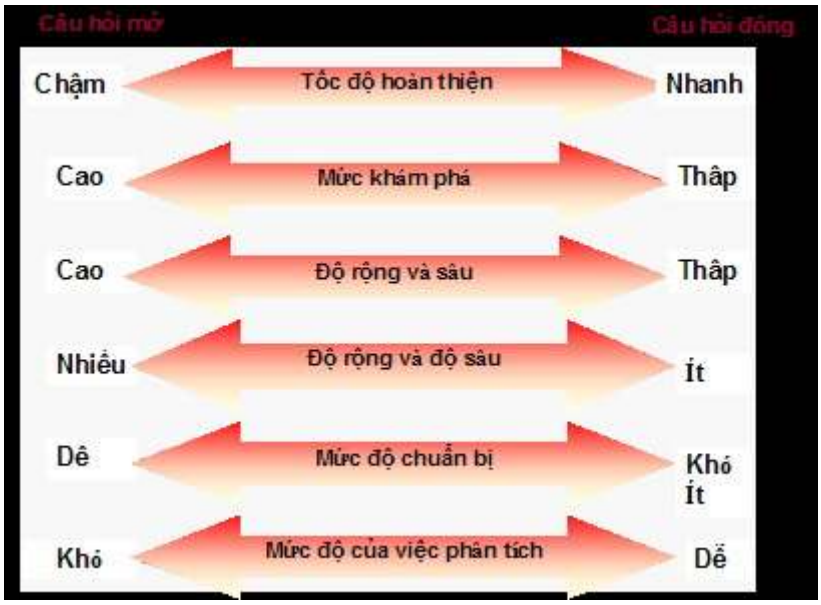
Câu hỏi mở:

- Cố gắng đoán trước câu trả lời sẽ nhận được
- Phù hợp để thu được các ý kiến

Câu hỏi đóng:

Sử dụng khi tất cả các lựa chọn đều liệt kê được

- Khi các lựa chọn loại trừ lẫn nhau



Hình 4-2 So sánh câu hỏi mở và câu hỏi đóng khi dùng phiếu hỏi

4.2.1. Thiết kế phiếu hỏi

- Ngôn ngữ dùng trong phiếu hỏi nên:

o Đơn giản

o Cụ thể

o Không thành kiến

o Không có vẻ bề trên

o Chính xác về mặt kỹ thuật

o Hướng đến những người có hiểu biết

o Phù hợp với khả năng đọc hiểu của người trả lời

- Phiếu hỏi phải chính xác và đáng tin cậy

o Tính tin cậy thể hiện sự nhất quán trong trả lời – nghĩa là thu được cùng các kết quả nếu như cùng một phiếu hỏi được phân phát trong cùng điều kiện

o Tính chính xác là mức độ câu hỏi đo được những gì người phân tích muốn đánh giá

- Tỉ lệ câu trả lời tốt có thể có được nhờ sự điều chỉnh phù hợp phiếu hỏi

o Để ra nhiều khoảng trống

o Bố trí khoảng trống lớn để viết/gõ câu trả lời

o Tạo điều kiện cho người trả lời dễ dàng bày tỏ rõ câu trả lời của họ

o Nhất quán về hình thức

- Thứ tự câu hỏi:

o Đặt các câu hỏi quan trọng nhất lên đầu tiên

o Nhóm các câu hỏi có cùng nội dung lại với nhau

o Đưa các câu hỏi ít gây tranh luận lên trên

4.2.2. Các phương pháp phát phiếu hỏi

- Tập hợp tất cả những người trả lời vào cùng một thời gian
- Phát phiếu hỏi cho từng cá nhân
- Gửi phiếu hỏi qua đường bưu điện
- Phát phiếu hỏi qua Web hoặc thư điện tử, có các ưu điểm:

o Giảm chi phí

o Thu thập và lưu trữ các kết quả dễ dàng hơn

- Phiếu hỏi dạng web thường gồm:

o Hộp văn bản đơn dòng

o Hộp văn bản cuộn, dùng một hoặc nhiều đoạn văn bản

o Hộp chọn dành cho các câu trả lời có/không hoặc đúng/sai

o Nút tùy chọn cho các câu trả lời mang tính loại trừ lẫn nhau có/không hoặc đúng/sai

o Menu thả để chọn từ một danh sách

o Nút Submit (xác nhận) hoặc Reset (xác lập lại)

4.3. Phương pháp lấy mẫu

- Lấy mẫu là quá trình lựa chọn một cách có hệ thống các phần tử đại diện của một mẫu. Thay vì nghiên cứu tất cả các thể hiện của các biểu mẫu và bản ghi trong các tệp hoặc cơ sở dữ liệu thì người phân tích chỉ cần sử dụng kỹ thuật lấy mẫu để chọn ra một phần đủ lớn các phần tử đại diện phục vụ cho việc xác định thông tin diễn ra trong hệ thống.
- Bao gồm hai quyết định quan trọng:

o Những tài liệu và website quan trọng nào nên được lấy mẫu

o Những người nào nên được phỏng vấn và gửi phiếu hỏi

- Lý do người phân tích cần lấy mẫu là:

o Giảm chi phí

o Tăng tốc quá trình thu thập dữ liệu

o Cải thiện hiệu quả

o Giảm việc tập trung thu thập dữ liệu

4.3.1. Các bước thiết kế mẫu

- Để thiết kế một mẫu tốt, một người phân tích hệ thống cần tuân theo bốn bước sau:

o Xác định dữ liệu cần được thu thập hoặc mô tả

o Xác định tập cần được lấy mẫu

o Chọn loại mẫu

o Quyết định kích thước mẫu

- Quyết định kích thước mẫu nên được thực hiện theo những điều kiện cụ thể mà người phân tích hệ thống làm việc:

o Lấy mẫu dữ liệu trên các thuộc tính

o Lấy mẫu dữ liệu trên các biến

o Lấy mẫu dữ liệu định tính

4.3.2. Các kiểu lấy mẫu

- **Lấy mẫu tùy ý:**

o Các mẫu không giới hạn, không mang tính xác suất

o Dễ sắp xếp

o Không đáng tin cậy nhất

- **Lấy mẫu có mục đích**

o Dựa trên sự đánh giá

o Người phân tích chọn nhóm các cá nhân để lấy mẫu

o Dựa trên các tiêu chuẩn

o Mẫu không mang tính xác suất

o Đáng tin cậy ở mức độ vừa phải

- **Lấy mẫu ngẫu nhiên đơn giản**

o Dựa trên danh sách các con số của tập lấy mẫu

o Mỗi người hoặc tài liệu đều có cơ hội được lựa chon ngang nhau

- **Lấy mẫu ngẫu nhiên phức tạp, có ba hình thức là:**

o Lấy mẫu có hệ thống

  - Là phương pháp đơn giản nhất của lấy mẫu theo xác suất
  - Chọn mọi cá nhân thứ k trong danh sách
  - Không hay nếu danh sách được sắp thứ tự

o Lấy mẫu phân tầng

  - Xác định các tập lấy mẫu con
  - Chọn các đối tượng hoặc con người để lấy mẫu từ tập lấy mẫu con
  - Bù vào số lượng không cân đối các nhân viên trong một nhóm nhất định
  - Chọn các phương pháp khác nhau để thu thập dữ liệu từ các nhóm con khác nhau
  - Là phương pháp quan trọng nhất đối với người phân tích

o Lấy mẫu theo nhóm

  - Chọn nhóm các tài liệu hoặc con người để nghiên cứu
  - Chọn các nhóm điển hình đại diện cho số còn lại

4.4. Phân tích tài liệu định lượng/định tính

4.4.1. Phân tích tài liệu định lượng

- Nghiên cứu dữ liệu cứng là một phương pháp hữu hiệu để người phân tích thu thập thông tin

- Dữ liệu cứng có thể thu thập từ:
    - Phân tích các tài liệu định lượng như các hồ sơ được sử dụng để ra quyết định
    - Các báo cáo thực thi
    - Các hồ sơ
    - Các mẫu thu thập dữ liệu
    - Các giao dịch nghiệp vụ

4.4.2. Phân tích tài liệu định tính

- Xem xét các tài liệu định tính để thu được:
    - Các thông tin tiềm ẩm quan trọng
    - Trạng thái tâm lý
    - Những gì được coi là tốt/xấu
    - Hình ảnh, logo, biểu tượng
- Tài liệu định tính bao gồm:
    - Các bản ghi nhớ
    - Dấu hiệu trên các bản tin
    - Website của tổ chức
    - Các tài liệu chỉ dẫn
    - Sổ tay về chính sách của tổ chức

4.5. Phương pháp quan sát

- Việc quan sát cung cấp sự hiểu biết về những gì các thành viên của tổ chức thực sự đang làm
- Nhìn nhận trực tiếp các quan hệ tồn tại giữa những người ra quyết định và các thành viên khác của tổ chức

Kỹ thuật STROBE:

Gọi là kỹ thuật quan sát môi trường có cấu trúc (**STR**uctured **OB**servation of the **E**nvironment). Là kỹ thuật quan sát môi trường của những người ra quyết định

- STROBE phân tích bảy phần tử môi trường:
    - Vị trí văn phòng
    - Vị trí bàn làm việc
    - Thiết bị văn phòng
    - Tài sản
    - Các nguồn thông tin bên ngoài
    - Màu sắc và ánh sáng văn phòng

- Trang phục của người ra quyết định

- Vị trí văn phòng
  - Những văn phòng dễ thâm nhập
    - Các hành lang chính, cửa mở thông nhau
    - Không gian đi lại lớn
    - Làm tăng tần suất tương tác và các thông điệp không chính thức
  - Những văn phòng khó thâm nhập
    - Có thể nhìn nhận hệ thống theo cách khác
    - Nằm cô lập so với các văn phòng khác
    - Vị trí bàn làm việc
  - Không gian kín, quay lưng vào tường, khoảng rộng sau bàn lớn
    - Thể hiện vị trí có sức mạnh lớn nhất
- Bàn quay mặt vào tường, ghế nằm về một phía
    - Khích lệ nhân viên
    - Khả năng trao đổi, giao tiếp ngang nhau
- Thiết bị văn phòng
  - Tủ hồ sơ và giá sách:
    - Nếu không có những thứ đó thì nhân viên chỉ lưu trữ một số mục thông tin mang tính cá nhân
    - Nếu có thì họ sẽ lưu trữ và khai thác thông tin
- Tài sản
  - Máy tính điện tử
  - Máy vi tính
  - Bút mực, bút chì, thước kẻ
  - Nếu có thì nhân viên sẽ xử lý dữ liệu một cách cá nhân
- Các nguồn thông tin bên ngoài
  - Báo hoặc tạp chí thương mại thể hiện rằng nhân viên khai thác các thông tin bên ngoài
  - Các báo cáo, sổ ghi nhớ, sổ tay chính sách của công ty thể hiện rằng con người khai thác các thông tin bên trong tổ chức

- Màu sắc và ánh sáng văn phòng
    - Ánh sáng chói, ấm áp thể hiện:
    - Khuynh hướng hướng tới giao tiếp cá nhân nhiều hơn
    - Nhiều cuộc giao tiếp không chính thức hơn
    - Màu tươi, sáng sủa thể hiện:
        - Nhiều sự giao tiếp chính thức hơn (vì vậy nên chú trọng vào sổ ghi nhớ, các báo cáo…)
- Trang phục
    - Nam giới
        - Complê trang trọng thể hiện khả năng đó là người có quyền lực lớn
        - Trang phục bình thường thể hiện nhiều khả năng đó là người tham gia vào việc ra quyết định
    - Nữ giới
        - Trang phục trang trọng thể hiện khả năng đó là người có quyền lực
- Có 5 biểu tượng dùng để đánh giá kết quả quan sát các phần tử của STROBE so với kết quả phỏng vấn thực tế là:
    - Một dấu check – kết quả phỏng vấn được xác nhận
    - Dấu "X" – kết quả phỏng vấn là ngược lại
    - Biểu tượng oval – cần phải xem xét kỹ hơn
    - Hình vuông – việc quan sát làm thay đổi kết quả phỏng vấn
    - Hình tròn – kết quả phỏng vấn được bổ sung bởi việc quan sát

Câu hỏi thảo luận

4.1. Nêu các mục tiêu của việc phỏng vấn.

4.2. Kể tên các dạng câu hỏi có thể dùng trong quá trình phỏng vấn.

4.3. Nêu ý nghĩa của việc lấy mẫu.

4.4. Nêu các phần tử môi trường được phân tích trong kỹ thuật STROBE.

Câu hỏi trắc nghiệm

1. Cácphương pháp thu thập thông tin là:

☐ a. Phỏng vấn, sử dụng phiếu hỏi, phân tích tài liệu định lượng/định tính, lấy mẫu, quan sát.

☐ b. Sử dụng phiếu hỏi, lấy mẫu.

- c. Quan sát, phỏng vấn

2. Khi phỏng vấn, nếu người được phỏng vấn là người rụt rè thì nên chọn thứ tự đặt câu hỏi theo kiểu:

- a. Cấu trúc hình phễu
- b. Cấu trúc kim tự tháp
- c. Cả hai kiểu trên

3. Khi phỏng vấn, nếu đối tượng là người bạo dạn và nhiệt tình thì nên chọn thứ tự đặt câu hỏi theo kiểu nào:

- a. Cấu trúc hình phễu hoặc cấu trúc kim cương
- b. Cấu trúc kim tự tháp
- c. Không đáp án nào đúng

Tổng kết chương 4

Chương 4 đã trình bày các phương pháp để thu thập thông tin trong thực tế.

Người học cần phân biệt được các dạng câu hỏi có thể sử dụng trong quá trình phỏng vấn:

- Câu hỏi mở
- Câu hỏi đóng
- Câu hỏi lưỡng cực
- Câu hỏi thăm dò

Bên cạnh đó, người học phải hiểu và phân biệt các cách cấu trúc một cuộc phỏng vấn:

- Cấu trúc kim tự tháp
- Cấu trúc hình phễu
- Cấu trúc kim cương

Đối với phương pháp dùng phiếu hỏi, người học cần nắm được cách thức thiết kế một phiếu hỏi và các hình thức phát phiếu hỏi.

Người học cần hiểu phương pháp phân tích tài liệu định lượng và phân tích tài liệu định tính.

Hướng dẫn bài tập lớn

Dựa vào những kiến thức đã học về thu thập thông tin, tuỳ theo chủ đề bài tập lớn đã chọn, người học tiến hành khảo sát thực tế, xác định các yêu cầu đối với hệ thống cần xây dựng. Người học thực hiện viết báo cáo cho bài tập lớn gồm các phần cơ bản:

- Đặt vấn đề
    - Mục đích
    - Giới thiệu bài toán
- Khảo sát thực tế và xác lập dự án
    - Hoạt động của hệ thống hiện tại
    - Các ưu nhược điểm của hệ thống hiện tại
    - Các yêu cầu đặt ra đối với hệ thống cần xây dựng

CHƯƠNG 5. MÔ HÌNH HÓA CHỨC NĂNG

Mục tiêu

Chương này tập trung vào việc mô hình hóa các chức năng trong quá trình phân tích. Nội dung bao gồm khái niệm và các nguyên tắc xây dựng biểu đồ phân cấp chức năng và biểu đồ luồng dữ liệu. Ngoài ra, chúng tôi cũng đưa vào những đoạn phim hướng dẫn các bạn cách vẽ các biểu đồ bằng phần mềm Microsoft Visio.

5.1. Mô hình hóa hệ thống

5.1.1. Các bước mô hình hóa hệ thống

Trong chương 3, bạn đã biết về các hoạt động phân tích hệ thống, những hoạt động đó là nhằm mục đích vẽ các mô hình hệ thống. Các mô hình hệ thống đóng vai trò quan trọng trong phát triển hệ thống. Dù là người sử dụng hay người phân tích hệ thống thì bạn đều phải giải quyết những vấn đề phi cấu trúc. Và một cách để cấu trúc vấn đề là vẽ các mô hình.

> **Mô hình** là một biểu diễn hình tượng của thực tế. Các mô hình có thể được xây dựng cho các hệ thống hiện có để giúp chúng ta hiểu kỹ hơn về những hệ thống đó. Hoặc cũng có thể xây dựng mô hình cho các hệ thống được đề xuất nhằm tài liệu hóa các yêu cầu nghiệp vụ hoặc thiết kế kỹ thuật.

- **Mô hình hóa chức năng** (Process Modeling) với biểu đồ luồng dữ liệu (Data Flow Diagram - DFD)

o Hệ thống làm gì?

Mô hình hóa chức năng là kỹ thuật dùng để tổ chức và tài liệu hóa cấu trúc và luồng dữ liệu xuyên qua các quá trình của một hệ thống và/hoặc các chức năng được thực hiện bởi các quá trình hệ thống.

- **Mô hình hóa dữ liệu** (Data Modeling) với biểu đồ quan hệ thực thể (Entity Relationship Diagram - ERD)

o Hệ thống có những dữ liệu nào?

Mô hình hóa dữ liệu là kỹ thuật dùng để tổ chức và mô hình hóa dữ liệu của một hệ thống nhằm xác định các yêu cầu nghiệp vụ cho một cơ sở dữ liệu. Đôi khi mô hình hóa dữ liệu còn được gọi là *mô hình hóa cơ sở dữ liệu*.

- **Mô hình hóa đối tượng** (Object Modeling) với ngôn ngữ mô hình hợp nhất (Unified Modeling Language - UML)

o Cái gì và tại sao? (lôgíc của hệ thống)

5.1.2. Mục đích của mô hình hóa hệ thống

- Để hiểu rõ hơn về hệ thống: các cơ hội để đơn giản hóa, tối ưu hóa (Tái cấu trúc quy trình)
- Để liên kết các hành vi và cấu trúc của hệ thống (các yêu cầu nghiệp vụ về: thông tin/dữ liệu và chức năng/quy trình)
- Để trực quan hóa và điều khiển kiến trúc hệ thống (thiết kế)
- Để kiểm soát những rủi ro trong quá trình phát triển

5.1.3. Các thao tác mô hình hóa chức năng

- Lập kế hoạch chiến lược hệ thống

  Các mô hình quá trình nghiệp vụ của tổ chức mô tả các chức năng nghiệp vụ quan trọng

- Tái cấu trúc quy trình nghiệp vụ

  Các mô hình chức năng "As is" làm đơn giản việc phân tích các điểm yếu (Hệ thống hiện tại).

  Các mô hình chức năng "To be" làm đơn giản việc cải thiện (Hệ thống mới được đề xuất).

- Phân tích hệ thống

  Mô hình hóa hệ thống hiện có bao gồm những thiếu sót của nó (DFD lôgíc)

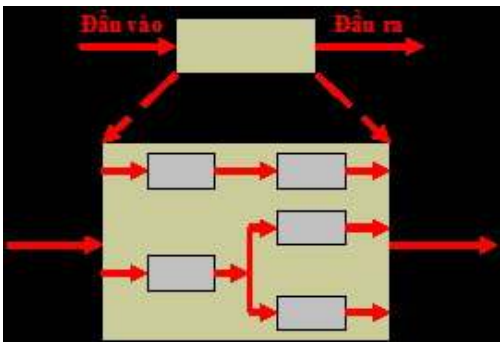
  Mô hình hóa các yêu cầu lôgíc (các quá trình và luồng dữ liệu cần có dù hệ thống được xây dựng thế nào – DFD lôgíc) của hệ thống được đề xuất.

  Mô hình hóa các giải pháp kỹ thuật đề cử (DFD vật lý)

  Mô hình hóa giải pháp được chọn (DFD vật lý)

5.1.4. Khái niệm hệ thống

Một hệ thống tồn tại bằng việc lấy đầu vào từ môi trường, biến đổi (xử lý) đầu vào này và tạo ra một đầu ra. Một hệ thống có thể được phân rã thành nhiều hệ thống con. Một hệ thống con có đầu vào và đầu ra của riêng nó. Đầu ra của một hệ thống con có thể trở thành đầu vào của những hệ thống con khác.



Hình 5-1 Hệ thống và các hệ thống con

- Hệ thống và quá trình
    - **Một hệ thống là một quá trình.** Nó thể hiện một chức năng nghiệp vụ.
    - Một **quá trình** là công việc được thực hiện trên hoặc đáp ứng cho các điều kiện hoặc luồng dữ liệu vào
    - Một quá trình (chức năng) có thể được phân rã thành các quá trình con (các chức năng con, các thao tác).

5.2. Mô hình logic

5.2.1. Phân biệt mô hình lôgíc và mô hình vật lý

- **Mô hình lôgíc** cho biết hệ thống là gì và làm gì. Nó độc lập với việc cài đặt kỹ thuật. Nó minh họa bản chất của hệ thống. Mô hình lôgíc còn có thể được gọi là *mô hình bản chất*, *mô hình khái niệmmô hình nghiệp vụ*.
- **Mô hình vật lý** không chỉ thể hiện hệ thống là gì và làm gì mà còn thể hiện cách thức hệ thống được cài đặt một cách vật lý và kỹ thuật. Nó phản ánh các lựa chọn công nghệ. Mô hình vật lý còn có thể được gọi là *mô hình cài đặt* hay *mô hình kỹ thuật*.

5.2.2. Sự cần thiết của mô hình lôgíc

Các nhà phân tích hệ thống đã nhận thấy giá trị của việc tách riêng việc nghiên cứu nghiệp vụ với việc nghiên cứu kỹ thuật. Đó là lý do tại sao họ sử dụng các mô hình hệ thống lôgíc để minh họa các yêu cầu nghiệp vụ và các mô hình hệ thống vật lý để minh họa các thiết kế kỹ thuật. Các hoạt động của người phân tích hệ thống tập trung chủ yếu vào các mô hình hệ thống lôgíc vì những lý do sau:

- Các mô hình lôgíc loại bỏ tư tưởng thiên lệch do ảnh hưởng bởi cách thức cài đặt hệ thống đã có hoặc ý kiến chủ quan của một người nào đó về cách cài đặt cho hệ thống. Do đó, chúng khuyến khích tính sáng tạo.
- Các mô hình lôgíc làm giảm khả năng bỏ sót các yêu cầu nghiệp vụ trong trường hợp con người bị chi phối quá nhiều vì các kết quả mang tính kỹ thuật. Nhờ việc tách biệt những gì hệ thống phải làm với cách thức hệ thống thực hiện mà chúng ta có thể phân tích tốt hơn các yêu cầu nhằm đảm bảo tính hoàn thiện, chính xác và nhất quán.
- Các mô hình lôgíc cho phép truyền đạt với người dùng cuối dưới dạng ngôn ngữ phi kỹ thuật hoặc ít kỹ thuật hơn.

5.3. Biểu đồ phân rã chức năng

5.3.1. Khái niệm BFD

BFD là sơ đồ phân rã có thứ bậc các chức năng của hệ thống từ tổng thể đến chi tiết. Mỗi chức năng có thể có một hoặc nhiều chức năng con, tất cả được thể hiện trong một khung của sơ đồ.

- **Ý nghĩa của BFD:**
    - Giới hạn phạm vi của hệ thống cần phải phân tích.
    - Tiếp cận hệ thống về mặt logic nhằm làm rõ các chức năng mà hệ thống thực hiện để phục vụ cho các bước phân tích tiếp theo.
    - Phân biệt các chức năng và nhiệm vụ của từng bộ phận trong hệ thống, từ đó lọc bỏ những chức năng trùng lặp, dư thừa.

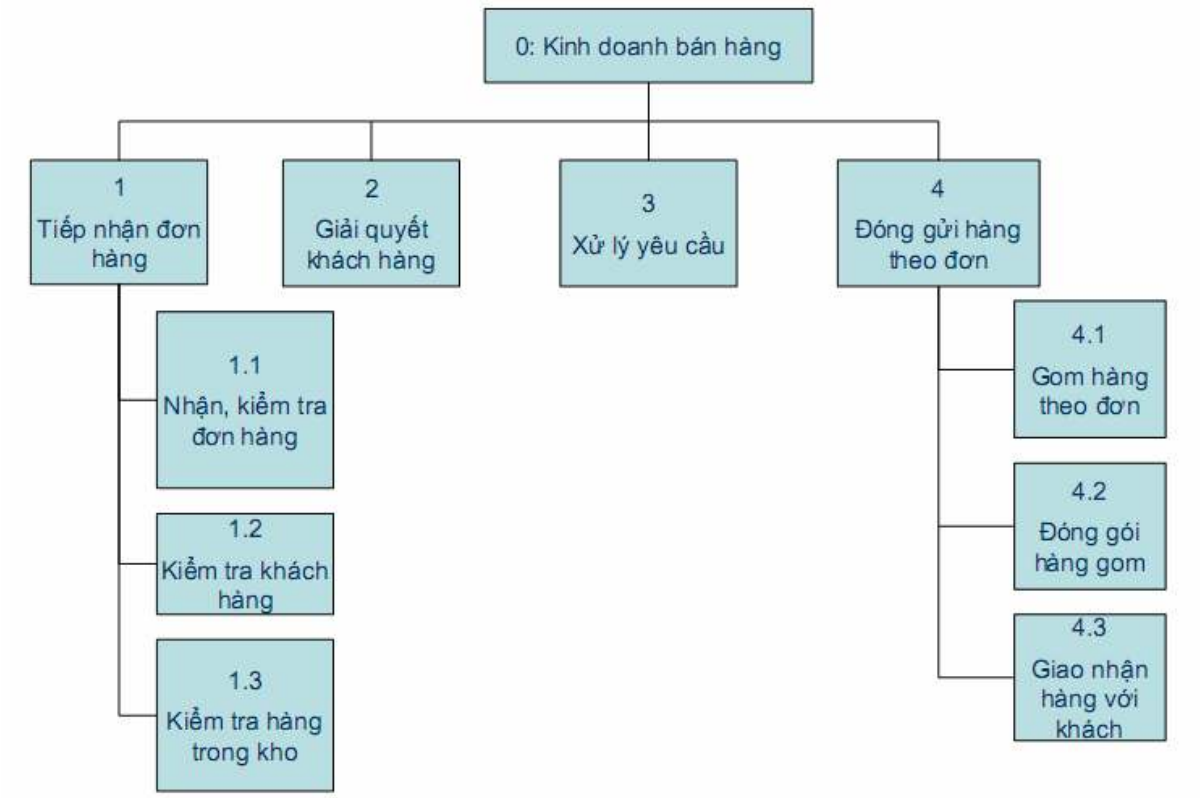
Tuy nhiên, BFD không có tính động, nó chỉ cho thấy các chức năng mà không thể hiện trình tự xử lý của các chức năng đó cũng như là sự trao đổi thông tin giữa các chức năng. Do đó, BFD thường được sử dụng làm mô hình chức năng trong bước đầu phân tích.

5.3.2. Phương pháp xây dựng BFD

a. Phân mức các chức năng

- Chú ý là với một sơ đồ, không nên có quá 6 mức, thông thường thì 3 mức là phù hợp với các hệ thống trung bình.
- Với mỗi chức năng không nên có quá 6 chức năng con vì như vậy sẽ làm sơ đồ trở nên phức tạp và khó kiểm soát. Nếu gặp trường hợp có quá nhiều chức năng con thì có thể giải quyết bằng cách tạo thêm mức trung gian để nhóm các chức năng con lại.
- Cần đảm bảo tính cân bằng của sơ đồ, nghĩa là các chức năng thuộc cùng một mức nên có sự tương đương nhau về kích thước và độ phức tạp.
- Mỗi chức năng phải mang một tên duy nhất, không trùng lặp với chức năng khác; tên phải thể hiện khái quát các chức năng con của nó, phản ánh được thực tế nghiệp vụ mà nó thực hiện. Tên của chức năng phải bắt đầu bằng động từ, ví dụ như “lập đơn hàng”.

Tuân thủ những nguyên tắc trên sẽ giúp cho việc xây dựng các mô hình dữ liệu tiếp theo được rõ ràng.



Hình 5-2 Minh họa cấu trúc sơ đồ phân mức chức năng bán hàng.

b. Xác định các chức năng

- Ở mức cao nhất của nghiệp vụ, chức năng chính có thể là một trong các loại sau:
    - Sản xuất sản phẩm.
    - Cung cấp dịch vụ (bán hàng, bảo dưỡng).
    - Quản lý tài nguyên (tài sản, nguồn nhân lực, con người…).

Khi đã xác định được loại mà nó thuộc vào thì sẽ đặt tên cho chức năng cao nhất này.

- Tiếp theo, để xác định các chức năng con thì từ chức năng chính, ta đặt nó trong chu kỳ sống gồm các giai đoạn:
    - Xác định nhu cầu.
    - Mua bán.
    - Bảo hành, bảo dưỡng.
    - Thanh lý hoặc chuyển nhượng.

Mỗi giai đoạn có thể có một hoặc nhiều chức năng con. Ví dụ, với chức năng Bán hàng thì ở giai đoạn xác định nhu cầu có thể có chức năng con là Quản lý thông tin khách hàng, ở giai đoạn mua bán thì có thể là Cập nhật đơn hàng…

- Người phân tích phải xác định được mức nào là thấp nhất, khi đó sẽ dừng việc phân tích chức năng. Để nhận biết một chức năng mức thấp nhất bằng cách xét xem có phải chức năng đó chỉ có một nhiệm vụ hoặc một nhóm các nhiệm vụ nhỏ.
- Khi xây dựng BFD cần đảm bảo tính đơn giản, rõ ràng và chính xác của sơ đồ. Với các hệ thống lớn, có thể trình bày BFD trên nhiều trang, trang 1 là BFD mức cao nhất (mức 0), tiếp theo ứng với mỗi chức năng sẽ được phân tích ở các trang sau tới chức năng mức thấp nhất thì dừng.

5.4. Biểu đồ luồng dữ liệu (DFD)

5.4.1. Mô hình hóa chức năng với DFD

Trong chương này, chúng ta tập trung vào việc mô hình hóa chức năng lôgíc trong giai đoạn phân tích hệ thống. Như trên đã nói, mô hình hóa chức năng là một kỹ thuật để tổ chức và tài liệu hóa cấu trúc và luồng dữ liệu cũng như lôgíc, đường lối và các thủ tục được thực hiện bởi các quá trình của một hệ thống. Một trong các mô hình chức năng phân tích hệ thống chính là sơ đồ luồng dữ liệu.

- Một sơ đồ luồng dữ liệu (Data Flow Diagram – DFD) là một công cụ đồ họa để mô tả luồng dữ liệu luân chuyển trong một hệ thống và những hoạt động xử lý được thực hiện bởi hệ thống đó. Sơ đồ luồng dữ liệu còn có các tên gọi khác là *biểu đồ bọt*, *biểu đồ biến đổi* và *mô hình chức năng*.
- Tại sao sử dụng DFD?
    - Sự mô tả bằng ngôn ngữ hướng tới sự giải thích, nó có thể bỏ sót những thông tin quan trọng.
    - Sự mô tả đồ họa minh họa được luồng dữ liệu trong một tổ chức thông qua DFD.

Biểu đồ luồng dữ liệu đã rất phổ biến hơn 20 năm nay nhưng lợi ích của DFD đã được đổi mới nhờ vào tính ứng dụng của nó trong việc tái cấu trúc quy trình nghiệp vụ (business process redesign – BPR). Khi mà tổ chức nhận thấy rằng hầu hết các hệ thống xử lý dữ liệu đã trở nên lỗi thời, không hiệu quả và rườm rà về thủ tục thì đó là lúc có thể thu lợi ích mới nhờ vào việc tổ chức lại các quy trình nghiệp vụ. Điều này được tiến hành trước tiên bằng việc mô hình hóa các quy trình nghiệp vụ nhằm mục đích phân tích, thiết kế lại và/hoặc cải thiện chúng. Tiếp theo, công nghệ thông tin có thể được áp dụng một cách sáng tạo cho các quy trình nghiệp vụ đã được cải thiện nhằm tối đa hóa giá trị thu về cho tổ chức.

5.4.2. Vai trò của DFD

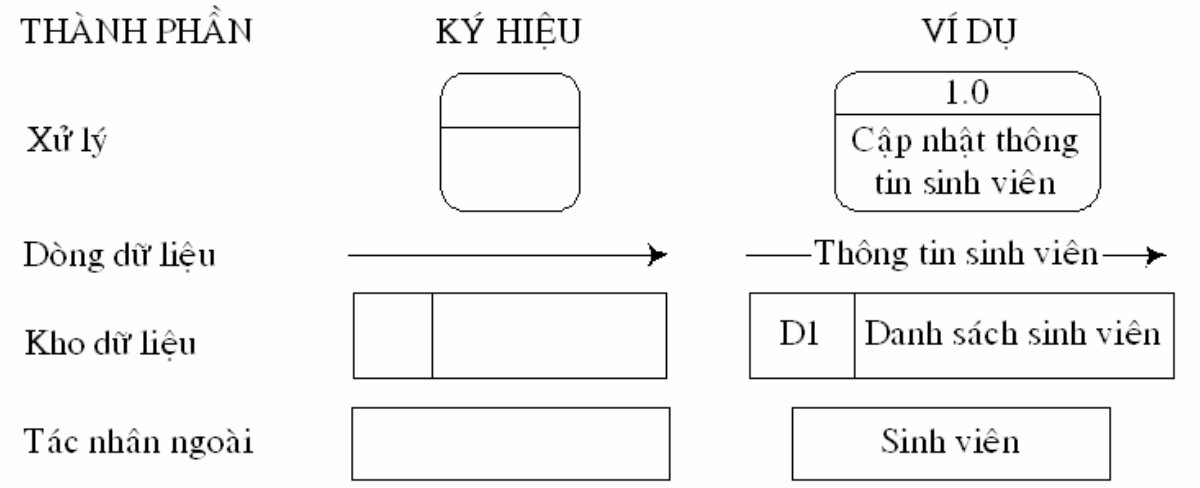
DFD tài liệu hóa một thao tác/hoạt động/chức năng nghiệp vụ của một hệ thống thành một quá trình. DFD mô tả cách thức dữ liệu được xử lý trong và tại biên giới của hệ thống. DFD thể hiện chi tiết sự phụ thuộc lẫn nhau giữa các quá trình của hệ thống, các sự dịch chuyển dữ liệu hoặc thông tin giữa các quá trình.

- DFD lôgíc mô tả luồng thông tin của một hệ thống
- DFD vật lý mô tả cách thức một hệ thống thông tin được cài đặt vật lý (ai làm, bằng cách nào, bằng công cụ nào)
- Mô tả cách thức một hệ thống thông tin được cài đặt vật lý (ai làm, bằng cách nào, bằng công cụ nào)

5.5. Các phần tử của DFD

5.5.1. Các bộ ký hiệu DFD

Có hai một vài bộ ký hiệu DFD mà phổ biến là Gane/Sarson và Demarco/Yourdon. Trong đó, bộ ký hiệu Gane & Sarson được sử dụng phổ biến hơn.



Hình 5-3 Các bộ ký hiệu DFD

5.5.2. Tác nhân ngoài

- Khái niệm:
  - Một tác nhân ngoài là một nguồn cung cấp hoặc nhận thông tin dữ liệu của hệ thống
  - Một tác nhân ngoài không phải là một phần của hệ thống, nó thể hiện mối quan hệ giữa hệ thống với môi trường bên ngoài
- Nhãn: Tên của tác nhân ngoài phải là một **danh từ**

Một tác nhân ngoài xác định một người, một đơn vị của tổ chức hay một tổ chức khác nằm ngoài phạm vi của dự án nhưng có tương tác với hệ thống đang được nghiên cứu.

Các tác nhân ngoài xác định "biên giới" hay phạm vi của hệ thống đang được mô hình hóa. Khi phạm vi thay đổi, các tác nhân ngoài có thể trở thành các quá trình và ngược lại

Tác nhân ngoài thường là:

- Phòng ban, bộ phận trong tổ chức nhưng nằm ngòai phạm vi hệ thống.
- Một chi nhánh hoặc tổ chức bên ngoài
- Một hệ thống thông tin khác của tổ chức
- Người dùng cuối hoặc người quản lý của hệ thống

5.5.3. Kho dữ liệu

- Khái niệm:
    - Một kho dữ liệu là một kho lưu trữ dữ liệu, nó chứa thông tin.
    - Kho chứa vật lý là phi vật chất, nó có thể là một tủ hồ sơ, sách hoặc file máy tính.
- Nhãn: Tên của kho dữ liệu phải bắt đầu bằng **danh từ**, nó nói lên nội dung thông tin.

Một kho dữ liệu là "dữ liệu tĩnh" khác với luồng dữ liệu là "dữ liệu chuyển động". Một kho dữ liệu cần biểu diễn cho "những thứ" mà tổ chức muốn lưu trữ dữ liệu, "những thứ" thường là:

- Con người: ví dụ như khách hàng, phòng, nhân viên, thầy giáo, sinh viên, nhà cung cấp…
- Các địa điểm: ví dụ như nơi sinh, tòa nhà, phòng, chi nhánh…
- Các đối tượng: ví dụ như sách, máy móc, sản phẩm, nguyên liệu thô, bản quyền phần mềm, gói phần mềm, công cụ, phương tiện vận tải…
- Các sự kiện (dữ liệu được thu thập về chúng): ví dụ như việc bán hàng, giải thưởng, sự trì hoãn, lớp học, chuyến bay, hóa đơn, đơn hàng, đăng ký, đặt chỗ…
- Các khái niệm (dữ liệu về chúng rất quan trọng): ví dụ như việc giảm giá, tài khoản, khóa học, chất lượng…

Có thể xác định các kho dữ liệu với các yếu tố Tài nguyên – Sự kiện – Tác nhân – Địa điểm. Các kho dữ liệu được mô tả trong một DFD chứa tất cả các thể hiện của các thực thể dữ liệu (được mô tả trong một biểu đồ quan hệ quan hệ ERD).

5.5.4. Luồng dữ liệu

- Khái niệm:

Một luồng dữ liệu biểu diễn một **sự di chuyển của dữ liệu (thông tin)** giữa các quá trình hoặc kho dữ liệu.

Một luồng dữ liệu **không** biểu diễn một tài liệu hay một vật thể vật lý: nó biểu diễn sự trao đổi **thông tin** trong tài liệu hoặc về vật thể.

Nhãn: Phải có tên và không trùng lặp với các luồng dữ liệu khác. Tên phải thể hiện logic của thông tin chứ không phải dạng vật lý của nó và phải bắt đầu bằng **danh từ**

Một luồng dữ liệu biểu diễn một đầu vào dữ liệu tới một quá trình hoặc đầu ra dữ liệu từ một quá trình.

Một luồng dữ liệu cũng có thể được dùng để biểu diễn việc tạo, đọc, xóa hoặc cập nhật dữ liệu trong một file hoặc cơ sở dữ liệu (được gọi là kho dữ liệu).

Một luồng dữ liệu ghép (gói) là một luồng dữ liệu chứa các luồng dữ liệu khác.

5.5.5. Quá trình

- Khái niệm:

Một quá trình là một hoạt động được thực hiện trên luồng dữ liệu vào để tạo một luồng dữ liệu ra.

Là chức năng được thực hiện bởi hệ thống để đáp ứng lại các luồng dữ liệu hoặc điều kiện vào.

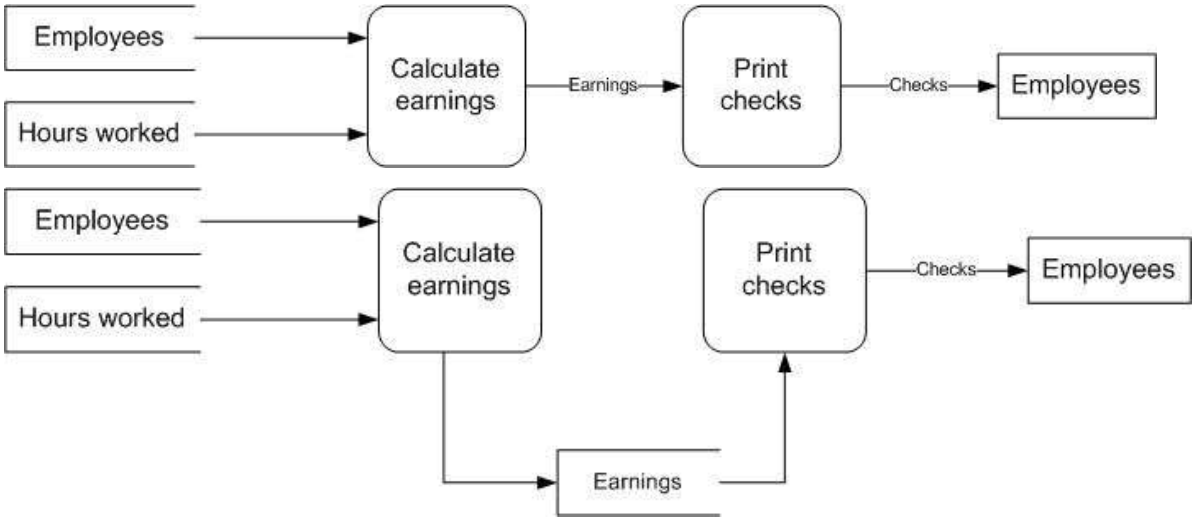
- Nhãn: Sử dụng một **động từ** để gán nhãn cho hành động thực hiện bởi quá trình (không phải là tên của người hay phòng ban thực hiện nó trong DFD vật lý).
- Một quá trình phải có **ít nhất một** luồng dữ liệu **vào** và **ít nhất một** luồng dữ liệu **ra**.
- Các loại quá trình lôgíc:
    - Chức năng: một tập các hoạt động tiếp diễn liên quan tới một nghiệp vụ; ví dụ như việc bán hàng
    - Sự kiện (hay giao dịch, hoạt động): một công việc phải được hoàn thành toàn bộ (hoặc như một phần của một chức năng); ví dụ như việc thu tiền thanh toán (là một công đoạn trong việc bán hàng)
    - Quá trình cơ bản (hay thao tác): một hoạt động hoặc thao tác chi tiết, rời rạc được yêu cầu để đáp lại một sự kiện. Thông thường, một số thao tác như vậy phải được hoàn thành để đáp ứng một sự kiện; ví dụ như ghi tiền thành toán
- Cách tách các quá trình

Mỗi hệ thống có thể được chia thành nhiều quá trình khác nhau bằng các cách khác nhau. Các quá trình có thể được tách nếu có một luồng thông tin đi giữa chúng.

Điều kiện để tách: nếu các tiến trình này không thực hiện đồng thới hoặc không cùng một nơi hoặc không do một người thực hiện. Khi đó, ta kiểm tra quá trình tách bằng cách cuối luồng dữ liệu ta đặt câu hỏi:

- Tiến trình tiếp theo có thể thực hiện ở thời gian khác được không?
- Tiến trình tiếp theo có thể thực hiện ở nơi khác được không?
- Tiến trình tiếp theo có thể được thực hiện bởi người khác được không?

Nếu một trong ba câu hỏi trên là có thì ta tách chúng bằng cách đặt một tệp dữ liệu ở giữa.



Hình 5-4 Ví dụ cách tách các quá trình

5.6. Biểu đồ luồng dữ liệu mức ngữ cảnh

Biểu đồ luồng dữ liệu mức ngữ cảnh (Context data flow diagram) là một mô hình chức năng được dùng để tài liệu hóa phạm vi của một hệ thống. Nó còn được gọi là *mô hình môi trường*. Để xây dựng biểu đồ ngữ cảnh, cần phải:

- Xác định biên giới của hệ thống
- Xác định các tác nhân ngoài
- Không chi tiết về các quá trình và kho dữ liệu của hệ thống
- Chiến lược cụ thể xây dựng biểu đồ ngữ cảnh:

o Coi cả hệ thống là một "hộp đen", xem nó là một chức năng duy nhất và chỉ quan tâm tới phần bên ngoài của nó mà không phải xét tới những hoạt động bên trong.

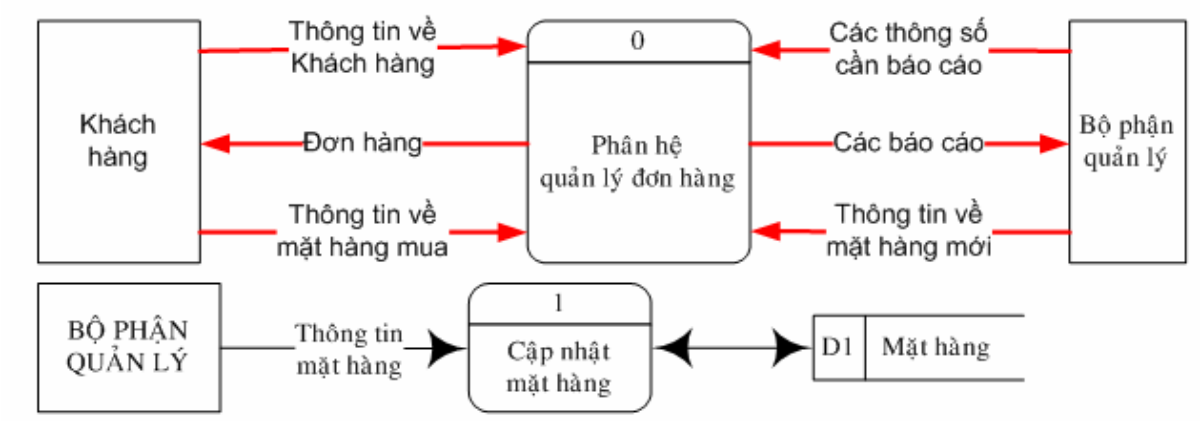
o Hỏi người sử dụng cuối của hệ thống về những giao dịch nghiệp vụ mà hệ thống phải đáp ứng. Đó chính là các luồng vào của hệ thống. Với mỗi luồng vào, cần xác

định nguồn tương ứng của nó. Các nguồn sẽ trở thành các tác nhân ngoài trong sơ đồ ngữ cảnh.

o Hỏi người sử dụng cuối về những đáp ứnh phải được sinh ra bởi hệ thống. Đó chính là các luồng ra của hệ thống, Với mỗi luồng ra, xác định đích của nó. Các đích đó cũng sẽ trở thành các tác nhân ngoài.

o Xác định các kho dữ liệu ngoài. Rất nhiều hệ thống đòi hỏi truy nhập vào các tệp hoặc cơ sở dữ liệu của hệ thống khác.

Vẽ một biểu đồ ngữ cảnh dựa trên tất cả các thông tin đã xử lý. Chú ý chỉ minh họa những luồng dữ liệu thể hiện những mục tiêu chính của hệ thống nhằm tránh việc biểu đồ ngữ cảnh có quá nhiều luồng dữ liệu vào/ra.



Hình 5-5 Biểu đồ ngữ cảnh của hệ thống quản lý bán điện

5.7. Trình tự và quy tắc xây dựng DFD

5.7.1. Các bước xây dựng DFD

Kỹ thuật phổ biến được dùng để xây dựng DFD là kỹ thuật phân mức. Dựa theo BFD của hệ thống, chúng ta sẽ xây dựng DFD theo nhiều mức, mỗi mức thể hiện trên một hoặc nhiều trang. Nên đặt tên cho mỗi trang bằng tên của chức năng đang được phân tích trên trang đó. Như vậy với trang phân tích tại mức 0 thì tên của nó chính là tên của hệ thống.

- Biểu đồ ngữ cảnh

o Xác định hệ thống và giới hạn của nó (ngữ cảnh)

o Xác định các tác nhân ngoài (người cung cấp, người nhận thông tin hệ thống của)

o Xác định các luồng dữ liệu ngoài (đầu vào, đầu ra)

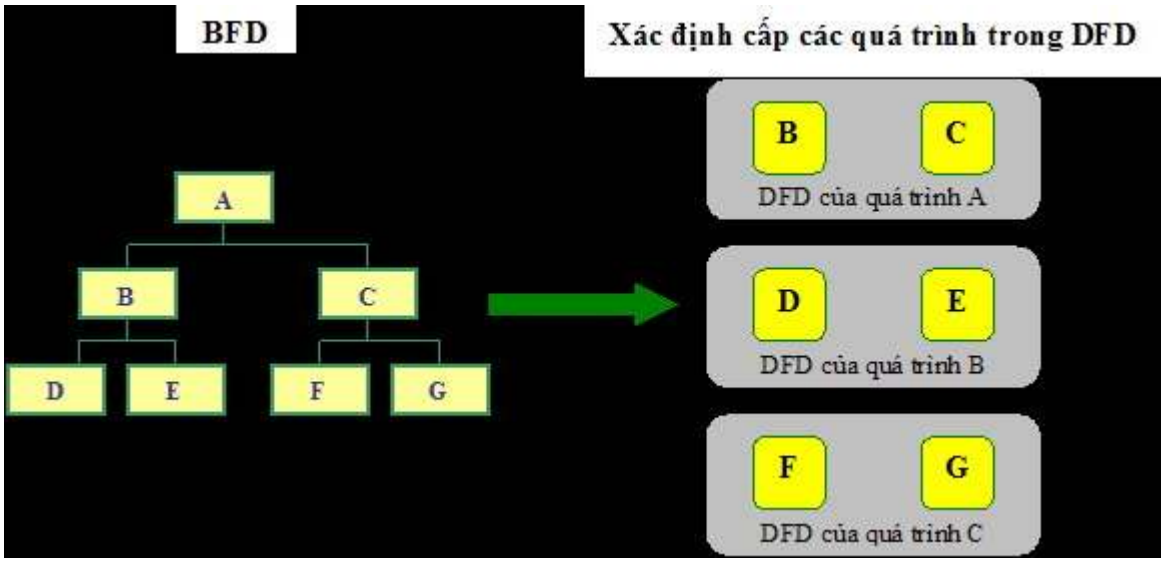
o Chú ý: bản thân toàn bộ hệ thống là một quá trình (nó nhận đầu vào và biến đổi nó thành đầu ra)

- DFD mức 0

- Xác định những gì đang được thực hiện ra giữa từng đầu vào và đầu ra tương ứng
- Xác định các quá trình
- Xác định các luồng dữ liệu ngoài giữa các tác nhân ngoài với các quá trình
- Xác định các luồng dữ liệu ngoài giữa các quá trình với các kho dữ liệu

- Các DFD mức 1

o Là các quá trình con của các quá trình mức 0



Hình 5-6 Kỹ thuật phân mức DFD dựa trên BFD

5.7.2. Các qui tắc xây dựng DFD

- Qui tắc 1: mỗi biểu tượng có riêng một nhãn duy nhất để tránh gây hiểu nhầm
- Qui tắc 2: Sử dụng một ĐỘNG TỪ để gán nhãn cho một quá trình (vì một quá trình là một hành động!!!)
- Qui tắc 3: Mỗi luồng dữ liệu phải được liên kết với ít nhất một quá trình
- Qui tắc 4: Phải có một góc tô đậm trong tất cả các thể hiện của một biểu tượng lặp trong cùng một biểu đồ
- Qui tắc 5: Một quá trình phải luôn có luồng dữ liệu vào và ra
- Qui tắc 6: Không cần có một luồng dữ liệu (mà không có sự biến đổi) liên kết với một quá trình (vì hoạt động như vậy là vô giá trị)
- Qui tắc 7: Các quá trình cha và các quá trình con tương ứng của nó phải có các luồng dữ liệu vào ra giống nhau (nhưng các quá trình con có thể có luồng dữ liệu của riêng nó)
- Qui tắc 8: Các luồng dữ liệu không thể tự phân tách được

- Qui tắc 9: Một gói dữ liệu có thể gồm nhiều phần tử dữ liệu được truyền đi đồng thời tới cũng một đích
- Qui tắc 10: Không được sử dụng mũi tên hai chiều vì luồng vào (cập nhật) và luồng ra (trích thông tin) của một kho dữ liệu mang nội dung thông tin khác nhau

5.7.3. Phân tích hướng cấu trúc cổ điển (top-down)

1. Vẽ DFD lôgíc dạng top-down của hệ thống hiện có

Vẽ các biểu đồ DFD vật lý dạng top-down để biểu diễn sự cài đặt vật lý hiện tại của hệ thống bao gồm các giới hạn của nó.

2. Chuyển đổi các DFD vật lý thành các DFD lôgíc tương ứng của nó.

3. Vẽ các DFD lôgíc dạng top-down biểu diễn một hệ thống được cải thiện.

4. Mô tả tất cả các luồng dữ liệu, kho dữ liệu, quy tắc và thủ tục trong một từ điển dữ liệu.

5. Một cách tuỳ chọn, đánh dấu các bản sao của DFD lôgíc để biểu diễn các giải pháp vật lý khác nhau.

6. Vẽ các DFD vật lý dạng top-down biểu diễn giải pháp được lựa chọn.

5.7.4. Phân tích hướng cấu trúc hiện đại (hướng sự kiện)

1. Vẽ một DFD ngữ cảnh để xác lập phạm vi ban đầu của dự án.

2. Vẽ một biểu đồ phân rã chức năng để phân chia hệ thống thành các hệ thống con.

3. Tạo một danh sách các đáp ứng sự kiện hay use-case cho hệ thống để xác định các sự kiện mà hệ thống phải có đáp ứng.

4. Vẽ một DFD sự kiện (hay bộ xử lý sự kiện) cho từng sự kiện.

5. Kết hợp các DFD sự kiện thành một biểu đồ hệ thống (hay đối với các hệ thống lớn thì là các biểu đồ hệ thống con).

6. Vẽ các DFD chi tiết cho các bộ xử lý sự kiện phức tạp hơn.

7. Tài liệu hoá các luồng dữ liệu và quá trình trong từ điển dữ liệu.

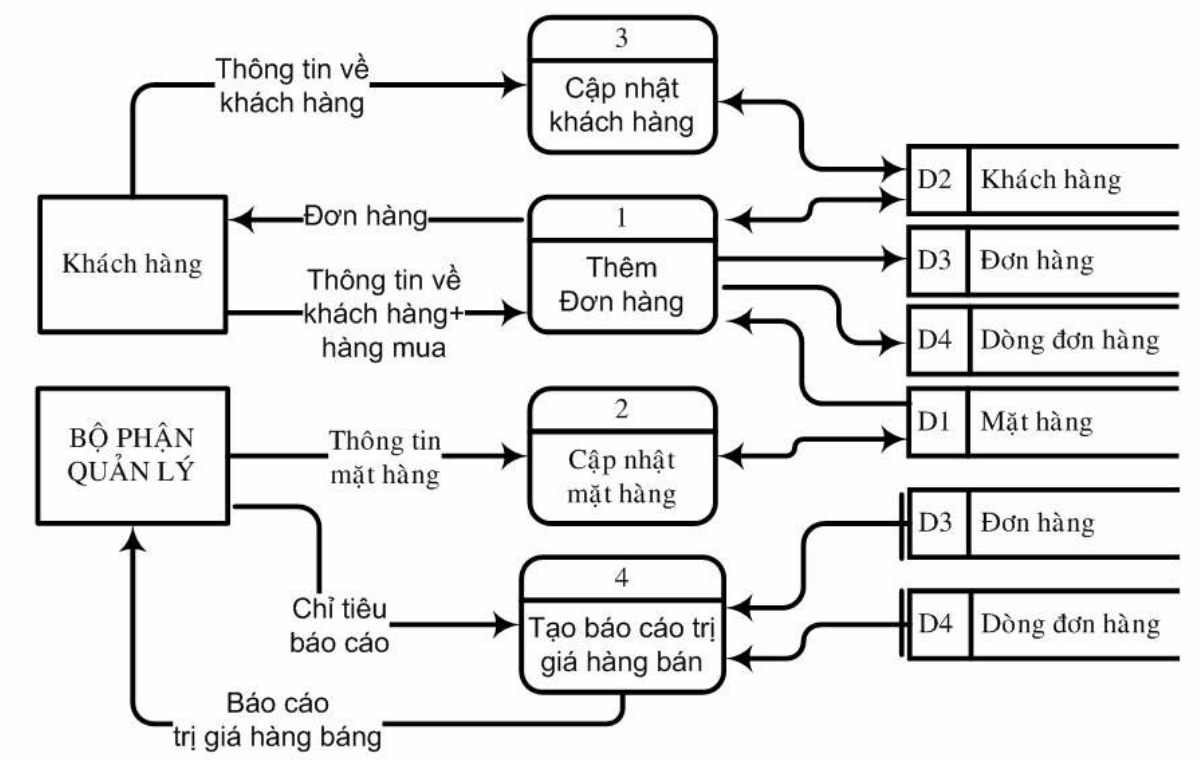
5.7.5. So sánh DFD với biểu đồ tiến trình (flowchart)

- Các quá trình trong DFD có thể thực hiện song song (vào cùng một thời gian)
    - Các quá trình trong flowchart thực hiện mỗi cái vào một thời điểm
- DFD thể hiện luồng dữ liệu xuyên suốt hệ thống
    - Flowchart thể hiện luồng điều khiển (trình tự và sự chuyển giao điều khiển)

- Các quá trình trong một DFD có thể có thời gian khác nhau đáng kể (hàng ngày, hàng tuần, cấp bách)
    - Các quá trình trong flowchart là một phần của một chương trình đơn nhất với thời gian nhất quán

5.7.6. Sự bảo toàn dữ liệu

- Là việc bảo đảm rằng một luồng dữ liệu chỉ chứa dữ liệu cần thiết cho quá trình nhận nó.
    - Xác định và loại bỏ những phần không hiệu quả
    - Đơn giản hoá việc giao tiếp giữa các quá trình
    - Phải xác định chính xác kết cấu dữ liệu của từng luồng dữ liệu, được biểu diễn dưới dạng các cấu trúc dữ liệu (trong phần Mô hình hoá dữ liệu)



Hình 5-7 DFD mức 0 của việc quản lý đơn hàng

Câu hỏi thảo luận

5.1. Hệ thống là gì?

5.2. Nêu mục đích của việc mô hình hóa hệ thống.

5.3. Phân biệt mô hình lôgíc và mô hình vật lý.

GỢI Ý: tham khảo mục Mô hình lôgíc.

5.4. Giải thích tại sao mô hình lôgíc lại đóng vai trò quan trọng trong phân tích hệ thống?

5.5. Ý nghĩa của biểu đồ phân cấp chức năng là gì?

5.6. Tại sao trong BFD, mỗi chức năng không nên có quá 6 chức năng con?

GỢI Ý: tham khảo mục Phương pháp xây dựng BFD.

5.7. Nêu vai trò của DFD.

5.8. Khi xây dựng DFD, có thể gặp những sai sót nào?

5.9. Thảo luận về BFD cho hệ thống quản lý bán điện trong hình 5-3.

5.10. Giải thích ví dụ về cách tách các quá trình trong hình 5-5.

5.11. Thảo luận về biểu đồ ngữ cảnh của hệ thống quản lý bán điện trong hình 5-7.

5.12. Thảo luận về DFD mức 0 của hệ thống quản lý bán điện trong hình 5-9.

5.13. Xây dựng các DFD còn lại cho hệ thống quản lý bán điện.

Câu hỏi trắc nghiệm

1. Trong BFD, tên của một chức năng phải bắt đầu bằng:

- a. Danh từ
- b. Tính từ.
- c. Động từ.

2. Trong DFD, tên của luồng dữ liệu phải bắt đầu bằng:

- a. Động từ.
- b. Tính từ.
- c. Danh từ.

3. Phát biểu nào sau đây là đúng?

- a. Một quá trình có thể chỉ có một luồng dữ liệu vào và không có luồng dữ liệu ra.

- b. Một quá trình có thể không có luồng dữ liệu đi vào/đi ra.
- c. Một quá trình phải có ít nhất một luồng dữ liệu vào và một luồng dữ liệu ra.

4. Phát biểu nào sau đây là đúng?

- a. Mỗi luồng dữ liệu có thể không liên kết với quá trình nào.
- b. Mỗi luồng dữ liệu phải liên kết với ít nhất một quá trình.
- c. Cả hai phát biểu trên đều sai.

Tổng kết chương 5

Chương 5 đã trình bày các khái niệm cơ bản về Mô hình hoá hệ thống và đặc biệt là mô hình hoá chức năng, cách thức xây dựng biểu đồ phân cấp chức năng và biểu đồ luồng dữ liệu.

Người học phải phân biệt được mô hình lôgíc và mô hình vật lý.

Người học cần nắm vững trình tự xây dựng biểu đồ phân cấp chức năng và có thể áp dụng để thực hiện xây dựng biểu đồ phân cấp chức năng cho các hệ thống cụ thể.

Người học phải hiểu và nắm vững các phần tử của một DFD:

- Tác nhân ngoài
- Kho dữ liệu
- Luồng dữ liệu
- Quá trình

Người học cần nẵm rõ khái niệm biểu đồ luồng dữ liệu ngữ cảnh, trình tự và qui tắc xây dựng biểu đồ luồng dữ liệu. Từ đó, có thể vận dụng để vẽ biểu đồ luồng dữ liệu cho các hệ thống cụ thể.

Hướng dẫn bài tập lớn

Vận dụng kiến thức đã học trong chương 5, người học xây dựng biểu đồ phân cấp chức năng, biểu đồ luồng dữ liệu ngữ cảnh và phân rã biểu đồ luồng dữ liệu cho hệ thống đã chọn làm bài tập lớn. Viết tiếp báo cáo bài tập lớn với các nội dung cơ bản:

- Biểu đồ phân cấp chức năng
- Biểu đồ luồng dữ liệu
    - Biểu đồ luồng dữ liệu mức ngữ cảnh
    - Biểu đồ luồng dữ liệu mức 0
    - Biểu đồ luồng dữ liệu mức 1
    - ...

CHƯƠNG 6. MÔ HÌNH HOÁ DỮ LIỆU

Mục tiêu

Chương 6 tập trung vào việc mô hình hóa dữ liệu bao gồm khái niệm và phương pháp xây dựng biểu đồ thực thể quan hệ và biểu đồ dữ liệu quan hệ.

6.1. Mô hình hóa dữ liệu

6.1.1. Khái niệm

- Mô hình hoá dữ liệu (mô hình hoá cơ sở dữ liệu, mô hình hoá thông tin) là một kỹ thuật để tổ chức và tài liệu hoá dữ liệu của hệ thống trong một mô hình. Kỹ thuật này xác định các yêu cầu nghiệp vụ đối với một cơ sở dữ liệu. Mô hình hóa dữ liệu thường được gọi là mô hình hóa cơ sở dữ liệu vì cuối cùng một mô hình dữ liệu luôn được cài đặt thành cơ sở dữ liệu.
- Biểu đồ quan hệ thực thể (Entity Relationship Diagram - ERD) mô tả dữ liệu dưới dạng các thực thể và các quan hệ được mô tả bởi dữ liệu. ERD xác định các đơn vị thông tin cơ sở cần thiết cho hệ thống (các thực thể) và các mối quan hệ giữa chúng. Nghĩa là tất cả các dữ liệu chỉ được lưu giữ một lần trong toàn bộ hệ thống.

6.1.2. Từ mô hình dữ liệu tới cài đặt cơ sở dữ liệu

- ERD: là một mô hình khái niệm của các thực thể dữ liệu, các thuộc tính (đặc điểm) và các quan hệ (với các thực thể khác) của chúng trong một hệ thống thông tin (độc lập kỹ thuật). (Phân tích)

Mô hình dữ liệu quan hệ (Relational Data Model - RDM): một bản thiết kế cho việc cài đặt của một mô hình dữ liệu khái niệm (ERD) trong môi trường cơ sở dữ liệu quan hệ (độc lập phần mềm). (Thiết kế)

- Sơ đồ quan hệ : một sơ đồ thể hiện cách thức một mô hình dữ liệu được cài đặt với hệ quản trị cơ sở dữ liệu (như Microsoft Access hay MS SQL Server…). (Cài đặt)

6.1.3. Vai trò của biểu đồ quan hệ thực thể

- Cơ sở dữ liệu = dữ liệu + quan hệ
- ERD được dùng để mô hình hoá dữ liệu và quan hệ của chúng
- ERD là một biểu diễn đồ hoạ của mô hình dữ liệu khái niệm
- ERD là độc lập tài nguyên: nó không gắn với bất cứ môi trường cơ sở dữ liệu cụ thể nào

6.2. Các phần tử của biểu đồ quan hệ thực thể (ERD)

6.2.1. Thực thể

Thực thể là một nhóm các thuộc tính tương ứng với một đối tượng khái niệm mà chúng ta cần thu thập và lưu trữ dữ liệu về nó

o Các vật thể, con người, địa điểm, sự kiện, khái niệm mà sự tồn tại của nó không phụ thuộc vào các thực thể khác

- Thực thể là một tập các thể hiện của đối tượng mà nó biểu diễn
- Thực thể phải có một tên duy nhất (một danh từ số ít), từ định danh duy nhất và ít nhất một thuộc tính (chính là từ định danh)
- Các loại thực thể có thể có:

o Con người: là những người thực hiện chức năng nào đó trong hoặc ngoài hệ thống. Ví dụ: công ty, khách hàng, phòng ban, bộ phận, nhân viên, giáo viên, sinh viên, nhà cung cấp…

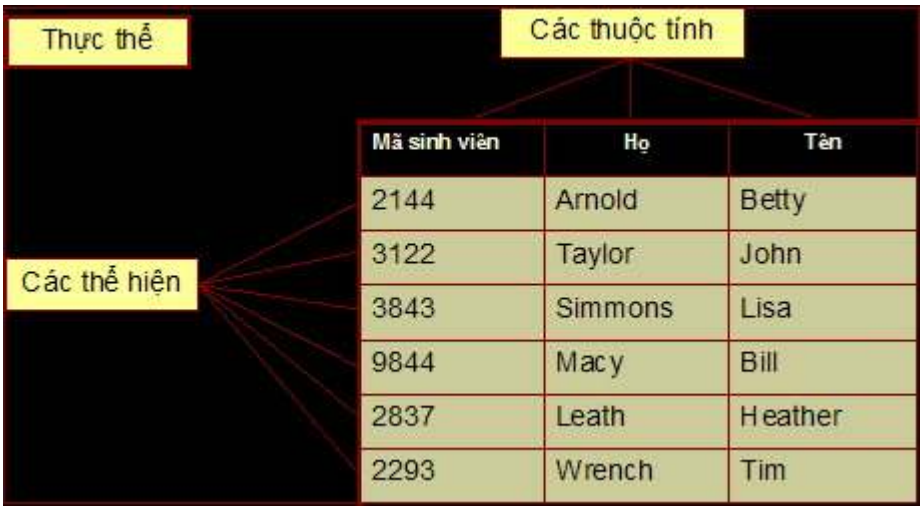
o Địa điểm: là nơi được sử dụng bởi con người. Ví dụ: nơi bán hàng, toà nhà, phòng, chi nhánh…

o Vật thể: là những đối tượng vật lý thấy rõ. Ví dụ: sách, tạp chí, sản phẩm, nguyên liệu thô, công cụ…

o Sự kiện: là những gì xảy ra theo thời gian hoặc theo một quy trình nhất định. Ví dụ: giải thưởng, sự huỷ bỏ, chuyến bay, giờ học, việc lập hoá đơn, việc đặt hàng, việc đăng ký, sự gia hạn, sự đặt chỗ, việc bán hàng…

o Khái niệm: là những gì không thể nhìn thấy được. Ví dụ: tài khoản, khoảng thời gian, khoá học, nguồn tài chính, quy tắc, luật lệ…

- Trong ERD, thực thể được ký hiệu là một hình chữ nhật, mỗi thực thể tương đương với một bảng dữ liệu trong cơ sở dữ liệu của hệ thống.
- Thể hiện của thực thể: là một thực thể cụ thể. Ví dụ thực thể SinhVien có thể có nhiều thể hiện như Betty, John, Lisa…



Mã sinh viên	Họ	Tên
2144	Arnold	Betty
3122	Taylor	John
3843	Simmons	Lisa
9844	Macy	Bill
2837	Leath	Heather
2293	Wrench	Tim

6.2.2. Thuộc tính

Mỗi thực thể bao gồm nhiều thông tin, mỗi thông tin là một thuộc tính của tập thực thể, ứng với một trường trong bảng dữ liệu tương ứng. Ví dụ: khách hàng Nguyễn Văn A có năm sinh là 1981, có số điện thoại là 8534… . Tập thực thể khách hàng sẽ có các thuộc tính "năm sinh", "số điện thoại". Một thuộc tính là một đặc tính mô tả hoặc đặc điểm quan tâm của một thực thể.

- Kiểu dữ liệu (Data type) của một thuộc tính xác định kiểu dữ liệu có thể lưu trữ được trong thuộc tính đó
- Phạm vi (Domain) của một thuộc tính xác định các giá trị mà thuộc tính đó có thể chứa một cách hợp lệ
    - Giá trị mặc định (default value) của một thuộc tính là giá trị sẽ được ghi vào nếu không được xác định bởi người dùng
- Kiểu dữ liệu

Các kiểu dữ liệu lôgíc điển hình cho các thuộc tính	
Kiểu dữ liệu lôgíc	**Ý nghĩa lôgíc**
NUMBER	Có thể là số thực hoặc số nguyên
TEXT	Một chuỗi ký tự, bao gồm cả các con số
MEMO	Tương tự như TEXT nhưng có kích thước không xác định. Một số hệ thống đòi hỏi khả năng này để lưu các văn bản dài trong bản ghi cơ sở dữ liệu
DATE	Ngày dưới bất kỳ định dạng nào

TIME	Giờ dưới bất kỳ định dạng nào
YES/NO	Một thuộc tính chỉ có thể nhận một trong hai giá trị bên
VALUE SET	Một tập hữu hạn các giá trị. Trong hầu hết các trường hợp, một lược đồ mã sẽ được xây dựng (ví dụ M=Nam giới, F=Nữ giới)
IMAGE	Các loại hình ảnh

Bảng 6-1 Các kiểu dữ liệu lôgíc điển hình cho các thuộc tính

- Phạm vi dữ liệu

Các phạm vi lôgíc điển hình cho các kiểu dữ liệu lôgíc		
Kiểu dữ liệu	**Phạm vi**	**Ví dụ**
NUMBER	Đối với số nguyên, xác định phạm vi Đối với số thực, xác định phạm vi và độ chính xác	{10-99} {1.000-799.999}
TEXT	Kích thước lớn nhất của thuộc tính Các giá trị thực tế thường là vô hạn; tuy nhiên, người dùng có thể xác định các hạn chế nào đó	Text(30)
DATE	Sự biến đổi trên các định dạng MMDDYYYY	MMDDYYYY MMYYYY
TIME	Đối với thời gian AM/PM: HHMMT Đối với thời gian 24 giờ: HHMM	HHMMT HHMM
YES/NO	{YES, NO}	{YES, NO} {ON, OFF}

VALUE SET	{giá trị#1, giá trị#2,…giá trị#n} {bảng các mã và ý nghĩa}	{M=Nam giới F=Nữ giới}

Bảng 6-2 Các phạm vi lôgíc điển hình cho các kiểu dữ liệu lôgíc

- Giá trị mặc định

Các giá trị mặc định chấp nhận được cho các thuộc tính		
Giá trị mặc định	**Diễn giải**	**Vi dụ**
Một giá trị hợp lệ từ phạm vi	Đối với một thể hiện của thuộc tính, nếu người dùng không xác định giá trị thì sử dụng giá trị này	0 1.00
NONE hoặc NULL	Đối với một thể hiện của thuộc tính, nếu người dùng không xác định giá trị thì để trống	NONE NULL
Required hay NOT NULL	Đối với một thể hiện của thuộc tính, đòi hỏi người dùng phải nhập vào một giá trị hợp lệ từ phạm vi. (Điều này được dùng khi không có giá trị nào trong phạm vi có thể làm giá trị mặc định nhưng lại nhất thiết phải có giá trị được nhập)	REQUIRED NOT NULL

Bảng 6-3 Các giá trị mặc định chấp nhận được cho các thuộc tính

- Có 3 loại thuộc tính:

o Thuộc tính khóa: gồm một hoặc nhiều thuộc tính trong thực thể được dùng để gán cho mỗi thể hiện thực thể một cách tham khảo duy nhất. Ví dụ thuộc tính Mã sinh viên trong thực thể Sinh viên

o Thuộc tính mô tả: là các thuộc tính dữ liệu mô tả về một đối tượng và không được chọn làm thuộc tính khóa. Ví dụ các thuộc tính Tên sinh viên, Địa chỉ…

o Thuộc tính kết nối: là thuộc tính mà với thực thể này thì là thuộc tính mô tả nhưng với thực thể khác thì là thuộc tính khóa, nó đóng vai trò kết nối các thực thể có quan hệ với nhau

6.2.3. Mối quan hệ

- Một quan hệ tài liệu hoá một liên kết giữa một, hai hoặc nhiều thực thể. Nó phải có một cái tên (và có thể mang dữ liệu).

Quan hệ 1-1:

- Là mối quan hệ trong đó một thực thể của tập thực thể này tương ứng với duy nhất một thực thể của tập thực thể kia và ngược lại. Ví dụ, một thực thể hóa đơn hàng chỉ ứng với duy nhất một thực thể chi tiết hóa đơn mô tả nó.
- Quan hệ 1-1 được biểu diễn bằng một mũi tên hai đầu hoặc là một đoạn thẳng…
- Quan hệ này sẽ dẫn tới việc nhập chung hai tập thực thể thành một tập thực thể, tập thực thể mới phải bao gồm các thuộc tính của hai tập thực thể cũ.

Quan hệ 1-n:

- Là mối quan hệ mà trong đó một thực thể của tập thực thể này có quan hệ với nhiều thực thể của tập thực thể kia. Ví dụ, một khách hàng có thể đặt nhiều đơn hàng nên một thực thể khách hàng trong tập thực thể khác hàng có quan hệ với nhiều thựuc thể đơn hàng trong tập thực thể đơn hàng.
- Quan hệ 1- nhiều được biểu diễn bằng một mũi tên 1 đầu hướng từ bên nhiều tới bên 1 hoặc là một đoạn thẳng với một đầu là trạc ba hướng về bên nhiều…
- Quan hệ này đóng vai trò rất quan trọng thể hiện mối liên hệ giữa các thực thể trong mô hình. Ở đây, thuộc tính khóa của bên một sẽ là thuộc tính kết nối của bên nhiều.

Quan hệ n-n:

- Là mối quan hệ mà trong đó một thực thể của tập thực thể này có quan hệ với nhiều thực thể của tập thực thể kia và ngược lại. Ví dụ, một nhà cung cấp trong tập thực thể nhà cung cấp có thể cung cấp nhiều loại hàng trong tập thực thể Hàng hóa và ngược lại một loại hàng có thể được cung cấp bởi nhiều nhà cung cấp.
- Quan hệ nhiều - nhiều được biểu diễn bằng một đoạn thẳng hoặc là một đoạn thẳng có trạc ba ở cả hai đầu…
- Quan hệ này không thể hiện được mối quan hệ giữa hai thực thể cũng như không cho thấy điều gì về mặt nghiệp vụ, nên thường tách thành hai quan hệ 1-n bằng cách tạo một tập thực thể trung gian có quan hệ 1- n với cả hai tập thực thể đã có. Ví dụ với quan hệ n-n giữa nhà cung cấp và hàng hóa, ta sẽ tạo tập thực thể nhà cung cấp/hàng hóa có quan hệ là một nhà cung cấp gồm nhiều

dòng nhà cung cấp/hàng hóa và một hàng hóa lại ứng với nhiều dòng nhà cung cấp/hàng hóa.

- Số yếu tố tài liệu hoá số lượng các thể hiện của một thực thể có thể có quan hệ với một thể hiện của thực thể khác trong một quan hệ

o Bao gồm số lớn nhất và nhỏ nhất các thể hiện

o Phản ánh quy tắc nghiệp vụ hoặc thực tế nghiệp vụ nó chung (ví dụ có bao nhiêu lớp học mà một sinh viên có thể tham gia, có bao nhiêu sinh viên có thể có trong một lớp học).

- Bậc của quan hệ xác định số lượng thực thể tham gia vào một quan hệ
  - Một ngôi
  - Hai ngôi
  - Ba ngôi
  - ...
- Quan hệ một ngôi (đệ qui) chỉ gồm một thực thể (các thể hiện trong cùng một thực thể)
- Quan hệ hai ngôi
- Quan hệ ba ngôi

6.3. Xây dựng biểu đồ quan hệ thực thể

6.3.1. Các bước mô hình hóa dữ liệu lôgíc

1. Mô hình dữ liệu ngữ cảnh
   - Để thiết lập phạm vi dự án
2. Mô hình dữ liệu dựa trên khoá
   - Loại bỏ các quan hệ không cụ thể
   - Thêm các thực thể có liên quan
   - Bao gồm các khoá chính
   - Xác định chính xác số yếu tố
3. Mô hình dữ liệu với thuộc tính đầy đủ
   - Tất cả các thuộc tính còn lại
   - Các tiêu chuẩn nhóm con
4. Mô hình dữ liệu được chuẩn hoá
   - Thế nào là một mô hình dữ liệu tốt?

o Đơn giản

- Các thuộc tính dữ liệu mô tả bất cứ thực thể đã cho nào thì chỉ nên mô tả thực thể đó thôi
- Mỗi thuộc tính của một thể hiện của thực thể chỉ có thể có một giá trị

o Không dư thừa

- Mỗi thuộc tính dữ liệu, không phải là khoá ngoại, mô tả tối đa một thực thể
- Tìm cùng một thuộc tính được ghi lại nhiều lần dưới các tên khác nhau

o Linh động và dễ điều chỉnh cho những nhu cầu phát sinh trong tương lai

6.3.2. Trình tự xây dựng ERD

- Xác định các thực thể (Top-down)
- Xác định bậc của các quan hệ giữa các thực thể (rõ ràng ngữ cảnh)
- Hoàn thiện các quan hệ với các số yếu tố (rõ ràng ngữ cảnh)
- Xây dựng mô hình

Xác định các thực thể

o **Cách 1:**

Một tập thực thể có thể thuộc một trong 3 loại sau đây.

- Thông tin liên quan tới một giao dịch chủ yếu của hệ thống, ví dụ như hóa đơn bán hàng thuộc về quá trình bán hàng, đơn đặt hàng thuộc về quá trình mua hàng.
- Thông tin liên quan tới thuộc tính hoặc tài nguyên của hệ thống, ví dụ khách hàng, nhà cung cấp, vị trí kho hàng…
- Thông tin đã được khái quát dưới dạng thống kê liên quan tới lập kế hoạch hoặc quản lý như bằng chấm công, lịch trực…

Để nhận ra tập thực thể, phải đặt câu hỏi để ghi nhận thông tin về thực thể:

- Cái gì mà ta cần lưu thông tin về nó?
- Cái gì là cốt yếu trong hệ thống?
- Cái gì mà ta nói về nó trong hệ thống?
- Cái gì có thể dùng để phân biệt sự kiện của tập thực thể này với sự kiện của một tập thực thể khác?

o **Cách 2:**

- Lấy một bản mô tả về hệ thống hiện tại hoặc cần có trong tương lai, xem xét các danh từ có trong đó xem có phải là thông tin cần lưu giữ không. Chú ý loại bỏ các từ đồng nghĩa. Lưu ý là có những danh từ mang tính mô tả nhưng lại

không trở thành một tập thực thể, một số khác lại có thể là tập thực thể tiềm năng.

- Ví dụ: chương trình quản lý kho hàng

o Theo dõi hàng tồn trong một kho nào đó tại một thời điểm nào đó.

o Theo dõi chi tiết xuất nhập tồn của mỗi loại hàng hóa.

o In chi tiết xuất nhập vật tư cho mỗi khách hàng.

Vậy các tập thực thể xác định được từ mô tả này chính là: vật tư, kho hàng, khách hàng.

- **Xác định mối quan hệ**

Quan hệ giữa các tập thực thể thường được diễn tả bởi các động từ, nó xác định sự tác động của các thực thể với nhau. Để xác định được các mối quan hệ giữa các tập thực thể, cần chú ý:

- Nếu cần phải lưu giữ thông tin về tập thực thể này trong tập thực thể kia thì sẽ có một quan hệ xuất hiện để tạo mối liên kết.
- Khi quan hệ giữa hai thực thể là gián tiếp thì ta không cần phải xây dựng mối quan hệ giữa chúng.
- **Ví dụ**:

Hệ thống thông tin quản lý tồn kho gồm 4 thực thể cơ bản sau:

Tên dữ liệu	**Bao gồm**
Danh mục tồn kho	+ Thẻ giá + Hồ sơ tồn kho + Báo cáo hàng tồn kho cũ + Danh sách đặt hàng bổ sung + Báo cáo mức tồn kho + Bản kiểm kê tồn kho
Danh mục hàng bán	+ Hoá đơn bán hàng + Báo cáo bán hàng
Danh mục nhà cung cấp	+ Hoá đơn nhà cung cấp + Đơn đặt hàng bổ sung
Danh mục bảo hành	+ Đơn bảo hành

Quan hệ giữa Danh mục tồn kho và Danh mục nhà cung cấp là quan hệ n-n vì:

- Mỗi loại sản phẩm trong danh mục tồn kho có thể có hơn một nhà cung cấp trong danh mục nhà cung cấp.
- Mỗi nhà cung cấp trong danh mục nhà cung cấp có thể cung cấp nhiều hơn một loại sản phẩm.

Chúng ta đưa ra tập thực thể mới là **Danh mục đặt hàng** có quan hệ 1 - n với cả hai tập thực thể ban đầu. Việc này nhằm tách 1 quan hệ n-n thành 2 quan hệ 1-n.

Tương tự, có quan hệ n – n giữa danh mục đặt hàng và danh mục tồn kho. Chúng ta đưa ra tập thực thể **Chi tiết đặt hàng**.

Chúng ta đã thu thập được danh sách một số phần tử dữ liệu, cần xác định những thực thể còn thiếu. Bắt đầu với dữ liệu bán hàng, có một vài trường của hoá đơn bán mô tả một khách hàng. Vậy khách hàng là một đối tượng mà dữ liệu tập hợp vào đó. Do vậy **Danh mục khách hàng** là một tập thực thể.

Nhận xét rằng quan hệ giữa danh mục khách hàng và danh mục tồn kho là quan hệ n-n vì:

- Một khách hàng có thể mua nhiều loại sản phẩm.
- Một loại sản phẩm có thể được bán cho nhiều khách hàng.

Với các thuộc tính còn lại trong dữ liệu bán hàng ta có **Danh mục bán hàng,** từ đó chúng ta có quan hệ n-n giữa Danh mục bán hàng với Danh mục tồn kho vì:

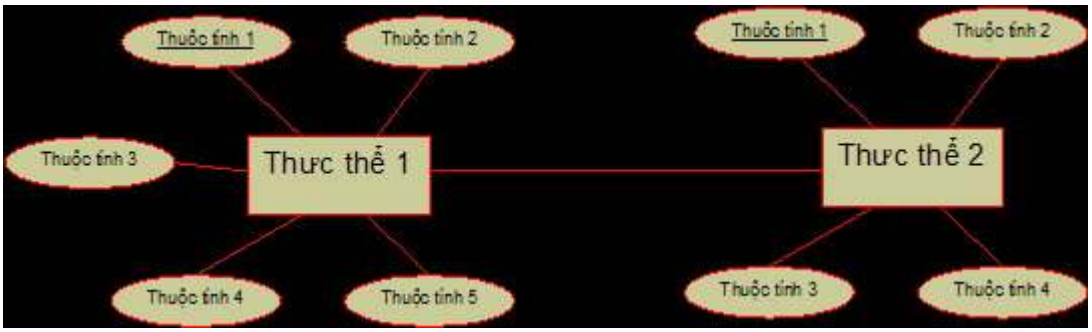
- Một giao dịch bán có thể có nhiều hơn một loại hàng.
- Một loại hàng có thể có trong nhiều giao dịch bán.

Do vậy , ta đưa ra tập thực thể mới là **Chi tiết bán hàng**

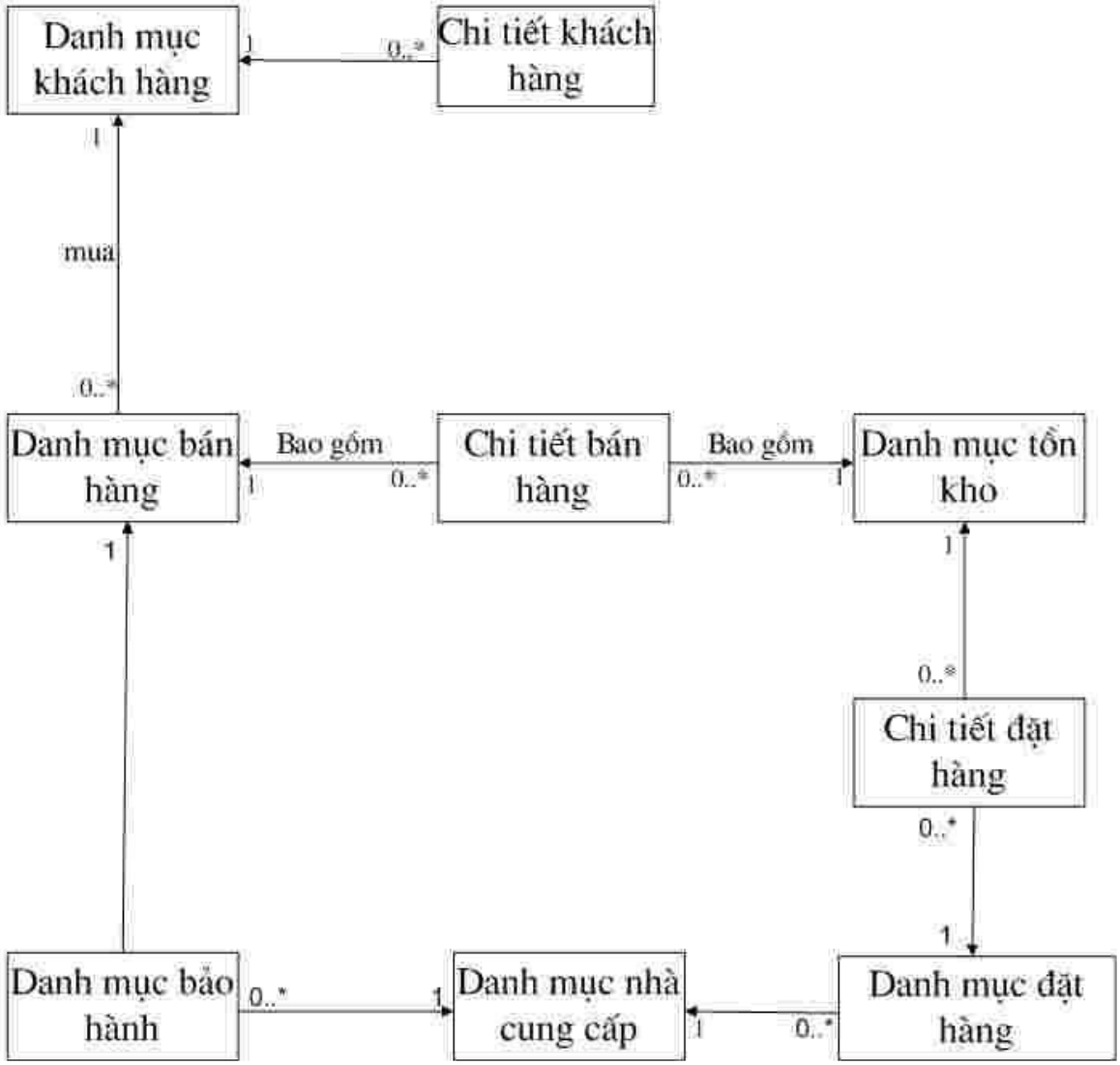
6.3.3. Qui tắc xây dựng ERD

- Mỗi thực thể phải có tên
- Mỗi thực thể phải có định danh
- Mỗi thể hiện không thể là một thực thể
- Mỗi quan hệ phải có tên (có thể mang hoặc không mang dữ liệu)
- Số yếu tố phải hợp lý (rõ ràng ngữ cảnh)

6.3.4. Các kiểu ký hiệu ERD khác



Sơ đồ thực thể liên kết (ERD)



Hình 6-4 Biểu đồ quan hệ thực thể của Hệ thống quản lý kho

6.4. Xây dựng biểu đồ dữ liệu quan hệ (RDM)

Mô hình dữ liệu quan hệ (RDM – Relational Data model) là công cụ tiếp theo sau ERD được dùng trong việc mô hình hóa dữ liệu nhằm mục đích xác định danh sách các thuộc tích của các thực thể.

Quá trình xây dựng RDM bao gồm các bước:

- Xác định các thuộc tính cần thiết.
- Chuẩn hóa các thực thể.
- Xác định các mối quan hệ giữa các thuộc tính của các thực thể.

6.4.1. Xác định thuộc tính

- Để xác định các thuộc tính của các thực thể, cần dựa vào những yếu tố sau:
    - Sự hiểu biết về hệ thống đang phân tích.
    - Quá trình phỏng vấn, trao đổi với người sử dụng.
    - Các báo cáo, biểu mẫu được dùng trong hệ thống hiện tại.
- Từ những thông tin thu thập được, ta sẽ lập danh sách các thuộc tính cho các thực thể đã được xác định trong ERD. Phân biệt các thuộc tính khóa bằng cách gạch dưới.

6.4.2. Phân tích và chuẩn hóa dữ liệu

- Phân tích dữ liệu là một quá trình chuẩn bị một mô hình dữ liệu cho việc cài đặt thành một cơ sở dữ liệu đơn giản, không dư thừa, mềm dẻo và dễ thích ứng. Kỹ thuật cụ thể được gọi là sự chuẩn hóa.
- Chuẩn hóa là một kỹ thuật tổ chức các thuộc tính dữ liệu sao cho chúng được nhóm thành các thực thể không dư thừa, ổn định, mềm dẻo và dễ thích ứng:

o Không có sự lặp lại các thuộc tính ở các bảng khác nhau, trừ thuộc tính khóa và thuộc tính kết nối

o Không chứa các thuộc tính có giá trị là kết quả tính được của các thuộc tính khác. Ví dụ, thuộc tính giá thành là kết quả của thuộc tính số lượng nhân với thuộc tính đơn giá nên cần phải loại bỏ.

o Không có vai trò giống nhau giữa các thực thể

- **Khái niệm phụ thuộc hàm**:

o Phụ thuộc hàm đơn trị: từ 1 giá trị của khóa trong bảng, ta chỉ xác định được 1 giá trị cho các thuộc tính khác.VD: với mỗi mã khách hàng, chỉ có duy nhất một giá trị Họ tên, số điện thoại, địa chỉ…

o Phụ thuộc hàm đa trị: 1 giá trị của khóa trong bảng lại ứng với nhiều giá trị của các thuộc tính khác. Ví dụ: ứng với một mã số học sinh lại có nhiều môn học khác nhau vì một học sinh có thể học nhiều môn học.

o Như vậy, nếu có thuộc tính không phụ thuộc hàm vào khóa thì nó phải nằm trong một thực thể khác. Quá trình chuẩn hóa được thực hiện dựa trên khái niệm phụ thuộc hàm nêu trên.

- **Chuẩn hóa dạng 1:**

o Yêu cầu: các thuộc tính nào có thể xuất hiện nhiều lần với cùng một thực thể thì loại bỏ ra.

o Các thuộc tính bị loại ra sẽ cùng với thuộc tính khóa của thực thể ban đầu tạo thành một tập thực thể mới.

- **Chuẩn hóa dạng 2:**

o Yêu cầu: tất cả các thuộc tính trong thực thể phải phụ thuộc hàm vào toàn bộ khóa.

o Đối với các thực thể chỉ có một trường là khóa thì đương nhiên thỏa mãn dạng chuẩn 2.

o Đối với các thực thể có khóa bao gồm 2 thuộc tính trở lên, nếu trong đó có những thuộc tính phụ thuộc hàm đơn trị vào một bộ phận của khóa thì tách các thuộc tính đó ra thành 1 thực thể mới với khóa là bộ phận khóa của thực thể ban đầu mà nó phụ thuộc hàm.

- **Chuẩn hóa dạng 3:**

o Yêu cầu: tất cả các thuộc tính phải phụ thuộc đơn trị vào khóa và không phụ thuộc hàm đơn trị vào bất kỳ thuộc tính nào không phải là khóa trong thực thể.

o Tách những thuộc tính phụ thuộc hàm đơn trị vào thuộc tính không phải là khóa, đưa chúng vào thực thể mới có khóa chính là thuộc tính mà nó phụ thuộc hàm.

Ví dụ 1: xét quá trình xây dựng các thuộc tính cho các tập thực thể dựa trên mẫu hóa đơn bán hàng của một công ty

Số HĐ: **HÓA ĐƠN BÁN** Ngày…… Họ tên khách hàng:…….Mã số khách hàng:……..

Địa chỉ:……

Mã sô mặt hàng	**Tên hàng**	**Số lượng**	**Đơn giá**	**Thành tiền**

- Quá trình chuẩn hóa diễn ra như sau:

Thuộc tính ban đầu chưa chuẩn hóa	Chuẩn hóa dạng 1 **1NF**	Chuẩn hóa dạng 2 **2NF**	Chuẩn hóa dạng 3 **3NF**
<u>Số hiệu đơn</u> Mã số khách hàng Ngày đặt hàng Tên khách hàng Địa chỉ Mã số mặt hàng Tên mặt hàng Số lượng Đơn giá	<u>Số hiệu đơn</u> Mã số khách hàng Ngày đặt hàng Tên khách hàng Địa chỉ <u>Số hiệu đơn</u> <u>Mã số mặt hàng</u> Tên mặt hàng Số lượng Đơn giá	<u>Số hiệu đơn</u> Mã số khách hàng Ngày đặt hàng Tên khách hàng Địa chỉ <u>Số hiệu đơn</u> <u>Mã số mặt hàng</u> Số lượng Đơn giá <u>Mã số mặt hàng</u> Tên mặt hàng	<u>Số hiệu đơn</u> Mã số khách hàng Ngày đặt hàng <u>Mã số khách hàng</u> Tên khách hàng Địa chỉ <u>Số hiệu đơn</u> <u>Mã số mặt hàng</u> Số lượng Đơn giá <u>Mã số mặt hàng</u> Tên mặt hàng

- Sau khi chuẩn hóa, thu được các thực thể sau:

Đơn hàng bán (<u>Số hiệu đơn hàng</u>, Mã số khách hàng, Ngày đặt hàng)

Khách hàng (Mã số khách hàng, Tên khách hàng, Địa chỉ khách hàng)

Dòng đơn hàng (Số hiệu đơn hàng, Mã số mặt hàng, Số lượng, Đơn giá)

Mặt hàng (Mã số mặt hàng, Tên mặt hàng)

- **Kết hợp các tập thực thể chung.**

Do việc chuẩn hóa xuất phát từ nhiều tài liệu khác nhau nên có thể sau khi chuẩn hóa sẽ xuất hiện các thực thể giống nhau. Cần phải hợp nhất chúng thành một thực thể mà chứa đủ các thuộc tính. Rất có thể sau giai đoạn này thì thực thể thu được sẽ không còn ở dạng chuẩn 3 nên cần phải thực hiện chuẩn hóa lại các thực thể mới.

Ví dụ 2:

Có 2 tập thực thể đơn đặt hàng được chuẩn hóa từ 2 tài liệu là đơn đặt hàng và tài liệu giao hàng như sau:

Đơn hàng (Số hiệu đơn hàng, Mã số khách hàng, Ngày đặt hàng)

Đơn đặt hàng (Số hiệu đơn hàng, Tình trạng đơn hàng, Địa chỉ giao hàng)

Sau khi kết hợp có:

Đơn hàng (Số hiệu đơn hàng, Mã số khách hàng, Ngày đặt hàng, Tình trạng đơn hàng, Địa chỉ giao hàng)

Thực thể mới không còn thỏa dạng chuẩn 3 vì địa chỉ giao nhận phụ thuộc hàm vào Mã số khách hàng là thuộc tính không phải là khóa của thực thể. Thực hiện chuẩn hóa,tách thuộc tính Địa chỉ giao nhận ra khỏi thực thể ta được thực thể mới:

Đơn hàng (Số hiệu đơn hàng, Mã số khách hàng, Ngày đặt hàng, Tình trạng đơn hàng)

6.4.3. Xác định các mối quan hệ

Theo ví dụ 6-1 và 6-2 sau khi chuẩn hóa, thu được các thực thể sau:

Đơn hàng bán (Số hiệu đơn hàng, Mã số khách hàng, Ngày đặt hàng)

Khách hàng (Mã số khách hàng, Tên khách hàng, Địa chỉ khách hàng)

Dòng đơn hàng (Số hiệu đơn hàng, Mã số mặt hàng, Số lượng, Đơn giá)

Mặt hàng (Mã số mặt hàng, Tên mặt hàng)

Giao nhận (Số hiệu giao nhận, Mã số khách hàng, Ngày giao)

Dòng giao hàng (Số hiệu giao nhận, Số hiệu đơn hàng, Mã số mặt hàng, Số lượng giao)

a. Ma trận thực thể/khóa.

- Để xác định các mối quan hệ giữa các thực thể , ta cần lập bảng ma trận thực thể/khóa. Trong đó, các cột liệt kê các tập thực thể, các hàng liệt kê các thuộc tính khóa của các thực thể.
- Ứng với mỗi ô giao giữa hàng và cột, nếu thuộc tính khóa có trong thực thể, ta đánh dấu X, nếu không là khóa của thực thể nhưng có xuất hiện trong đó thì đánh dấu O. Ví dụ:

Thực thể Thuộc tính khóa	Đơn hàng	Khách hàng	Dòng đơn hàng	Mặt hàng	Giao nhận	Dòng giao hàng
Số hiệu đơn hàng	X		X			X
Mã số khách hàng	O	X			X	
Mã số mặt hàng			X	X		X
Số hiệu giao nhận					X	X

b. Thiết lập các mối quan hệ.

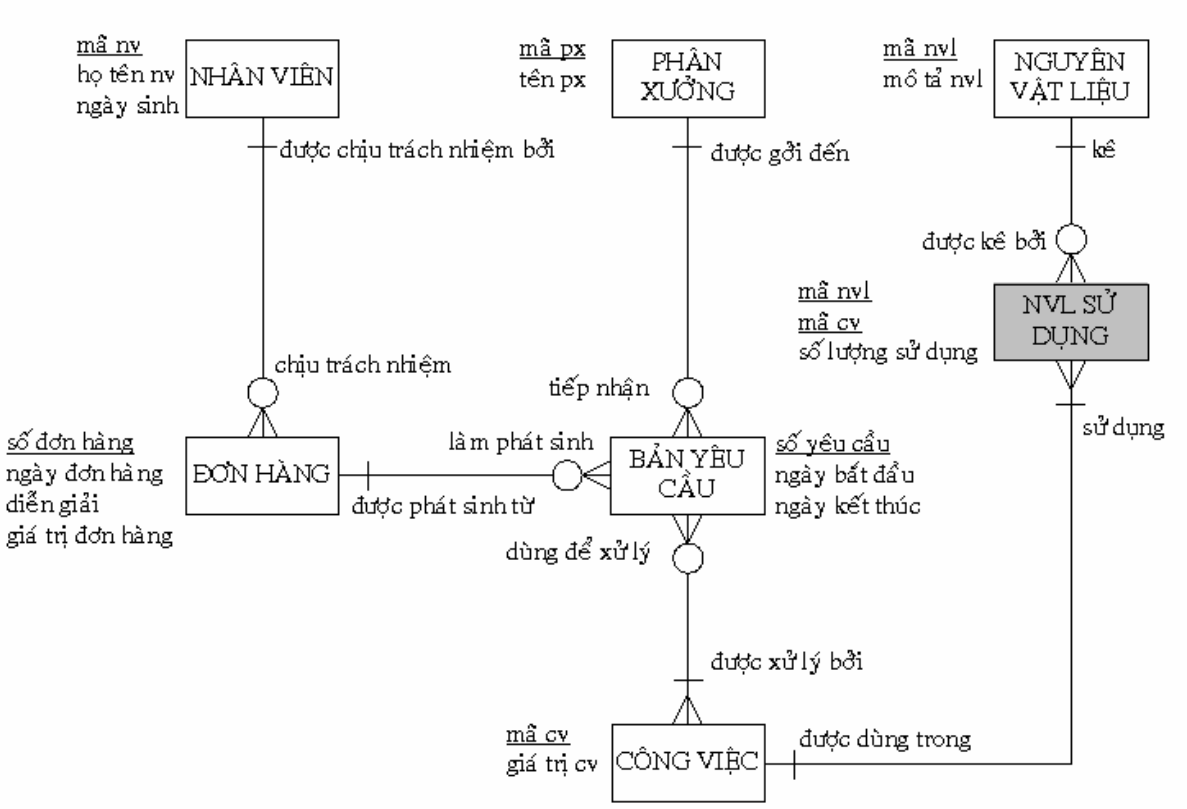
- Dựa theo bảng ma trận thực thể/khóa, ta xác định các mối quan hệ bằng cách bắt đầu từ cột thứ nhất, từ ô chứa khóa của nó, ta chiếu qua các ô kế tiếp trên cùng một hàng để xem nếu ô nào có chứa dấu X hoặc O thì ta có một liên kết giữa thực thể đang xét với thực thể có ô chứa dấu trên hàng đó.

6.4.4. Xây dựng mô hình RDM

- Sau khi đã thực hiện xong 3 bước trên, ta lập mô hình quan hệ.
- Tiếp theo, cần điều chỉnh để có được mô hình dữ liệu hoàn chỉnh:

o So sánh ERD được xây dựng ban đầu với RDM để chỉnh lý những điểm khác biệt sao cho hai mô hình phải phản ánh chính xác lẫn nhau.

o Trong một số trường hợp, nhà phân tích có thể đưa vào hoặc loại bỏ những quan hệ phụ để làm trong sáng mô hình.



Hình 6-5 Biểu đồ quan hệ dữ liệu của hệ thống quản lý phân xưởng

6.5. Từ điển dữ liệu

6.5.1. Khái niệm

Từ điển dữ liệu là bộ phận của tư liệu trong phân tích thiết kế, nó mô tả nội dung của các đối tượng theo định nghĩa có cấu trúc

Trong DFD các chức năng xử lý, kho dữ liệu, luồng dữ liệu chỉ mô tả ở mức khái quát thường là tập hợp các khoản mục riêng lẻ. Các khái quát này cần được mô tả chi tiết hoá hơn qua công cụ từ điển dữ liệu

Từ điển dữ liệu là sự liệt kê có tổ chức các phần tử dữ liệu thuộc hệ thống, liệt kê các mục từ chỉ tên gọi theo một thứ tự nào đó và giải thích các tên một cách chính xác chặt chẽ ngắn gọn để cho cả người dùng và người phân tích hiểu chung cái vào, cái ra, cái luân chuyển.

6.5.2. Kí hiệu mô tả nội dung cho từ điển dữ liệu:

Kết cấu dữ liệu	Ký pháp	Ý nghĩa
Định nghĩa	=	Được tạo từ
Tuần tự	+	Và
Tuyển chọn	[|]	Hoặc
Lặp	{ } n	Lặp n lần
Lựa chọn	( )	Dữ liệu tuỳ chọn
Giải thích	* Lời chú thích *	Giới hạn chú thích

6.5.3. Ví dụ một từ điển dữ liệu:

- Xác định một tờ hoá đơn:

Hoá đơn = Số HD + Ngày bán + Khách hàng +

+ Hàng * n

+ Số lượng * n

+ Thành tiền * n

+ Tổng cộng + KT trưởng + Người bán.

- Xác định thông tin về khách hàng

Khách hàng = Họ tên Khách + Tài khoản + Địa chỉ + Điện thoại

- Xác định thông tin về từng mặt hàng

Hàng = Mã hàng + Tên quy cách + Đơn vị tính + Đơn giá

- Họ tên khách cần được tách tên để thuận tiện đối với tên Tiếng Việt

Họ tên khách = Họ đệm + tên

Câu hỏi thảo luận

6.1. Phân biệt thuộc tính khóa và thuộc tính kết nối.

6.2. Quan hệ 1 - 1 xuất hiện trong các trường hợp nào? Hãy cho ví dụ.

6.3. Quan hệ nhiều - nhiều được xử lý như thế nào trong mô hình ERD?

6.4. Quan hệ 1 - nhiều thường xuất hiện trong những trường hợp nào? Hãy cho ví dụ.

6.5. Nêu ý nghĩa của số yếu tố trong ERD?

6.6. Phân biệt phụ thuộc hàm đơn trị và phụ thuộc hàm đa trị.

6.7. Phân tích mối quan hệ giữa các thực thể trong ERD của hệ thống quản lý kho ở hình 6-4.

6.8. Phân tích RDM của hệ thống quản lý bán điện trong hình 6-5.

6.9. Xây dựng RDM của hệ thống quản lý kho.

Câu hỏi trắc nghiệm

1. Thuộc tính kết nối là:

- a. Thuộc tính mô tả về một đối tượng và không được chọn làm thuộc tính khóa.
- b. Thuộc tính mà với thực thể này thì là thuộc tính mô tả nhưng với thực thể khác thì là thuộc tính khóa.
- c. Gồm một hoặc nhiều thuộc tính trong thực thể được dùng để gán cho mỗi thể hiện thực thể một cách tham khảo duy nhất.

2. Mối quan hệ mà trong đó một thực thể của tập thực thể này có quan hệ với nhiều thực thể của tập thực thể kia là:

- a. Quan hệ nhiều - nhiều.
- b. Quan hệ một - một.
- c. Quan hệ một - nhiều.

3. Khi gặp quan hệ nhiều - nhiều, thường thực hiện:

- a. Tách thành hai quan hệ một - nhiều.
- b. Gộp hai tập thực thể thành một tập thực thể.
- c. Cả hai câu trên đều sai.

4. Quan hệ một ngôi bao gồm mấy thực thể?

- a. Hai.
- b. Ba.
- c. Một.

5. Thế nào là một mô hình dữ liệu tốt?

- a. Đơn giản và không dư thừa.
- b. Linh động và dễ điều chỉnh.
- c. Cả hai ý trên.

Tổng kết chương 6

Chương 6 đã nêu các khái niệm về mô hình hoá dữ liệu, cách thức xây dựng biểu đồ thực thể quan hệ và biểu đồ dữ liệu quan hệ.

Người học cần hiểu và nắm rõ các thành phần của biểu đồ quan hệ thực thể:

- Thực thể
- Thuộc tính
- Mối quan hệ

Người học phải nắm vững trình tự và qui tắc xây dựng ERD, RDM, đặc biệt là các bước chuẩn hoá dữ liệu. Từ đó có thể vận dụng để vẽ ERD và RDM cho các hệ thống cụ thể.

Hướng dẫn bài tập lớn

Vận dụng kiến thức đã học để xây dựng biểu đồ quan hệ thực thể và biểu đồ dữ liệu quan hệ cho hệ thống đã chọn làm bài tập lớn. Viết tiếp báo cáo bài tập lớn với các nội dung cơ bản:

- Biểu đồ thực thể liên kết
- Biểu đồ quan hệ thực thể
- Từ điển dữ liệu (nếu cần)

PHẦN 3. PHƯƠNG PHÁP THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG

Nội dung

Phần này trình bày về phương pháp thiết kế và xây dựng hệ thống, bao gồm các nội dung:

CHƯƠNG 7. TỔNG QUAN VỀ THIẾT KẾ HỆ THỐNG

Mục tiêu

Chương này giới thiệu các hướng tiếp cận cơ bản trong thiết kế hệ thống, các công việc cần thực hiện trong giai đoạn này.

7.1.Các hướng tiếp cận thiết kế hệ thống

7.1.1. Các tiếp cận hướng mô hình

Thiết kế hướng mô hình (Model-driven) là một cách tiếp cận thiết kế hệ thống nhấn mạnh vào việc vẽ các mô hình hệ thống để tài liệu hóa các khía cạnh cài đặt và kỹ thuật của một hệ thống. Các mô hình thiết kế thường được dẫn xuất từ các mô hình lôgíc được phát triển trước đó theo cách phân tích hướng mô hình. Cuối cùng thì các mô hình thiết kế hệ thống sẽ trở thành các bản thiết kế phục vụ cho việc xây dựng và cài đặt hệ thống mới.

Trong tiếp cận hướng mô hình có 3 kỹ thuật là thiết kế hướng cấu trúc, kỹ thuật thông tin và thiết kế hướng đối tượng. Ngày nay, các tiếp cận hướng mô hình thường được củng cố nhờ vào việc sử dụng các công cụ tự động hóa. Các công cụ thường dùng:

- Công cụ đi kèm bộ công cụ lập trình: Oracle Designer
- Các công cụ đơn giản: MS.Word, MS.Visio, Smartdraw...
- Các công cụ chuyên dụng: Rational Rose, Rational XDE for platforms...
- **Thiết kế hướng cấu trúc hiện đại** (Modern Structured Design):

o Là kỹ thuật phân rã chức năng hệ thống ra thành nhiều phần, mỗi thành phần lại được thiết kế chi tiết hơn ở các bước sau. Thiết kế hướng cấu trúc còn được gọi là thiết kế chương trình từ tổng quan đến chi tiết (top- down).

o Mỗi modun ở mức thấp nhất chỉ thực hiện một phần việc nhất định, ít liên quan đến công việc của các modun khác.

o Thường được sử dụng vì đơn giản, dễ hiểu, thuận tiện trong triển khai và nâng cấp.

o Mô hình phần mềm được dẫn xuất từ thiết kế hướng cấu trúc được gọi là biểu đồ cấu trúc (structure chart). Biểu đồ này được xây dựng từ các luồng dữ liệu trong chương trình. Thiết kế hướng cấu trúc được thực hiện trong giai đoạn phân tích hệ

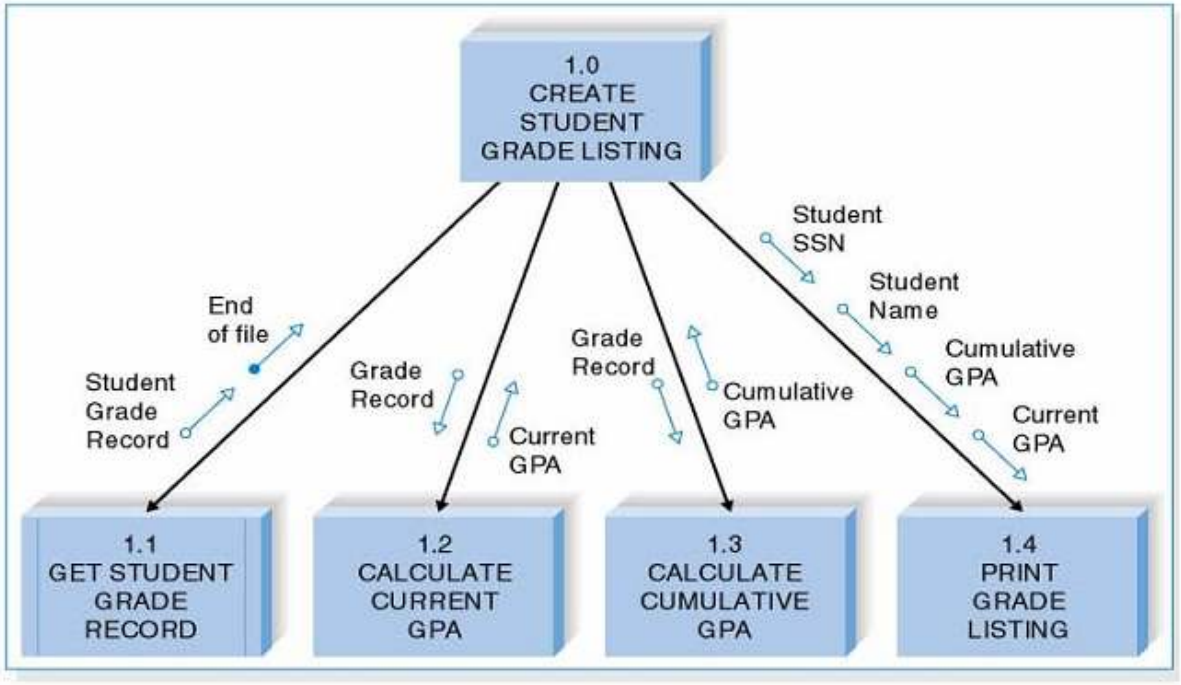
thống. Tuy nhiên, nó không bao trùm mọi khía cạnh của việc thiết kế, như thiết kế đầu vào/đầu ra hay cơ sở dữ liệu.

- Các ký hiệu trong biểu đồ cấu trúc:

o Môđun: được biểu diễn bằng hình chữ nhật có nhãn là tên của môđun.

o Dữ liệu được chuyển giao giữa các môđun: biểu diễn bởi mũi tên có đầu tròn rỗng.

o Thông tin điều khiển: biểu diễn bằng mũi tên với đầu tròn đặc.



Hình 7-1 Ví dụ biểu đồ cấu trúc

- **Kỹ thuật thông tin** (Information Engineering):
  - Là cách tiếp cận hướng mô hình và lấy dữ liệu làm trung tâm nhưng , chú trọng đến việc tổ chức các thông tin:
    - nội dung thông tin
    - quan hệ giữa các thành phần thông tin.
    - công cụ chủ yếu là sơ đồ mô hình dữ liệu
    - được sử dụng khi thiết kế chương trình dựa trên mô hình cơ sở dữ liệu quan hệ.
- **Làm bản mẫu (Prototyping)**
  - Bản mẫu là một chương trình nhỏ, chưa hoàn chỉnh nhưng đủ để cho người xem hình dung về chức năng, hoạt động của chương trình cần thực hiện:

- Làm bản mẫu là phương pháp mô hình hoá trên mã nguồn chứ không trên bản vẽ
- Thuận tiện để làm bản demo cho người dùng cuối xem (không đòi hỏi phải hiểu những ngôn ngữ mô hình hoá)
- Sớm phát hiện những sai khác về nghiệp vụ
- Chỉ phù hợp với các dự án nhỏ, ít phức tạp

- **Thiết kế hướng đối tượng (Object Oriented Design):**
  - Sử dụng cách tiếp cận theo tư duy hướng đối tượng - phân biệt rõ ràng hai yếu tố:
    - dữ liệu (thuộc tính)
    - chức năng (hành vi) và các mối tương tác (sự kiện)
  - Là bước tiếp theo của Phân tích hệ thống hướng đối tượng, do đó đòi hỏi những kết quả từ bước trước: định nghĩa đối tượng, thuộc tính, hành vi, sự kiện...

7.1.2. Phát triển ứng dụng nhanh

Kỹ thuật xây dựng ứng dụng nhanh chóng bằng cách phối hợp sử dụng nhiều kỹ thuật:

Tổ hợp thông tin

Làm bản mẫu

Kỹ thuật phát triển ứng dụng kết hợp (Joint Application Development): phát triển ứng dụng bằng cách gộp chung hai giai đoạn phân tích và thiết kế. Nhấn mạnh sự tham gia đồng thời của nhà phân tích, thiết kế, người dùng cuối, chuyên gia hệ thống trong quá trình xây dựng. Thường được dùng phát triển các ứng dụng nhỏ trong thời gian ngắn.

7.2. Các công việc cụ thể trong giai đoạn thiết kế

- Thiết kế kiến trúc ứng dụng

  Lựa chọn công nghệ sử dụng cho dự án, Đưa ra mô hình vật lý của hệ thống.

- Thiết kế cơ sở dữ liệu

  Đưa ra mô hình dữ liệu.

  Lựa chọn hệ quản trị CSDL và tối ưu hoá mô hình dữ liệu theo hệ quản trị đã lựa chọn

- Thiết kế giao diện hệ thống: đầu ra, đầu vào, giao diện người dùng, báo cáo...
- Đưa ra các đặc tả hệ thống cho lập trình viên

Câu hỏi thảo luận

7.1. Đặc điểm của thiết kế hướng cấu trúc hiện đại là gì?

7.2. Nêu ý nghĩa của việc làm bản mẫu.

7.3. Nêu các công việc cụ thể trong giai đoạn thiết kế.

Tổng kết chương 7

Chương 7 trình bày các hướng tiếp cận thiết kế hệ thống và các công việc cụ thể trong giai đoạn này.

Người học cần nắm vững:

- Các hướng tiếp cận hướng mô hình
- Hướng tiếp cận phát triển ứng dụng nhanh

CHƯƠNG 8. KIẾN TRÚC ỨNG DỤNG VÀ VIỆC MÔ HÌNH HOÁ

Mục tiêu

Chương này tập trung vào các khái niệm về kiến trúc ứng dụng và việc mô hình hoá kiến trúc hệ thống thông qua biểu đồ luồng dữ liệu vật lý.

8.1. Kiến trúc ứng dụng

Kiến trúc hệ thống thông tin (KTHT) là một đặc tả về mặt công nghệ của một hệ thống thông tin. KTHT dùng làm phương tiện để:

- Trao đổi về đặc tính của hệ thống (tập trung hay phân tán, CSDL, tính tích hợp, giao diện hệ thống...).
- Cơ sở để triển khai hệ thống theo thiết kế.
- Cơ sở để bảo trì hệ thống sau này.

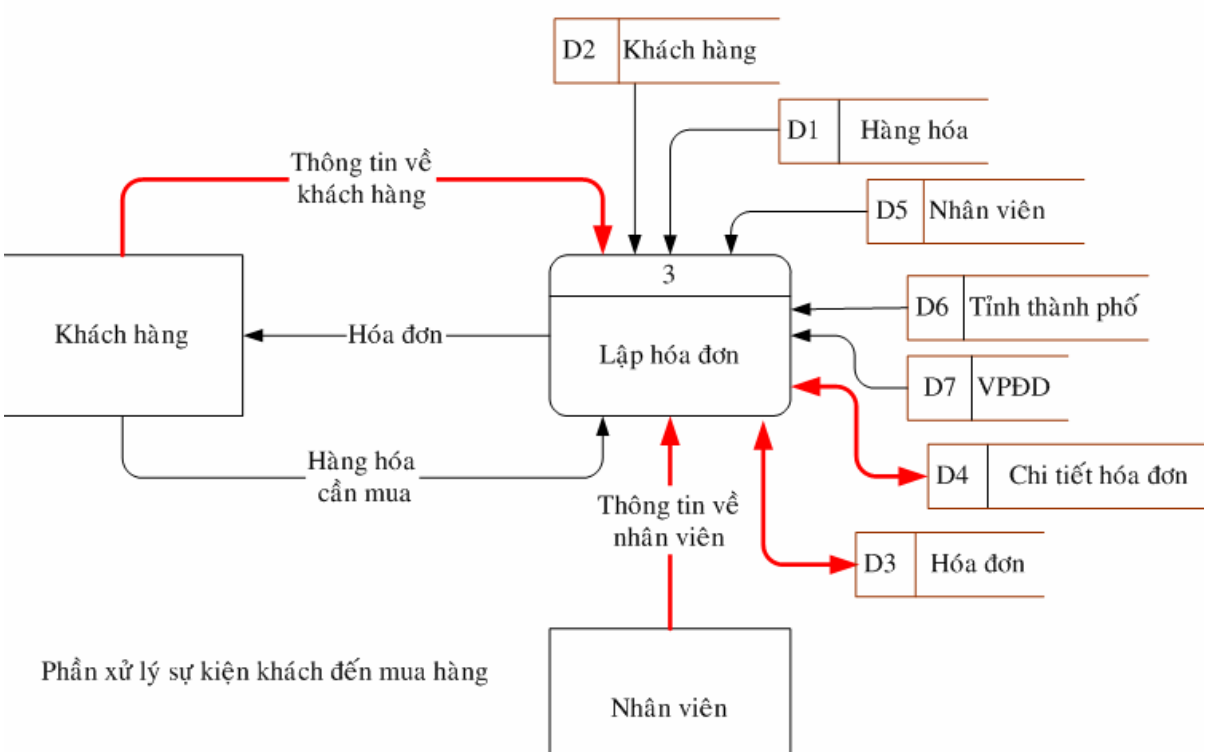
8.2. Biểu đồ luồng dữ liệu vật lý

Biểu đồ luồng dữ liệu vật lý (Physical Data Flow Diagram - PDFD) là mô hình chức năng dùng để mô hình hoá kiến trúc hệ thống. PDFD biểu diễn các thuộc tính của từng thành phần trong KTHT cũng như mô tả mối quan hệ, tương tác giữa các thành phần đó. Dưới đây sẽ giới thiệu cách PDFD diễn tả từng đối tượng trong KTHT.

8.2.1. Chức năng vật lý

- Chức năng vật lý là nơi thực hiện các công việc của hệ thống, đó có thể là con người, máy tính cá nhân, server, máy tính cầm tay...
- Mỗi hệ thống cần một hoặc một số chức năng vật lý
- PDFD giúp chúng ta thấy rõ: thông tin được xử lý bởi con người hay máy móc, thông tin được xử lý bởi công nghệ nào...
- Ký hiệu

- Ví dụ PDFD biểu diễn một chức năng vật lý



8.2.2. Luồng dữ liệu vật lý

- Luồng dữ liệu vật lý:
  - o Mô tả các luồng dữ liệu đi luân chuyển trong hệ thống
  - o Các lệnh tương tác với CSDL: tạo, đọc, cập nhật, xoá các đối tượng csdl
  - o Nhập/xuất các phần tử dữ liệu giữa các thành phần trong mạng

8.2.3. Kho dữ liệu vật lý

- Các kho dữ liệu vật lý dùng để mô tả
  - o Một cơ sở dữ liệu
  - o Một bảng trong cơ sở dữ liệu
  - o Một file máy tính
  - o File tạm
  - o Một phương tiện lưu trữ dự phòng
  - o Một dạng lưu trữ dữ liệu phi máy tính (mã vạch, RFID, thẻ từ...)

8.3. Kiến trúc công nghệ thông tin

Kiến trúc công nghệ thông tin (Information technology architecture) là một chủ đề phức tạp. Trong mục này, chúng tôi chỉ tóm tắt những xu thế công nghệ thông tin hiện đại có tác động tới các quyết định trong giai đoạn thiết kế.

8.3.1. Hệ thống phân tán

Hệ phân tán (Distributed system) là hệ thống trong đó các thành phần phân tán giữa những địa điểm, mạng, máy tính khác nhau: tính toán lưới (grid-computing, mạng máy tính dựa trên PC...). Đối lập với hệ phân tán là **hệ tập trung** (Centralized system) là hệ thống trong đó các thành phần, các tác vụ xử lý tập trung tại một nơi (Mainframe). Các hệ thống hiện đại là các hệ phân tán, nó giúp phân phối dữ liệu và các dịch vụ đến gần người dùng cuối hơn, cắt giảm sự phức tạp và chi phí đầu tư, bảo trì. Có 3 loại kiến trúc hệ thống phân tán:

- **Kiến trúc máy chủ tệp** (File server architecture)

Là một mạng cục bộ (LAN) trong đó có một máy chủ chứa dữ liệu của một hệ thống thông tin. Mạng LAN là mạng nội bộ kết nối các máy tính(PC, Server, PDA...) trong một phạm vi hẹp (văn phòng, toà nhà...). Mạng LAN giúp tổng hợp năng lực các máy tính đơn lẻ trong mạng khi cho phép bất kỳ máy nào cũng có thể là máy chủ, bất kể máy nào cũng có thể là máy khách.

Kiến trúc này cho phép nhiều máy tính cá nhân và máy chạm được kết nối để chia sẻ dữ liệu và giao tiếp với nhau.

- **Kiến trúc khách/chủ** (Client/Server architecture)

Là kiến trúc trong đó có một hay nhiều máy tính đóng vai trò máy chủ cung cấp các dịch vụ, dữ liệu cho một hay nhiều máy khách.

Máy chủ cơ sở dữ liệu (Database server): là máy chủ logic lưu trữ một hay nhiều cơ sở dữ liệu đồng thời cung cấp một hệ thống các câu lệnh cho phép thao tác với những cơ sở dữ liệu nói trên.

Máy chủ ứng dụng (Application server): là máy chủ logic lưu trữ phần xử lý logic của một hay nhiều ứng dụng, cho phép các máy khách truy nhập vào để thực thi ứng dụng.

Máy chủ nhắn tin hoặc phần mềm nhóm (Message hoặc Groupware server): là máy chủ logic cung cấp các dịch vụ như email, lịch làm việc, các chức năng hỗ trợ làm việc nhóm.

Máy chủ web (Webserver): là máy chủ logic lưu trữ và vận hành các website trên mạng internet hoặc intranet.

- **Kiến trúc tính toán dựa trên Internet (Internet-Based computing architecture)**

Là một dạng khác của kiến trúc phân tán đang góp phần định hình lại ý tưởng thiết kế của các nhà phân tích hệ thống và chuyên gia thông tin.

Một hệ thống tính toán mạng là hệ thống trong đó các hệ thống thông tin đều chạy trên trình duyệt (ví dụ như hệ thống tài chính, hệ thống quản lý nhân sự…), lấy dữ liệu từ máy chủ web.

8.3.2. Kiến trúc dữ liệu

- Cơ sở dữ liệu quan hệ lưu trữ dữ liệu dưới dạng bảng. Mỗi bảng bao gồm nhiều cột (giống các trường trong cơ sở dữ liệu dựa trên file), giao giữa các dòng và cột là các bản ghi (tương tự khái niệm bản ghi trên cơ sở dữ liệu file). Cơ sở dữ liệu quan hệ có một cơ sở toán học vững chắc và được dùng làm cơ sở dữ liệu của hầu hết các hệ thống hiện nay.
- Cơ sở dữ liệu quan hệ phân tán là cơ sở dữ liệu quan hệ trong đó một hay nhiều bảng được nhân rộng và phân tán trên nhiều máy chủ cơ sở dữ liệu ở các nơi khác nhau.
- Hệ quản trị cơ sở dữ liệu (HQTCSDL) là hệ thống quản lý việc lưu trữ, truy vấn, phân quyền truy nhập một hay nhiều cơ sở dữ liệu. HQTCSDL phân tán là một HQTCSDL làm thêm chức năng quản lý sự đồng bộ, kiểm soát truy nhập đối với các bảng dữ liệu phân tán. Có 2 kỹ thuật:

Data partitioning: phân mảnh và phân tán một hay nhiều trường dữ liệu giữa các server mà không có hoặc có rất ít sự trùng lặp.

Data Replication: không phân mảnh mà nhân bội một hay nhiều trường rồi phân tán giữa các server.

8.3.3. Kiến trúc giao diện

- Là các kênh giao tiếp giữa các trung tâm xử lý trong hệ thống hay giữa các hệ thống máy tính với nhau. Các kiểu giao tiếp: dữ liệu vào ra trực tuyến/ theo bó, nhập liệu không cần bàn phím (mã vạch, thẻ từ, RFID), nhập liệu bằng bút cảm ứng, dữ liệu EDI, dữ liệu có được thông qua nhận dạng (vân tay, scan...), thông qua middleware...

Câu hỏi thảo luận

8.1. Ý nghĩa của biểu đồ luồng dữ liệu vật lý.

Tổng kết chương 8

Chương 8 trình bày khái niệm về kiến trúc ứng dụng, biểu đồ luồng dữ liệu và kiến trúc công nghệ thông tin.

Người học cần nắm được các thành phần của một PDFD:

- Chức năng vật lý
- Luồng dữ liệu vật lý

- Kho dữ liệu vật lý

Người học nắm được các xu thế công nghệ hiện tại, có thể tác động tới quá trình thiết kế:

- Hệ thống phân tán
- Kiến trúc dữ liệu
- Kiến trúc giao diện

CHƯƠNG 9. THIẾT KẾ CƠ SỞ DỮ LIỆU

Mục tiêu

Chương 9 bao gồm các nội dung liên quan tới việc thiết kế cơ sở dữ liệu như các khái niệm về cơ sở dữ liệu và việc mô hình hoá dữ liệu.

9.1. Các phương thức lưu trữ dữ liệu

- Có hai phương thức lưu trữ dữ liệu phổ biến:
    - File
    - Cơ sở dữ liệu

9.1.1. File

Là một tập hợp của các bản ghi tương tự nhau. Các file không có liên quan với nhau trừ khi được liên kết trong code của chương trình ngoài

- Ưu điểm:
    - Dễ dàng thiết kế nếu chỉ dùng cho một ứng dụng
    - Tối ưu về hiệu năng nếu chỉ dùng cho một ứng dụng
- Nhược điểm:
    - Khó thích ứng hoặc khó dùng chung giữa nhiều ứng dụng
    - Hay bị dư thừa dữ liệu (cùng một thông tin lại được lưu trữ trên nhiều file khác nhau)

9.1.2. Cơ sở dữ liệu

Là một tập hợp của nhiều files (bảng) có quan hệ với nhau. Bản ghi của một file (hay bảng) có thể có mối quan hệ vật lý với một hay nhiều bản ghi ở các file (hay bảng) khác.

- Ưu điểm:
    - Tách biệt dữ liệu khỏi logic chương trình do đó tăng tính thích ứng, khả chuyển của chương trình.
    - Kiểm soát được quy mô, độ lớn của dữ liệu
    - Tối ưu trong việc chia sẻ dùng chung giữa nhiều ứng dụng
    - Giảm thiểu dư thừa dữ liệu
- Nhược điểm:
    - Phức tạp hơn công nghệ file rất nhiều
    - Ở khía cạnh nào đó truy xuất cơ sở dữ liệu thường chậm hơn so với truy xuất file

- Cần tuân thủ nhiều nguyên tắc khi thiết kế để có thể khai thác được lợi ích của cơ sở dữ liệu quan hệ
- Cần có chuyên gia sử dụng hệ quản trị cơ sở dữ liệu

9.2. Kiến trúc dữ liệu

- Kiến trúc dữ liệu mô tả cách thức:
  - sử dụng file/cơ sở dữ liệu để lưu trữ dữ liệu
  - công nghệ file/cơ sở dữ liệu được lựa chọn sử dụng
  - cơ cấu quản lý được thiết lập để quản lý các nguồn dữ liệu
- Thông thường dữ liệu được lưu trữ đồng thời bởi nhiều cách thức, phương tiện:
  - Các files,
  - Cơ sở dữ liệu cá nhân, cơ sở dữ liệu chung của nhóm, cơ sở dữ liệu giao dịch,
  - Nhà kho dữ liệu (tổng hợp các nguồn)...
- Hệ quản trị CSDL:
  - Là một phần mềm dùng để quản lý việc tạo, truy nhập, kiểm soát, quản lý các đối tượng dữ liệu của một hay nhiều cơ sở dữ liệu.
  - Phần nền tảng của một HQTCSDL là một bộ máy dữ liệu - data engine
  - Ngôn ngữ định nghĩa dữ liệu (Data Definition Language - DDL) là một phần của bộ máy dùng để định nghĩa các bảng, trường, quan hệ
  - Ngôn ngữ thao tác dữ liệu (Data Manipulation Language - DML) dùng để thêm, sửa, xoá và di chuyển giữa các trường trong cơ sở dữ liệu

9.3. Triển khai mô hình dữ liệu logic dựa trên một cơ sở dữ liệu quan hệ

9.3.1. Cơ sở dữ liệu quan hệ

Là cơ sở dữ liệu lưu trữ và quản lý dữ liệu trong những bảng 2 chiều. Các bảng này có thể có quan hệ với nhau thông qua các trường khoá

- Đặc thù của cơ sở dữ liệu quan hệ:
  - Mô hình dữ liệu vật lý (Schema)
  - DDL và DML được thể hiện bởi ngôn ngữ SQL
  - Triggers là các chương trình được nhúng cùng cơ sở dữ liệu và tự động thực thi khi cơ sở dữ liệu được cập nhật
  - Thủ tục thường trú (Stored procedure) là chương trình được nhúng cùng cơ sở dữ liệu và thực thi từ câu lệnh của ứng dụng

9.3.2. Mô hình hoá dữ liệu

- Một mô hình dữ liệu tốt là mô hình trong đó:
  - Mỗi thuộc tính mô tả một và chỉ một thực thể
  - Mỗi thuộc tính chỉ tồn tại ở duy nhất một thực thể (trừ thuộc tính khoá ngoại)
- Để có được một mô hình dữ liệu tốt, ta tiến hành các bước chuẩn hoá (xem thêm phần phân tích hệ thống)
- Chuẩn hoá dữ liệu - Một thực thể logic hay một bảng vật lý được gọi là:
  - Ở dạng chuẩn thứ nhất nếu không có thuộc tính (trường) nào có hai giá trị trong cùng một thể hiện
  - Ở dạng chuẩn thứ hai nếu nó đã ở dạng chuẩn thứ nhất và giá trị các trường không phải là khoá chính hoàn toàn phụ thuộc vào khoá chính.
  - Ở dạng chuẩn thứ ba nếu nó đã ở dạng chuẩn thứ hai và giá trị các trường không phải khoá chính không phụ thuộc các trường không phải khoá chính khác
- Các bước tạo mô hình dữ liệu vật lý
  - Xem lại mô hình dữ liệu logic
  - Tạo bảng cho mỗi thực thể
  - Tạo trường cho mỗi thuộc tính
  - Tạo chỉ mục (index) cho mỗi khoá
  - Thiết kế khoá ngoại cho các quan hệ
  - Định nghĩa kích thước/kiểu dữ liệu, thuộc tính null, giá trị mặc định
  - Đánh giá và thiết lập các ràng buộc
  - Chú ý các công nghệ khác nhau cho các kiểu dữ liệu khác nhau
- Tạo câu lệnh SQL
  - Tuỳ vào từng hệ quản trị CSDL, tạo CSDL tương ứng trên ngôn ngữ DDL, ví dụ: CREATE TABLE(makh int, hoten varchar(25))…

Câu hỏi thảo luận

9.1. Nêu ưu nhược điểm của phương thức lưu trữ dữ liệu dạng file.

9.2. Nêu ưu nhược điểm của phương thức lưu trữ dữ liệu sử dụng cơ sở dữ liệu.

Tổng kết chương 9

Chương 9 đề cập tới các phương thức lưu trữ dữ liệu, kiến trúc dữ liệu.

Người học cần phân biệt được 2 phương thức lưu trữ dữ liệu phổ biến là:

- File
- Cơ sở dữ liệu

Người học cần nắm được các bước xây dựng mô hình dữ liệu vật lý:

- Xem lại mô hình dữ liệu logic
- Tạo bảng cho mỗi thực thể
- Tạo trường cho mỗi thuộc tính
- Tạo chỉ mục (index) cho mỗi khoá
- Thiết kế khoá ngoại cho các quan hệ
- Định nghĩa kích thước/kiểu dữ liệu, thuộc tính null, giá trị mặc định
- Đánh giá và thiết lập các ràng buộc

Hướng dẫn bài tập lớn

Với kiến thức đã học, xây dựng cơ sở dữ liệu cho hệ thống đã chọn làm bài tập lớn. Tiếp tục viết báo cáo bài tập lớn với các nội dụng cơ bản:

- Cấu trúc các bảng dữ liệu (tên trường, kiểu dữ liệu...)
- Mô hình quan hệ giữa các bảng dữ liệu

CHƯƠNG 10. THIẾT KẾ ĐẦU VÀO

Mục tiêu

Chương này trình bày cac phương thức nhập liệu, các nguyên tắc thiết kế đầu vào cho một hệ thống và các điều khiển có thể sử dụng trong quá trình thiết kế đầu vào.

10.1. Tổng quan về thiết kế đầu vào

10.1.1. Các khái niệm

Có thể phân loại các phương thức nhập liệu theo hai đặc trưng: (1) cách thức dữ liệu được thu thập, đưa vào và xử lý và (2) phương pháp và công nghệ được dùng để thu thập và nhập dữ liệu.

- Thu thập dữ liệu (data capture): nhận dạng và tạo dữ liệu mới từ nguồn tạo tin
- Nhập liệu (data entry): chuyển dữ liệu từ nguồn tạo tin vào máy tính
- Xử lý dữ liệu (data processing): là quá trình biến đổi trực tiếp trên dữ liệu trước khi đưa nó về dạng máy tính có thể đọc được. Xử lý bó là thu thập 1 khối lượng dữ liệu và xử lý đồng thời cả bó. Xử lý trực tuyến là xử lý ngay lập tức dữ liệu vừa thu thập được.

10.1.2. Các phương thức nhập liệu

- Bàn phím
- Chuột
- Màn hình cảm ứng (màn hình tương tác)
- Nhận dạng âm thanh, tiếng nói
- Tự động nhập liệu: mã vạch, nhận dạng quang học, mực từ, thẻ từ, thẻ thông minh, sinh trắc học...

10.1.3. Các nguyên tắc thiết kế đầu vào

Nên tuân theo những nguyên tắc dưới đây khi thiết kế phương thức nhập liệu:

- Không nên nhập những dữ liệu có thể tính toán được từ những dữ liệu khác. Ví dụ: Số lượng x Đơn giá = Thành tiền
- Không nhập những dữ liệu có thể lưu trong máy tính như những hằng số.
- Sử dụng mã lấy từ cơ sở dữ liệu đối với những thuộc tính phù hợp.
- Sử dụng các chỉ dẫn nhập liệu khi thiết kế các form nhập liệu (tooltip).
- Giảm thiểu số lượng ký tự gõ vào để tránh gây sai sót. Thay vào đó, cố gắng dùng các hộp check chọn càng nhiều càng tốt.
- Dữ liệu nhập vào theo trình tự từ trên xuống dưới, trái qua phải.

10.1.4. Kiểm soát nhập liệu

Việc kiểm soát dữ liệu đầu vào rất cần thiết trong tất cả các hệ thống ứmg dụng trên máy tính. Các điều khiển đầu vào đảm bảo rằng dữ liệu đầu vào là chính xác và hệ thống được bảo vệ khỏi các lỗi vô ý hoặc hữu ý.

- Số lượng đầu vào cần phải được theo dõi, đặc biệt là trong trường hợp nhập dữ liệu theo bó:
    - Lưu mã số giao dịch cho bó các dữ liệu nhập liệu theo bó.
    - Ghi các log file cho các dữ liệu được nhập trực tuyến

Phải kiểm soát tính đúng đắn của dữ liệu nhập vào. Phải làm các kiểm tra về: trùng lặp thực thể, kiểu dữ liệu, định dạng, tính ràng buộc với các dữ liệu khác. Ví dụ: Khi nhập liệu thành phố và quốc gia cho một hồ sơ nhân sự, nếu đã chọn quốc gia là Việt Nam thì chỉ cho phép chọn thành phố là Hà Nội, Huế hoặc các thành phố khác ở Việt Nam ... chứ không cho phép chọn thành phố thuộc quốc gia khác như Tokyo chẳng hạn.

10.2. Các điều khiển giao diện cho thiết kế đầu vào

10.2.1. Một số điều khiển phổ biến

- Hộp văn bản (Text box): chứa một hộp hình chữ nhật kèm theo tên, cho phép nhập dữ liệu vào.
- Nút chọn loại trừ (Radio button): chứa một hình trong nhỏ kèm theo một đoạn văn bản mô tả tương ứng với giá trị lựa chọn. Trong một nhóm các nút này thì chỉ cho phép chọn một nút mà thôi.
- Hộp chọn kiểm tra (Check box): chứa một hộp hình vuông kèm theo đoạn văn bản mô tả trường dữ liệu vào, người dùng sẽ chọn giá trị Yes/No. Trong một nhóm các hộp chọn thì có thể chọn nhiều hộp.
- Hộp danh sách (List box): là một hình chữ nhật chứa một hoặc nhiều dòng dữ liệu.
- Danh sách thả (Drop down list): chứa hộp chọn hình chữ nhật và một nút bên cạnh. Khi nhấn vào nút đó thì danh sách sẽ được thả xuống.
- Hộp thả kết hợp (Combination box): cũng là một danh sách thả nhưng cho phép người dùng nhập thêm dữ liệu ngoài những dữ liệu có sẵn trong đó.
- Nút lệnh (Button): các nút lệnh không phải là điều khiển vào. Chúng không dành cho việc lựa chọn dữ liệu vào. Mục đích của chúng là cho phép người dùng xác nhận rằng tất cả các dữ liệu cần được xử lý hay hủy bỏ một giao dịch

hoặc cần gọi chức năng trợ giúp… Tóm lại, nút lệnh đóng vai trò gọi tới một chức năng nào đó.

10.2.2. Một số điều khiển cao cấp

- Hộp lịch thả (Drop down calendar): là một ô dữ liệu có chứa một nút mũi tên. Khi nhấn chuột vào đó thì hộp lịch được thả xuống để chọn ngày.
- Điều khiển hiệu chỉnh trượt (Slider edit control): cho phép lựa chọn giá trị bằng cách trượt con trỏ.
- Điều khiển hiệu chỉnh mặt nạ (Masked edit control): điều khiển này tạo ra định dạng để buộc dữ liệu nhập vào phải tuân theo.
- Hộp danh sách chọn (Check list box): điều khiển này được dùng để kết hợp nhiều hộp chọn kiểm tra.
- Hộp cây chọn (Check tree box): điều khiển này được dùng để biểu diễn các lựa chọn dữ liệu dưới dạng cây phân cấp.

10.3. Quy trình thiết kế đầu vào

- Xác định các dữ liệu đầu vào của hệ thống và các yêu cầu nhập liệu logic.
- Lựa chọn các điều khiển thích hợp.
- Thiết kế, lập cơ chế kiểm soát nhập liệu, lưu vết
- Nếu cần thiết, lập hồ sơ đặc tả đầu vào.

Câu hỏi thảo luận

10.1. Nêu tóm tắt các phương thức nhập liệu.

10.2. Nêu các nguyên tắc thiết kế nhập liệu.

Tổng kết chương 10

Chương 10 trình bày các phương thức thiết kế đầu vào, các nguyên tắc thiết kế và quy trình thiết kế đầu vào.

Người học cần nắm rõ các phương thức thiết kế đầu vào:

- Bàn phím
- Chuột
- Màn hình cảm ứng
- Âm thanh
- Tiếng nói
- Quang học

- Mực từ
- Điện từ
- Thẻ thông minh
- Sinh trắc học

Người học cần phân biệt được các điều khiển giao diện đầu vào thường được sử dụng.

CHƯƠNG 11. THIẾT KẾ ĐẦU RA

Mục tiêu

Chương 11 giới thiệu các phương thức thiết kế đầu ra, các nguyên tắc và quy trình thiết kế đầu ra cho một hệ thống.

11.1. Tổng quan về thiết kế đầu ra

11.1.1. Phân loại đầu ra

Một cách để phân loại đầu ra là dựa vào hình thức phân phối chúng trong hay ngoài tổ chức và đối tượng người sẽ đọc và sử dụng chúng. Hình thức đầu ra chủ yếu là dưới dạng các báo cáo.

- Báo cáo nội bộ: là các báo cáo được cung cấp cho người dùng hệ thống trong tổ chức
- Báo cáo chi tiết: thông tin trực tiếp truy xuất từ dữ liệu hệ thống, ví dụ: danh sách khách hàng.
- Báo cáo tóm lược: thông tin sau khi truy xuất đã được sắp xếp theo thứ tự thuận tiện cho người dùng quan sát, đôi khi kết quả được thể hiện dưới dạng đồ hoạ, ví dụ: khách hàng theo khu vực
- Báo cáo ngoại lệ: thông tin cảnh báo, đột xuất theo sự kiện thay đổi về chất lượng, điều kiện của hệ thống.
- Báo cáo bên ngoài là các báo cáo cung cấp cho khách hàng, nhà cung cấp, cơ qua pháp luật...
- Báo cáo quay vòng là các loại báo cáo bên ngoài sau đó lại trở về hệ thống như là một phương thức thu thập dữ liệu, chẳng hạn bản điều tra, hoá đơn...

11.1.2. Các phương thức cài đặt đầu ra

- In ra trên giấy
- Hiển thị trên màn hình, trên trang web
- Xuất dưới dạng đa phương tiện
- Gửi thư trực tiếp
- Tạo các đường liên kết
- ...

11.2. Cách thức thiết kế đầu ra

11.2.1. Các nguyên tắc thiết kế đầu ra

- Báo cáo phải đơn giản, dễ hiểu, dễ giải thích:
    - Bao gồm tiêu đề

    - Ghi rõ ngày giờ phát hành
    - Có các phần ghi thông tin chung
- Thông tin phải được thể hiện ở dạng người dùng bình thường không được tuỳ ý sửa chữa.
- Thông tin hiển thị phải hài hoà giữa các trang
- Cung cấp cách di chuyển giữa các ô thật sự đơn giản
- Thời gian xuất báo cáo phải được kiểm soát
- Một số hình thức báo cáo phải được sự đồng ý của công ty

11.2.2. Quy trình thiết kế đầu ra

- Xem xét tất cả các đầu ra của hệ thống và các yêu cầu logic.
- Làm rõ đặc tả (vật lý) của các yêu cầu báo cáo.
- Thiết kế các bản mẫu trước (nếu cần).
- Thiết kế, kiểm thử và kiểm tra đầu ra của báo cáo

Câu hỏi thảo luận

11.1. Nêu các phương thức cài đặt đầu ra.

11.2. Nêu các nguyên tắc thiết kế đầu ra.

Tổng kết chương 11

Chương 11 đề cập tới các phương thức cài đặt đầu ra, các nguyên tắc và quy trình thiết kế đầu ra.

Người học cần nắm rõ các loại đầu ra:

- Báo cáo nội bộ
- Báo cáo bên ngoài
- Báo cáo quay vòng

Người học có thể phân biệt các phương thức cài đặt đầu ra:

- Máy in
- Màn hình
- Đa phương tiện
- Thư điện tử
- Siêu liên kết
- Tấm vi phim

Người học phải nắm vững và có thể vận dụng các nguyên tắc và quy trình thiết kế đầu ra.

CHƯƠNG 12. THIẾT KẾ GIAO DIỆN NGƯỜI DÙNG

Mục tiêu

Chương này trình bày tổng quan về thiết kế giao diện, các kỹ thuật sử dụng trong thiết kế và các phong cách thiết kế giao diện phổ biến.

12.1. Tổng quan về giao diện người dùng

- Giao diện người dùng hiệu quả phải phù hợp với trình độ và kinh nghiệm của người dùng. Những nguyên nhân sau đây khiến cho người dùng sử dụng sai hay cảm thấy nhàm chán, lẫn lộn thậm chí hoảng sợ quay sang chối bỏ phần mềm:
    - Sử dụng nhầm lẫn các thuật ngữ, khái niệm
    - Giao diện không trực quan
    - Cách tiếp cận giải quyết vấn đề bị lẫn lộn
    - Thiết kế giao diện rắc rối
- Các nguyên tắc nên áp dụng khi thiết kế giao diện người dùng:
    - Phải hiểu rõ trình độ người sử dụng cũng như đặc thù các công việc của họ
    - Lôi kéo người dùng vào việc thiết kế giao diện
    - Kiểm tra và thử nghiệm việc thiết kế trên người dùng thật
    - Áp dụng các quy ước, thói quen trong thiết kế giao diện, tuân thủ style chung cho toàn chương trình.
    - Người dùng cần được chỉ dẫn những công việc họ sẽ đối mặt tiếp theo:
- Chỉ cho người dùng hệ thống đang mong đợi họ làm gì
- Chỉ cho người dùng dữ liệu họ nhập đúng hay sai
- Giải thích cho người dùng hệ thống đang đứng yên do có công việc cần xử lý chứ không treo
- Khẳng định với người dùng hệ thống đã hay chưa hoàn thành một công việc nào đó
    - Nên định hình giao diện sao cho các thông điệp, chỉ dẫn luôn xuất hiện tại cùng một vị trí
    - Định hình các thông điệp và chỉ dẫn đủ dài để người dùng có thể đọc được, đủ ngắn để họ có thể hiểu được
    - Các giá trị mặc định cần được hiển thị
    - Lường trước những sai sót người dùng có thể gặp phải để phòng tránh

- Không cho phép xử lý tiếp nếu lỗi chưa được khắc phục

12.2. Kỹ thuật giao diện người dùng

12.2.1. Hệ điều hành và trình duyệt web

Những hệ điều hành đồ họa phổ biến cho các máy khách hiện nay là Windows, Macintosh, Unix, Linux và cho các máy cầm tay là Palm OS, Windows CE. Tuy nhiên, hệ điều hành ngày càng không còn là nhân tố chính trong thiết kế giao diện người dùng nữa. Các ứng dụng Internet và Intranet chạy trên các trình duyệt web. Hầu hết các trình duyệt có thể chạy trên nhiều hệ điều hành. Điều này cho phép thiết kế giao diện người dùng ít phụ thuộc vào hệ điều hành. Tính năng này được gọi là độc lập nền tảng (platform independence). Thay vì viết giao diện riêng cho từng hệ điều hành thì chỉ cần viết giao diện cho một hoặc hai trình duyệt. Hiện tại, hai trình duyệt phổ biến nhất là Microsoft Internet Explorer và Netscape Navigator nhưng vẫn còn tồn tại một khó khăn khác đó là vấn đề về các phiên bản trình duyệt.

12.2.2. Màn hình hiển thị

Kích thước vùng hiển thị là vấn đề then chốt khi thiết kế giao diện. Không phải màn hình hiển thị nào cũng là dạng màn hình máy tính cá nhân. Có rất nhiều thiết bị hiển thị không phải là máy tính cá nhân.

Đối với màn hình máy tính cá nhân, chúng ta có đơn vị đo lường là độ phân giải đồ họa. Độ phân giải đồ họa được tính theo pixel, đó là số điểm sáng phân biệt được hiển thị trên màn hình. Hiện nay, độ phân giải phổ biến là 800.000 pixel theo chiều ngang và 600.000 pixel theo chiều dọc trong một màn hình 17 inch. Những kích thước hiển thị lớn hơn hỗ trợ nhiều pixel hơn; tuy nhiên, người thiết kế nên thiết kế giao diện theo loại màn hình có độ phân giải phổ biến nhất.

Rõ ràng, các máy tính cầm tay và một số thiết bị hiển thị đặc biệt (ví dụ như màn hình máy rút tiền tự động ATM) hỗ trợ màn hình hiển thị nhỏ hơn nhiều cũng phải được xem xét khi thiết kế giao diện.

Cách thức thể hiện vùng hiển thị đối với người dùng được điều khiển bởi cả khả năng kỹ thuật của màn hình và khả năng của hệ điều hành, Hai cách tiếp cận phổ biến nhất là paging và scrolling. **Paging** hiển thị một màn hình hoàn chỉnh các ký tự vào cùng một lần. Toàn bộ vùng hiển thị được gọi là một trang (hay màn hình). Các trang được hiển thị theo nhu cầu của người dùng bằng cách nhấn nút lệnh, tương tự như lật các trang trong một cuốn sách. **Scrolling** dịch chuyển phần thông tin hiển thị lên hoặc xuống trên màn hình, thường là mỗi lần 1 dòng. Các màn hình máy tính cá nhân còn cho phép nhiều tùy chọn paging và scrolling.

12.2.3. Bàn phím và các thiết bị trỏ

Hầu hết (nhưng không phải tất cả) các thiết bị hiển thị và màn hình đều được tích hợp với bàn phím. Những tính năng chủ yếu của bàn phím là tập ký tự và các khóa chức năng.

Tập ký tự của hầu hết các máy tính cá nhân đều theo chuẩn. Những tập ký tự đó có thể được mở rộng với phần mềm để hỗ trợ thêm các ký tự và biểu tượng. Các khóa chức năng nên được sử dụng một cách nhất quán. Nghĩa là, bất kỳ chương trình nào cũng nên sử dụng nhất quán các khóa chức năng cho cùng mục đích. Ví dụ, F1 thường được dùng để gọi chức năng trợ giúp trong cả hệ điều hành và các ứng dụng.

Hầu hết các giao diện (bao gồm các hệ điều hành và trình duyệt) đều sử dụng thiết bị trỏ như chuột, bút và màn hình cảm ứng. Tất nhiên, thiết bị trỏ phổ biến nhất vẫn là chuột.

Bút đang trở nên quan trọng trong các ứng dụng chạy trên các thiết bị cầm tay. Bởi lý do là những thiết bị đó thường không có bàn phím. Do đó, giao diện có thể cần được thiết kế để cho phép "gõ" trên một bàn phím được hiển thị trên màn hình hoặc sử dụng một chuẩn viết tay như Graffiti hoặc Jot.

12.3. Các phong cách thiết kế giao diện người dùng

12.3.1. Giao diện dựa trên cửa sổ và frame

Phần cơ bản nhất của một giao diện là cửa sổ. Một cửa sổ có thể nhỏ hoặc lớn hơn vùng màn hình hiển thị. Nó thường chứa các điều khiển chuẩn ở góc trên bên phải như phóng to, thu nhỏ hay đóng cửa sổ.

Phần dữ liệu hiển thị bên trong cửa sổ có thể lớn hoặc nhỏ hơn kích thước cửa sổ. Trong trường hợp lớn hơn, có thể dùng thanh cuộn để dịch chuyển.

Một cửa sổ có thể được chia thành các vùng gọi là frame. Mỗi frame có thể hoạt động độc lập với các các frame khác trong cùng một cửa sổ. Mỗi frame có thể được xác định để phục vụ cho một mục đích nhất định.

Trong một cửa sổ, chúng ta có thể sử dụng tất cả các điều khiển giao diện đã giới thiệu trong các chương 9 và 10.

12.3.2. Giao diện dựa trên menu

Chiến lược đối thoại phổ biến nhất và cổ điển nhất là menu. Có nhiều loại menu nhưng tư tưởng chung đều là yêu cầu người dùng chọn một hành động từ menu:

- o Menu kéo thả, menu xếp tầng
- o Menu pop-up
- o Thanh công cụ và menu icon

o Menu siêu liên kết

12.3.3. Giao diện dựa trên dòng lệnh

Thay cho menu hoặc cũng có thể bổ sung thêm cho menu, một số ứng dụng được thiết kế sử dụng đối thoại dựa trên tệp lệnh (còn gọi là giao diện ngôn ngữ lệnh – command language interface). Tuy nhiên, người sử dụng phải học cú pháp tập lập nên cách tiếp cận này chỉ phù hợp với đối tượng người dùng chuyên gia. Có 3 loại cú pháp, lựa chọn loại nào là phụ thuộc vào công nghệ có thể dùng:

- Cú pháp dựa trên ngôn ngữ (ví dụ như SQL)
- Cú pháp mnemonic: người sử dụng được cung cấp một màn hình giao tiếp trong đó họ có thể nhập các lệnh gọi tới các hành động. Các câu lệnh phải có nghĩa với người sử dụng.
- Cú pháp ngôn ngữ tự nhiên: cho phép người dùng nhập các câu hỏi các lệnh bằng ngôn ngữ tự nhiên. Hệ thống thông dịch các lệnh đó theo cú pháp đã biết và có thể yêu cầu người dùng nhập lại rõ ràng hơn nếu nó không hiểu được ý muốn của người dùng.

Cách thiết kế giao diện dựa trên dòng lệnh từng phổ biến trong các ứng dụng máy tính lớn và các ứng dụng máy tính cá nhân trên DOS trước đây. Nhưng phong cách tương tác này vẫn được sử dụng trong một số ứng dụng hiện nay. Ví dụ như Microsoft Access có phần soạn thảo câu truy vấn như hình dưới đây:

Thiết kế giao diện người dùng là việc đặc tả sự đối thoại giữa người sử dụng chương trình và máy tính. Sự đối thoại này thường cho kết quả là dữ liệu đầu vào và thông tin đầu ra. Có một số phong cách thiết kế giao diện người dùng. Trước đây, những hình thức đó được xem là loại bỏ nhau nhưng ngày nay, chúng đang pha trộn lẫn nhau. Mục này giới thiệu tổng quan một số phong cách và chiến lược được dùng để thiết kế giao diện người dùng và cách thức chúng được kết hợp vào các ứng dụng.

12.3.4. Đối thoại hỏi – đáp

Hình thức đối thoại hỏi đáp được dùng chủ yếu để hỗ trợ cho đối thoại dựa trên menu hoặc dựa trên câu lệnh. Người dùng được gợi ý bằng câu hỏi mà họ cần cho câu trả lời. Câu hỏi đơn giản nhất là Yes/No. Chiến lược này yêu cầu chúng ta phải xét mọi câu trả lời đúng có thể có và chuẩn bị mọi hành động nếu xuất hiện câu trả lời sai. Rõ ràng đây là một hình thức giao diện khó thiết kế. Tuy nhiên, hình thức này phổ biến trong các ứng dụng trên web.

12.3.5. Một số tính năng đặc biệt

- Xác thực và phân quyền

Trong hầu hết các hệ thống, người sử dụng phải được xác thực trước khi họ được phép sử dụng hệ thống. Nói một cách khác, người sử dụng phải đăng nhập vào hệ

thống. Hầu hết việc đăng nhập đều yêu cầu tên người dùng (username) và mật khẩu (password). Mỗi người dùng được cấp một quyền hạn sử dụng một số chức năng nhất định. Người dùng có quyền cao nhất thường là người quản trị hệ thống.

- Trợ giúp

o Tooltip: xuất hiện khi người dùng đưa chuột vào vị trí biểu tượng (hoặc đối tượng) trên màn hình. Tooltip chứa một đoạn mô tả ngắn gọn về chức năng thể hiện bởi đối tượng tương ứng.

o Help Wizard: hướng dẫn người sử dụng thông qua một quá trình phức tạp bằng cách đưa ra một chuỗi các hộp đối thoại yêu cầu người dùng phải đưa ra đầu và và trả về phản hồi của hệ thống.

o Tác tử (Agents): là các đối tượng phần mềm có thể hoạt động trên nhiều ứng dụng phần mềm và thậm chí là trên các mạng. Ví dụ như tác tử trợ giúp của Microsoft (có thể hiểu là trợ lý) cung cấp một trợ lý chung trong các ứng dụng Office. Nó cho phép người dùng đặt câu hỏi bằng một đoạn ngôn ngữ tự nhiên. Đoạn ngôn ngữ sẽ được thông dịch bởi tác tử để đưa ra đáp ứng phù hợp. Sau đó, người dùng có thể lựa chọn một trong các đáp ứng để chọn ra chỉ mục trợ giúp chi tiết hơn nữa.

12.4. Cách thức thiết kế giao diện người dùng

12.4.1. Các công cụ tạo giao diện

- Microsoft Access
- CASE Tools
- Visual Basic
- Visio
- …

12.4.2. Quy trình thiết kế giao diện người dùng

o Bước 1 - Lập sơ đồ phân cấp giao tiếp người dùng hoặc sử dụng lược đồ biến đổi trạng thái

o Bước 2 - Lập bản mẫu đối thoại và giao diện người dùng

o Bước 3 - Tham khảo và tiếp thu ý kiến phản hồi của người dùng. Nếu cần thiết quay trở lại bước 1 và bước 2.

Câu hỏi thảo luận

12.1. Nêu các nguyên tắc tổng thiết kế giao diện.

12.2. Nêu các kỹ thuật giao diện.

12.3. So sánh các phong cách giao diện.

Tổng kết chương 12

Chương 12 trình bày về kỹ thuật giao diện người dùng, các phong cách và cách thức thiết kế giao diện người dùng.

Người học cần nắm rõ các kỹ thuật giao diện người dùng:

- Hệ điều hành và trình duyệt web
- Màn hình hiển thị
- Bàn phím và các thiết bị trỏ

Người học phải phân biệt được các phong cách thiết kế giao diện:

- Giao diện dựa trên cửa sổ và frame
- Giao diện dựa trên menu
- Giao diện dựa trên dòng lệnh

Hướng dẫn bài tập lớn

Thiết kế đầu ra cho hệ thống mà bạn đang thực hiện phân tích thiết kế, viết tiếp báo cáo bài tập lớn với các nội dung:

- Các phương thức cài đặt đầu vào
- Các hình thức đầu ra
- Các biểu mẫu giao diện trong chương trình (có hình ảnh)

CHƯƠNG 13. XÂY DỰNG VÀ TRIỂN KHAI HỆ THỐNG

Mục tiêu

Chương này tập trung vào việc xây dựng và phát triển hệ thống.

- **Xây dựng hệ thống (**Systems construction**)** là việc phát triển, cài đặt và kiểm thử các thành phần của hệ thống. Xây dựng hệ thống thường được hiểu và gọi bằng thuật ngữ không phù hợp là *phát triển hệ thống* (cụm từ này thường được dùng để nói tới toàn bộ vòng đời hệ thống).
- **Triển khai hệ thống (**Systems implementation**)** là việc cài đặt và đưa toàn bộ hệ thống vào hoạt động.

13.1. Giai đoạn xây dựng

Mục tiêu của giai đoạn xây dựng là phát triển và kiểm thử một hệ thống chức năng đáp ứng các yêu cầu nghiệp vụ và thiết kế, đồng thời cài đặt giao diện giữa hệ thống mới và hệ thống đã có. Trong mục này, chúng tôi giới thiệu các bước trong giai đoạn xây dựng của một dự án phát triển hệ thống thông thường.

13.1.1. Bước 1 – Xây dựng và kiểm thử mạng (nếu cần thiết)

Thông thường các hệ thống được xây dựng trên những mạng sẵn có. Trong trường hợp, hệ thống đòi hỏi hệ thống mạng mới thì cần phải xây dựng và kiểm thử mạng mới sao cho phù hợp với chương trình sẽ sử dụng nó.

- Phân bổ nhiệm vụ:

o Người thiết kế mạng:

- Thiết kế kết nối LAN và WAN

o Người quản trị mảng xây dựng và kiểm thử:

- Các chuẩn kiến trúc mạng
- Bảo mật

o Người phân tích hệ thống

- Đóng vai trò diễn giải, làm đơn giản hoá các yêu cầu của hệ thống
- Đảm bảo rằng các yêu cầu nghiệp vụ không gây tác động xấu

13.1.2. Bước 2 – Xây dựng và kiểm thử cơ sở dữ liệu

- Cài đặt lược đồ cơ sở dữ liệu
- Kiểm thử với dữ liệu mẫu
- Đưa ra kết quả là cấu trúc dữ liệu rỗng
- Phân bổ nhiệm vụ:

o Người sử dụng hệ thống:

- Cung cấp và/hoặc phê chuẩn dữ liệu kiểm thử

o Người thiết kế cơ sở dữ liệu và người lập trình

- Xây dựng các bảng, views, thủ tục thường trú (nếu là cơ sở dữ liệu quan hệ)

o Người quản trị cơ sở dữ liệu

- "Điều chỉnh" cơ sở dữ liệu để đạt hiệu suất tối ưu
- Bảo mật
- Sao lưu và phục hồi

o Người phân tích hệ thống

- Xây dựng cơ sở dữ liệu hướng ứng dụng và không kết hợp
- Đảm bảo việc tuân theo các yêu cầu nghiệp vụ

13.1.3. Bước 3 - Cài đặt và kiểm thử gói phần mềm mới (nếu cần)

- Nếu hệ thống đòi hỏi phải có những phần mềm được thuê hoặc mua về thì chúng cần phải được cài đặt và kiểm thử
- Phân bổ nhiệm vụ:

o Người phân tích hệ thống

- Làm sáng sủa các yêu cầu nghiệp vụ

o Người thiết kế hệ thống

- Làm sáng sủa các yêu cầu tích hợp

o Người quản trị mạng

- Cài đặt gói phần mềm

o Người bán phần mềm hoặc người tư vấn

- Trợ giúp cài đặt và kiểm thử

o Người lập trình ứng dụng

- Kiểm thử dựa theo các yêu cầu tích hợp

13.1.4. Bước 4 – Viết và kiểm thử các chương trình mới

- Tái sử dụng các thành phần phần mềm có thể có trong thư viện
- Viết các thành phần mới
- Kiểm thử
- Tài liệu hoá
- Phân bố nhiệm vụ:

o Người phân tích hệ thống

- Làm sáng sủa các yêu cầu nghiệp vụ

o Người thiết kế hệ thống

- Làm sáng sủa thiết kế chương trình và các yêu cầu tích hợp

o Đội lập trình ứng dụng

- Viết và kiểm thử phần mềm
- Các mức độ kiểm thử:

o **Kiểm thử một phần** (Stub test) – việc kiểm thử được thực hiện đối với một số môđun của chương trình. Nói một cách khác, đó là cách kiểm thử một tập con độc lập của chương trình.

o **Kiểm thử chương trình** (Unit or program test) – việc kiểm thử được thực hiện đối với toàn bộ một chương trình. Mức độ kiểm thử này được tiến hành khi các mô đun đã được lập trình và việc kiểm thử một phần đã hoàn thành.

o **Kiểm thử hệ thống** (Systems test) – việc kiểm thử được thực hiện đối với toàn bộ hệ thống nhằm đảm bảo rằng các chương trình ứng dụng được viết và kiểm thử độc lập đều hoạt động tốt khi chúng được tích hợp vào hệ thống tổng hợp.

13.2. Giai đoạn triển khai

Hệ thống chức năng thu được từ giai đoạn xây dựng chính là đầu vào then chốt cho giai đoạn triển khai. Kết quả của giai đoạn triển khai là hệ thống hoạt động là đầu vào của giai đoạn vận hành và bảo trì trong vòng đời hệ thống. Trong mục này, chúng tôi trình bày các bước trong giai đoạn triển khai cho một dự án phát triển hệ thống thông thường.

13.2.1. Bước 1 - Kiểm thử hệ thống lần cuối

- Kiểm tra mạng, cơ sở dữ liệu, phần mềm được mua về, phần mềm mới xây dựng và phần mềm đã có để đảm bảo rằng chúng đều có thể hoạt động cùng nhau.
- Phân bổ nhiệm vụ:

o Người phân tích hệ thống

- Phát triển dữ liệu kiểm thử cho hệ thống
- Truyền đạt các vấn đề khó khăn

o Người xây dựng hệ thống (lập trình viên ứng dụng, lập trình viên cơ sở dữ liệu và chuyên viên mạng)

- Giải quyết các vấn đề xuất hiện trong quá trình kiểm thử

o Người sở hữu và người sử dụng hệ thống

- Kiểm chứng xem hệ thống hoạt động có chính xác hay không
- Kết quả thu được có thể là việc phải quay trở lại giai đoạn xây dựng để sửa đổi hệ thống

o Lặp lại tới khi việc kiểm thử hệ thống thành công

13.2.2. Bước 2 – Lập kế hoạch chuyển đổi

- Người phân tích hệ thống sẽ xây dựng một kế hoạch chuyển đổi chi tiết từ hệ thống cũ sang hệ thống mới, trong đó xác định rõ:

o Cách thức cài đặt cơ sở dữ liệu

o Cách thức đào tạo người dùng

o Các tài liệu cần xây dựng

o Chiến lược chuyển đổi

- Phân bố nhiệm vụ:

o Người phân tích hệ thống/người phân tích dự án

- Phát triển kế hoạch chuyển đổi chi tiết

o Ban lãnh đạo

- Phê chuẩn kế hoạch và thời gian biểu

13.2.3. Bước 3 – Cài đặt cơ sở dữ liệu

- Nạp cơ sở dữ liệu mới từ dữ liệu sẵn có từ hệ thống cũ

o Thông thường là phải tái cấu trúc dữ liệu khi nạp vào

o Phải đảm bảo rằng dữ liệu được chuyển đổi một cách chính xác

- Phân bố nhiệm vụ:

o Lập trình viên ứng dụng:

- Viết (hoặc sử dụng) các chương trình đặc biệt để trích rút dữ liệu từ các cơ sở dữ liệu sẵn có và nạp vào cơ sở dữ liệu mới

o Người phân tích và người thiết kế hệ thống

- Tính toán kích thước cơ sở dữ liệu và ước lượng thời gian cài đặt

13.2.4. Bước 4 – Huấn luyện người dùng

- Đào tạo và cung cấp tài liệu cho người sử dụng
- Phân bố nhiệm vụ:

o Người phân tích hệ thống

- Lập kế hoạch đào tạo

- Theo dõi việc đào tạo
- Viết tài liệu
- Trợ giúp người dùng trong thời gian học

o Người sở hữu hệ thống

- Phê duyệt thời gian hoàn thiện việc đào tạo

o Người sử dụng hệ thống

- Tham dự khóa đào tạo
- Tiếp nhận hệ thống

13.2.5. Bước 5 – Chuyển đổi sang hệ thống mới

- Hệ thống được bàn giao từ người phát triển tới cho người dùng cuối
- Phân bố nhiệm vụ:

o Người phân tích hệ thống/Người quản lý dự án

- Thực hiện kế hoạch chuyển đổi
- Điều chỉnh lỗi
- Đo mức độ hài lòng đối với hệ thống

o Người sở hữu hệ thống/người sử dụng hệ thống

- Cung cấp ý kiến phản hồi

Câu hỏi thảo luận

13.1. Nêu các bước cần thực hiện trong giai đoạn xây dựng hệ thống.

13.2. Nêu các bước cần thực hiện trong giai đoạn triển khai.

Tổng kết chương 13

Chương này trình bày về xây dựng và triển khai hệ thống

Người học cần nắm vững các bước trong quá trình xây dựng hệ thống:

- Xây dựng và kiểm thử mạng
- Xây dựng và kiểm thử cơ sở dữ liệu
- Cài đặt và kiểm thử gói phần mềm mới
- Viết và kiểm thử các chương trình mới

Người học cần nắm rõ các bước trong giai đoạn triển khai một hệ thống:

- Kiểm thử hệ thống lần cuối
- Lập kế hoạch chuyển đổi
- Cài đặt cơ sở dữ liệu

- Huấn luyện người dùng
- Chuyển đổi sang hệ thống mới

Hướng dẫn bài tập lớn

Tiến hành cài đặt, chạy thử hệ thống đã xây dựng được, nhận xét về hệ thống. Viết tiếp báo cáo bài tập lớn với các nội dung cơ bản:

- Yêu cầu phần cứng để cài đặt hệ thống
- Nhận xét về hiệu quả, độ chính xác của hệ thống

CHƯƠNG 14. VẬN HÀNH VÀ HỖ TRỢ HỆ THỐNG

Mục tiêu

Chương này trình bày tổng quan về việc đưa hệ thống vào hoạt động cũng như thực hiện bảo trì, hỗ trợ hệ thống.

14.1. Tổng quan về vận hành và hỗ trợ hệ thống

Hỗ trợ hệ thống (Systems support) là việc hỗ trợ về mặt kỹ thuật cho người sử dụng cũng như việc bảo trì để sửa lỗi hoặc đáp ứng những yêu cầu mới xuất hiện.**Vận hành hệ thống** (Systems operation) là việc thực thi hàng ngày của một hệ thống thông tin.

- Các hoạt động hỗ trợ hệ thống:

o **Bảo trì hệ thống** (Program maintenance) sửa các lỗi xuất hiện trong quá trình phát triển hệ thống

o **Phục hồi hệ thống** (System recovery) là việc phục hồi lại hệ thống và dữ liệu sau khi xảy ra sự cố

o **Hỗ trợ kỹ thuật** (Technical support) là bất kỳ sự trợ giúp nào được cung cấp cho người sử dụng trong trường hợp cần thiết

o **Nâng cấp hệ thống** (System enhancement) là việc cải thiện hệ thống để xử lý các vấn đề nghiệp vụ hoặc vấn đề kỹ thuật hay yêu cầu kỹ thuật mới

14.2. Bảo trì hệ thống

- Các nguyên nhân gây lỗi:

o Các yêu cầu không được xem xét kỹ

o Các yêu cầu không được truyền đạt chính xác

o Các yêu cầu bị hiểu sai

o Các yêu cầu hoặc bản thiết kế bị cài đặt không chính xác

o Sử dụng sai mục đích chương trình

- Các mục tiêu của việc bảo trì hệ thống:

o Tạo các thay đổi được dự đoán trước đối với hệ thống hiện có để hiệu chỉnh lỗi

o Duy trì các bộ phận vẫn hoạt động tốt của chương trình và tránh trường hợp những thay đổi có thể gây ảnh hưởng bất lợi tới các bộ phận đó

o Tránh nhiều nhất có thể việc làm giảm hiệu suất của hệ thống

o Hoàn thành nhiệm vụ nhanh nhất có thể để tránh việc hy sinh chất lượng và tính tin cậy của hệ thống

14.2.1. Bước 1 – Xác định vấn đề

Các dự án bảo trì hệ thống nhỏ thường được kích hoạt bởi việc phát hiện ra lỗi. Hầu hết các lỗi đều được phát hiện bởi người sử dụng khi họ nhận thấy một số bộ phận của hệ thống hoạt động không chính xác. Nhiệm vụ đầu tiên của người phân tích hệ thống hoặc lập trình viên là xác định vấn đề. Nếu đúng là có lỗi thì việc bảo trì phải được thực hiện trên bản sao của chương trình. Còn chương trình gốc vẫn được sử dụng tới khi sửa xong lỗi.

14.2.2. Bước 2 – Đánh dấu chương trình

Không phải phần nào của chương trình cũng hỏng. Do đó, trước khi thực hiện bất cứ thay đổi nào đối với chương trình thì cần kiểm thử chương trình để phân định ranh giới giữa phần chương trình có lỗi cần sửa với phần chương trình có thể xem xét muộn hơn.

14.2.3. Bước 3 – Nghiên cứu và bắt lỗi chương trình

Tri thức về chương trình thì có được từ mã nguồn. Để có thể hiểu được chương trình thì cần không ít thời gian. Hoạt động này có thể bị chậm trễ do một số hạn chế có thể có sau:

- Cấu trúc chương trình tồi
- Lôgíc không có cấu trúc
- Những hoạt động bảo trì không tốt trước đó (chẳng hạn như việc sửa đổi nhanh chóng và phần mở rộng được thiết kế tồi)
- Mã chết (là những câu lệnh không thể thực hiện được – thường bị bỏ qua trong các lần kiểm thử và bắt lỗi trước đây)
- Tài liệu không tốt hoặc không đầy đủ

14.2.4. Kiểm thử chương trình

Một phiên bản đề cử của chương trình phải được kiểm thử trước khi được đưa vào vận hành. Dưới đây là những bước kiểm thử cần thiết hoặc tùy chọn:

o **Kiểm thử bộ phận** (Unit testing) cần thiết để đảm bảo rằng một chương trình độc lập đã được sửa lỗi mà không gây ảnh hưởng ngoài dự đoán tới chương trình.

o **Kiểm thử hệ thống** (System testing) cần thiết để đảm bảo rằng toàn bộ ứng dụng (trong đó chương trình đã được sửa chữa và kiểm thử là một phần) vẫn hoạt động tốt trên phạm vi toàn hệ thống.

o **Kiểm thử hồi quy** (Regression testing) ngoại suy ảnh hưởng của các thay đổi tới hiệu suất hệ thống (thông lượng và thời gian đáp ứng) bằng cách phân tích hiệu suất trước và sau khi thực hiện thay đổi.

- **Kiểm soát phiên bản** (Version control) là quá trình trong đó ta theo dõi những thay đổi đã có đối với chương trình để làm thuận tiện việc tìm hiểu chương trình về sau.

14.3. Phục hồi hệ thống

Sự cố hệ thống là điều không thể tránh khỏi. Người phân tích hệ thống thường sửa lỗi hệ thống hoặc haọt động như một cầu nối giữa người sử dụng và người có trách nhiệm sửa lỗi. Các hoạt động phục hồi hệ thống có thể tóm tắt như sau:

- Trong nhiều trường hợp, người phân tích hệ thống có thể ngồi tại máy tính của người dùng để phục hồi hệ thống. Họ có thể cần phải cung cấp các chỉ dẫn để hiệu chỉnh cho người dùng để tránh lặp lại sự cố.
- Trong một số trường hợp, người phân tích phải liên hệ với nhân lực vận hành hệ thống để sửa lỗi. Điều này thường cần thiết khi hệ thống có sử dụng máy chủ. Thường thì người quản trị mạng hoặc quản trị viên cơ sở dữ liệu hay người quản trị web sẽ xem xét các vấn đề của máy chủ.
- Trong một số trường hợp, người phân tích có thể cần phải nhờ người quản trị dữ liệu phục hồi các tệp dữ liệu hoặc cơ sở dữ liệu đã mất hoặc bị hỏng. Khi phục hồi dữ liệu nghiệp vụ thì không chỉ cần khôi phục cơ sở dữ liệu.

o Bất cứ giao dịch nào xuất hiện trong khoảng thời gian kể từ lần sao lưu cuối cùng tới lúc phục hồi dữ liệu đều phải được xử lý lại.

o Nếu sự cố xảy ra trong khi diễn ra một giao dịch và giao dịch đó đã được hoàn thành một phần thì bất cứ việc cập nhật giao dịch nào diễn ra trước sự cố đều phải được hủy bỏ trước khi xử lý lại toàn bộ giao dịch.

- Trong một số trường hợp, người phân tích có thể phải nhờ người quản trị mạng sửa một lỗi trên mạng cục bộ hay mạng diện rộng.
- Trong một số trường hợp, người phân tích có thể phải nhờ chuyên viên kỹ thuật giải quyết các vấn đề nảy sinh về phần cứng.
- Trong một số trường hợp, người phân tích sẽ phát hiện được lỗi phần mềm có thểgây ra sự cố. Họ sẽ cố gắng cô lập lỗi nhanh chóng để nó không thể gây ra sự cố khác. Tiếp theo là việc xử lý lỗi.

14.4. Hỗ trợ kỹ thuật

Một hoạt động thường xuyên khác trong giai đoạn hỗ trợ hệ thống đó là hỗ trợ kỹ thuật. Cho dù người dùng có được đào tạo tốt như thế nào hay các tài liệu được viết kỹ thế nào thì người dùng vẫn cần có sự trợ giúp. Người phân tích hệ thống thường sẵn sàng đợi yêu cầu của người dùng. Những công việc thường phải làm là:

- Đều đặn theo dõi việc sử dụng hệ thống
- Theo dõi kết quả thăm dò mức độ hài lòng của người dùng và gặp gỡ họ
- Thay đổi thủ tục nghiệp vụ cho phù hợp với hoạt động của hệ thống phần mềm
- Cung cấp thêm các hoạt động đào tạo người dùng
- Lưu lại các yêu cầu và ý tưởng về việc nâng cấp hệ thống

14.5. Nâng cấp hệ thống

14.5.1. Động lực của việc nâng cấp hệ thống

Hầu hết các hoạt động nâng cấp hệ thống đều xuất phát từ việc đáp ứng các sự kiện sau:

- *Các vấn đề về nghiệp vụ mới phát sinh*: một vấn đề nghiệp vụ mới nảy sinh khiến cho một phần của hệ thống hiện tại trở nên không sử dụng được hoặc không hiệu quả.
- *Các yêu cầu nghiệp vụ mới xuất hiện*: một yêu cầu nghiệp vụ mới (ví dụ loại báo cáo hay giao dịch hoặc chính sách mới) cần có để duy trì giá trị sử dụng của hệ thống hiện tại.
- *Các yêu cầu công nghệ mới xuất hiện*: một quyết định về việc sử dụng một công nghệ mới (bao gồm cả việc nâng cấp phần cứng và phần mềm) trong hệ thống hiện có cần được thực hiện.
- *Các yêu cầu thiết kế mới*: một phần tử của hệ thống hiện có cần được thiết kế lại (ví dụ thêm bảng hoặc trường vào cơ sở dữ liệu, thay đổi giao diện người dùng…)

14.5.2. Bước 1 – Phân tích yêu cầu nâng cấp

Bước này nhằm xác định các hoạt động cần thực hiện để đáp ứng yêu cầu nâng cấp của hệ thống. Nhiệm vụ đầu tiên là phân tích một yêu cầu trong mối tương quan với các yêu cầu thay đổi nổi bật khác để xác đinh thứ tự ưu tiên.

Nếu cần phải thay đổi ngay lập tức thì yêu cầu được chọn cần phải được định hướng giải pháp tùy theo loại thay đổi cần thực hiện:

- *Các vấn đề nghiệp vụ mới* cần được hướng tới phiên bản thu nhỏ của giai đoạn phân tích vấn đề. Từ đó, việc nâng cấp sẽ được định hướng thông qua các phiên bản thu nhỏ tương ứng của các giai đoạn phân tích yêu cầu, phân tích quyết định, thiết kế,xây dựng và triển khai.
- *Các yêu cầu nghiệp vụ mới* cần được hướng tới các phiên bản thu nhỏ của các giai đoạn phân tích yêu cầu, phân tích quyết định, thiết kế, xây dựng và triển khai.

- *Các yêu cầu kỹ thuật mới* cần được hướng tới giai đoạn phân tích quyết định trước khi diễn ra giai đoạn thiết kế, xây dựng và triển khai. Việc phân tích quyết định sẽ xác định xem liệu công nghệ được đề xuất có khả thi trong hệ thống mới hay không. Điều này đặc biệt quan trọng bởi việc thay đổi công nghệ có thể tốn kém và phức tạp.
- *Các yêu cầu thiết kế mới* phải được hướng tới giai đoạn thiết kế, xây dựng và triển khai.

14.5.3. Bước 2 – Thực hiện sửa chữa nhanh

Một số kiểu nâng cấp hệ thống có thể được thực hiện nhanh chóng bằng cách viết những chương trình đơn giản mới hoặc thựuc hiện các thay đổi đơn giản đối với các chương trình đang có. Các chương trình và thay đổi đơn giản có thể được thực hiện mà không cần phải tái cấu trúc dữ liệu (tức là thay đổi cấu trúc cơ sở dữ liệu), không phải cập nhật dữ liệu hay nhập dữ liệu mới. Nói một cách khác, các chương trình đó sinh ra các báo các và đầu ra mới.

14.5.4. Bước 3 – Phục hồi hệ thống vật lý hiện có

Đôi khi xảy ra việc hệ thống được nâng cấp nhưng tài liệu về nó lại không được cập nhật. Và có trường hợp các hệ thống được phát triển không theo quy trình nghiêm ngặt nào. Khi đó, người phân tích có thể được yêu cầu phục hồi cấu trúc vật lý hiện có của một hệ thống để làm cơ sở cho việc nâng cấp hệ thống về sau.

Câu hỏi thảo luận

14.1. Nêu các bước cần thực hiện khi bảo trì hệ thống.

14.2. Nêu các bước cần thực hiện khi nâng cấp hệ thống.

Tổng kết chương 14

Chương 14 trình bày về bảo trì, phục hồi hệ thống, hỗ trợ về kỹ thuật và nâng cấp hệ thống.

Người học cần nắm vững các bước trong giai đoạn bảo trì hệ thống:

- Xác định vấn đề
- Đánh dấu chương trình
- Nghiên cứu và bắt lỗi chương trình
- Kiểm thử chương trình

Người học nắm rõ các bước trong quá trình nâng cấp hệ thống:

- Phân tích yêu cầu nâng cấp
- Thực hiện sửa chữa nhanh

- Phục hồi hệ thống vật lý hiện có

Hướng dẫn bài tập lớn

Xem lại và hoàn thiện chương trình, báo cáo bài tập lớn, nộp cho giáo viên

Phần IV: PHÂN TÍCH THIẾT KẾ HỆ THỐNG HƯỚNG ĐỐI TƯỢNG

CHƯƠNG 15. TỔNG QUAN VỀ THIẾT KẾ HƯỚNG ĐỐI TƯỢNG

15.1. Phân tích hướng đối tượng (Object Oriented Analysis - OOA):

Là giai đoạn phát triển một mô hình chính xác và súc tích của vấn đề, có thành phần là các đối tượng và khái niệm đời thực, dễ hiểu đối với người sử dụng.

Trong giai đoạn OOA, vấn đề được trình bày bằng các thuật ngữ tương ứng với các đối tượng có thực. Thêm vào đó, hệ thống cần phải được định nghĩa sao cho người không chuyên Tin học có thể dễ dàng hiểu được.

Dựa trên một vấn đề có sẵn, nhà phân tích cần ánh xạ các đối tượng hay thực thể có thực như khách hàng, ô tô, người bán hàng, … vào thiết kế để tạo ra được bản thiết kế gần cận với tình huống thực. Mô hình thiết kế sẽ chứa các thực thể trong một vấn đề có thực và giữ nguyên các mẫu hình về cấu trúc, quan hệ cũng như hành vi của chúng. Nói một cách khác, sử dụng phương pháp hướng đối tượng chúng ta có thể mô hình hóa các thực thể thuộc một vấn đề có thực mà vẫn giữ được cấu trúc, quan hệ cũng như hành vi của chúng.

Đối với ví dụ một phòng bán ô tô, giai đoạn OOA sẽ nhận biết được các thực thể như:

- Khách hàng
- Người bán hàng
- Phiếu đặt hàng
- Phiếu (hoá đơn) thanh toán
- Xe ô tô

Tương tác và quan hệ giữa các đối tượng trên là:

- Người bán hàng dẫn khách hàng tham quan phòng trưng bày xe.
- Khách hàng chọn một chiếc xe
- Khách hàng viết phiếu đặt xe
- Khách hàng trả tiền xe
- Xe ô tô được giao đến cho khách hàng

Đối với ví dụ nhà băng lẻ, giai đoạn OOA sẽ nhận biết được các thực thể như:

- Loại tài khoản: ATM (rút tiền tự động), Savings (tiết kiệm), Current (bình thường), Fixed (đầu tư), ...

- Khách hàng
- Nhân viên
- Phòng máy tính.

Tương tác và quan hệ giữa các đối tượng trên:

- Một khách hàng mới mở một tài khoản tiết kiệm
- Chuyển tiền từ tài khoản tiết kiệm sang tài khoản đầu tư
- Chuyển tiền từ tài khoản tiết kiệm sang tài khoản ATM

Xin chú ý là ở đây, như đã nói, ta chú ý đến cả ***hai*** khía cạnh: thông tin và cách hoạt động của hệ thống (tức là những gì có thể xảy ra với những thông tin đó).

Lối phân tích bằng kiểu ánh xạ "đời thực" vào máy tính như thế thật sự là ưu điểm lớn của phương pháp hướng đối tượng.

15.2. Thiết kế hướng đối tượng (Object Oriented Design - OOD):

Là giai đoạn tổ chức chương trình thành các tập hợp đối tượng cộng tác, mỗi đối tượng trong đó là thực thể của một lớp. Các lớp là thành viên của một cây cấu trúc với mối quan hệ thừa kế.

Mục đích của giai đoạn OOD là tạo thiết kế dựa trên kết quả của giai đoạn OOA, dựa trên những quy định phi chức năng, những yêu cầu về môi trường, những yêu cầu về khả năng thực thi, ... OOD tập trung vào việc cải thiện kết quả của OOA, tối ưu hóa giải pháp đã được cung cấp trong khi vẫn đảm bảo thỏa mãn tất cả các yêu cầu đã được xác lập.

Trong giai đoạn OOD, nhà thiết kế định nghĩa các chức năng, thủ tục (operations), thuộc tính (attributes) cũng như mối quan hệ của một hay nhiều lớp (class) và quyết định chúng cần phải được điều chỉnh sao cho phù hợp với môi trường phát triển. Đây cũng là giai đoạn để thiết kế ngân hàng dữ liệu và áp dụng các kỹ thuật tiêu chuẩn hóa.

Về cuối giai đoạn OOD, nhà thiết kế đưa ra một loạt các biểu đồ (diagram) khác nhau. Các biểu đồ này có thể được chia thành hai nhóm chính là Tĩnh và động. Các biểu đồ tĩnh biểu thị các lớp và đối tượng, trong khi biểu đồ động biểu thị tương tác giữa các lớp và phương thức hoạt động chính xác của chúng. Các lớp đó sau này có thể được nhóm thành các gói (Packages) tức là các đơn vị thành phần nhỏ hơn của ứng dụng.

15.3. Lập trình hướng đối tượng (Object Oriented Programming - OOP):

Giai đoạn xây dựng phần mềm có thể được thực hiện sử dụng kỹ thuật lập trình hướng đối tượng. Đó là phương thức thực hiện thiết kế hướng đối tượng qua việc sử dụng một ngôn ngữ lập trình có hỗ trợ các tính năng hướng đối tượng. Một vài ngôn

ngữ hướng đối tượng thường được nhắc tới là C++ và Java. Kết quả chung cuộc của giai đoạn này là một loạt các code chạy được, nó chỉ được đưa vào sử dụng sau khi đã trải qua nhiều vòng quay của nhiều bước thử nghiệm khác nhau.

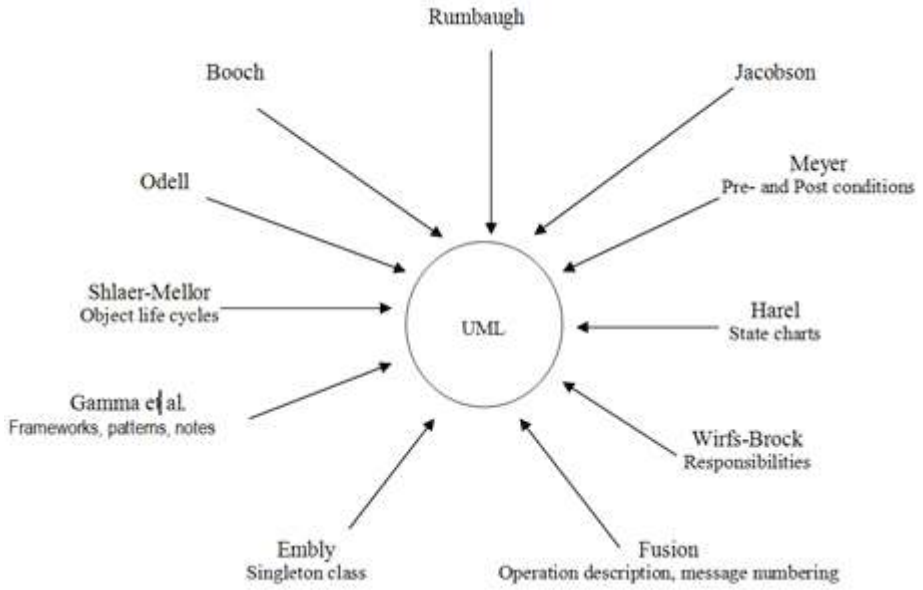
CHƯƠNG 16. LỊCH SỬ PHÁT TRIỂN CỦA UML

16.1. Lịch sử phát triển của UML

Những năm đầu của thập kỷ 90 có rất nhiều phương pháp phân tích, thiết kế hệ thống hướng đối tượng và cùng với chúng là các ký hiệu riêng cho từng phương pháp. Số lượng các phương pháp trong khoảng từ 10 đã lên đến gần 50 trong những năm từ 1989 đến 1994. Ba phương pháp phổ biến nhất là OMT (Object Modeling Technique)[James Rumbaugh], Booch91 [Grady Booch] và OOSE (Object-Oriented Software Enginering)[Ivar Jacobson]. Mỗi phương pháp đều có những điểm mạnh và yếu. Như OMT mạnh trong phân tích và yếu ở khâu thiết kế, Booch91 thì mạnh ở thiết kế và yếu ở phân tích. OOSE mạnh ở phân tích các ứng xử, đáp ứng của hệ thống mà yếu trong các khâu khác.

Do các phương pháp chưa hoàn thiện nên người dùng rất phân vân trong việc chọn ra một phương pháp phù hợp nhất để giải quyết bài toán của họ. Hơn nữa, việc các ký hiệu khác nhau của các phương pháp đã gây ra những sự mập mờ, nhầm lẫn khi mà một ký hiệu có thể mang những ý nghĩa khác nhau trong mỗi phương pháp. Ví dụ như một hình tròn được tô đen biểu hiện một multiplicity trong OMT lại là một aggregation trong Booch). Thời kỳ này còn được biết đến với tên gọi là cuộc chiến giữa các phương pháp. Khoảng đầu năm 94, Booch đã cải tiến phương pháp của mình trong đó có ứng dụng những ưu điểm của các phương pháp của Rumbaugh và Jacobson. Tương tự Rumbaugh cũng cho đăng một loạt các bài báo được biết đến với tên gọi phương pháp OMT-2 cũng sử dụng nhiều ưu điểm của phương pháp của Booch. Các phương pháp đã bắt đầu hợp nhất, nhưng các kí hiệu sử dụng ở các phương pháp vẫn còn nhiều điểm khác biệt.

Cuộc chiến này chỉ kết thúc khi có sự ra đời của UML - một ngôn ngữ mô hình hóa hợp nhất. Tại sao lại là hợp nhất? Đó là do có sự hợp nhất các cách kí hiệu của Booch, OMT và Objectory cũng như các ý tưởng tốt nhất của một số phương pháp khác như hình vẽ sau:



Hình 16.1

Bằng cách hợp nhất các kí hiệu sử dụng trong khi phân tích, thiết kế của các phương pháp đó, UML cung cấp một nền tảng chuẩn trong việc phân tích thiết kế. Có nghĩa là các nhà phát triển vẫn có thể tiến hành theo phương pháp mà họ đang sử dụng hoặc là có thể tiến hành theo một phương pháp tổng hợp hơn(do thêm vào những bước ưu điểm của từng phương pháp). Nhưng điều quan trọng là các ký hiệu giờ đây đã thống nhất và mỗi ký hiệu chuẩn của tổ chức OMG (Object Management Group) vào tháng 7-1997.

16.2. Unified Modeling Language là gì?

UML là một ngôn ngữ dùng để

- Trực quan hóa
- Cụ thể hóa
- Sinh mã ở dạng nguyên mẫu
- Lập và cung cấp tài liệu

UML là một ngôn ngữ bao gồm một bảng từ vựng và các quy tắc để kết hợp các từ vựng đó phục vụ cho mục đích giao tiếp. Một ngôn ngữ dùng cho việc lập mô hình là ngôn ngữ mà bảng từ vựng(các ký hiệu) và các quy tắc của nó tập trung vào việc thể hiện về mặt khái niệm cũng như vật lý của một hệ thống.

Mô hình hóa mang lại sự hiểu biết về một hệ thống. Một mô hình không thể giúp chúng ta hiểu rõ một hệ thống, thường là phải xây dựng một số mô hình xét từ những góc độ khác nhau. Các mô hình này có quan hệ với nhau.

UML sẽ cho ta biết cách tạo ra và đọc hiểu được một mô hình đươc cấu trúc tốt, nhưng nó không cho ta biết những mô hình nào nên tạo ra và khi nào tạo ra chúng. Đó là nhiệm vụ của quy trình phát triển phần mềm.

16.2.1. UML là ngôn ngữ dùng để trực quan hóa

Đối với nhiều lập trình viên, không có khoảng cách nào giữa ý tưởng để giải quyết một vấn đề và việc thể hiện điều đó thông qua các đoạn mã. Họ nghĩ ra và họ viết mã. Trên thực tế, điều này gặp một số vấn đề. Thứ nhất, việc trao đổi về các ý tưởng giữa những người lập trình sẽ gặp khó khăn, trừ khi tất cả đều nói cùng một ngôn ngữ. Thậm chí ngay cả khi không gặp trở ngại về ngôn ngữ thì đối với từng công ty, từng nhóm cũng có những "ngôn ngữ" riêng của họ. Điều này gây trở ngại cho một người mới vào để có thể hiểu được những việc đang được tiến hành. Hơn nữa, trong lĩnh vực phần mềm, nhiều khi khó có thể hiểu được nếu chỉ xem xét các đoạn mã lệnh. Ví dụ như sự phân cấp của các lớp, ta có thể phải duyệt rất nhiều đoạn lệnh để hiểu được sự phân cấp của các lớp. Và nếu như người lập trình không mô tả các ý tưởng mà anh ta đã xây dựng thành mã lệnh thì nhiều khi cách tốt nhất là xây dựng lại trong trường hợp một người khác đảm nhận tiếp nhiệm vụ khi anh ta rời khỏi nhóm.

Xây dựng mô hình sử dụng ngôn ngữ UML đã giải quyết được các khó khăn trên.

Khi trở thành một chuẩn trong việc lập mô hình, mỗi kí hiệu mang một ý nghĩa rõ ràng và duy nhất, một nhà phát triển có thể đọc được mô hình xây dựng bằng UML do một người khác viết.

Những cấu trúc mà việc nắm bắt thông qua đọc mã lệnh là khó khăn nay đã được thể hiện trực quan.

Một mô hình rõ ràng, sáng sủa làm tăng khả năng giao tiếp, trao đổi giữa các nhà phát triển.

16.2.2. UML là ngôn ngữ dùng để chi tiết hóa

Có nghĩa là xây dựng các mô hình một các tỉ mỉ, rõ ràng, đầy đủ ở các mức độ chi tiết khác nhau. Đặc biệt là UML thực hiện việc chi tiết hoá tất cả các quyết định quan trọng trong phân tích, thiết kế và thực thi một hệ thống phần mềm.

16.2.3. UML là ngôn ngữ dùng để sinh ra mã ở dạng nguyên mẫu

Các mô hình xây dựng bởi UML có thể ánh xạ tới một ngôn ngữ lập trình cụ thể như : Java, C++... thậm chí cả các bảng trong một CSDL quan hệ hay CSDL hướng đối tượng.

Việc các yêu cầu có khả năng thường xuyên thay đổi trong quá trình phát triển hệ thống dẫn đến việc các cấu trúc và hành vi của hệ thống được xây dựng có thể khác

mô hình mà ta đã xây dựng. Điều này có thể làm cho một mô hình tốt trở nên vô nghĩa vì nó không còn phản ánh đúng hệ thống nữa. Cho nên phải có một cơ chế để đồng bộ hóa giữa mô hình và mã lệnh.

UML cho phép cập nhật một mô hình từ các mã thực thi.(ánh xạ ngược). Điều này tạo ra sự nhất quán giữa mô hình của hệ thống và các đoạn mã thực thi mà ta xây dựng cho hệ thống đó.

16.2.4. UML là ngôn ngữ dùng để lập và cung cấp tài liệu

Một tổ chức phần mềm ngoài việc tạo ra các đoạn mã lệnh(thực thi) thì còn tạo ra các tài liệu sau:

- *Ghi chép về các yêu cầu của hệ thống*
- *Kiến trúc của hệ thống*
- *Thiết kế*
- *Mã nguồn*
- *Kế hoạch dự án*
- *Tests*
- *Các nguyên mẫu*
- *...*

16.2.5. Ứng dụng của UML

Mục đích chính của UML là để xây dựng mô hình cho các hệ thống phần mềm, nó có thể được sử dụng một cách hiệu quả trong nhiều lĩnh vực như:

- *Hệ thống thông tin doanh nghiệp (enterprise)*
- *Ngân hàng và dịch vụ tài chính*
- *Viễn thông*
- *Giao thông*
- *Hàng không và quốc phòng*
- *Máy móc điện tử dùng trong y tế*
- *Khoa học*
- *Các ứng dụng phân tán dựa trên Web*

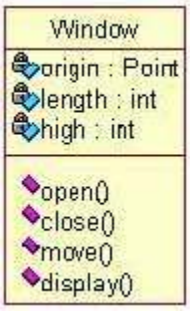
UML không chỉ giới hạn trong lĩnh vực phần mềm. Nó còn có thể dùng để lập mô hình cho các hệ thống không phải là phần mềm như hệ thống pháp luật (luồng công việc - workflow), thiết kế phần cứng, ...

CHƯƠNG 17.CÁC THÀNH PHẦN CỦA UML

17.1. Các phần tử mang tính cấu trúc

Lớp (Class)

Là một tập hợp các đối tượng có cùng một tập thuộc tính, các hành vi, các mối quan hệ với những đối tượng khác.



Hợp tác (Collaboration)

Thể hiện một giải pháp thi hành bên trong hệ thống, bao gồm các lớp/ đối tượng mối quan hệ và sự tương tác giữa chúng để đạt được một chức năng mong đợi của Use case.

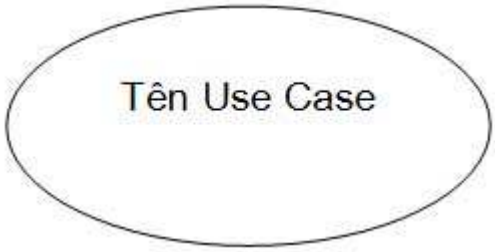
Giao diện (Interface)

Là một tập hợp các phương thức (operation) tạo nên dịch vụ của một lớp hoặc một thành phần (component). Nó chỉ ra một tập các operation ở mức khai báo chứ không phải ở mức thực thi (implementation).

Use case

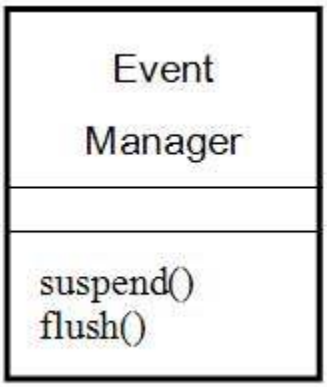
là mô tả một tập hợp của nhiều hành động tuần tự mà hệ thống thực hiện để đạt được một kết quả có thể quan sát được đối với một actor cụ thể nào đó. Actor là những gì ở bên ngoài mà tương tác với hệ thống. Use case mô tả sự tương tác giữa actor và hệ

thống. Nó thể hiện chức năng mà hệ thống sẽ cung cấp cho actor. Tập hợp các Use case của hệ thống sẽ tạo nên tất cả các trường hợp mà hệ thống có thể được sử dụng.



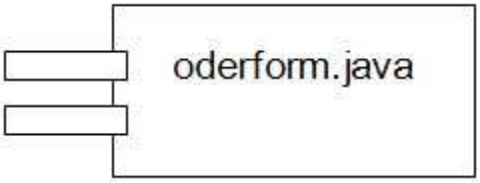
Lớp tích cực (Acitive class)

là một lớp mà các đối tượng của nó thực hiện các hoạt động điều khiển. Lớp tích cực cũng giống như lớp bình thường ngoại trừ việc các đối tượng của nó thể hiện các phần tử mà ứng xử của chúng có thể thực hiện đồng thời với các phần từ khác. Lớp này thường dùng để biểu diễn tiến trình(process) và luồng(thread)



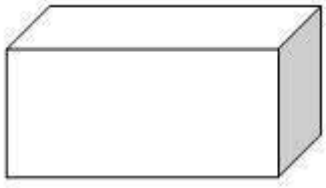
Thành phần (Component)

là biểu diễn vật lý của mã nguồn. Trong hệ thống ta sẽ thấy các kiểu khác nhau của component như các thành phần COM+ hay JavaBeans cũng như là các thành phần như các file mã nguồn, các file nhị phân tạo ra trong quá trình phát triển hệ thống.



Nodes

là thể hiện một thành phần vật lý như là một máy tính hay một thiết bị phần cứng.



17.2. Các quy tắc của UML

Các thành phần của UML không thể ngẫu nhiên đặt cạnh nhau. Như bất cứ một ngôn ngữ nào, UML có những quy tắc chỉ ra rằng một mô hình tốt sẽ như thế nào. Một mô hình tốt là mô hình mang tính nhất quán và có sự kết hợp hài hòa giữa các mô hình có liên quan của nó.

UML có một số quy tắc dành cho việc:

- ***Đặt tên:*** *để có thể truy xuất các phần tử của mô hình thì phải đặt tên cho chúng như tên của các quan hệ, biểu đồ...*
- ***Xác định phạm vi:*** *ngữ cảnh mang lại một ý nghĩa cụ thể cho một cái tên*
- ***Tính nhìn thấy được:*** *để có được sự đơn giản và dễ kiểm soát thì ở những ngữ cảnh khác nhau cần chỉ ra rằng một cái tên là hiện hữu và được sử dụng bởi những đối tượng khác như thế nào.*
- ***Tính toàn vẹn:*** *mọi thứ quan hệ một cách đúng đắn và nhất quán với nhau như thế nào.*

17.3. Các kỹ thuật chung của UML

17.3.1. Cụ thể hóa

Như đã trình bày ở phần trên, việc thể hiện trực quan giúp chúng ta hiểu vấn đề dễ dàng hơn chứ không có nghĩa là các mô tả bằng lời là không có ích.Cho nên UML không chỉ là một tập các kí hiệu đồ họa. Bên cạnh các kí hiệu đồ họa còn có các phát biểu bằng lời để chỉ rõ ngữ nghĩa của các kí hiệu đó. Ví dụ như trong kí hiệu của một lớp(một hình chữ nhật) còn có thể được chỉ rõ ra các thuộc tính, các phương thức của lớp đó.

17.3.2 Trang trí

Tất cả các phần tử trong UML đều có một hình dạng phân biệt đối với các phần tử khác. Đồng thời chúng cũng được thiết kế để thể hiện những mặt quan trọng nhất của đối tượng. Ví dụ như kí hiệu cho một lớp là một hình chữ nhật rất dễ vẽ bởi vì lớp là một thành phần quan trọng, xuất hiên rất nhiều trong các mô hình hướng đối tượng. Và kí hiệu này thể hiện được cả 3 thành phần quan trọng của lớp đó là tên lớp, các thuộc tính và các phương thức của nó. Ngoài ra nó còn bao gồm các chi tiết như: lớp đó có phải là lớp trừu tượng không, các thuộc tính, phương thức của nó thuộc loại gì

(public, private hay protected). Nói tóm lại các kí hiệu trong UML giúp ta nhận biết các đặc điểm quan trọng của đối tượng, khái niệm được mô tả một cách dễ dàng và nhanh chóng.

17.3.3. Phân chia

Phân biệt rõ phần trừu tượng và cụ thể.

Trước tiên là lớp và đối tượng. Một lớp là một sự trừu tượng hóa, một đối tượng là một thể hiện cụ thể của sự trừu tượng đó. Trong UML ta có thể mô hình lớp và đối tượng.

Có rất nhiều thứ tương tự. Ví dụ như một Use case và một thể hiện của Use case, một component và một thể hiện của component

17.3.4 Kỹ thuật mở rộng

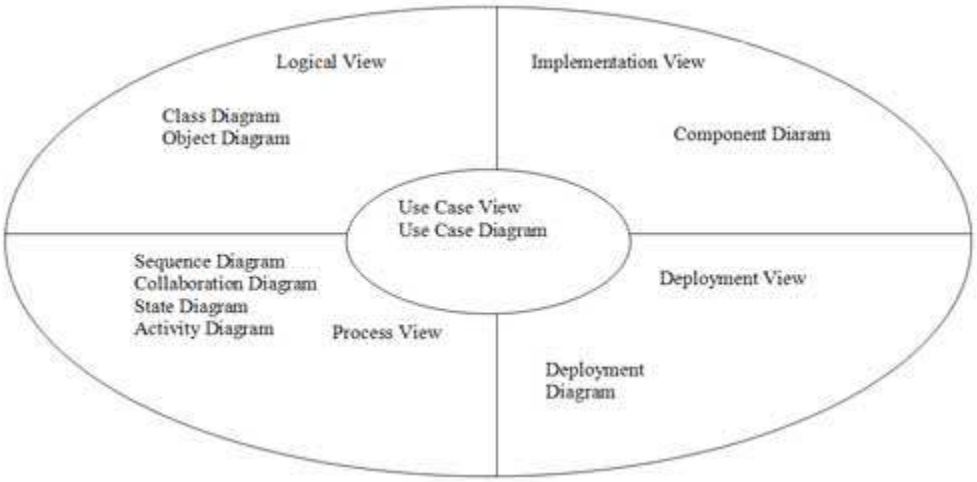
UML cung cấp những thành phần cơ bản để lập nên một mô hình cho một phần mềm. Nhưng nó không thể nào bao quát hết theo thời gian mọi mô hình trong mọi lĩnh vực. Do đó UML được thiết kế mở theo nghĩa là người dùng có thể mở rộng một số thành phần để có thể áp dụng một cách tốt nhất cho hệ thống của họ mà lại không phải thay đổi hay thiết kế lại các thành phần cơ sở của UML. Cơ chế đó bao gồm:

- ***Stereotypes (khuôn mẫu):*** *mở rộng tập từ vựng của UML, cho phép tạo những thành phần mới kế thừa những đặc điểm của những thành phần đã có đồng thời chứa thêm những đặc điểm riêng gắn với một bài toán cụ thể nào đó.*
- ***Tagged values (giá trị thẻ):*** *mở rộng thuộc tính của các thành phần của UML, nó cho phép ta tạo thêm những thông tin mới về một phần tử. Ví dụ như khi làm việc hợp tác để tạo ra một sản phẩm, ta muốn chỉ ra các phiên bản và tác giả của một đối tượng nào đó. Điều này không được xây dựng sẵn trong UML mà có thể thực hiện thông qua việc thêm vào một giá trị thẻ.*
- ***Constraints (ràng buộc):*** *mở rộng ngữ nghĩa của các thành phần của UML, cho phép tạo ra những quy tắc mới hoặc sửa chữa những quy tắc đã có.*

17.4. Kiến trúc của hệ thống

Khi xem xét một hệ thống, chúng ta cần xây dựng các mô hình từ những khía cạnh khác nhau, xuất phát từ thực tế là những người làm việc với hệ thống với những vai trò khác nhau sẽ nhìn hệ thống từ những khía cạnh khác nhau.

UML xét hệ thống trên 5 khía cạnh:



Hình 17.2

17.4.1. Use-Case View

Bao gồm các Use Case mô tả ứng xử của hệ thống theo cách nhìn nhận của người dùng, người phân tích hệ thống. Nó không chỉ ra cách cấu trúc của hệ thống phần mềm, nó chỉ dùng để nhìn nhận một cách tổng quát những gì mà hệ thống sẽ cung cấp, thông qua đó người dùng có thể kiểm tra xem các yêu cầu của mình đã được đáp ứng đầy đủ hay chưa hoặc có chức năng nào của hệ thống là không cần thiết. Biểu đồ dùng đến là biểu đồ Use Case.

17.4.2. Logical View

Được dùng để xem xét các phần tử bên trong hệ thống và mối quan hệ, sự tương tác giữa chúng để thực hiện các chức năng mong đợi của hệ thống.

17.4.3. Process View

Chia hệ thống thành các tiến trình(process) và luồng(thread), mô tả việc đồng bộ hóa và các xử lý đồng thời. Dùng cho người phát triển và tích hợp hệ thống, bao gồm các biểu đồ sequence, collaboration, activity và state.

17.4.4. Implementation View

Bao gồm các component và file tạo nên hệ thống vật lý. Nó chỉ ra sự phụ thuộc giữa các thành phần này, cách kết hợp chúng lại với nhau để tạo ra một hệ thống thực thi.

17.4.5. Deployment View

Chỉ ra cấu hình phần cứng mà hệ thống sẽ chạy trên đó. Nó thể hiện sự phân tán, cài đặt các phần mà tạo nên kiến trúcvật lý của hệ thống. Biểu đồ được sử dụng là biểu đồ Deployment.

Trong chương trước, các bài viết đã đề cập tới tầm quan trọng của việc lập mô hình và sự hỗ trợ của UML trong việc lập mô hình như thế nào. Tuy nhiên nhiệm vụ chính

của UML là đóng vai trò một ngôn ngữ mô hình hóa thống nhất, trực quan, chuẩn hóa các kí hiệu, ngữ nghĩa của các mô hình và các biểu đồ khi thể hiện các đối tượng, các sự kiện trong thế giới thực và trong lĩnh vực máy tính chứ không chỉ ra cho người dùng biết việc lập mô hình cho một hệ thống phải theo các bước như thế nào. Đó chính là mục đích của một phương pháp phân tích, thiết kế hướng đối tượng.

Hướng đối tượng là một cách tiếp cận khác với cách tiếp cận có cấu trúc truyền thống. Với cách tiếp cận hướng đối tượng, ta chia ứng dụng thành các đối tượng, tương đối độc lập với nhau. Sau đó ta có thể xây dựng hệ thống bằng cách kết hợp chúng lại với nhau. Một trong những ưu điểm của phương pháp này là tính sử dụng lại. Ta có thể xây dựng các đối tượng một lần và dùng chúng trong nhiều ứng dụng. Hơn thế nữa các đối tượng này đã qua một quá trình thử nghiệm và kiểm tra nên các rủi ro về lỗi là rất ít.

Vậy phương pháp hướng đối tượng khác phương pháp có cấu trúc ở điểm nào? Theo cách tiếp cận có cấu trúc thì chúng ta tập trung vào các thông tin mà hệ thống sẽ lưu giữ. Chúng ta hỏi người dùng về các thông tin mà họ cần, thiết kế cơ sở dữ liệu để lưu trữ các thông tin này, lập các màn hình để nhập và hiển thị thông tin, tạo các báo cáo để in thông tin. Nói một cách khác, chúng ta tập trung vào thông tin mà ít chú trọng tới cái gì được thực hiện với các thông tin đó tức là ứng xử của hệ thống. Cách tiếp cận này còn được gọi là hướng dữ liệu và đã được dùng để tạo ra hàng nghìn ứng dụng trong nhiều năm qua.

Hướng dữ liệu áp dụng tốt trong việc thiết kế cơ sở dữ liệu và nắm bắt thông tin, tuy nhiên cách tiếp cận này gặp phải một số khó khăn. Một trong những thách thức lớn nhất đó là việc thay đổi các yêu cầu của người dùng. Một hệ thống được xây dựng hướng dữ liệu có thể điều khiển việc thay đổi cơ sở dữ liệu một cách dễ dàng nhưng những thay đổi về quy tắc nghiệp vụ (business rules) sẽ không dễ dàng để thực thi.

Khái niệm hướng đối tượng đã được phát triển để giải quyết vấn đề này. Nó sẽ tập trung vào cả dữ liệu và các thao tác trên các dữ liệu đó. Do đó hệ thống sẽ linh hoạt hơn và dễ dàng thay đổi khi dữ liệu và ứng xử trên dữ liệu thay đổi.

UML không chỉ thuần túy là một ngôn ngữ mô hình hóa. Nó được phát triển bởi các chuyên gia hàng đầu trong lĩnh vực hướng đối tượng, những người đã đề xuất ra những phương pháp phân tích thiết kế hướng đối tượng hay được dùng nhất, như kỹ thuật phân tích Use case của Ivar Jacobsson, biểu đồ chuyển trạng thái của Harel... do đó nếu những người phân tích tiếp cận việc xây dựng các phần tử của mô hình đã được định nghĩa trong UML một cách hợp lý và có hệ thống thì họ sẽ thu được một phương pháp phân tích, thiết kế hướng đối tượng tốt.

Thông thường việc phân tích và thiết kế hệ thống được thực hiện theo các bước sau:

- Phân tích yêu cầu: Dùng phương pháp phân tích Use case để nắm bắt các yêu cầu của khách hàng. Đây là một bước quan trọng và sự thành công của bước này sẽ quyết định sự thành công của dự án. Bởi vì một hệ thống dù có xây dựng tốt đến đâu nhưng không đáp ứng được những nhu cầu của khách hàng hệ thống sẽ thất bại.

- Phân tích: Sau khi đã biết được người dùng muốn gì, chúng ta tập trung mô tả lại hệ thống, các khái niệm chính trong lĩnh vực của hệ thống cần xây dựng, trong hướng đối tượng gọi là các lớp lĩnh vực (domain class), mối quan hệ và sự tương tác giữa các đối tượng đó. Mục đích chính là hiểu hệ thống hoạt động như thế nào.

- Thiết kế: ở bước này sử dụng kết quả thu được ở các bước trước để mở rộng thành một giải pháp kỹ thuật, thêm vào các lớp thuộc về kỹ thuật như các lớp giao diện, các lớp điều khiển...Tập trung mô tả cấu trúc bên trong của hệ thống, sự tương tác của tập hợp các đối tượng để đạt được những chức năng mà hệ thống cần có.

Mặc dù UML không bắt buộc phải sử dụng một quy trình phát triển phần mềm cụ thể nào những nó được khuyến khích sử dụng với quy trình lặp và tăng dần.

Việc phân tích thiết kế hướng đối tượng được hệ thống hóa như sau:

1. Phân tích Use case :
    1. Tìm Actor
    2. Tìm Use case
    3. Xây dựng biểu đồ Use case
2. Tìm lớp:
    1. Lớp
    2. Gói
    3. Xây dựng biểu đồ lớp
    4. Xây dựng biểu đồ đối tượng
3. Phân tích sự tương tác giữa các đối tượng
    1. Kịch bản
    2. Xây dựng biểu đồ trình tự
    3. Xây dựng biểu đồ hợp tác
4. Xác định quan hệ giữa các đối tượng
    1. Quan hệ Association
    2. Quan hệ Generalization
    3. Quan hệ Dependency

4. Quan hệ Realization

5. Thêm vào các thuộc tính và phương thức cho các lớp
6. Xác định ứng xử của đối tượng
    1. Xây dựng biểu đồ chuyển trạng
    2. Xây dựng biểu đồ hoạt động
7. Xác định kiến trúc của hệ thống
    1. Xây dựng biểu đồ thành phần
    2. Xây dựng biểu đồ triển khai.

Kiểm tra lại mô hình.

CHƯƠNG 18.USE CASE

Ứng xử của hệ thống, tức là những chức năng mà hệ thống cung cấp sẽ được mô tả trong mô hình Use case. Trong đó mô tả những chức năng (Use case), những thành phần ở bên ngoài(Actor) tương tác với hệ thống và mối quan hệ giữa Use case và Actor (biểu đồ Use case).

Mục đích quan trọng nhất của mô hình Use case là phục vụ cho việc trao đổi thông tin. Nó cung cấp phương tiện để khách hàng, những người dùng tương lai của hệ thống và những người phát triển hệ thống có thể trao đổi với nhau và biến những yêu cầu về mặt nghiệp vụ của người dùng thành những yêu cầu cụ thể mà lập trình viên có thể hiểu một cách rõ ràng.

18.1. Actor

18.1.1. Định nghĩa actor

Actor không phải là một phần của hệ thống. Nó thể hiện một người hay một hệ thống khác tương tác với hệ thống. Một Actor có thể:

- Chỉ cung cấp thông tin cho hệ thống.
- Chỉ lấy thông tin từ hệ thống.
- Nhận thông tin từ hệ thống và cung cấp thông tin cho hệ thống

18.1.2. Mô tả

Thông thường, các actor được tìm thấy trong phát biểu bài toán bởi sự trao đổi giữa phân tích viên với khách hàng và các chuyên gia trong lĩnh vực(domain expert). Các câu hỏi thường được sử dụng để xác định actor cho một hệ thống là:

- *Đối với một vấn đề cụ thể nào đó thì Ai là người quan tâm ?*
- *Hệ thống được dùng ở nơi nào trong tổ chức?*
- *Ai là người được lợi khi sử dụng hệ thống?*
- *Ai là người cung cấp thông tin cho hệ thống, sử dụng thông tin của hệ thống và xóa các thông tin đó?*
- *Ai là người hỗ trợ và bảo trì hệ thống?*
- *Hệ thống có sử dụng nguồn lực nào từ bên ngoài?*
- *Có người nào đóng một vài vai trò trong hệ thống? Có thể phân thành 2 actor*
- *Có vai trò nào mà nhiều người cùng thể hiện? Có thể chỉ là một actor*
- *Hệ thống có tương tác với các hệ thống nào khác không?*

Có 3 loại Actor chính là:

- *Người dùng*. Ví dụ: sinh viên, nhân viên, khách hàng...
- *Hệ thống khác*.
- *Sự kiện thời gian*. Ví dụ: Kết thúc tháng, đến hạn...

Điều gì tạo nên một tập hợp Actor tốt?

Cần phải cân nhắc kỹ lưỡng khi xác định actor của hệ thống. Công việc này thường được thực hiện lặp đi lặp lại. Danh sách đầu tiên về các actor hiếm khi là danh sách cuối cùng.

Ví dụ như trong bài toán đăng kí các môn học của một trường đại học, có một câu hỏi là liệu các sinh viên mới vào trường là một actor và sinh viên cũ là một actor khác? Giả sử câu trả là có thì bước tiếp theo là xác định xem cách thức mà hai actor này tương tác với hệ thống. Nếu chúng sử dụng hệ thống theo những cách khác nhau thì chúng là hai actor ngược lại sẽ chỉ là một actor mà thôi.

Mô tả Actor:

Việc mô tả một cách ngắn gọn về mỗi actor cần thêm vào mô hình. Mô tả này cần chỉ rõ vai trò của actor khi tương tác với hệ thống. Ví dụ:

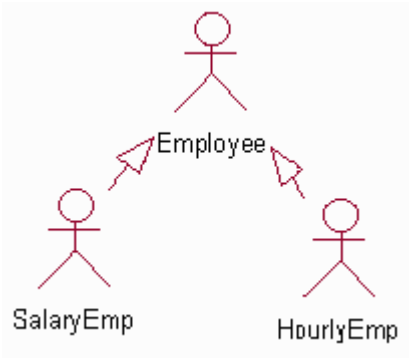
Sinh viên: là những người đăng kí học các lớp ở trường đại học.

18.1.3. Kí hiệu

Actor cũng có mối quan hệ kế thừa. Ví dụ như có thể có hai actor là nhân viên trả lương tháng, nhân viên làm hợp đồng. Cả hai đều thuộc một kiểu là Nhân viên. Actor Nhân viên là một actor trừu tượng vì nó không có một thể hiện nào trong thực tế, nó được dùng để chỉ ra rằng có một số điểm chung giữa hai actor trên.

Nói chung việc mô tả quan hệ kế thừa giữa các Actor là không cần thiết, trừ trường hợp chúng thực hiện những tương tác khác nhau đối với hệ thống.

Ví dụ:



18.2. Use case

18.2.1. Định nghĩa

Là một khối chức năng được thực hiện bởi hệ thống để mang lại một kết quả có giá trị đối với một actor nào đó.

18.2.2. Mô tả

Use case mô tả sự tương tác đặc trưng giữa người dùng và hệ thống. Nó thể hiện ứng xử của hệ thống đối với bên ngoài, trong một hoàn cảnh nhất định, xét từ quan điểm của người sử dụng. Nó mô tả các yêu cầu đối với hệ thống, có nghĩa là những gì hệ thống phải làm chứ không phải mô tả hệ thống làm như thế nào. Tập hợp tất cả Use case của hệ thống sẽ mô tả tất cả các trường hợp mà hệ thống có thể được sử dụng.

Một Use case có thể có những biến thể. Mỗi một biến thể được gọi là một kịch bản (scenario). Phạm vi của một Use case thường được giới hạn bởi các hoạt động mà người dùng thực hiện trên hệ thống trong một chu kì hoạt động để thực hiện một sự kiện nghiệp vụ.

Một Use case mô tả một nghiệp vụ thông thường. Nghiệp vụ này bao gồm các bước riêng rẽ, còn được gọi là các hoạt động. Khi các bước được mô tả dưới dạng văn bản thì việc chỉ ra sự phụ thuộc giữa các bước là một việc mất nhiều thời gian. Việc thể hiện các bước dưới dạng kí hiệu là dễ dàng và dễ hiểu hơn. Do đó Use case thường được mô tả chi tiết thông qua các biểu đồ mô tả hành vi (behavior) như biểu đồ hoạt động (activity diagram), biểu đồ trình tự (sequence diagram), biểu đồ hợp tác(collaboration diagram).

Use case cũng có thể được mô tả thông qua các thiết kế nguyên mẫu màn hình, các ví dụ về biểu mẫu báo cáo. Điều này giúp cho người dùng dễ dàng mường tượng hệ thống sẽ làm việc như thế nào, qua đó có thể kiểm tra tính đúng đắn của Use case.

Các câu hỏi thường được sử dụng để xác định Use Case cho một hệ thống là:

- *Nhiệm vụ của mỗi actor là gì?*
- *Có actor nào sẽ tạo, lưu trữ, thay đổi, xóa hoặc đọc thông tin trong hệ thống?*
- *Có actor nào cần báo tin cho hệ thống về một thay đổi đột ngột từ bên ngoài?*
- *Có actor nào cần được thông báo về một sự việc cụ thể xảy ra trong hệ thống?*
- *Trường hợp sử dụng nào sẽ hỗ trợ và bảo trì hệ thống?*
- *Tất cả các yêu cầu về mặt chức năng có được thể hiện hết thông qua các trường hợp sử dụng chưa?*

Điều gì tạo nên một Use Case tốt

Có một câu hỏi thường xuyên được đặt ra về mức độ chi tiết của Use case. Nó nên ở mức độ nào là tốt. Có lẽ không có câu trả lời hoàn toàn đúng, nhưng có một số nhận xét như sau: "*Một Use case thường biểu hiện một chức năng được thực hiện trọn vẹn (không ngắt quãng) từ đầu đến cuối. Một Use case phải mang lại một điều gì đó có giá trị đối với actor*".

Mô tả Use case

Use case cần có một vài câu ngắn gọn mô tả mục đích của Use case, cho ta biết chức năng do Use case cung cấp.

18.2.3. Kí hiệu

Một Use case được thể hiện bởi một hình ellip kèm theo tên của Use case. Ngoài ra còn có thể có thêm các chú thích để mô tả chi tiết hơn về ý nghĩa của Use case. Mỗi Use case trong hệ thống có tên phân biệt duy nhất. Use case có thể được đánh số để thuận tiện cho việc tra cứu nhanh trên biểu đồ hoặc trong tài liệu mô tả.

Ví dụ:

18.2.4. Luồng sự kiện cho một Use case (The Flow of events)

Use case chỉ cung cấp một khung nhìn ở mức cao, tổng quát. Để hiểu rõ hơn hệ thống cần phải làm gì thì cần phải mô tả chi tiết hơn, gọi là luồng sự kiện. Nó là một tài liệu mô tả các hoạt động cần thiết để đạt được ứng xử mong đợi của Use case.

Tuy là mô tả chi tiết nhưng luồng sự kiện vẫn được viết sao cho *có thể chỉ ra những gì hệ thống cần làm chứ không phải chỉ ra hệ thống làm như thế nào*.

Ví dụ: trong luồng sự kiện chúng ta nói "*Kiểm tra mã của người dùng*" chứ không nói rằng việc đó phải thực hiện bằng cách xem xét ở trong một bảng nào đó trong cơ sở dữ liệu. Nó mô tả chi tiết những gì người dùng của hệ thống sẽ làm và những gì hệ thống sẽ làm. Nó cần phải đề cập tới:

- *Use case bắt đầu và kết thúc khi nào và như thế nào*
- *Có những sự tương tác nào giữa Use case và actor để thực hiện chức năng đó.*
- *Những dữ liệu nào cần thiết cho Use case*
- *Thứ tự thực hiện thông thường của các sự kiện*
- *Các mô tả về các luồng ngoại lệ hoặc rẽ nhánh.*

Mỗi dự án cần có một mẫu chuẩn cho việc tạo tài liệu về luồng sự kiện. Có thể dùng theo mẫu đơn giản như sau:

- ***X****. Luồng sự kiện cho Use case ABC*
- ***X1****. Điều kiện bắt đầu: danh sách những điều kiện phải thỏa mãn trước khi Use case được thực hiện. Ví dụ như: một Use case khác phải thực hiện trước khi Use case này được thực hiện hay người dùng phải có đủ quyền để thực hiện Use case này. Không nhất thiết mọi Use case đều phải có điều kiện bắt đầu.*
- ***X2****. Luồng chính: mô tả những bước chính sẽ xẩy ra khi thực hiện Use case.*
- ***X3****. Các luồng phụ(luồng con).*
- ***X4****. Các luồng rẽ nhánh.*

Trong đó **X** là số thự tự của Use case trong hệ thống.

Ví dụ: Luồng sự kiện mô tả Use case cho hệ thống rút tiền tự động như sau:

1.1 Điều kiện bắt đầu.

1.2 Luồng chính:

1.2.1 Người dùng đưa thẻ vào máy.

1.2.2. Máy hiển thông báo chào mừng và yêu cầu nhập mã số

1.2.3 Người dùng nhập mã số

1.2.4 Máy xác nhận mã số đúng. Nếu nhập sai mã số, luồng rẽ nhánh E-1 được thực hiện.

1.2.5 Máy hiện ra ba lựa chọn:

- Rút tiền: luồng con A-1
- Chuyển tiền: luồng con A-2
- Thêm tiền vào tài khoản: luồng con A-3

1.2.6 Người dùng chọn rút tiền

1.3. Luồng con:

1.3.1 Luồng con A-1:

1.3.1.1 Máy hỏi số lượng tiền cần rút

1.3.1.2 Người dùng nhập số tiền cần rút

Máy kiểm tra trong tài khoản có đủ tiền không. Nếu không đủ luồng rẽ nhánh E-2 được thực hiện

....

1.4. Luồng rẽ nhánh:

1.4.1 E-1: Người dùng nhập sai mã số

Máy thông báo là người dùng đã nhập sai mã số yêu cầu người dùng nhập lại hoặc hủy bỏ giao dịch.

1.4.2 E-2: Không đủ tiền trong tài khoản...

18.2.5.Các mối quan hệ

Quan hệ giữa Use case và Actor:

Thường gọi là quan hệ tương tác vì nó thể hiện sự tương tác giữa một actor và một Use case. Mối quan hệ này có thể là hai chiều (từ Actor đến Use case và ngược lại), nó cũng có thể chỉ là một chiều, lúc đó chiều của quan hệ sẽ chỉ ra rằng ai là người khởi tạo liên lạc (communicate). Quan hệ này thể hiện bởi một đường thẳng nối giữa actor và Use case (quan hệ hai chiều) hay một mũi tên (quan hệ một chiều).

Quan hệ giữa Use case với Use case:

Có ba loại quan hệ sau: ***uses***, ***extends*** và ***generalization***.

Quan hệ Uses (sử dụng):

Có thể có nhiều Use case có chung một số chức năng nhỏ. Khi đó nên tách chức năng đó thành một Use case riêng hơn là mô tả nó trong tất cả các Use case mà cần chức năng đó. Khi đó có một quan hệ Uses giữa các Use case trên và Use case vừa tạo ra.

Ví dụ: trong hệ thống quản lý thư viện, mọi Use case đều bắt đầu bằng việc kiểm tra định danh của người dùng. Chức năng này có thể mô tả trong một Use case tên là "Đăng nhập hệ thống", sau đó các Use case khác sẽ sử dụng Use case này khi cần thiết.

Quan hệ Extends (mở rộng):

Không giống như quan hệ Uses trong đó nói rằng khi một Use case A sử dụng Use case B có nghĩa là trong khi thực hiện Use case A phải thực hiện Use case B, quan hệ Extends dùng để chỉ:

- *Các hành vi tùy chọn*: có thể thực hiện hoặc không. Ví dụ: khi gửi email có thể thực hiện các thao tác bảo mật nội dung thư hoặc là không. Ta có Use case "Bảo mật" có quan hệ extends với Use case "Gửi email".

- *Các hành vi mà chỉ thực hiện trong một số điều kiện nhất định*. Ví dụ như: Khi thêm sách mới trong thư viện thì phải nhập các từ khóa cho nó, nếu từ khóa chưa có phải thực hiện thêm từ khóa rồi mới tiếp tục thực hiện thêm các thông tin về sách. Ta có Use case "Thêm từ khóa" có quan hệ extends Use case "Thêm sách".
- *Một số hành vi khác sẽ được thực hiện phụ thuộc vào sự lựa chọn của người dùng*.

  Ví dụ như: người dùng của hệ thống rút tiền tự động có thể chọn Rút tiền nhanh hoặc Rút tiền theo cách bình thường. Ta có Use case "Rút tiền nhanh" có quan hệ extends với Use case "Rút tiền".

Quan hệ Generalization (thừa kế):

Cũng giống như quan hệ thừa kế giữa hai lớp, quan hệ thừa kế giữa use case A và use case B nói lên rằng use case B kế thừa những đặc điểm của use case A ngoài ra nó cũng có thể có thêm những đặc trưng riêng của nó.

Ví dụ: như kiểm tra định danh người dùng có thể theo nhiều cách: Kiểm tra mã số, kiểm tra dấu vân tay...

Khi đó cả hai đều thực hiện một số hành động tương đối giống nhau của một lớp hành động gọi là "Kiểm tra định danh người dùng".

18.3. Biểu đồ use case (Use case Diagram)

18.3.1. Định nghĩa

Là biểu đồ thể hiện sự tương tác, mối quan hệ giữa các Use case và actor trong hệ thống.

18.3.2. Mô tả

Mỗi hệ thống thường có một biểu đồ Use case chính thể hiện phạm vi của hệ thống và các chức năng chính của hệ thống. Số lượng các Use case khác được tạo ra sẽ tùy thuộc vào yêu cầu. Có thể là:

- Một biểu đồ thể hiện tất cả các Use case liên quan đến một actor nào đó
- Một biểu đồ thể hiện tất cả các Use case được cài đặt trong một giai đoạn phát triển.
- Một biểu đồ thể hiện một Use case và tất cả các mối quan hệ của nó.

Tuy nhiên nên cân nhắc để các biểu đồ thể hiện đủ các thông tin cần thiết, nếu quá nhiều biểu đồ sẽ gây ra sự nhầm lẫn và mất đi lợi ích của việc đơn giản hóa. Tập hợp

các Use case giúp cho khách hàng dễ dàng xem xét ở mức tổng quát hệ thống mà ta sẽ xây dựng. Một hệ thống thông thường có từ 20 đến 50 Use case.

18.3.3. Kí hiệu

Một biểu đồ Use case bao gồm một tập các Use case và actor. Giữa Use case và actor có một đường nối nếu như actor đó khởi đầu một Use case.

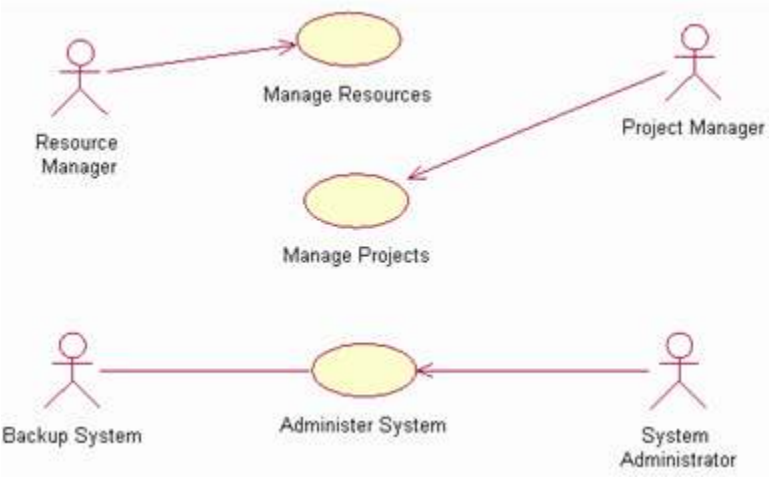
Biểu Use case có thể lồng nhau, có nghĩa là một Use case trong một biểu đồ Use case có thể được phân nhỏ ra thành những Use case khác, nằm trong một biểu đồ Use case khác.

Ví dụ:

Hệ thống quản lý dự án và nguồn nhân lực. Có bốn Actor là Resource Manager (Người quản lý nguồn nhân lực), Project Manager (Người quản lý dự án), System Administrator (Người quản trị hệ thống) và Backup System(hệ thống sao lưu dữ liệu).

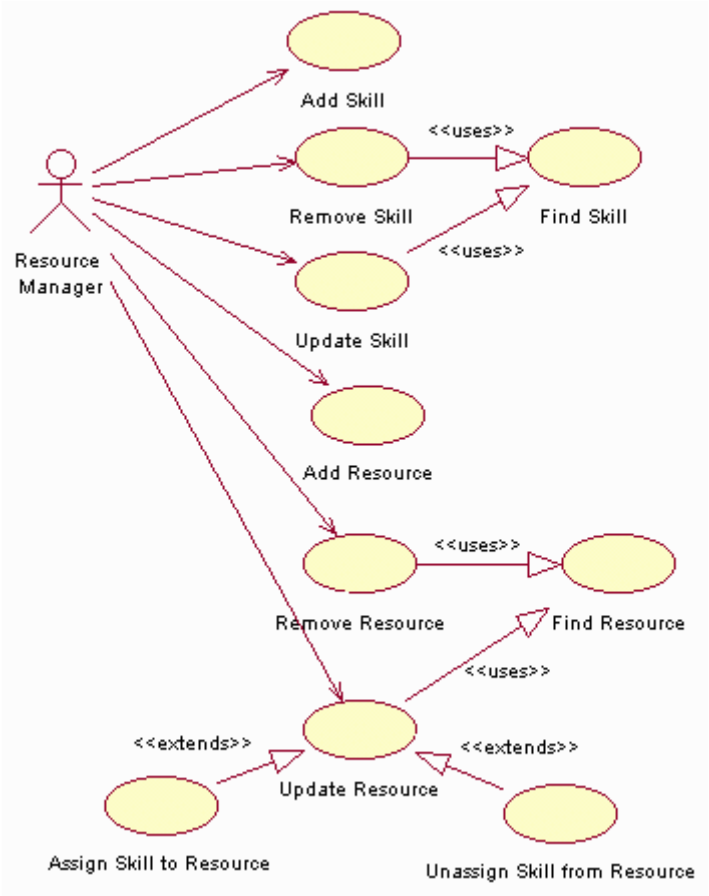
Hình 1-1 là biểu đồ use case ở mức tổng quát, cung cấp một bức tranh toàn cảnh về các actor và use case của hệ thống. Hình 1-2 chi tiết hóa use case "*Quản lý nguồn nhân lực*" bằng cách chỉ ra các use case mà actor Resource Manager mong muốn ở hệ thống. Resource Manager có thể thêm mới, sửa, xóa các thông tin về kĩ năng của nhân viên. Một kĩ năng phải được tìm ra trong cơ sở dữ liệu trước khi nó được xóa hoặc sửa nên use case FindSkill được tạo ra. Hai use case UpdateSkill và RemoveSkill đều sử dụng chức năng của use case FindSkill nên chúng có quan hệ uses với use case này.

Resource Manager cũng có thể thêm, xóa, sửa các thông tin về nhân viên. Khi cập nhật thông tin về một nhân viên, Resource Manager có thể lựa chọn: thêm kĩ năng cho một nhân viên hay xóa bỏ một kĩ năng của một nhân viên. Do đó hai use case UnassignSkill from Resource và use case AssignSkill to Resource có quan hệ extends với use case UpdateResource để chỉ ra chúng là hai khả năng lựa chọn của use case này.



Hình 18.1: biểu đồ Use case ở mức tổng quát.

Ta có thể xây dựng thêm các biểu đồ chi tiết hơn.



Hình 18.2: Biểu đồ Use case Manage Resource ở mức chi tiết hơn.

Nhìn vào biểu đồ trên ta thấy rõ tác dụng của nó trong việc trao đổi thông tin với khách hàng. Khách hàng có thể biết rõ những chức năng nào sẽ được hệ thống cung cấp. Nhìn vào các actor họ có thể biết chính xác ai sẽ tương tác với hệ thống. Việc này sẽ giúp họ tìm ra các chức năng còn thiếu. Ví dụ như: Khách hàng có thể nói rằng: " ồ không, các chức năng trên rất hay nhưng tôi còn muốn xem 10 nhân viên làm việc lâu

năm nhất trong công ty". Và như vậy các chức năng của hệ thống sẽ dễ dàng nắm bắt và đạt được sự nhất trí với khách hàng mà không phải bắt khách hàng đọc quá nhiều tài liệu kỹ thuật như trước.

18.4. Lớp (Class)

18.4.1.Định nghĩa

Lớp là định nghĩa của một tập hợp các đối tượng có chung các thuộc tính, các ứng xử và ngữ nghĩa. Như vậy lớp là một khuôn mẫu để tạo ra đối tượng. Mỗi đối tượng là một thể hiện của một lớp và một đối tượng không thể là thể hiện của nhiều hơn một lớp.

18.4.2.Mô tả

Lớp là khái niệm quan trọng nhất trong hướng đối tượng. Xây dựng được một tập hợp lớp tốt sẽ tạo nên một hệ thống tốt. Tuy nhiên việc tìm lớp khi phân tích một hệ thống không phải là việc đơn giản. Không có một phương pháp hoàn chỉnh để tìm lớp. Tuy nhiên có một cách rất hiệu quả để tìm các lớp của một hệ thống. Đó là việc tìm các lớp ***Thực thể*** (Entity), lớp ***Ngoại biên*** (Boundary) và lớp ***Điều khiển*** (Control).

18.4.3.Lớp thực thể (Entity Class)

Lớp thực thể dùng để mô hình hóa các thông tin lưu trữ lâu dài trong hệ thống. Nó thường độc lập với các đối tượng khác ở xung quanh, có nghĩa là nó không quan tâm tới việc các đối tượng xung quanh tương tác với hệ thống như thế nào. Do đó nó thường có khả năng sử dụng lại. Ví dụ như lớp Sinh viên, lớp này có thể có trong hệ thống quản lý điểm, hệ thống Đăng kí học, hệ thống quản lý thư viện... của một trường đại học.

Các danh từ, cụm danh từ mô tả về các trách nhiệm (responsibility) trong luồng sự kiện là một nơi dễ phát hiện lớp thực thể. Danh sách các danh từ ban đầu có thể được xem xét để loại bỏ ra những danh từ ở bên ngoài lĩnh vực bài toán, những danh từ trùng lặp...Các lớp thực thể thường được gọi là lớp lĩnh vực bởi vì nó thường dùng để mô tả các đối tượng, các khái niệm liên quan đến lĩnh vực của hệ thống đang xây dựng.

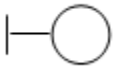
Kí hiệu:

18.4.4.Lớp biên (Boundary Class)

Dùng để nắm giữ sự tương tác giữa phần bên ngoài với phần bên trong của hệ thống. Chúng cung cấp giao diện cho một người dùng hay một hệ thống khác để tương tác

với hệ thống. Mỗi một tương tác giữa cặp Actor/ Use case đòi hỏi ít nhất là một lớp biên.

Kí hiệu:



18.4.5.Lớp điều khiển (Control Class)

Thể hiện trình tự ứng xử của hệ thống trong một hay nhiều Use case. Lớp này dùng để điều phối các hoạt động cần thực hiện để hiện thực hóa chức năng của một Use case.

Cần thận trọng trong việc sử dụng lớp Điều khiển. Nếu một lớp Điều khiển làm nhiều hơn việc điều phối các hoạt động thì nó đã được thiết kế sai với bản chất nó.

Kí hiệu:

Ngoài ra còn có cách phân loại như sau: lớp thông thường, lớp trừu tượng (abstract class), lớp tham số (parameterized class), lớp thể hiện (instantiated class), lớp tiện ích (utilities class), lớp tiện ích tham số (parameterized utilities class), lớp thể hiện tiện ích (instantiated utilities class).

18.4.6.Lớp tham số (parameterized class):

là lớp dùng để tạo ra một họ các lớp có các ứng xử có chung ý nghĩa nhưng thực hiện trên các tập dữ liệu khác nhau.

Ví dụ :

18.4.7.Lớp thể hiện (instantiated class):

khi ta gán một giá trị cụ thể cho tham số của lớp tham số, ta được một lớp thể hiện. Như ở trên ta có lớp List dùng để mô tả một danh sách và các phép toán liên quan tới danh sách như thêm một phần tử vào danh sách, xóa một phần tử khỏi danh sách, duyệt danh sách. Bây giờ ta cho một giá trị cụ thể đó là nhân viên, ta có danh sách nhân viên.

18.4.8.Lớp tiện ích (utilities class):

là một tập hợp các phép toán. Ví dụ như ta có một số hàm toán học : lấy bình phương, lấy căn... mà được dùng ở nhiều nơi trong hệ thống, khi đó các hàm này được nhóm lại và đóng kín trong một lớp gọi là lớp tiện ích. Lớp tiện ích thường được dùng để mở rộng tính năng của ngôn ngữ lập trình, lưu giữ các hàm có thể tái sử dụng cho nhiều hệ thống.

18.4.9.Lớp tiện ích tham số (parameterized utilities class):

cũng giống như lớp tiện ích, nó bao gồm một tập hợp các hàm hay dùng nhưng để chỉ một lớp tác động tổng quát chứ không chỉ rõ kiểu dữ liệu mà nó sẽ thao tác.

18.4.10.Lớp thể hiện tiện ích (instantiated utilities class):

khi cho một giá trị cụ thể cho lớp tiện ích tham số ta có một lớp thể hiện tiện ích. Ví dụ

18.4.11.Lớp trừu tượng (abstract class):

là lớp được thiết kế ở mức độ trừu tượng cao nhất, nó chứa những thuộc tính, những hành vi chung cho nhiều lớp con khác. Lớp trừu tượng được tạo ra chỉ để cho các lớp khác kế thừa nó, những phương thức khai báo trong lớp trừu tượng không được cài đặt mà chúng chỉ được cài đặt ở các lớp con. Cho nên không có một đối tượng nào được tạo ra từ lớp trừu tượng.

18.5. Phân bổ trách nhiệm giữa các lớp

Mô hình là một tập hợp của rất nhiều lớp, chúng ta cần đảm bảo rằng có một sự phân bổ trách nhiệm tương đối công bằng giữa các lớp. Điều đó có nghĩa là không có lớp nào quá lớn hoặc quá nhỏ. Mỗi lớp cần phải làm tốt một công việc. Nếu có nhiều lớp quá lớn, chúng ta sẽ thấy rằng mô hình rất khó thay đổi và sử dụng lại. Nếu có nhiều lớp quá nhỏ, chúng ta sẽ khó có khả năng kiểm soát và hiểu hết ý nghĩa của chúng.

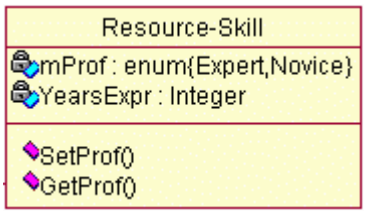
Để giải quyết vấn đề này, chúng ta nên thực hiện các bước sau:

- Xác định một tập hợp các lớp mà công việc tương đối liên quan với nhau để thực hiện một số ứng xử nào đó.
- Xác định một tập hợp các trách nhiệm cho mỗi lớp.
- Xem xét từng lớp một, nếu lớp nào quá lớn thì tách nó ra thành những lớp nhỏ hơn, tập hợp những lớp nhỏ thành một lớp lớn hơn và phân phối trách nhiệm một cách hợp lý giữa các lớp.
- Cân nhắc cách thức mà những lớp này hợp tác với những lớp khác, phân phối lại các trách nhiệm nếu thấy cần thiết. Công việc này thực hiện lặp đi, lặp lại cho tới lúc cảm thấy tương đối phù hợp, nó phụ thuộc nhiều vào kinh nghiệm thực tế.

Mô tả lớp

Trong quá trình phân tích, có nhiều lớp được tạo ra, do đó cần có một mô tả cho mỗi lớp để hiểu rõ mục đích của lớp là để làm gì, tránh sự nhầm lẫn. Mô tả lớp cần chỉ ra mục đích của lớp chứ không phải cấu trúc của lớp.

Kí hiệu:



Được thể hiện bởi một hình chữ nhật, có các phần ngăn cách giữa tên, thuộc tính, phương thức của lớp.

Ví dụ:

Lớp "Người đọc": Lớp này chứa các thông tin cần thiết về người đọc, phục vụ cho việc mượn sách. Người đọc là người đã đăng kí với thư viện và mượn sách của thư viện.

Một mô tả tồi sẽ như sau:

Lớp "Người đọc": Lớp này gồm có tên người đọc, địa chỉ...

Gói (Packages)

Nếu hệ thống chỉ có một vài lớp thì ta có thể dễ dàng quản lý chúng. Tuy nhiên hầu hết các hệ thống đều có khá nhiều lớp và do đó ta cần có một cơ chế để nhóm chúng lại cho dễ sử dụng, quản lý và sử dụng lại. Một gói(package) là một tập hợp các lớp hay các gói có liên quan với nhau. Qua việc nhóm lớp lại theo gói, ta có thể nhìn mô hình ở mức tổng quát hơn và khi cần ta có thể xem chi tiết các lớp trong một gói.

Trong UML một gói kí hiệu như sau:

18.6. Biểu đồ lớp (Class Diagram)

Khi có nhiều lớp thêm vào mô hình, biểu đồ lớp được tạo ra để cung cấp một bức tranh mô tả một số hoặc tất cả các lớp trong mô hình. Thường có một biểu đồ chính thể hiện các gói trong mô hình. Mỗi gói lại có một biểu đồ chính của gói để mô tả các lớp trong gói và mối quan hệ giữa chúng. Số lượng biểu đồ lớp là tuỳ ý.

Thông thường có một số cách dùng như sau:

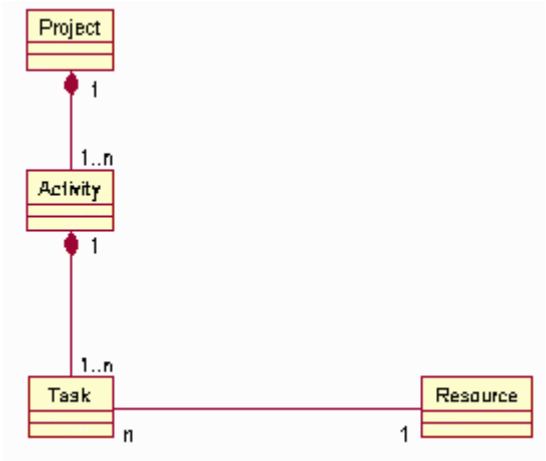
- Thể hiện cấu trúc và ứng xử của một hay nhiều lớp.

- Thể hiện mối quan hệ thừa kế giữa các lớp.

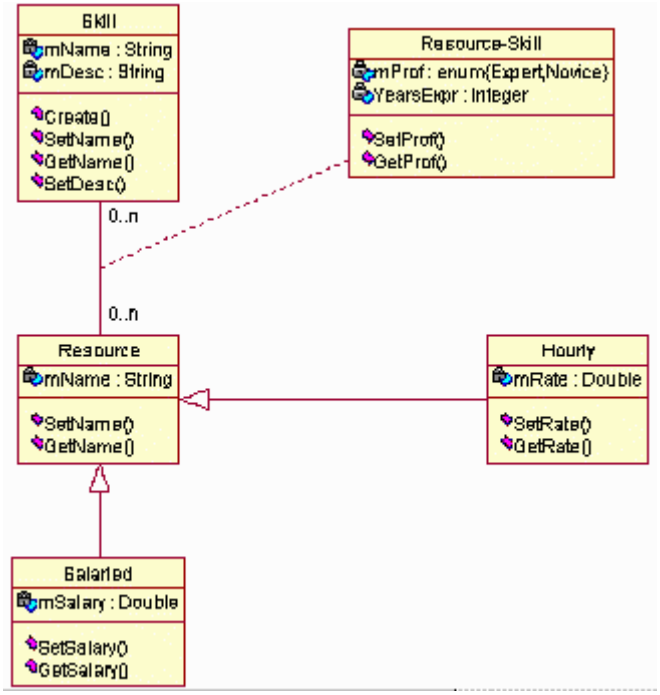
Biểu đồ lớp là một công cụ hữu hiệu trong việc thiết kế. Nó giúp cho lập trình viên xem xét và lên thiết kế về cấu trúc của hệ thống trước khi viết mã lệnh.

Ví dụ:

Một dự án có nhiều hoạt động (activity) và một hoạt động có nhiều nhiệm vụ(task). Quan hệ gộp (composition) giữa dự án và hoạt động chỉ ra rằng các hoạt động phải gắn với một dự án, nếu dự án bị hủy bỏ thì các hoạt động cũng bị hủy bỏ.



Hình 18.3.Biểu đồ lớp ở dạng tổng quát.



Hình 18.4.*Biểu đồ lớp ở mức chi tiết*

Những người phát triển sử dụng biểu đồ lớp để xây dựng các lớp. Một số công cụ CASE sẽ giúp tạo ra mã khung cho các lớp và người phát triển sẽ chi tiết hóa bằng ngôn ngữ lập trình mà họ chọn. Phân tích viên sẽ dùng biểu đồ lớp để xem hệ thống ở

mức chi tiết. Các kiến trúc sư hệ thống sẽ xem thiết kế của hệ thống. Nếu có một lớp có quá nhiều chức năng, họ có thể cân nhắc để tách lớp đó ra thành các lớp con.

CHƯƠNG 19: MÔ HÌNH ĐỘNG

19.1. Sự cần thiết có mô hình động (Dynamic model)

Mô hình đối tượng và quá trình phát triển nó là trọng tâm của những cuộc thảo luận trong chương trước. Mô hình đối tượng định nghĩa hệ thống theo khái niệm các thành phần tĩnh. Mô hình đối tượng miêu tả ứng xử mang tính cấu trúc và chức năng của các lớp. Mặc dầu vậy, để mô hình hóa sự hoạt động thật sự của một hệ thống và trình bày một hướng nhìn đối với hệ thống trong thời gian hệ thống hoạt động, chúng ta cần tới **mô hình động (dynamic model)**.

Trong UML, mô hình động đề cập tới các trạng thái khác nhau trong vòng đời của một đối tượng thuộc hệ thống. Phương thức ứng xử của một hệ thống tại một thời điểm cụ thể sẽ được miêu tả bằng các điều kiện khác nhau ấn định cho sự hoạt động của nó.

Một yếu tố hết sức quan trọng là cần phải hiểu cho được hệ thống sẽ đáp lại những kích thích từ phía bên ngoài ra sao, có nghĩa là chúng ta cần phải xác định và nghiên cứu những chuỗi các thủ tục sẽ là hệ quả của một sự kích thích từ ngoài. Cho việc này, ta cần tới mô hình động bởi trọng tâm của mô hình này là lối ứng xử phụ thuộc vào thời gian của các đối tượng trong hệ thống.

Chúng ta cần tới mô hình động bởi chúng ta cần thể hiện sự thay đổi xảy ra trong hệ thống dọc theo thời gian chạy. Công cụ miêu tả mô hình động là không thể thiếu ví dụ trong trường hợp các đối tượng trải qua nhiều giai đoạn khác nhau trong thời gian hệ thống hoạt động. Điều đó có nghĩa là mặc dù đối tượng được tạo ra một lần, nhưng các thuộc tính của chúng chỉ dần dần từng bước nhận được giá trị. Ví dụ như một tài khoản đầu tư có kỳ hạn được tạo ra, nhưng tổng số tiền lãi cộng dồn của nó chỉ được tăng lên dần dần theo thời gian.

Các mô hình động cũng là yếu tố hết sức cần thiết để miêu tả ứng xử của một đối tượng khi đưa ra các yêu cầu hoặc thực thi các tác vụ. Cả tác vụ lẫn dịch vụ, theo định nghĩa, đều là các hoạt động động và vì thế mà chỉ có thể được biểu diễn qua một mô hình động.

19.2. Các thành phần của mô hình động

Đối tượng trong các hệ thống giao tiếp với nhau, chúng gửi thông điệp (message) đến nhau. Ví dụ một đối tượng khách hàng là John gửi một thông điệp mua hàng đến người bán hàng là Bill để làm một việc gì đó. Một thông điệp thường là một lệnh gọi thủ tục mà một đối tượng này gọi qua một đối tượng kia. Các đối tượng giao tiếp với nhau ra sao và hiệu ứng của sự giao tiếp như thế được gọi là **khía cạnh động** của một hệ thống, ý nghĩa của khái niệm này là câu hỏi: các đối tượng cộng tác với nhau qua

giao tiếp như thế nào và các đối tượng trong một hệ thống thay đổi trạng thái ra sao trong thời gian hệ thống hoạt động. Sự giao tiếp trong một nhóm các đối tượng nhằm tạo ra một số các lệnh gọi hàm được gọi là **tương tác** (**interaction**), tương tác có thể được thể hiện qua ba loại biểu đồ: biểu đồ tuần tự (sequence Diagram), biểu đồ cộng tác (collaboration Diagram) và biểu đồ hoạt động (activity Diagram).

Trong chương này, chúng ta sẽ đề cập tới bốn loại biểu đồ động của UML:

- **Biểu đồ trạng thái:** miêu tả một đối tượng có thể có những trạng thái nào trong vòng đời của nó, ứng xử trong các trạng thái đó cũng như các sự kiện nào gây ra sự chuyển đổi trạng thái, ví dụ, một tờ hóa đơn có thể được trả tiền (trạng thái đã trả tiền) hoặc là chưa được trả tiền (trạng thái chưa trả tiền).

- **Biểu đồ tuần tự:** miêu tả các đối tượng tương tác và giao tiếp với nhau ra sao. Tiêu điểm trong các biểu đồ tuần tự là thời gian. Các biểu đồ tuần tự chỉ ra chuỗi của các thông điệp được gửi và nhận giữa một nhóm các đối tượng, nhằm mục đích thực hiện một số chức năng.

- **Biểu đồ cộng tác:** cũng miêu tả các đối tượng tương tác với nhau ra sao, nhưng trọng điểm trong một biểu đồ cộng tác là sự kiện. Tập trung vào sự kiện có nghĩa là chú ý đặc biệt đến mối quan hệ (nối kết) giữa các đối tượng, và vì thế mà phải thể hiện chúng một cách rõ ràng trong biểu đồ.

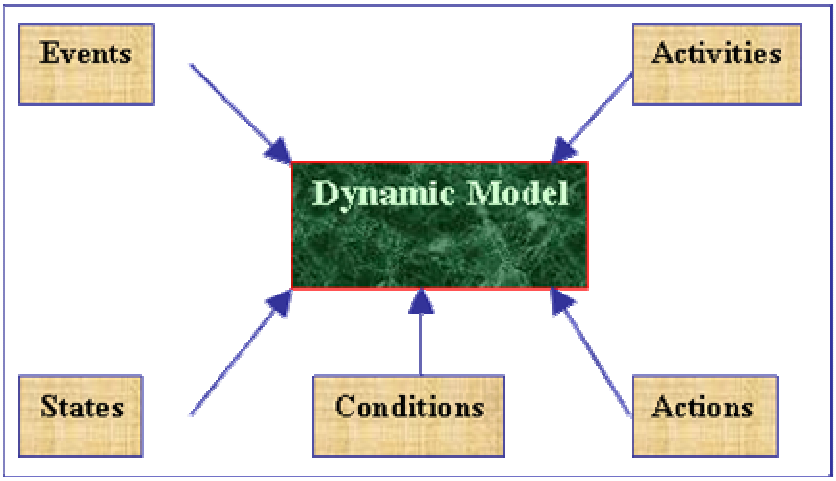
- **Biểu đồ hoạt động:** là một con đường khác để chỉ ra tương tác, nhưng chúng tập trung vào công việc. Khi các đối tượng tương tác với nhau, các đối tượng cũng thực hiện các tác vụ, tức là các hoạt động. Những hoạt động này cùng thứ tự của chúng được miêu tả trong biểu đồ hoạt động.

Vì biểu đồ tuần tự, biểu đồ cộng tác lẫn biểu đồ hoạt động đều chỉ ra tương tác nên thường bạn sẽ phải chọn nên sử dụng biểu đồ nào khi lập tài liệu cho một tương tác. Quyết định của bạn sẽ phụ thuộc vào việc khía cạnh nào được coi là quan trọng nhất.

Ngoài cấu trúc tĩnh và ứng xử động, hướng nhìn chức năng cũng có thể được sử dụng để miêu tả hệ thống. Hướng nhìn chức năng thể hiện các chức năng mà hệ thống sẽ cung cấp. Trường hợp sử dụng chính là các lời miêu tả hệ thống theo chức năng; chúng miêu tả các tác nhân có thể sử dụng hệ thống ra sao. Như đã đề cập từ trước, trường hợp sử dụng bình thường ra được mô hình hóa trong những giai đoạn đầu tiên của quá trình phân tích, nhằm mục đích miêu tả xem tác nhân có thể muốn sử dụng hệ thống như thế nào. Mô hình trường hợp sử dụng chỉ nên nắm bắt duy nhất khía cạnh tác nhân sử dụng hệ thống, không nên đề cập khía cạnh hệ thống được xây dựng bên trong ra sao. Lớp và các tương tác trong hệ thống thực hiện trường hợp sử dụng. Tương tác được miêu tả bởi các biểu đồ tuần tự, biểu đồ cộng tác và hoặc/và biểu đồ

hoạt động, tức là có một sự nối kết giữa hướng nhìn chức năng và hướng nhìn động của hệ thống. Các lớp được sử dụng trong việc thực thi các trường hợp sử dụng được mô hình hóa và miêu tả qua các biểu đồ lớp và biểu đồ trạng thái (một biểu đồ trạng thái sẽ được đính kèm cho một lớp, một hệ thống con hoặc là một hệ thống). Trường hợp sử dụng và các mối quan hệ của chúng đến tương tác đã được miêu tả trong chương 3 (trường hợp sử dụng).

Nhìn chung, một mô hình động miêu tả năm khía cạnh căn bản khác nhau:



Hình 19.1.Các thành phần của mô hình động

Các thành phần kể trên sẽ được đề cập chi tiết hơn trong các phần sau.

Ngoài ra, một mô hình động cũng còn được sử dụng để xác định các nguyên tắc chuyên ngành (business rule) cần phải được áp dụng trong mô hình. Nó cũng được sử dụng để ấn định xem các nguyên tắc đó được đưa vào những vị trí nào trong mô hình.

Một vài ví dụ cho những nguyên tắc chuyên ngành cần phải được thể hiện trong mô hình động:

- Một khách hàng không được quyền rút tiền ra nếu không có đủ mức tiền trong tài khoản.

- Những món tiền đầu tư có kỳ hạn không thể chuyển sang một tên khác trước khi đáo hạn.

- Giới hạn cao nhất trong một lần rút tiền ra bằng thẻ ATM là 500 USD.

19.3. Ưu điểm của mô hình động:

Bất cứ khi nào có những ứng xử động cần phải được nghiên cứu hoặc thể hiện, chúng ta sẽ phải dùng đến mô hình động.

Mô hình động đóng một vai trò vô cùng quan trọng trong những trường hợp như:

- Các hệ thống mang tính tương tác cao

- Hệ thống có sử dụng các trang thiết bị ngoại vi có thể gọi nên các ứng xử của hệ thống.

Mô hình động không tỏ ra thật sự hữu hiệu trong trường hợp của các hệ thống tĩnh. Ví dụ một hệ thống chỉ nhằm mục đích nhập dữ liệu để lưu trữ vào một ngân hàng dữ liệu.

Một mô hình động tập trung vào các chuỗi tương tác (biểu đồ cộng tác) và vào yếu tố thời gian của các sự kiện (biểu đồ tuần tự). Một mô hình động có thể được sử dụng cho mục đích thể hiện rõ ràng theo thời gian hoạt động của hệ thống nếu trong thời gian này có những đối tượng:

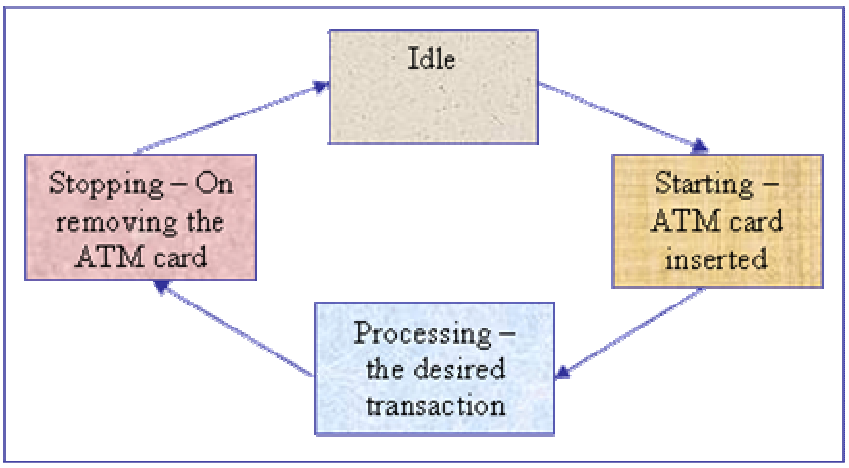
- Được tạo ra
- Bị xóa đi
- Được lưu trữ
- Bị hủy

Hãy quan sát trường hợp rút tiền mặt và tương tác của khách hàng đối với nhà băng:

- Khách hàng điền tất cả các chi tiết cần thiết vào giấy yêu cầu rút tiền mặt.
- Khách hàng đưa giấy yêu cầu đó cho một nhân viên phát thẻ xếp hàng.
- Nhân viên phát thẻ ghi số của giấy yêu cầu rút tiền vào danh sách.
- Động tác ghi số của giấy yêu cầu rút tiền được thực hiện tuần tự, tương ứng với những số thẻ tuần tự được phát ra.
- Một tấm thẻ xếp hàng (token) được trao cho khách hàng.
- Khách hàng đi vào hàng xếp, chờ nhân viên bên casse gọi đúng số thẻ của mình.
- Song song với quá trình chờ của khách hàng, giấy yêu cầu rút tiền của anh ta trải qua nhiều giai đoạn trong nội bộ nhà băng.
- Chữ ký của khách hàng trên giấy yêu cầu rút tiền được thẩm tra.
- Giấy yêu cầu được xem xét về phương diện số tài khoản và mức tiền trong tài khoản.
- Nếu một trong hai điều kiện trên không được thỏa mãn, quá trình rút tiền mặt sẽ bị chặn lại hoặc là được sửa đổi và tiếp tục.
- Khi cả hai điều kiện nêu trên được thỏa mãn, giấy yêu cầu rút tiền mặt sẽ được đưa đến cho nhân viên ngồi bên casse, nơi khách hàng sẽ được gọi tới tuần tự dựa theo số thẻ xếp hàng.
- Nhân viên bên casse đưa tiền mặt cho khách hàng.

Lối ứng xử trong việc rút tiền mặt là mang tính động. Suốt quá trình rút tiền mặt, tương tác và trình tự của quá trình phụ thuộc vào một số các điều kiện xác định. Loại ứng xử này không thể được thể hiện qua mô hình đối tượng, đây là trường hợp ta cần đến mô hình động.

Mô hình động cũng tỏ ra hữu dụng trong trường hợp có những trang thiết bị trải qua tuần tự các bước trong một vòng lặp và tiến trình phụ thuộc vào một số điều kiện nhất định. Ví dụ một đối tượng mô hình hóa lối ứng xử của một máy rút tiền mặt tự động (ATM). Máy ATM lần lượt đi qua các bước của một vòng lặp mang tính thủ tục (chức năng), bắt đầu từ việc một thẻ ATM được đút vào trong máy, xử lý các yêu cầu do khách hàng đưa ra, dừng lại và chờ yêu cầu giao dịch khác, rồi sau đó quay trở lại trạng thái ban đầu (đứng yên) sau khi thẻ ATM đã được rút ra ngoài.



Hình 19.2. Mô hình động của máy rút tiền ATM

19.4. Sự kiện và thông điệp (Event & Message)

19.4.1.Sự kiện (Event):

Một trong những thành phần quan trọng bậc nhất của một đối tượng là sự kiện. Một sự kiện là một sư kích thích được gửi từ đối tượng này sang đối tượng khác.

Một sự kiện là một việc sẽ xảy ra và có thể gây ra một hành động nào đó. Ví dụ như khi bạn bấm lên nút *Play* trên máy CD-Player, nó sẽ bắt đầu chơi nhạc (giả sử rằng CD-Player có điện, trong máy có đĩa CD và nói chung là dàn CD-Player hoạt động tốt). Sự kiện ở đây là bạn nhấn lên nút *Play*, và hành động ở đây là bắt đầu chơi nhạc. Nếu có một sự nối kết được định nghĩa rõ ràng giữa sự kiện và hành động, người ta gọi nó là **quan hệ nhân quả** (**Causality**). Trong công nghệ phần mềm, chúng ta thường chỉ mô hình hóa các hệ thống mang tính nhân quả, nơi sự kiện và hành động được nối kết với nhau. Một phản ví dụ của quan hệ nhân quả: bạn lái xe trên xa lộ với tốc độ quá nhanh, cảnh sát ngăn xe lại. Đây không phải là nhân quả bởi hành động ngăn bạn lại của cảnh sát không chắc chắn bao giờ cũng xảy ra; vì thế mà không có

một sự nối kết được định nghĩa rõ ràng giữa sự kiện (lái xe quá nhanh) và hành động (ngăn xe). Trong mô hình hóa, vậy là ta quan tâm đến sự kiện theo nghĩa là bất kỳ hành động nào khiến hệ thống phản ứng theo một cách nào đó.

Quan sát ví dụ một nhà băng lẻ, ta có một vài ví dụ về sự kiện như sau:

- Điền một tờ giấy yêu cầu rút tiền.
- Sự đáo hạn một tài khoản đầu tư có kỳ hạn.
- Kết thúc một hợp đồng trước kỳ hạn.
- Điền một giấy yêu cầu mở tài khoản.

UML biết đến tất cả bốn loại sự kiện:

- Một điều kiện trở thành được thỏa mãn (trở thành đúng)
- Nhận được một tín hiệu ngoại từ một đối tượng khác
- Nhận được một lời gọi thủ tục từ một đối tượng khác (hay từ chính đối tượng đó).
- Một khoảng thời gian xác định trước trôi qua.

Xin chú ý rằng cả các lỗi xảy ra cũng là sự kiện và có thể mang tính hữu dụng rất lớn đối với mô hình.

19.4.2. Sự kiện độc lập và sự kiện phụ thuộc

Các sự kiện có thể mang tính độc lập hay liên quan đến nhau. Có một số sự kiện, theo bản chất, phải đi trước hoặc là xảy ra sau các sự kiện khác. Ví dụ:

- Điền các chi tiết trong một tờ yêu cầu rút tiền mặt sẽ dẫn tới việc nhận được một số thẻ xếp hàng.
- Sự đáo hạn của một tài khoản đầu tư có kỳ hạn sẽ dẫn đến động tác gia hạn hoặc rút tiền mặt.
- Điền các chi tiết trong một giấy yêu cầu mở tài khoản sẽ dẫn tới việc phải nộp một khoản tiền tối thiểu (theo quy định) vào tài khoản.

Các sự kiện độc lập là những sự kiện không được nối kết với nhau trong bất kỳ một phương diện nào. Ví dụ:

- Rút tiền mặt và đưa tiền vào tài khoản là các sự kiện độc lập với nhau.
- Mở một tài khoản đầu tư có kỳ hạn và mở một tài khoản giao dịch là độc lập với nhau.
- Kết thúc trước kỳ hạn một tài khoản đầu tư và việc mở một tài khoản đầu tư có kỳ hạn khác là độc lập với nhau.

Các sự kiện độc lập còn có thể được gọi là các sự kiện song song hay đồng thời. Bởi chúng không phụ thuộc vào nhau, nên các sự kiện này có thể xảy ra tại cùng một thời điểm.

Trong nhiều trường hợp, một sự kiện riêng lẻ trong phạm vi vấn đề sẽ được chuyển tải thành nhiều sự kiện trong hệ thống. Ví dụ: đưa giấy yêu cầu rút tiền mặt cho nhân viên phát thẻ xếp hàng sẽ có kết quả là một loạt các sự kiện nối tiếp.

Có những tình huống nơi một sự kiện riêng lẻ sẽ được nhận bởi nhiều đối tượng khác nhau và khiến cho chúng phản ứng thích hợp. Ví dụ như một lời đề nghị ngăn một tờ séc có thể đồng thời được gửi đến cho nhân viên thu ngân và nhân viên kiểm tra séc.

19.4.3.Sự kiện nội (internal) và sự kiện ngoại (external):

Sự kiện nội là các sự kiện xảy ra trong nội bộ hệ thống. Đây là các sự kiện do một đối tượng này gây ra đối với đối tượng khác. Ví dụ, tính toán tiền lãi cho một tài khoản đầu tư có kỳ hạn sẽ được nội bộ hệ thống thực hiện, tuân theo một đối tượng quan sát ngày tháng.

Sự kiện ngoại là những sự kiện được kích nên từ phía bên ngoài biên giới của hệ thống, ví dụ như sự kết thúc trước kỳ hạn một tài khoản đầu tư.

19.4.4.Sự kiện và lớp sự kiện:

Lớp sự kiện đối với sự kiện cũng như lớp đối với đối tượng bình thường. Lời định nghĩa xác định một loại sự kiện được gọi là một lớp sự kiện.

Lớp sự kiện ngoài ra còn có thể được phân loại:

- *Các tín hiệu đơn giản*: Lớp sự kiện trong trường hợp này sẽ được thực thể hóa để chỉ ra một sự kiện hoặc là một tín hiệu của một sự kiện.

- *Các sự kiện chuyển tải dữ liệu*: thường thì một sự kiện có khả năng và chuyển tải dữ liệu. Tất cả các sự kiện cần phải "biết đến" các đối tượng sẽ nhận được sự kiện này. Thông tin về người nhận sự kiện được gọi là thông tin nhận diện. Nói một cách khác, yếu tố nhận diện xác định các đối tượng sẽ nhận sự kiện. Bên cạnh đó, còn có thể có các dữ liệu bổ sung thuộc về các đối tượng khác, không nhất thiết phải là đối tượng gửi hay nhận sự kiện.

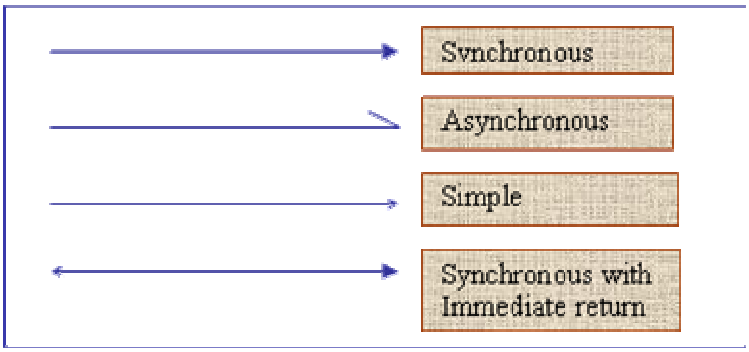
Về mặt nguyên tắc, các sự kiện thuộc dạng phát tin (Broadcast) sẽ được truyền đến cho tất cả các đối tượng. Nếu sự kiện này là không quan trọng đối với đối tượng nào đó trong trạng thái hiện thời của nó thì đối tượng sẽ bỏ qua sự kiện.

19.5. Thông điệp (Message):

Trong lập trình hướng đối tượng, một tương tác giữa hai đối tượng được thực thi dưới dạng thông điệp được gửi từ đối tượng này sang đối tượng khác. Trong ngữ cảnh này,

yếu tố quan trọng là không nên hiểu danh từ "thông điệp" quá chính xác theo nghĩa văn học bình thường. Một thông điệp ở đây thường được thực hiện qua một lệnh gọi thủ tục đơn giản (một đối tượng này gọi một thủ tục của một đối tượng khác); khi thủ tục đã được thực hiện xong, quyền điều khiển được trao trở về cho đối tượng gọi thủ tục cùng với giá trị trả về. Một thông điệp mặt khác cũng có thể là một thông điệp thực thụ được gửi qua một số cơ chế giao tiếp nào đó, hoặc là qua mạng hoặc là nội bộ trong một máy tính, đây là điều thường xảy ra trong các hệ thống thời gian thực. Thông điệp được thể hiện trong tất cả các loại biểu đồ động (tuần tự, cộng tác, hoạt động và trạng thái) theo ý nghĩa là sự giao tiếp giữa các đối tượng. Một thông điệp được vẽ là một được thẳng với mũi tên nối giữa đối tượng gửi và đối tượng nhận thông điệp. Loại mũi tên sẽ chỉ rõ loại thông điệp.

Hình dưới chỉ rõ các loại thông điệp được sử dụng trong UML.



Hình 19.3.Các ký hiệu của các kiểu thông điệp

***- Thông điệp đơn giản (simple)*:** Chỉ miêu tả đơn giản chiều điều khiển. Nó chỉ ra quyền điều khiển được trao từ đối tượng này sang cho đối tượng khác mà không kèm thêm lời miêu tả bất kỳ một chi tiết nào về sự giao tiếp đó. Loại thông điệp này được sử dụng khi người ta không biết các chi tiết về giao tiếp hoặc coi chúng là không quan trọng đối với biểu đồ.

***- Thông điệp đồng bộ* (*synchronous*):** thường được thực thi là một lệnh gọi thủ tục. Thủ tục xử lý thông điệp này phải được hoàn tất (bao gồm bất kỳ những thông điệp nào được lồng vào trong, được gửi như là một thành phần của sự xử lý) trước khi đối tượng gọi tiếp tục thực thi. Quá trình trở về có thể được chỉ ra dưới dạng thông điệp đơn giản.

***- Thông điệp không đồng bộ* (*asynchronous*):** đây là dạng điều khiển trình tự không đồng bộ, nơi không có một sự trở về đối với đối tượng gọi và nơi đối tượng gửi thông điệp tiếp tục quá trình thực thi của mình sau khi đã gửi thông điệp đi, không chờ cho tới khi nó được xử lý xong. Loại thông điệp này thường được sử dụng trong các hệ thống thời gian thực, nơi các đối tượng thực thi đồng thời.

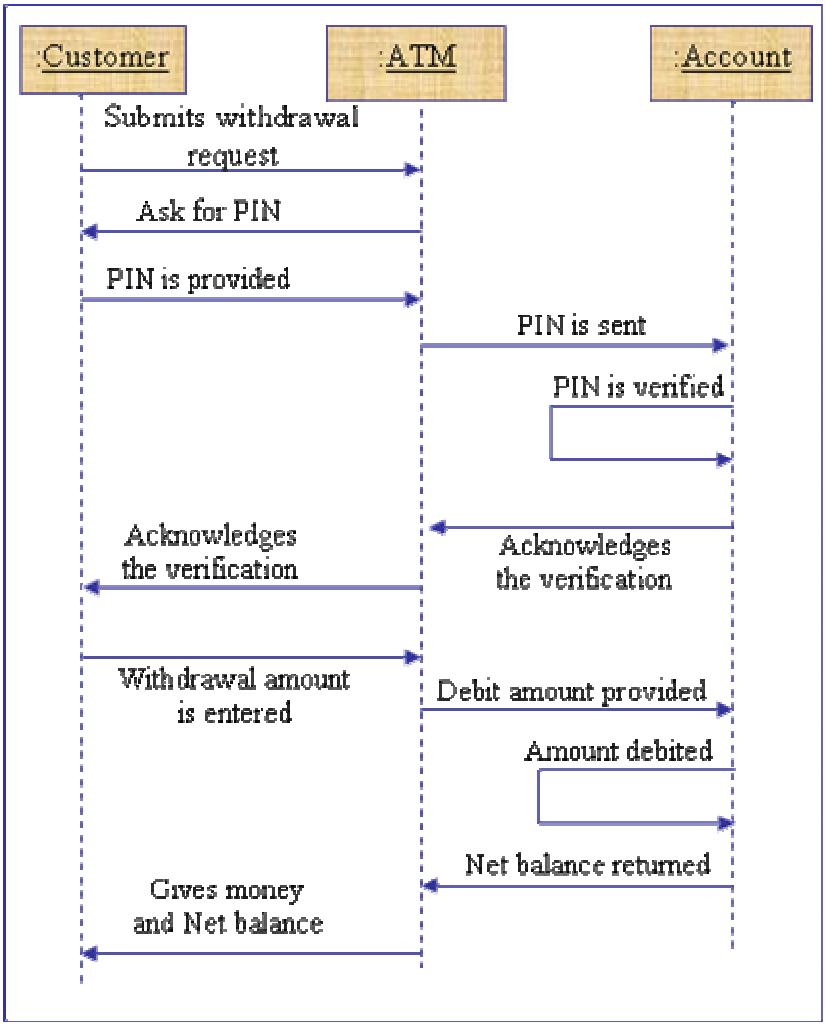
Thông điệp đơn giản và thông điệp đồng bộ có thể được kết hợp với nhau trong chỉ một đường thẳng chỉ thông điệp với mũi tên chỉ thông điệp đồng bộ ở một phía và mũi tên chỉ thông điệp đơn giản ở phía kia. Điều này chỉ rõ rằng sự trả về được xảy ra hầu như ngay lập tức sau lệnh gọi hàm.

19.6. Biểu đồ tuần tự (Sequence diagram)

Biểu đồ tuần tự minh họa các đối tượng tương tác với nhau ra sao. Chúng tập trung vào các chuỗi thông điệp, có nghĩa là các thông điệp được gửi và nhận giữa một loạt các đối tượng như thế nào. Biểu đồ tuần tự có hai trục: trục nằm dọc chỉ thời gian, trục nằm ngang chỉ ra một tập hợp các đối tượng. Một biểu đồ tuần tự cũng nêu bật sự tương tác trong một cảnh kịch (scenario) – một sự tương tác sẽ xảy ra tại một thời điểm nào đó trong quá trình thực thi của hệ thống.

Từ các hình chữ nhật biểu diễn đối tượng có các đường gạch rời (dashed line) thẳng đứng biểu thị đường đời đối tượng, tức là sự tồn tại của đối tượng trong chuỗi tương tác. Trong khoảng thời gian này, đối tượng được thực thể hóa, sẵn sàng để gửi và nhận thông điệp. Quá trình giao tiếp giữa các đối tượng được thể hiện bằng các đường thẳng thông điệp nằm ngang nối các đường đời đối tượng. Mỗi tên ở đầu đường thẳng sẽ chỉ ra loại thông điệp này mang tính đồng bộ, không đồng bộ hay đơn giản. Để đọc biểu đồ tuần tự, hãy bắt đầu từ phía bên trên của biểu đồ rồi chạy dọc xuống và quan sát sự trao đổi thông điệp giữa các đối tượng xảy ra dọc theo tiến trình thời gian.

Ví dụ hãy quan sát một cảnh kịch rút tiền mặt tại một máy ATM của một nhà băng lẻ:



Hình 19.4.Biểu đồ cảnh kịch rút tiền mặt tại máy ATM

Biểu đồ trên có thể được diễn giải theo trình tự thời gian như sau:

- Có ba lớp tham gia cảnh kịch này: khách hàng, máy ATM và tài khoản.
- Khách hàng đưa yêu cầu rút tiền vào máy ATM
- Đối tượng máy ATM yêu cầu khách hàng cung cấp mã số
- Mã số được gửi cho hệ thống để kiểm tra tài khoản
- Đối tượng tài khoản kiểm tra mã số và báo kết quả kiểm tra đến cho ATM
- ATM gửi kết quả kiểm tra này đến khách hàng
- Khách hàng nhập số tiền cần rút.
- ATM gửi số tiền cần rút đến cho tài khoản
- Đối tượng tài khoản trừ số tiền đó vào mức tiền trong tài khoản. Tại thời điểm này, chúng ta thấy có một mũi tên quay trở lại chỉ vào đối tượng tài khoản. Ý nghĩa của nó là đối tượng tài khoản xử lý yêu cầu này trong nội bộ đối tượng và không gửi sự kiện đó ra ngoài.

- Đối tượng tài khoản trả về mức tiền mới trong tài khoản cho máy ATM.

- Đối tượng ATM trả về mức tiền mới trong tài khoản cho khách hàng và dĩ nhiên, cả lượng tiền khách hàng đã yêu cầu được rút.

Đối tượng tài khoản chỉ bắt đầu được sinh ra khi đối tượng ATM cần tới nó để kiểm tra mã số và đối tượng tài khoản tiếp tục sống cho tới khi giao dịch được hoàn tất. Sau đó, nó chết đi. Bởi khách hàng có thể muốn tiếp tục thực hiện các giao dịch khác nên đối tượng khách hàng và đối tượng máy ATM vẫn tiếp tục tồn tại, điều này được chỉ ra qua việc các đường đời đối tượng được kéo vượt quá đường thẳng thể hiện sự kiện cuối cùng trong chuỗi tương tác.

Loại tương tác này là rất hữu dụng trong một hệ thống có một số lượng nhỏ các đối tượng với một số lượng lớn các sự kiện xảy ra giữa chúng. Mặc dù vậy, khi số lượng các đối tượng trong một hệ thống tăng lên thì mô hình này sẽ không còn mấy thích hợp.

Để có thể vẽ biểu đồ tuần tự, đầu tiên hãy xác định các đối tượng liên quan và thể hiện các sự kiện xảy ra giữa chúng.

Khi vẽ biểu đồ tuần tự, cần chú ý:

- Sự kiện được biểu diễn bằng các đường thẳng nằm ngang.

- Đối tượng bằng các đường nằm dọc.

- Thời gian được thể hiện bằng đường thẳng nằm dọc bắt đầu từ trên biểu đồ. Điều đó có nghĩa là các sự kiện cần phải được thể hiện theo đúng thứ tự mà chúng xảy ra, vẽ từ trên xuống dưới.

19.7. Biểu đồ cộng tác (Collaboration Diagram)

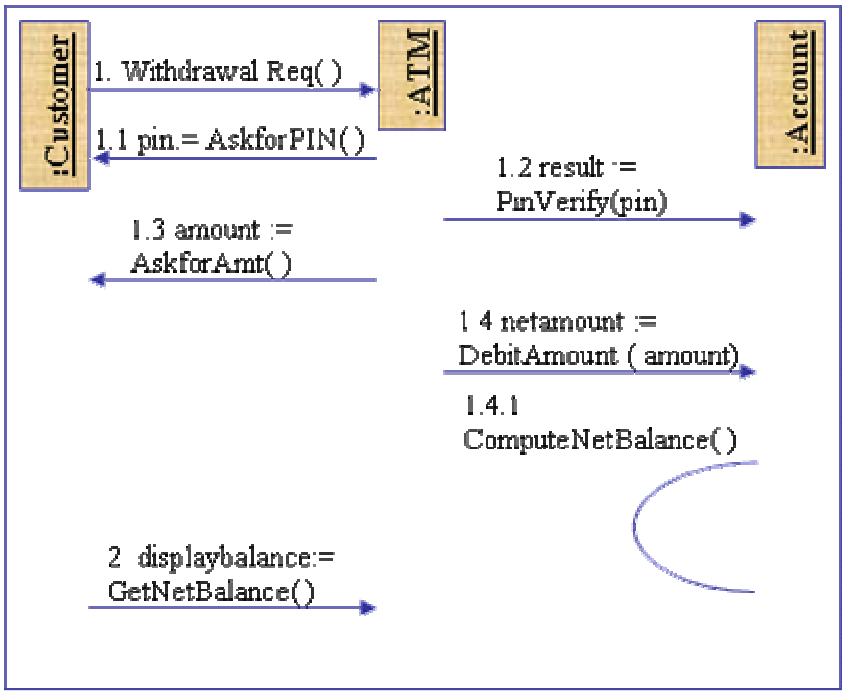
Một biểu đồ cộng tác miêu tả tương tác giữa các đối tượng cũng giống như biểu đồ tuần tự, nhưng nó tập trung trước hết vào các sự kiện, tức là tập trung chủ yếu vào sự tương tác giữa các đối tượng.

Trong một biểu đồ cộng tác, các đối tượng được biểu diễn bằng kí hiệu lớp. Thứ tự trong biểu đồ cộng tác được thể hiện bằng cách đánh số các thông điệp. Kỹ thuật đánh số được coi là hơi có phần khó hiểu hơn so với kỹ thuật mũi tên sử dụng trong biểu đồ tuần tự. Nhưng ưu điểm của biểu đồ cộng tác là nó có thể chỉ ra các chi tiết về các lệnh gọi hàm (thủ tục), yếu tố được né tránh trong biểu đồ tuần tự.

Biểu đồ sau đây là một ví dụ cho một biểu đồ cộng tác, được chuẩn bị cũng cho một cảnh kịch rút tiền mặt như trong biểu đồ tuần tự của phần trước. Hãy quan sát các thứ tự số trong biểu đồ. Đầu tiên thủ tục WithdrawalReq() được gọi từ lớp khách hàng. Đó là lệnh gọi số 1. Bước tiếp theo trong tuần tự là hàm AskForPin(), số 1.1, được gọi

từ lớp ATM. Thông điệp trong biểu đồ được viết dưới dạng pin:= AskForPin(), thể hiện rằng "giá trị trả về" của hàm này chính là mã số mà lớp khách hàng sẽ cung cấp.

Hình cung bên lớp tài khoản biểu thị rằng hàm ComputeNetBalance() được gọi trong nội bộ lớp tài khoản và nó xử lý cục bộ. Thường thì nó sẽ là một thủ tục riêng (private) của lớp.



Hình 19.5.Một biểu đồ cộng tác của kích cảnh rút tiền ở máy ATM

19.8. Biểu đồ trạng thái (State Diagram)

Biểu đồ trạng thái nắm bắt vòng đời của các đối tượng, các hệ thống con (Subsystem) và các hệ thống. Chúng cho ta biết các trạng thái mà một đối tượng có thể có và các sự kiện (các thông điệp nhận được, các khoảng thời gian đã qua đi, các lỗi xảy ra, các điều kiện được thỏa mãn) sẽ ảnh hưởng đến những trạng thái đó như thế nào dọc theo tiến trình thời gian. Biểu đồ trạng thái có thể đính kèm với tất cả các lớp có những trạng thái được nhận diện rõ ràng và có lối ứng xử phức tạp. Biểu đồ trạng thái xác định ứng xử và miêu tả nó sẽ khác biệt ra sao phụ thuộc vào trạng thái, ngoài ra nó cũng còn miêu tả rõ những sự kiện nào sẽ thay đổi trạng thái của các đối tượng của một lớp.

19.8.1.Trạng thái và sự biến đổi trạng thái (State transition)

Tất cả các đối tượng đều có trạng thái; trạng thái là một kết quả của các hoạt động trước đó đã được đối tượng thực hiện và nó thường được xác định qua giá trị của các thuộc tính cũng như các nối kết của đối tượng với các đối tượng khác. Một lớp có thể có một thuộc tính đặc biệt xác định trạng thái, hoặc trạng thái cũng có thể được xác

định qua giá trị của các thuộc tính “bình thường" trong đối tượng. Ví dụ về các trạng thái của đối tượng:

- Hóa đơn (đối tượng) đã được trả tiền (trạng thái).
- Chiếc xe ô tô (đối tượng) đang đứng yên (trạng thái).
- Động cơ (đối tượng) đang chạy (trạng thái).
- Jen (đối tượng) đang đóng vai trò người bán hàng (trạng thái).
- Kate (đối tượng) đã lấy chồng (trạng thái).

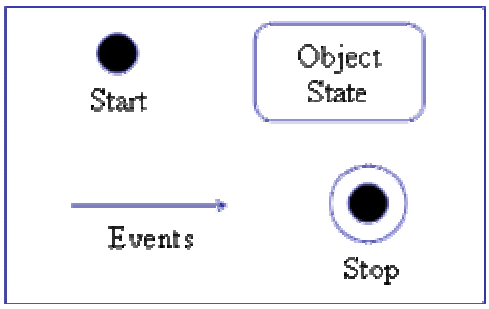
Một đối tượng sẽ thay đổi trạng thái khi có một việc nào đó xảy ra, thứ được gọi là sự kiện; ví dụ có ai đó trả tiền cho hóa đơn, bật động cơ xe ô tô hay là lấy chồng lấy vợ. Khía cạnh động có hai chiều không gian: tương tác và sự biến đổi trạng thái nội bộ. Tương tác miêu tả lối ứng xử đối ngoại của các đối tượng và chỉ ra đối tượng này sẽ tương tác với các đối tượng khác ra sao (qua việc gửi thông điệp, nối kết hoặc chấm dứt nối kết). Sự biến đổi trạng thái nội bộ miêu tả một đối tượng sẽ thay đổi các trạng thái ra sao – ví dụ giá trị các thuộc tính nội bộ của nó sẽ thay đổi như thế nào. Biểu đồ trạng thái được sử dụng để miêu tả việc bản thân đối tượng phản ứng ra sao trước các sự kiện và chúng thay đổi các trạng thái nội bộ của chúng như thế nào, ví dụ, một hóa đơn sẽ chuyển từ trạng thái chưa trả tiền sang trạng thái đã trả tiền khi có ai đó trả tiền cho nó. Khi một hóa đơn được tạo ra, đầu tiên nó bước vào trạng thái chưa được trả tiền.

19.8.2.Biểu đồ trạng thái

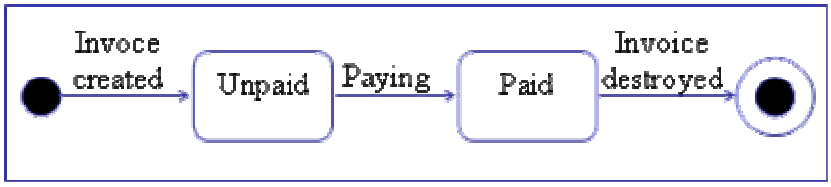
Biểu đồ trạng thái thể hiện những khía cạnh mà ta quan tâm tới khi xem xét trạng thái của một đối tượng:

- Trạng thái ban đầu
- Một số trạng thái ở giữa
- Một hoặc nhiều trạng thái kết thúc
- Sự biến đổi giữa các trạng thái
- Những sự kiện gây nên sự biến đổi từ một trạng thái này sang trạng thái khác

Hình sau sẽ chỉ ra các kí hiệu UML thể hiện trạng thái bắt đầu và trạng thái kết thúc, sự kiện cũng như các trạng thái của một đối tượng.

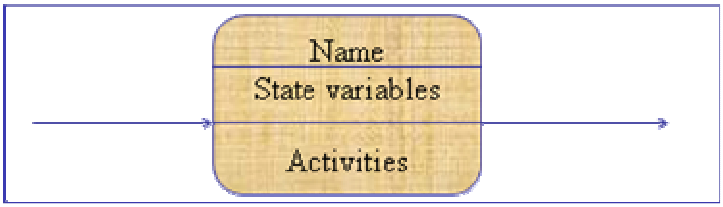


Các ký hiệu UML thể hiện bắt đầu, kết thúc, sự kiện và trạng thái của một đối tượng.



Biểu đồ trạng thái thực hiện hoá đơn.

Một trạng thái có thể có ba thành phần, như được chỉ trong hình sau :



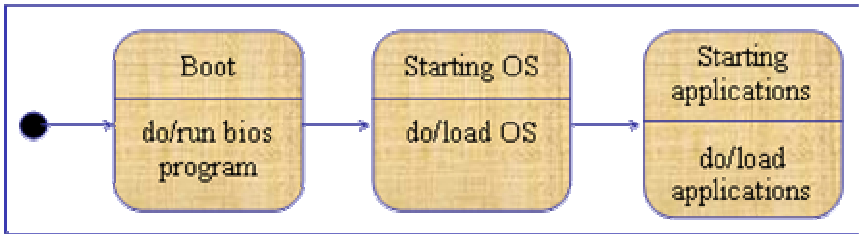
Các ngăn Tên, Biến trạng thái và hành động

Phần thứ nhất chỉ ra tên của trạng thái, ví dụ như chờ, đã được trả tiền hay đang chuyển động. Phần thứ hai (không bắt buộc) dành cho các biến trạng thái. Đây là những thuộc tính của lớp được thể hiện qua biểu đồ trạng thái; nhiều khi các biến tạm thời cũng tỏ ra rất hữu dụng trong trạng thái, ví dụ như các loại biến đếm (counter). Phần thứ ba (không bắt buộc) là phần dành cho hoạt động, nơi các sự kiện và các hành động có thể được liệt kê. Có ba loại sự kiện chuẩn hóa có thể được sử dụng cho phần hành động: *entry (đi vào), exit (đi ra),* và *do (thực hiện)*. Loại *sự kiện đi vào* được sử dụng để xác định các hành động khởi nhập trạng thái, ví dụ gán giá trị cho một thuộc tính hoặc gửi đi một thông điệp. *Sự kiện đi ra* có thể được sử dụng để xác định hành động khi rời bỏ trạng thái. *Sự kiện thực hiện* được sử dụng để xác định hành động cần phải được thực hiện trong trạng thái, ví dụ như gửi một thông điệp, chờ, hay tính toán. Ba loại sự kiện chuẩn này không thể được sử dụng cho các mục đích khác.

Một sự biến đổi trạng thái thường có một sự kiện đi kèm với nó, nhưng không bắt buộc. Nếu có một sự kiện đi kèm, sự thay đổi trạng thái sẽ được thực hiện khi sự kiện kia xảy ra. Một hành động loại *thực hiện* trong trạng thái có thể là một quá trình đang tiếp diễn (ví dụ chờ, điều khiển các thủ tục,...) phải được thực hiện trong khi đối

tượng vẫn ở nguyên trong trạng thái này. Một hành động *thực hiện* có thể bị ngắt bởi các sự kiện từ ngoài, có nghĩa là một sự kiện kiện gây nên một sự biến đổi trạng thái có thể ngưng ngắt một hành động thực hiện mang tính nội bộ đang tiếp diễn.

Trong trường hợp một sự biến đổi trạng thái không có sự kiện đi kèm thì trạng thái sẽ thay đổi khi hành động nội bộ trong trạng thái đã được thực hiện xong (hành động nội bộ kiểu đi vào, đi ra, thực hiện hay các hành động do người sử dụng định nghĩa). Theo đó, khi tất cả các hành động thuộc trạng thái đã được thực hiện xong, một sự thay đổi trạng thái sẽ tự động xảy ra mà không cần sự kiện từ ngoài.



Biến đổi trạng thái không có sự kiện từ ngoài. Sự thay đổi trạng thái xảy ra khi các hoạt động trong mỗi trạng thái được thực hiện xong.

19.8.3.Nhận biết trạng thái và sự kiện

Quá trình phát hiện sự kiện và trạng thái về mặt bản chất bao gồm việc hỏi một số các câu hỏi thích hợp:

- Một đối tượng có thể có những trạng thái nào?: Hãy liệt kê ra tất cả những trạng thái mà một đối tượng có thể có trong vòng đời của nó.

- Những sự kiện nào có thể xảy ra?: Bởi sự kiện gây ra việc thay đổi trạng thái nên nhận ra các sự kiện là một bước quan trọng để nhận diện trạng thái.

- Trạng thái mới sẽ là gì?: Sau khi nhận diện sự kiện, hãy xác định trạng thái khi sự kiện này xảy ra và trạng thái sau khi sự kiện này xảy ra.

- Có những thủ tục nào sẽ được thực thi?: Hãy để ý đến các thủ tục ảnh hưởng đến trạng thái của một đối tượng.

- Chuỗi tương tác giữa các đối tượng là gì?: Tương tác giữa các đối tượng cũng có thể ảnh hưởng đến trạng thái của đối tượng.

- Qui định nào sẽ được áp dụng cho các phản ứng của các đối tượng với nhau?: Các qui định kiềm tỏa phản ứng đối với một sự kiện sẽ xác định rõ hơn các trạng thái.

- Những sự kiện và sự chuyển tải nào là không thể xảy ra?: Nhiều khi có một số sự kiện hoặc sự thay đổi trạng thái không thể xảy ra. Ví dụ như bán một chiếc ô tô đã được bán rồi.

- Cái gì khiến cho một đối tượng được tạo ra?: Đối tượng được tạo ra để trả lời cho một sự kiện. Ví dụ như một sinh viên ghi danh cho một khóa học.

- Cái gì khiến cho một đối tượng bị hủy?: Đối tượng sẽ bị hủy đi khi chúng không được cần tới nữa. Ví dụ khi một sinh viên kết thúc một khóa học.

- Cái gì khiến cho đối tượng cần phải được tái phân loại (reclassfied)?: Những loại sự kiện như một nhân viên được tăng chức thành nhà quản trị sẽ khiến cho động tác tái phân loại của nhân viên đó được thực hiện.

19.8.4. Một số gợi ý cho việc tạo dựng biểu đồ trạng thái

Một khi mô hình đã được tạo nên, hãy nêu ra các câu hỏi và kiểm tra xem mô hình có khả năng cung cấp tất cả các câu trả lời. Qui trình sau đây cần phải được nhắc lại cho mỗi đối tượng.

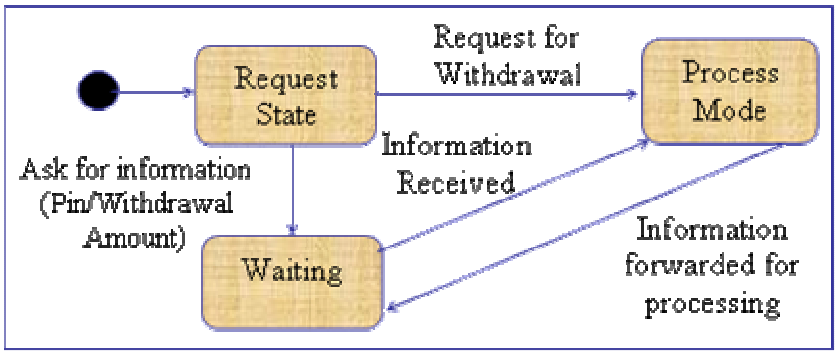
Chuyển biểu đồ tuần tự thành biểu đồ trạng thái

Hãy dõi theo một chuỗi các sự kiện được miêu tả trong biểu đồ, chuỗi này phải mang tính tiêu biểu cho các tương tác trong hệ thống. Hãy quan sát các sự kiện ảnh hưởng đến đối tượng mà ta đang nghiên cứu.

Hãy sắp xếp các sự kiện thành một đường dẫn, dán nhãn input (hoặc entry) và output (exit) cho các sự kiện. Khoảng cách giữa hai sự kiện này sẽ là một trạng thái.

Nếu cảnh kịch có thể được nhắc đi nhắc lại rất nhiều lần (vô giới hạn), hãy nối đường dẫn từ trạng thái cuối cùng đến trạng thái đầu tiên.

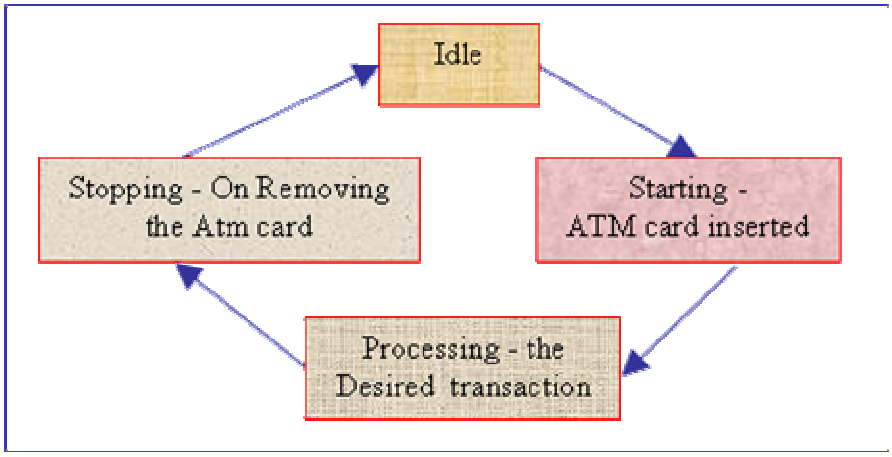
Biểu đồ sau đây chỉ ra biểu đồ trạng thái của một lớp máy ATM, được chiết suất từ biểu đồ tuần tự hoặc biểu đồ cộng tác đã được trình bày trong các phần trước.



Hình 19.6. Chuyển một biểu đồ tuần tự sang biểu đồ trạng thái

Nhận ra các vòng lặp (loop)

Một chuỗi sự kiện có thể được nhắc đi nhắc lại vô số lần được gọi là vòng lặp (loop).



Hình 19.7. Biểu đồ lặp

Chú ý:

- Trong một vòng lặp, chuỗi các sự kiện được nhắc đi nhắc lại cần phải đồng nhất với nhau. Nếu có một chuỗi các sự kiện khác chuỗi khác thì trường hợp đó không có vòng lặp.

- Lý tưởng nhất là một trạng thái trong vòng lặp sẽ có sự kiện kết thúc. Đây là yếu tố quan trọng, nếu không thì vòng lặp sẽ không bao giờ kết thúc.

Bổ sung thêm các điều kiện biên và các điều kiện đặc biệt

Sau khi đã hoàn tất biểu đồ trạng thái cho mọi đối tượng cần thiết trong hệ thống, đã đến lúc chúng ta cần kiểm tra, đối chứng chúng với điều kiện biên và các điều kiện đặc biệt khác, những điều kiện rất có thể đã chưa được quan tâm đủ độ trong thời gian tạo dựng biểu đồ trạng thái. Điều kiện biên là những điều kiện thao tác trên giá trị, đây là những giá trị nằm bên ranh giới của một điều kiện để quyết định về trạng thái của đối tượng, ví dụ như quy định về kỳ hạn của một tài khoản là 30 ngày thì ngày thứ 31 đối với tài khoản này sẽ là một điều kiện biên. Các điều kiện đặc biệt là những điều kiện ngoại lệ, ví dụ ngày thứ 30 của tháng 2 năm 2000 (nếu có một điều kiện thật sự như vậy tồn tại ngoài đời thực).

Trộn lẫn các cảnh kịch khác vào trong biểu đồ trạng thái

Một khi biểu đồ trạng thái cho một đối tượng đã sẵn sàng, chúng ta cần phải trộn những chuỗi sự kiện có ảnh hưởng đến đối tượng này vào trong biểu đồ trạng thái. Điều này có nghĩa là chúng ta cần phải quan sát các hiệu ứng gián tiếp của các sự kiện khác đối với đối tượng đang là chủ đề chính của biểu đồ trạng thái. Đây là việc quan trọng, bởi các đối tượng trong một hệ thống tương tác với nhau và vì các đối tượng khác cũng có khả năng gây nên sự kiện cho một đối tượng định trước, nên lối ứng xử này cũng cần phải được thể hiện trong biểu đồ trạng thái.

Điểm bắt đầu cho công việc này là:

- Ấn định một điểm bắt đầu chung cho tất cả các chuỗi sự kiện bổ sung.

- Xác định điểm nơi các ứng xử bắt đầu khác biệt với những ứng xử đã được mô hình hóa trong biểu đồ trạng thái.

Bổ sung thêm sự các biến đổi mới từ trạng thái này, trong tư cách một đường dẫn thay thế. Cần để ý đến những đường dẫn có vẻ ngoài đồng nhất nhưng thật ra có khác biệt trong một tình huống nhất định nào đó.

Hãy chú ý đến các sự kiện xảy ra trong những tình huống bất tiện. Mỗi sự kiện do khách hàng hay người sử dụng gây nên đều có thể sa vào trạng thái của các sự kiện bất tiện. Hệ thống không nắm quyền điều khiển đối với người sử dụng và người sử dụng có thể quyết định để làm nảy ra một sự kiện tại một thời điểm tiện lợi đối với anh ta. Ví dụ như khách hàng có thể quyết định kết thúc trước kỳ hạn một tài khoản đầu tư.

Một trường hợp khác cũng cần phải được xử lý là sự kiện do người sử dụng gây nên không thể xảy ra. Có một loạt các lý do (người sử dụng thiếu tập trung, buồn nản, lơ đãng...) khiến cho sự kiện loại này không xảy ra. Cả trường hợp này cũng phải được xử lý thấu đáo. Ví dụ một khách hàng thất bại trong việc báo cho nhà băng biết những mệnh lệnh của anh ta về kỳ hạn của tài khoản, một tấm séc được viết ra nhưng lại không có khả năng giải tỏa mức tiền cần thiết.

Nhìn theo phương diện các biểu đồ trạng thái như là một thành phần của mô hình động, cần chú ý những điểm sau:

- Biểu đồ trạng thái chỉ cần được tạo dựng nên cho các lớp đối tượng có ứng xử động quan trọng.

- Hãy thẩm tra biểu đồ trạng thái theo khía cạnh tính nhất quán đối với những sự kiện dùng chung để cho toàn bộ mô hình động được đúng đắn.

- Dùng các trường hợp sử dụng để hỗ trợ cho quá trình tạo dựng biểu đồ trạng thái.

- Khi định nghĩa một trạng thái, hãy chỉ để ý đến những thuộc tính liên quan.

19.9. Biểu đồ hoạt động (Activity Diagram)

Biểu đồ hoạt động nắm bắt hành động và các kết quả của chúng. Biểu đồ hoạt động tập trung vào công việc được thực hiện trong khi thực thi một thủ tục (hàm), các hoạt động trong một lần thực thi một trường hợp sử dụng hoặc trong một đối tượng. Biểu đồ hoạt động là một biến thể của biểu đồ trạng thái và có một mục tiêu tương đối khác, đó là nắm bắt hành động (công việc và những hoạt động phải được thực hiện) cũng như kết quả của chúng theo sự biến đổi trạng thái. Các trạng thái trong biểu đồ

hoạt động (được gọi là các trạng thái hành động) sẽ chuyển sang giai đoạn kế tiếp khi hành động trong trạng thái này đã được thực hiện xong (mà không xác định bất kỳ một sự kiện nào theo như nội dung của biểu đồ trạng thái). Một sự điểm phân biệt khác giữa biểu đồ hoạt động và biểu đồ trạng thái là các hành động của nó được định vị trong các ***luồng*** (***swimlane***). Một luồng sẽ gom nhóm các hoạt động, chú ý tới khái niệm người chịu trách nhiệm cho chúng hoặc chúng nằm ở đâu trong một tổ chức. Một biểu đồ hoạt động là một phương pháp bổ sung cho việc miêu tả tương tác, đi kèm với trách nhiệm thể hiện rõ các hành động xảy ra như thế nào, chúng làm gì (thay đổi trạng thái đối tượng), chúng xảy ra khi nào (chuỗi hành động), và chúng xảy ra ở đâu (luồng hành động).

Biểu đồ hoạt động có thể được sử dụng cho nhiều mục đích khác nhau, ví dụ như:

- Để nắm bắt công việc (hành động) sẽ phải được thực thi khi một thủ tục được thực hiện. Đây là tác dụng thường gặp nhất và quan trọng nhất của biểu đồ hoạt động.

- Để nắm bắt công việc nội bộ trong một đối tượng.

- Để chỉ ra một nhóm hành động liên quan có thể được thực thi ra sao, và chúng sẽ ảnh hưởng đến những đối tượng nằm xung quanh chúng như thế nào.

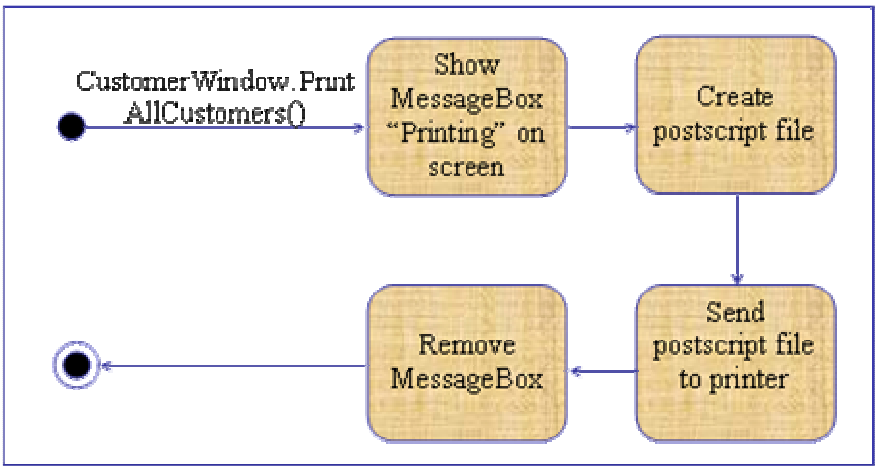
- Để chỉ ra một trường hợp sử dụng có thể được thực thể hóa như thế nào, theo khái niệm hành động và các sự biến đổi trạng thái của đối tượng.

- Để chỉ ra một doanh nghiệp hoạt động như thế nào theo các khái niệm công nhân (tác nhân), qui trình nghiệp vụ (workflow), hoặc tổ chức và đối tượng (các khía cạnh vật lý cũng như tri thức được sử dụng trong doanh nghiệp).

Biểu đồ hoạt động có thể được coi là một loại Flow chart. Điểm khác biệt là Flow Chart bình thường ra chỉ được áp dụng đối với các qui trình tuần tự, biểu đồ hoạt động có thể xử lý cả các các qui trình song song.

Hành động và sự thay đổi trạng thái

Một hành động được thực hiện để sản sinh ra một kết quả. Việc thực thi của thủ tục có thể được miêu tả dưới dạng một tập hợp của các hành động liên quan, sau này chúng sẽ được chuyển thành các dòng code thật sự. Theo như định nghĩa ở phần trước, một biểu đồ hoạt động chỉ ra các hành động cùng mối quan hệ giữa chúng và có thể có một điểm bắt đầu và một điểm kết thúc. Biểu đồ hoạt động sử dụng cũng cùng những ký hiệu như trong biểu đồ trạng thái bình thường.



Hình 19.8. Khi một người gọi tác vụ PrintAllCustomer (trong lớp CustomerWindow), các hành động khởi động. hành động đầu tiên là hiện một hộp thông báo lên màn hình; hành động thứ hai là tạo một tập tin postscript; hành động thứ ba là gởi file postscript đến máy in; và hành động thứ tư là xóa hộp thông báo trên màn hình. Các hành động được chuyển tiếp tự động; chúng xảy ra ngay khi hành động trong giai đoạn nguồn được thực hiện.

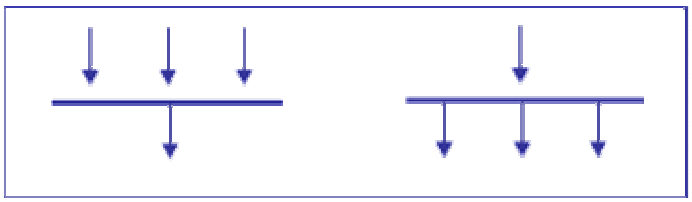
Các sự thay đổi có thể được bảo vệ bởi các điều kiện canh giữ (Guard-condition), các điều kiện này phải được thỏa mãn thì sự thay đổi mới nổ ra. Một ký hiệu hình thoi được sử dụng để thể hiện một quyết định. Ký hiệu quyết định có thể có một hoặc nhiều sự thay đổi đi vào và một hoặc nhiều sự thay đổi đi ra được dán nhãn đi kèm các điều kiện bảo vệ. Bình thường ra, một trong số các sự thay đổi đi ra bao giờ cũng được thỏa mãn (là đúng).

Một sự thay đổi được chia thành hai hay nhiều sự thay đổi khác sẽ dẫn đến các hành động xảy ra song song. Các hành động được thực hiện đồng thời, mặc dù chúng cũng có thể được thực hiện lần lượt từng cái một. Yếu tố quan trọng ở đây là tất cả các thay đổi đồng thời phải được thực hiện trước khi chúng được thống nhất lại với nhau (nếu có). Một đường thẳng nằm ngang kẻ đậm (còn được gọi là thanh đồng hộ hóa – Synchronisation Bar) chỉ rằng một sự thay đổi được chia thành nhiều nhánh khác nhau và chỉ ra một sự chia sẻ thành các hành động song song. Cũng đường thẳng đó được sử dụng để chỉ ra sự thống nhất các nhánh.

Kí hiệu UML cho các thành phần **căn bản của biểu đồ hoạt động:**

- *Hoạt động* (*Activity*): là một qui trình được định nghĩa rõ ràng, có thể được thực thi qua một hàm hoặc một nhóm đối tượng. Hoạt động được thể hiện bằng hình chữ nhật bo tròn cạnh.

- Thanh đồng bộ hóa (Synchronisation bar): chúng cho phép ta mở ra hoặc là đóng lại các nhánh chạy song song nội bộ trong tiến trình.



Hình 19.9. Thanh đồng bộ hóa

- *Điều kiện canh giữ (Guard Condition)*: các biểu thức logic có giá trị hoặc đúng hoặc sai. Điều kiện canh giữ được thể hiện trong ngoặc vuông, ví dụ:

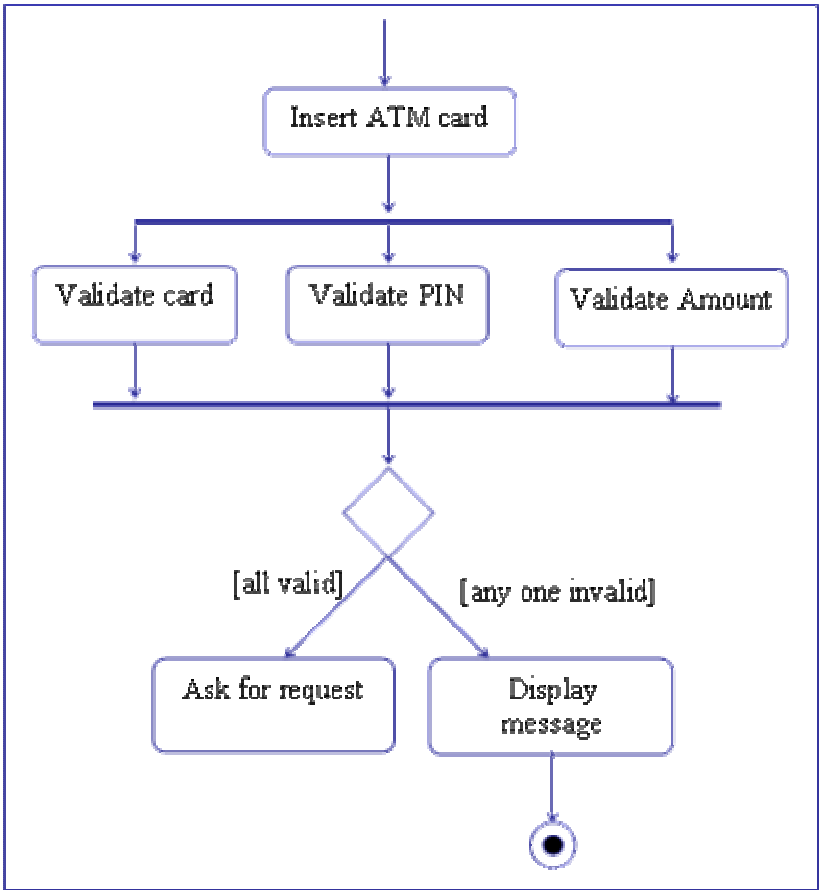
[Customer existing].

- *Điểm quyết định (Decision Point)*: được sử dụng để chỉ ra các sự thay đổi khả thi. Kí hiệu là hình thoi.

Hình sau đây miêu tả một đoạn biểu đồ hoạt động của máy ATM. Sau khi thẻ được đưa vào máy, ta thấy có ba hoạt động song song:

- Xác nhận thẻ
- Xác nhận mã số PIN
- Xác nhận số tiền yêu cầu được rút

Chỉ khi sử dụng biểu đồ hoạt động, các hoạt động song song như vậy mới có thể được miêu tả. Mỗi một hoạt động xác nhận bản thân nó cũng đã có thể là một quá trình riêng biệt.



Hình 19.10**.** Biểu đồ hoạt động của máy ATM

19.10. Vòng đời đối tượng (Object lifecycle)

Vòng đời mà một đối tượng đi qua phụ thuộc vào loại đối tượng. Có hai loại vòng đời khác nhau đối với một đối tượng: vòng đời sinh ra rồi chết đi; và vòng đời vòng lặp.

19.10.1 Vòng đời sinh ra và chết đi:

Trong một vòng đời sinh ra rồi chết đi:

- Sẽ có một hay nhiều trạng thái nơi đối tượng bắt đầu tồn tại. Những trạng thái này được gọi là trạng thái tạo ra đối tượng.

- Sẽ có một hay nhiều trạng thái đóng tư cách là điểm kết thúc cho vòng đời của một đối tượng. Những trạng thái này được gọi là trạng thái kết thúc.

Có hai loại trạng thái kết thúc. Một dạng trạng thái kết thúc là loại nơi đối tượng bị hủy và không tiếp tục tồn tại nữa. Loại thứ hai là dạng trạng thái kết thúc mà sau đó đối tượng sẽ được lưu trữ lại hoặc chuyển sang trạng thái im lặng. Đối tượng tiếp tục tồn tại nhưng không tiếp tục thể hiện ứng xử động.

Sau trạng thái khởi tạo và trước trạng thái kết thúc, đối tượng có thể đi qua một hoặc là nhiều trạng thái trung gian. Tại mỗi một thời điểm, đối tượng phải ở một trạng thái hiện thời.

Không có một điểm nào sau trạng thái khởi tạo và trước trạng thái kết thúc mà đối tượng lại không có trạng thái.

Ví dụ cho đối tượng có vòng đời sinh ra và chết đi: khách hàng, tài khoản.

19.10.2. Vòng đời lặp

Khác với trường hợp sinh ra và chết đi, trong vòng đời vòng lặp:

- Đối tượng được coi là đã luôn luôn tồn tại ở đây và sẽ còn mãi mãi tiếp tục tồn tại.

- Không có trạng thái khởi tạo cũng không có trạng thái kết thúc.

Mặc dù thật ra đối tượng đã được tạo ra tại một thời điểm nào đó và cũng sẽ thật sự bị hủy diệt tại một thời điểm nào đó, nhưng nó vẫn được coi là luôn luôn tồn tại và có mặt. Thường thì những đối tượng tỏ ra có một vòng đời vòng lặp sẽ được tạo ra tại thời điểm cài đặt hệ thống và sẽ chết đi khi hệ thống kết thúc.

Một đối tượng với vòng đời vòng lặp sẽ có một hoặc là nhiều trạng thái "ngủ yên". Đó là những trạng thái nơi đối tượng nằm chờ một sự kiện xảy ra. Bên cạnh đó, đối tượng cũng luôn luôn ở trạng thái hiện thời.

Ví dụ cho đối tượng có vòng đời lặp lại: máy rút tiền tự động (ATM), nhân viên thu ngân.

19.11. Xem xét lại mô hình động

19.11.1. Thẩm vấn biểu đồ trạng thái

Sau khi đã hoàn tất các thành phần căn bản của mô hình động như các biểu đồ tuần tự, biểu đồ cộng tác, biểu đồ trạng thái và biểu đồ hoạt động, nhóm phát triển có thể phác thảo biểu đồ thành phần và biểu đồ triển khai. Biểu đồ triển khai có thể được coi là biểu đồ cuối cùng trong mô hình động. Tới thời điểm này, có thể coi là ta đã hoàn tất một phiên bản của mô hình động.

Phần quan trọng nhất trong mô hình này là biểu đồ trạng thái. Hãy tìm câu trả lời cho một loạt các câu hỏi để xác định xem biểu đồ trạng thái đã đúng đắn và có một mức độ chi tiết thích hợp hay chưa. Công việc này cần nhắm tới hai mục đích:

- Kiểm tra tính trọn vẹn của mô hình

- Đảm bảo mọi điều kiện gây lỗi đã được xử lý

Trong giai đoạn này, có thể sẽ có các cảnh kịch (scenario) mới xuất hiện và gia nhập phạm vi quan sát của chúng ta, nếu trước đó có một số trạng thái chưa được xử lý. Những tình huống loại này là loại vấn đề có thể được giải quyết, song có thể né tránh qua việc xác định thật đầy đủ các sự kiện và trạng thái.

19.11.2. Phối hợp sự kiện

Bước cuối cùng là một vòng kiểm tra bổ sung nhằm đảm bảo tính đúng đắn của mô hình động:

- Kiểm tra để đảm bảo mỗi thông điệp đều có đối tượng gửi và đối tượng nhận. Trong một số trường hợp, số liệu chính xác của những đối tượng nhận sự kiện có thể không được biết tới, nhưng chúng ta phải đảm bảo rằng chúng ta biết những lớp nào sẽ xử lý những sự kiện này.

- Hãy nghiên cứu mô hình theo khía cạnh trạng thái, tìm ra những trạng thái không có trạng thái dẫn trước và không có trạng thái tiếp theo. Những trạng thái thái này rất có thể là trạng thái khởi đầu hoặc trạng thái kết thúc. Mặc dù vậy, nếu trạng thái đó không thuộc về một trong hai loại trạng thái kia, rất có thể đây là một triệu chứng cho thấy mô hình còn thiếu điều gì đó.

- Nhìn chung, tất cả các trạng thái thường đều có trạng thái dẫn trước và trạng thái tiếp sau.

- Hãy lần theo hiệu ứng của các sự kiện đi vào (entry) để đảm bảo là chúng tương thích với các trường hợp sử dụng nơi chúng xuất phát. Để làm điều này, hãy lần theo một sự kiện từ một đối tượng này đến đối tượng khác, kiểm tra xem mỗi sự kiện có phù hợp với trường hợp sử dụng hay không. Trong trường hợp có mâu thuẫn, hãy sửa lại biểu đồ trạng thái hoặc trường hợp sử dụng để đảm bảo sự nhất quán.

- Kiểm tra lại những lỗi đồng bộ, có thể chúng là kết quả của một sự kiện không chờ đợi.

19.11.3. Bao giờ thì sử dụng biểu đồ nào

Không cần phải vẽ tất cả các loại biểu đồ động cho tất cả các loại hệ thống. Mặc dù vậy, trong một số trường hợp khác nhau chúng ta nhất thiết phải cần đến một số loại biểu đồ động nhất định. Sau đây là một vài lời mách bảo có thể giúp giải thích một vài điều còn chưa thông tỏ về việc sử dụng các loại biểu đồ động.

Biểu đồ tuần tự và biểu đồ cộng tác được vẽ khi chúng ta muốn xem xét ứng xử động của nhiều đối tượng/ lớp trong nội bộ một cảnh kịch của một trường hợp sử dụng. Biểu đồ tuần tự và biểu đồ cộng tác rất hữu dụng trong việc chỉ ra sự cộng tác giữa các đối tượng, nhưng chúng lại không hữu dụng khi muốn miêu tả ứng xử chính xác của một đối tượng.

Biểu đồ trạng thái được sử dụng để thể hiện ứng xử chính xác của một đối tượng.

Biểu đồ hoạt động được sử dụng để thể hiện lối ứng xử xuyên suốt nhiều trường hợp sử dụng hoặc các tiểu trình xảy ra song song của một lần thực thi.Biểu đồ thành phần và biểu đồ triển khai được sử dụng để chỉ ra mối quan hệ vật lý giữa phần mềm và các thành phần phần cứng trong hệ thống.

19.11.4. Lớp con và biểu đồ trạng thái

Tất cả các lớp con đều thừa kế cả thuộc tính cũng như các thủ tục của lớp cha. Vì vậy, một lớp con cũng sẽ thừa kế cả mô hình động của lớp cha.

Ngoài biểu đồ trạng thái được thừa kế, lớp con cũng có biểu đồ trạng thái riêng của nó. Biểu đồ trạng thái của một lớp cha sẽ được mở rộng để bao chứa lối ứng xử chuyên biệt của lớp con.

Biểu đồ trạng thái của lớp con và biểu đồ trạng thái của lớp cha phải được bảo trì riêng biệt và độc lập. Biểu đồ trạng thái của lớp con cần phải được định nghĩa sử dụng các thuộc tính của lớp con chứ không phải chỉ bằng các thuộc tính của lớp cha. Mặt khác, vẫn có một sự móc nối ngoài ý muốn của lớp cha đến với lớp con thông qua các thuộc tính mà chúng sử dụng chung, ví dụ chỉ nên xem xét biểu đồ trạng thái cho các tài khoản có kỳ hạn theo phương diện sự thay đổi của chính các thuộc tính của chúng, chứ không phải là thuộc tính của lớp cha. Ta phải thực hiện như vậy để né tránh trường hợp trộn lẫn thuộc tính của lớp con và lớp cha.

Việc tuân thủ quy tắc kể trên trong quá trình vẽ biểu đồ trạng thái cho một lớp con sẽ đảm bảo tính môđun cho động tác mở rộng của bạn.

19.12. Phối hợp mô hình động và mô hình đối tượng

Khi kết hợp giữa các mô hình đối tượng và mô hình động, mỗi sự kiện trong mô hình động cần phải tương thích với một thủ tục trong mô hình đối tượng. Từ đó suy ra, mỗi sự thay đổi về mặt trạng thái trong mô hình động cần phải phù hợp với một thủ tục của đối tượng. Hành động phụ thuộc vào trạng thái của đối tượng và vào sự kiện.

Mối quan hệ giữa mô hình đối tượng và mô hình động có thể được miêu tả như sau:

- Mô hình đối tượng là cơ cấu (framework) cho mô hình động.

- Mô hình động xác định các chuỗi thay đổi được phép xảy ra đối với các đối tượng trong mô hình đối tượng.

- Mô hình động bị hạn chế chỉ trong những đối tượng có mặt trong mô hình đối tượng cũng như cấu trúc của chúng.

- Không thể có một mô hình động cho một đối tượng không tồn tại trong mô hình đối tượng. Có một mối quan hệ 1-1 giữa mô hình đối tượng và mô hình động.

- Mô hình động chính là mô hình đối tượng cộng thêm với phần ứng xử "sống".

- Mô hình đối tượng miêu tả sự khác biệt giữa các đối tượng như là sự khác biệt giữa các lớp. Khi một đối tượng ứng xử khác một đối tượng khác thì mỗi đối tượng trong số đó sẽ có một lớp riêng.

- Mặc dù vậy, trong mô hình động, sự khác biệt trong ứng xử động sẽ được mô hình hóa thành các trạng thái khác nhau của cùng một lớp.

19.13. Tóm tắt về mô hình động

Tất cả các hệ thống đều có cấu trúc tĩnh và có ứng xử động. Cấu trúc có thể được miêu tả qua các phần tử mô hình tĩnh, ví dụ như lớp, quan hệ giữa các lớp, nút mạng và thành phần. Khái niệm ứng xử miêu tả các phần tử mô hình trong nội bộ cấu trúc sẽ tương tác với nhau dọc theo tiến trình thời gian ra sao. Đó thường là những tương tác được xác định trước và có thể được mô hình hóa. Mô hình hóa ứng xử động của một hệ thống gọi là mô hình động, được UML hỗ trợ. Có tất cả bốn loại biểu đồ khác nhau, mỗi loại với một mục đích khác nhau: biểu đồ trạng thái , biểu đồ tuần tự, biểu đồ cộng tác và biểu đồ hoạt động.

Biểu đồ trạng thái được sử dụng để miêu tả lối ứng xử cũng như các trạng thái nội bộ trong một lớp (nó cũng có thể được sử dụng cho các hệ thống con hoặc cho toàn bộ hệ thống). Nó tập trung vào khía cạnh các đối tượng theo tiến trình thời gian sẽ thay đổi các trạng thái của chúng ra sao tùy theo những sự kiện xảy ra, lối ứng xử cũng như các hành động được thực hiện trong các trạng thái, và bao giờ thì sự thay đổi trạng thái xảy ra. Một sự kiện có thể nổ ra khi một điều kiện trở thành được thỏa mãn, khi nhận một tín hiệu hoặc lệnh gọi thủ tục, hoặc là khi một khoảng thời gian định trước qua đi.

Biểu đồ tuần tự được sử dụng để miêu tả một nhóm các đối tượng sẽ tương tác với nhau trong một cảnh kịch riêng biệt như thế nào. Nó tập trung vào chuỗi thông điệp, tức là câu hỏi các thông điệp được gửi và nhận giữa một nhóm các đối tượng như thế nào. Biểu đồ tuần tự có hai trục; trục dọc chỉ thời gian và trục nằm ngang chỉ ra các đối tượng tham gia cảnh kịch. Khía cạnh quan trọng nhất của một biểu đồ tuần tự là thời gian.

Biểu đồ cộng tác được sử dụng để miêu tả các đối tượng tương tác với nhau trong không gian bộ nhớ (space), có nghĩa là bên cạnh các tương tác động, nó còn miêu tả rõ ràng các đối tượng được nối kết với nhau như thế nào. Trong biểu đồ cộng tác không có trục cho thời gian; thay vào đó, các thông điệp sẽ được đánh số để tạo chuỗi.

Biểu đồ hoạt động được sử dụng để miêu tả sự việc xảy ra ra sao, công việc được thực hiện như thế nào. Biểu đồ hoạt động cũng có thể được sử dụng cho các thủ tục, các lớp, các trường hợp sử dụng, và cũng có thể được sử dụng để chỉ ra các quy trình nghiệp vụ (workflow).